Faculty of Engineering and Information Technology
University of Technology Sydney

Single Image Super-Resolution via Deep Dense Network

A thesis submitted in partial fulfillment of
the requirements for the degree of
Master of Engineering

by

Jialiang Shen

November 2019

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I, Jialiang Shen declare that the thesis, is submitted in fulfilment of the requirements for the award of Master of Engineering, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature of Candidate

Production Note:
Signature removed prior to publication.

Date: 6/11/2019

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Associate Prof. Jian Zhang and my senior fellow Yucheng Wang in Baidu for the continuous support of my master study and research, for their patience, motivation, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having better advisors and mentors for my master study.

I also would like to appreciate my co-supervisor Dr. JingSong Xu for his expert and sincere help in the project and providing me with continuous support throughout my master study and research.

I thank my fellow labmates in Global Big Data and Technology Center: Zhibin Li, Xiaoshui Huang and Muming Zhao for the stimulating discussions, and supportive helps in the research periods and thesis writing.

Last but not the least, I would like to thank my parents, for their unconditional support, both financially and emotionally throughout the whole master studying.

Jialiang Shen
June 2019 @ UTS

Contents

List of Figures

List of Tables

List of Publications

Journals

- Yucheng Wang, **Jialiang Shen**, Jian Zhang (2019), ASDN: One Deep Convolutional Network for Any-Scale Image Super-Resolution. *submitted to* 'Transactions on Multimedia (**TMM**)'. IEEE.

Conferences

- **Jialiang Shen**, Yucheng Wang, Jian Zhang (2018), Memory-Optimized Deep Dense Network for Super-Resolution. *in* 'International Conference on Digital Image Computing: Techniques and Applications (**DICTA**)'. IEEE.
- Yucheng Wang, **Jialiang Shen**, Jian Zhang (2018), Deep Bi-Dense Network for Image Super-Resolution. *in* 'International Conference on Digital Image Computing: Techniques and Applications (**DICTA**)'. IEEE.

Abstract

Image Super-Resolution (SR) is a research field of computer vision, which enhances the resolution of an imaging system. The need for high resolution is common in computer vision applications for better performance in pattern recognition and analysis of images. However, recovering of the HR image from LR image is a highly ill-posed problem. In this thesis, the image SR problem is solved from three aspects with deep dense network models, including improving reconstruction accuracy, optimizing model training-time memory consumption, and extending effective SR scale ranges. Chapter 1 introduces the importance of image SR reconstruction and summarizes the challenges of image SR problem. Chapter 2 reviews the existing image SR methods, analyses their limitations and explains some related fundamental theories. Chapter 3 proposes a bi-dense model to improve image SR performance based on the dense connections for feature reuse. The bi-dense network does not only reuse local feature layers in the dense block, but also reuses the block information in the network to archive excellent performance with a moderate number of parameters. Chapter 4 evaluates the memory consumption of the vanilla dense model for image SR. For solving this problem, we introduce shared memory strategy into image SR by proposing a memory-optimized deep dense network. Chapter 5 discovers most of the deep SR methods are inefficient or impractical for generating SR of any scale factor, and proposes a novel Any-Scale Deep Network (ASDN), which requires few training scales to achieve one unified network for any-scale SR. In order to design such a powerful network architecture, we propose Laplacian

Frequency Representation to predict SR results of the small ratio range and Recursive Deployment for SR of any larger scale. In this way, the required training data and update periods are substantially decreased to optimize the any-scale SR network. All these algorithms are aimed to solve the single image SR problem. These algorithms are tested on many public datasets and the results on those datasets demonstrate superior performance of our approach over the state-of-the-art methods and validate the effectiveness and correctness of our methods.

Chapter 1

Introduction

1.1 Background

Single-image super-resolution (SR) is a branch of image reconstruction that generates a plausible and visually pleasing high-resolution (HR) output image from a low-resolution (LR) input image. The desire for high image resolution from two main application areas: improvement of pictorial information for human understanding and helping representation for automatic machine perception. Image resolution describes the details contained in an image, the higher the resolution, the more image details. The image resolution is initially restricted by the image sensors or the imaging acquisition devices. The number of sensor elements per unit area determines image pixel density. Therefore, an imaging device with inadequate sensors will generate low resolution images with blurring effects. In order to increase the image resolution, one brutal force way is to increase the sensor density by reducing the sensor size. However, as the sensor size decreases, the amount of light incidenting on each sensor also decreases, causing shot noise. Therefore, the hardware limitation of the size of the sensor restricts the image quality, which needs some image technical methods known as super-resolution (SR) to reconstruct HR images from LR observations.

In SR it is generally assumed that the LR image is a bicubic downsam-

pled version of HR. SR methods are mainly concerned with upscaling the LR image without losing the sharpness of the original HR image. Notorious super-resolution algorithms were published in the literatures, including interpolation algorithms, optimization algorithms and learning based methods. Interpolation-based methods (Keys 1981), such as nearest neighbouring interpolation, bilinear interpolation, and bicubic interpolation , which assume that the intensity at a position on the HR image can be interpolated from its neighbouring pixels on the corresponding LR image. Optimisation based (Peleg & Elad 2014) methods often model the problem by finding the optimum of an energy function, which consists of a data observation term from LR image and a regularisation term from various assumptions and hypotheses. Learning based methods (Tai, Yang & Liu n.d.), directly learn the mapping function from the LR image to the HR image. This thesis aims to adopt novel deep learning models to solve image SR tasks from three aspects with three kinds of solutions.

Driven by the emergence of large-scale data sets and fast development of computation power, learning-based methods especially using deep neural networks have significantly improved performance in terms of peak signal-to noise ratio (PSNR) (Hore & Ziou 2010) in the SR problem. Convolutional neural network (CNN) was initially introduced in image super-resolution to learn the mapping functions between LR and HR images by (Dong, Loy, He & Tang 2014) and achieved excellent results compared to other former handcrafted algorithms. Then, the deeper structure networks VDSR (Kim, Kwon Lee & Mu Lee 2016*a*) and EDSR (Lim, Son, Kim, Nah & Lee 2017) were proposed by creating short skip connections from the early to later layers to boost the reconstruction accuracy. Later on, more skip connection models were proposed to further improve SR performance by enhancing the gradient and information proceeding and propagation. Inspired by the connection structure, we embed them into the model design and focus on the three problems confronting efficient SR methods:

(1) Reconstruction Accuracy: SR is an ill-posed inverse transformation,

which requires precise scientific model and statistic calculation. The reconstruction accuracy is always the main evaluation metric of SR methods and hundreds of methods have been proposed to update the benchmark each year. In order to prove the effectiveness of proposed network structure, we need to compare with the start-of-art performance to claim the basic achievement.

(2) Memory Consumption: As the network goes deeper for the higher reconstruction accuracy, the implementation of models can require a significant amount of GPU memory, which is expensive and limits the further development of the larger models. However many intermediate feature maps (e.g. ReLU, Batch Normalization etc.) responsible for most of the memory consumption are relatively cheap to compute.

(3) Scale Limitation: Most of the deep SR methods require the scale-specific networks trained independently for various scales, which are inefficient and impractical for SR of any scale factor. It takes countless training samples and optimization time to generate one network for any-scale SR with the existing deep SR methods. However, the real-life applications always upscale the LR images into arbitrary size HR images, which need efficient any-scale SR methods.

In addition, these three problems contain each other. For example, network goes deeper and wider to improve image reconstruction accuracy and the required GPU memory also increases. When one network is able to embed any-scale SR information, training data of scales in a large range will slow down network convergence, which influences the SR reconstruction accuracy on specific scales. In this thesis, we propose three individual methods for these problems and minimize the regression of performance.

In order to achieve such a tradeoff among performance, memory, and scale range, dense network maybe a good choice. Dense connection is a connection method to further improve SR performance (Tong, Li, Liu & Gao 2017), which makes each layer obtain additional inputs from the concatenation of all the preceding layers and passes its own feature-maps to all the subsequent layers. Therefore, each layer is fully connected with others by the

dense connection and the learning process at each contextual level can encode more useful information to help improve the performance. In this thesis, we will solve all the interested problems with dense mechanism, by proposing a novel network structure embedding dense connections, arguing the memory consumption of the a dense SR network and constructing an any-scale SR network with a dense model.

1.2 Research Issues

Based on the aforementioned current research limitations, we present these following research issues:

1.2.1 How to choose the network structure

Image Super-Resolution is a highly ill-posed problem, which requires a large amount of hidden parameters to approximate the mapping function between LR and HR images. The existing SR methods have evaluated the relation between deep network structure and learning performance. It is a general impression that networks having the deeper structure and larger parameters will lead to the better reconstruction accuracy. However, merely stacking more convolutional layers in the deep network is unable to archive better performance than the relative shallow network. Recently skip connections were proposed to help feature flow across the network. Therefore, in order to improve the reconstruction accuracy, it is important to elaborately design the deep network structure. The model does not only have strong learning abilities, but also meets the criteria of the deeper network, the better performance.

1.2.2 How to reduce the memory consumption

As SR network goes deeper, it is a common phenomenon that GPU memory may become a limited resource for the network development. Many existing

researches observed that the intermediate feature maps are responsible for the most of the memory consumption, such as Concat, ReLU and Batch Normalization (BN) layers. Many deep learning frameworks keep these intermediate feature maps allocated in GPU memory for use during back-propagation. Therefore, once a convolutional layer is added in the network, it will at least take triple times memory in GPU for storing the intermediate ReLU and BN results and network parameters increase exponentially. Furthermore, deep learning methods for image SR require more GPU memory than high-level computer vision tasks since the size of feature maps will not decrease as they are propagated to the network end. Thus it is important to reduce the memory consumption by minimizing the storage for intermediate results, which can be stored in a shared memory allocation and re-computed before back-propagation.

1.2.3 How to be adaptive to input with different scales

Many existing SR methods are based on image context to learn the mapping function between LR and HR scale patches, which need the image information of all the interested scales (e.g. dictionary learning (Yang, Wright, Huang & Ma 2010), self-example learning (Zhu, Guo, Yu & Chen 2014), random forest (Schulter, Leistner & Bischof 2015), and convolutional neural network (Dong et al. 2014)). However, it is common for some image applications to zoom images into random sizes to observe the details. Therefore, one model for any-scale image SR is needed. But it is inefficient to prepare all training samples of these scales and impractical to optimize a network with all these scales to generate such a model for any-scale SR. Thus, it is important to use a small amount of training scales to effectively generate the any scale SR model.

1.3 Research Contributions

- Proposed a novel model called DBDN for image SR task. The model does not only reuse local feature layers in the dense block, but also reuses the block information in the network to archive excellent performance with moderate number of parameters (chapter 3);
- Proposed a compact intra-dense block where each layer is connected to other layers to learn local features. Then, in order to preserve feature information and keep the model compact, we use the compressed concatenation of the output of all layers in the block as the output.(chapter 3) ;
- Introduced an inter-block dense net for high-level feature learning. Since the features learned in the early blocks of the network matter to the task of SR, we use inter-block dense connection to allow all of the early block information to be reused to learn the later block features.(chapter 3);
- Proposed a novel SR model with dense blocks to reduce redundant features learning and minimize the model size. (chapter 4);
- Train the SR network adopting shared memory allocation strategy to store the concatenated features and Batch Normalization features to reduce the training memory cost of network (chapter 4);
- Proposed a Laplacian Frequency Representation mechanism, which theoretically and experimentally solves the continuous scale SR reconstruction problem. The HR images of the continuous scales are based on the weighted interpolation of their two neighbouring Laplacian frequency levels. (chapter 5);
- Introduced Recursive Deployment for generating the HR images of any upsampling ratio. The HR images of the larger scales can be gradually upsampled and recursively deployed. (chapter 5);

- Proposed an Any-Scale Deep Network (ASDN), which combines the bi-dense structure in the first work, channel and spatial attention modules (Hu, Li, Huang & Gao 2018) to highlight high-frequency components for improving the SR reconstruction accuracy. Our model can generate SR predictions of any random scale with one unified network. However, the existing CNN-based SR methods generally ignore some possible scales for any-scale SR in the theory and experiment (chapter 5).

1.4 Thesis Structure

The thesis is structured as follow:

The chapter 2 provides the literature review of the definition of image super-resolution and various image super-resolution methods, including interpolation-based, optimization-based and learning-based models. We then focus on deep learning based models for image SR. The development and relationship of the models are also explored, specially about the dense structure networks for image SR. Besides, we review the some fundamental theories related to our methods in the thesis, including memory optimization, Laplacian frequency representation, recursive deployment.

The chapter 3 presents the deep bi-dense model for image SR. The model is designed to improve SR performance with compact parameters by enhancing feature reusing across networks via inter-block dense connections and intra-block dense connections. In comparison to recent dense models, the experimental results demonstrate that our model has the better quality of PSNR and SSIM with moderate parameter size.

The chapter 4 proposes to use the memory sharing strategy to reduce GPU memory consumption of dense models in SR tasks. To reduce the memory consumption, a memory-economic dense model and shared memory allocation strategy are proposed. The experimental results show the proposed network achieves promising performance on the benchmark datasets with less

training memory.

The chapter 5 presents the any-scale SR model via deep dense model. This model is able to generate SR predictions of arbitrary scales with one unified network by training with SR samples of only several scales. Our method is compared with the fixed integer scales and any decimal scale on the commonly used benchmarks, which shows its superior to the existing deep methods on SR of fixed integer scales and arbitrary decimal scales.

The chapter 6 concludes the thesis and outlines the contribution of the work.

Chapter 2

Literature Review and Mathematical Background

Image Super-Resolution has been an active topic over the past two decades, and there are hundreds of thousands of methods were proposed to address this problem. The chapter reviews the existing related works, in regarding with different image Super-Resolution types, deep convolutional neural networks, dense-based models, and the foundational theories related to our methods.

2.1 Techniques for Image SR

Single Image Super-Resolution (SISR) has endured much development for a long period. There are many techniques to tackle this problem. The preview works can be classified into the following types:

2.1.1 Interpolation-Based Methods

Interpolation-based methods usually represent the missing pixels in the high-resolution image with a function. Bilinear and bicubic are the commonly used vanilla methods, which are theoretically simple and computationally efficient. However, these methods always generate blurring and jagged artifacts, due to the over-smooth assumptions. To overcome the blurring artifacts,

edge-directed interpolation methods were proposed to sharp the reconstructed images. Directional Bilinear Interpolation (Yang, Kim & Jeong 2008) and Fast edge-oriented algorithm (Chen, Huang & Lee 2005) estimate the edge orientation and interpolate along the edge orientation. The interpolation accuracy is influenced by the estimation of edge orientation, however, edges in natural images appear to be blurred, aliased or noisy. To improve the estimation accuracy of edge orientation, many methods were proposed to fuse the edge with statistical-cased approaches, such as (Zhang & Wu 2006). Later on, new edge-directed interpolation, Wiener filter (Li & Orchard 2001) was designed to estimate the HR interpolation with weighted nearest neighbors based on the geometric duality of edges. Then soft-decision adaptive interpolation (SAI) (Hung & Siu 2012*a*) was proposed to interpolate a block of pixels at one time using statistical consistency. Robust soft-decision adaptive interpolation (RSAI) (Hung & Siu 2012*b*) adds the weights in the cost function of SAI to adaptively weight the outliers for image interpolation. The SAI and RSAI are able to produce great reconstruction quality, but the computation is cost. To reduce the computation, bilateral soft-decision adaptive interpolation (BSAI) (Hung & Siu 2012*a*) was proposed to replace the least squares estimation with bilateral filter and reduce to a single parameter estimation. But the computational cost is still not enough for real-time applications (e.g. high-definition TVs upsampling LR video sequence). For the real time applications, nonlocal means interpolation (weighted sum filter) (Hung & Siu 2011) was directly used to upsample videos, due to the combination of computing simplicity and high performance. After fast upsampling, some simple restoration techniques are adopted to deblur the video, which are other types of SR methods.

2.1.2 Optimization-Based Methods

Optimisation-based methods often model the problem by finding the optimum of an energy function, which consists of a prior observation term from LR image and a regularisation term forming global constraints in or-

der to ensure the fidelity between the newly reconstructed HR image and the original LR image. Various image prior have been proposed to preserve edges (e.g. edge-directed priors (Tai, Liu, Brown & Lin 2010), gradient profile prior (Sun, Xu & Shum 2008), total variation prior (Babacan, Molina & Katsaggelos 2008), self-similarity prior (Irani 2009) and Bayesian prior (Tipping & Bishop 2003)). Although these methods are beneficial for edge preservation, they are generally inconvenient for adding the lost details for HR image reconstruction. Therefore, many methods were proposed to overcome this limitation. Zhang (Zhang, Gao, Li & Xia 2016) employed the non-local steering kernel regression (NLSKR) model as the regularization term to preserve edges and produce details in the resultant image. Fractional order total variation (TV) (Ren, He & Zhang 2013) regularization term was proposed based on the fractional differential, which can preserve the low-frequency contour feature and texture details in the smooth areas and high-frequency marginal feature in the edge areas. Xian (Xian & Tian 2016) introduced internal across-scale gradient similarity to predict texture details, which extract similar gradient image patches and restore the target HR image based on the constructed HR gradients. Sun (Sun et al. 2008) described the spatial layout of image gradients by gradient profiles, and infers HR gradient field from the LR image by gradient field transformation to reduce jaggy or ringing artifacts. These methods are based on the image gradient features, which cause a higher deviation for images having discontinuous gradient or non-smooth gradient.

2.1.3 Learning-Based Methods

Learning-based image SR methods, also known as example-based methods, predict HR image pixels by matching the image pairs to a universal set of training samples with machine learning models. Therefore, large sets of patches are needed to precisely predict reconstructed pixels and heavy computation is required to learn the mapping function. Various methods have been proposed to speed up calculation and improve performance. William

(William, Venkateswaran, Narayanan & Ramachandran 2016) learned the SR function as optimizing matrix-value operator with less cost of computation. Change et al. (Chang, Yeung & Xiong 2004) constructed the target HR image by enforcing local compatibility and smoothness constraints with the embedded K nearest patches using the manifold learning method Locally Linear Embedding (LLE). Another example of neighbouring embedding approach is Nonnegative Neighbor Embedding (Bevilacqua, Roumy, Guillemot & Alberi-Morel 2012), which learns the HR space by the local nonnegative least squares decomposition weights over the local neighborhood in LR spaces. These approaches use a dictionary of sampled patches from LR and HR images, which can quickly grow larger when adding more training images. To compact the dictionary, sparse coding was proposed by Yang et al. (Yang et al. 2010) for SR, which learns sparse dictionaries with LR and HR image pairs. However the sparse constraint is still a bottleneck on training dictionaries considering the computation. Timofte et al. proposed an anchored neighborhood regression (ANR) (Timofte, De Smet & Van Gool 2014) framework to speed up prediction. The above-mentioned learning methods have the complexity to find the similar patches by comparing the input patch with all the dictionary items. Schulter et al. (Schulter et al. 2015) adopted random forest (RF) for SISR, which has the clustering-regression mechanism to learn the leafs for SR mapping. the RF-based algorithm can outperform ANR with less computational complexity. Recently, deep learning has achieved promising performance on image SR (Dong et al. 2014), which mainly adopts convolutional neural networks for learning an end-to-end SR mapping model. Due to the efficient computation and accurate performance, our proposed methods are based on the deep learning frameworks to solve the image SR task.

2.2 Deep Convolutional Neural Network

2.2.1 Development of CNN in Image SR

Deep convolutional neural network (CNN) is a deep learning framework and has shown superiority in computer vision area. The development of computation hardware and scalable optimization algorithms make CNNs a powerful tool for solving pixel-level prediction tasks. The milestone of CNN-based SR method is SRCNN (Dong et al. 2014), which is a three-layer CNN training different scales SR with individual models. However, SRCNN can not generate superior results with the deeper model. He et al. (Huang, Sun, Liu, Sedra & Weinberger 2016) solved this problem by residually skip connecting layers inside the network (ResNet) to help the gradient flow across the deeper models. Many variations of ResNet (i.e. DCRN (Kim, Kwon Lee & Mu Lee 2016*b*) and EDSR (Lim et al. 2017)) were proposed in SR methods to improve the SR performance. Later on, more skip connection structure, dense connection (Huang, Liu, Van Der Maaten & Weinberger 2017) (DenseNet) was proved to accelerate network convergence by feature reusing across the layers. RDN (Zhang, Tian, Kong, Zhong & Fu 2018) and SR-DenseNet (Tong et al. 2017), embed the dense convolutional neural network into image SR to further improve image reconstruction accuracy. Then, attention modules were adopted into the SR to help the network focus on the high-frequency feature learning. Liu et al. (Liu, Wang, Li, Cheng, Zhang, Huang & Lu 2018) introduced spatial attention to mask out the high-frequency component location in the HR images and RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018) replaced normal feature layers with residual channel layers to adaptively rescale channel-wise features to reduce the unnecessary computations for abundant low-frequency features. However, we find most SR models treat different scale factors as independent problems, for example training three different models for handling $2\times, 3\times$ and $4\times$ SR, which waste a lot of computation time.

VDSR (Kim et al. 2016*a*) discovered the inter-scale correlation of deep CNN models, which train a single model to handle multiple scale SR problems

efficiently. Later on, a new multi-scale network structure was introduced in MDSR (Lim et al. 2017). Instead of processing the inputs of different scales with one end-to-end network as VDSR, MDSR uses multiple parallel single upsampling (Haris, Shakhnarovich & Ukita 2018) reconstruction layers for different SR of scales. However, SR of the larger scale factor (e.g. $8\times$) is still a challenging problem. To solve this problem, plausible priors are imposed to restrict the solution space, for example basing on the SR of small scales to gradually increase resolution by training SR samples of the larger scales. Lai et al (Burt & Adelson 1987) proposed Laplacian pyramid structure (LapSRN) for reconstructing HR outputs. Iterative back projection (Dai, Han, Wu & Gong 2007) was also adopted to tackle big-scale SR problems which iteratively computes the reconstruction error then feeds it back to tune the HR results. Haris et al (Haris et al. 2018) proposed DBPN with CNN models to simulate iterative back-projection to further improves performance on $\times 8$. But to cope with any-scale SR, it is inconvenient to train the models or upsampling layers independently for different scales. Until very recently, Meta-SR (Hu, Mu, Zhang, Wang, Sun & Tan 2019) proposed a meta-upscale module to dynamic magnify image with decimal scales with one model, by training and testing with 40 different scales at the stride of 0.1. However, for any-scale SR, Meta-SR ignores the SR problems of many other possible scales, only evaluating the SR values of these 40 scales. Futhermore Meta-SR adopts the method to train all the interested scales for any-scale SR, it entails countless training scale sets of SR samples for preparation and storage, due to the undetermined decimal precision and scale range. Besides, it takes too much time to optimize the network with all the possible training scales. In summary, none of the existing CNN-based works are designed to deal with any-scale SR problem. The existing algorithms can solve part of the problems to the any-scale SR problem, such as multi-scale learning, dynamic upsampling, and large scale SR, but none of them are able to generate HR images of any random scale which not trained before. In this thesis, we will introduce several proposed novel network frameworks for SR problem and

explore any-scale SR tasks with the proposed model.

2.2.2 Dense Convolutional Neural Network

In this section we will focus on the three dense models that are most related to ours: DenseNet (Huang et al. 2017), SRDenseNet (Tong et al. 2017), and RDN (Zhang, Tian, Kong, Zhong & Fu 2018). To not lose generality, only two blocks are shown in Figure 2.1 to describe these models.

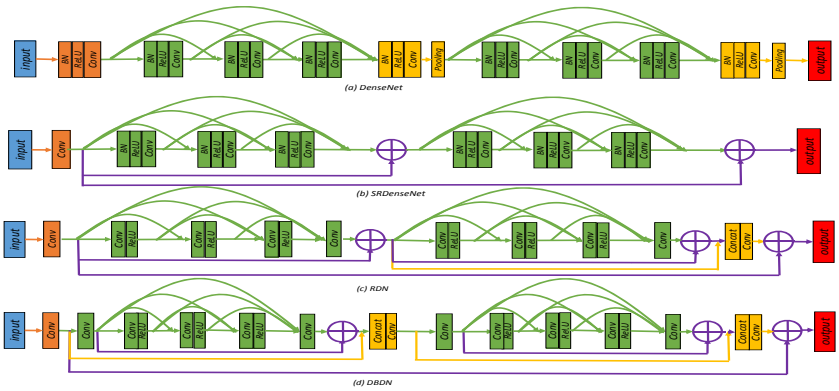

Figure 2.1: Simplified structure of (a) DenseNet. The green lines and layers denote the connections and layers in the dense block, and the yellow layers denote the transition and pooling layer. (b) SRDenseNet. The green layers are the same dense block structures as those in DenseNet. The purple lines and element-wise $\bigoplus$ refer to the skip conection. (c) RDN. The yellow lines denote concatenating the blocks output for reconstruction and the green layers are residual dense blocks. (d) DBDN. The yellow lines and layers denote the inter-block dense connection and the green layers are the intra dense blocks. The output goes into upsampling reconstruction layers. The orange layers after input are the feature extraction layers in all models.

DenseNet

The main idea of DenseNet (Huang et al. 2017) is to connect each layer to the other layers in a feed-forward network, in order to alleviate the vanishing gradient problems, strengthen feature propagation, and encourage feature reusing in the training of very deep networks. To implement the idea, the author proposes a dense connectivity mechanism as shown in Figure 2.1(a), where each layer obtains additional inputs from all the preceding layers and passes on its own feature maps to all subsequent layers. Denoting the input of each block as x_0, and the ℓ_{th} layer in the dense block as H_ℓ, then the output of the ℓ_{th} layer x_ℓ in the block can be described as:

$$x_\ell = H_\ell([x_0, x_1, ..., x_{\ell-1}]) \tag{2.1}$$

Here $[x_0, x_1, ..., x_{\ell-1}]$ refers to the concatenation of layers produced before the ℓ_{th} layer. $H_\ell(.)$ is a composite function of three consecutive operations: Batch Normalization (BN) (Ioffe & Szegedy 2015), followed by a rectified linear unit (ReLU) (Nair & Hinton 2010), and a 3×3 Convolution (Conv). Supposing the b_{th} dense block structure has ℓ layers, then the output of the b_{th} dense block is:

$$B_b = [x_0, x_1, ..., x_{\ell-1}, x_\ell] \tag{2.2}$$

Transition layers and pooling layers connect such dense blocks into line and change the block feature dimensions to construct the network. Although the dense connectivity in the dense blocks fulfills the idea of connecting each layer to all the other layers in the blocks. Stacking the dense blocks to construct DenseNet, restricts the block information flow. Furthermore, the dense block in DenseNet is designed for image recognition task, but it is not suitable for image SR and therefore needs to be adjusted for SR design.

SRDenseNet

Observing the efficiency of the dense block in DenseNet, SRDenseNet first introduces the dense connectivity into image SR. As shown in Figure 2.1(b),

the dense block in SRDenseNet has the same structure as DenseNet. However, SRDenseNet removes the transition layers and pooling layers. To keep the image detail information, removing the pooling layers is necessary in SR tasks. However, removing the transition layers will limit the setting of the feature layer channels. Then SRDenseNet adds skip connection between each block output and the network input, which helps to improve the network performance compared to only stacking the dense blocks. Therefore the output of the b_{th} dense block can be defined as

$$B_b = [x_0, x_1, ..., x_{l-1}, x_l] + x \tag{2.3}$$

Here x is the low level extracted features after convolutional layer. Furthermore, SRDenseNet has proven that adding a reasonable amount of skip connection between the blocks in the DenseNet can improve the SR reconstruction performance. For this reason, ensuring more block information flow will potentially boost the image reconstruction accuracy.

RDN

Compared with SRDenseNet and DenseNet, RDN is more suitable for SR tasks. There are several notes for RDN: (1) Unlike SRDenseNet and DenseNet that use a composite function of three consecutive operations $H_\ell(.)$: BN, ReLU and Conv for each layer in dense block, inspired by EDSR, RDN removes the BN module in the layer. The adjusted dense layers achieve better performance with fewer parameters compared with the original dense layers design in the SRDenseNet and DenseNet. (2) RDN uses a larger growth rate in covolutional layers, which allows it to learn more features compared with SRDenseNet. In SRDenseNet, the growth rate is 16, but RDN has 64 growth rates. The wider network design benefits RDN and helps it to achieve higher performance. (3) They concatenate all the block output for reconstruction, which alleviates the gradient vanishing problems in the deep network structure. Therefore, the output of the b_{th} residual dense block structure B_b can

be defined as

$$B_b = F_b([x_0, x_1, ..., x_{l-1}, x_l]) + x_0 \tag{2.4}$$

, Here F_b denotes the feature fusion function in the b_{th} residual dense block. And The output for the reconstruction layer R can be described as

$$R = F([B_0, B_1, ..., B_{b-1}, B_b]) + B_0 \tag{2.5}$$

, Here F denotes the global feature fusion function in the network. Although RDN concatenates all of the blocks' output for reconstruction at the end of the network, they ignore reusing the block information by dense connection to learn features. Since the block only receives one early block information, and lefts all the early blocks information. It may lead to the block information and gradient loss during propagation. In this reason, we propose Deep Bi-dense Network (DBDN) to reinforce the block information flow in the network.

2.3 Memory Optimization

Directly applying dense block in image super-resolution will generate a large number of network parameters, this is because each layer in dense block uses all previous feature maps as input. As a result, the number of parameters increases quadratically with network depth. Therefore the development of the deeper network is commonly limited by GPU memory. To reduce the training-time memory, many researchers (Chen, Xu, Zhang & Guestrin 2016) find that when training a deep convolutional network, a large proportion of memory is used to store the intermediate outputs and backward gradients. Chen (Chen, Li, Li, Lin, Wang, Wang, Xiao, Xu, Zhang & Zhang 2015) implemented a 1000 layer memory-efficient ResNet (He, Zhang, Ren & Sun 2016) model using one GPU by dropping the intermediate results into share memory allocations during training. The ResNet model is construced by composite Conv-BN-Relu layers, so the algorithm only keeps the result of convolution and drops the result of batch normalization and

activation function within each composite layer. The dropped results are recomputed before each composite layer back-propagation. Plesis et al (Pleiss, Chen, Huang, Li, van der Maaten & Weinberger 2017), observed the feature reusing mechanism of DenseNet (Tong et al. 2017) causes large memory consumption. The intermediate feature maps in DenseNet, such as Batch Normalization and Concatenation, are responsible for the most of the memory consumption and can be recomputed in the shared memory allocations before back-propagation. However, these models are only designed for image classification, which are not suitable for image super-resolution tasks. Furthermore, deep learning methods for image super-resolution require more GPU memory than image classification since the size of feature maps will not decrease as they are propagated to the network end. Therefore it is very important to have memory optimized training algorithms for image SR.

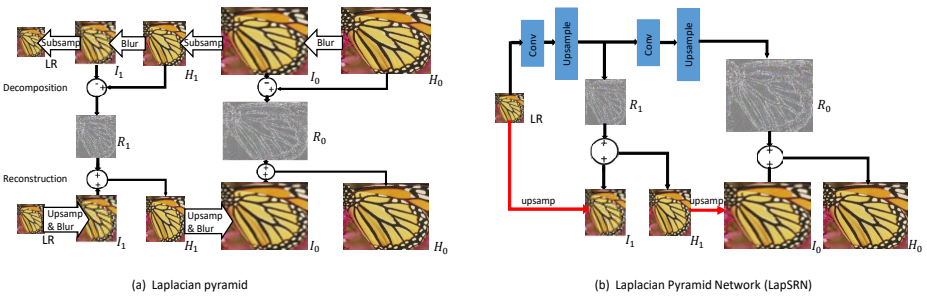

Figure 2.2: Comparison of two-level Laplacian Pyramids. (a) Laplacian Pyramid. The decomposition step produces two residual images R_1, R_0 by subtracting H_1 with I_1, H_0 with I_0 to preserve the high-frequency information, which then added with the interpolated LR I_1, I_0 respectively to reconstruct the $2\times, 4\times$ frequency levels H_1, H_0. (b) LapSRN. The residual images R_1, R_0 are progressively learnt by the networks and upsampled at each level, then added with LR images for HR images.

2.4 Laplacian Pyramid Structure

The Laplacain Pyramid (Burt & Adelson 1987) is used for HR image restoring by preserving multi-scale image residual information. As shown in Figure 2.2(a), the decomposition step firstly preserves the residual information as image is downscaled. Then the kept edge information will be stored back by adding with the low-resolution image to reconstruct the initial HR image. With the development of deep learning, many models (Karras, Aila, Laine & Lehtinen 2017) adopted Laplacian Pyramid structure as the main mapping framework, which construct progressive upsampling networks for image SR. Such as LapSRN (Lai, Huang, Ahuja & Yang 2017*a*) in Figure 2.2(b), a multi-phase network and each phase learns the residual information with convolutional layers, which is the most related to our work. However, our proposed Laplacian Frequency Representation differs from LapSRN in two aspects.

Basically, the objectives of these two Laplacian structures are different. The LapSRN was proposed to effectively predict SR of large scale factors (e.g. $8\times$) based on the pyramid reconstruction principle. On the contrary, our Laplacian Frequency Representaton is designed to predict SR results of continuous scales with SR of several discrete scales.

Then, the objective difference results in the various architecture design. First, the LapSRN progressively reconstruct each frequency levels at the interval of 2 times, for $2\times, 4\times$, and $8\times$ SR respectively. However, the Laplacian frequency levels are parallelly allocated and each level has the interval of 0.1 in the scale, e.g. $1.1\times, 1.2\times$, SR respectively. Second, the LapSRN upscales images at the end of each level with transposed convolutional layers. But such post-upsampling design will fix the frequency levels for the specific scales, and thus our model upsamples input images into the desired size before applying convolution at each level. Third, the LapSRN is customised for the fixed-scale HR image reconstruction, which predict the HR image of a specific scale by summing up the learnt residual information with the interpolated LR images as illustrated in Figure 2.2(b). While the Laplaican Frequen-

cy Representation is designed for predicting SR of any scale in a continuous range, and the HR image of any scale is difficult to be directly learnt through the network. We find the lost edge information can be presented by the two neighbouring frequency levels. In our design, the residual information of HR image is predicted according to the weighted interpolation of the two Laplcain frequency levels neighbouring the testing scale. Here, we assume the weight is determined by the distance proportion between the testing scale point and the two selected levels. Then the HR image of that scale is reconstructed by adding up the weighted residual information with the lower frequency level. The Laplacian Frequency Representation entails the fewer training SR samples to generate HR images of scales in the continuous ratio range, which reduces the undesired training data storage space and the length of optimization period to accelerate the network convergence.

2.5 Recursive Deployment

Recursive mechanism is firstly introduced into network as Recurrent Neural Network (RNN) (LeCun, Bengio & Hinton 2015), which is a kind of deep network created by applying the same set of weights and network structures recursively. RNNs have demonstrated strong ability to learn continuous features representations, for instance, learning sequence representations. Later on, many CNN-based SR methods adopt the recursive mechanism to enlarge receptive field and reduce model parameters for learning SR mapping function, such as DRCN (Kim et al. 2016*b*) and DRRN (He et al. 2016). Furthermore, the recursive mechanism can be developed to progressively reconstruct SR results, such as MS-LapSRN (Lai, Huang, Ahuja & Yang 2017*b*) which is a recursive version of LapSRN (Lai et al. 2017*a*). MS-LapSRN (Lai et al. 2017*b*) shares the network parameters and structure across pyramid levels which predicts the residual images at $2\times$ resolution. Therefore, the SR of $4\times$, and $8\times$ are gradually upscaled and recursively deployed with the scale $2\times$ twice and thrice. However, MS-LapSRN is designed for SR of integer

scales, and not applicable for any-scale SR. To copy with SR of any upsampling ratio, our recursive deployment is conducted with scales r in a small range and we reuse the network multiple times to reduce the computation memory costs, instead of progressively allocate the network as MS-LapSRN.

2.6 Applications of Image Super-Resolution

Regular video information enhancement: SR techniques can convert LR video images to high-definition images, for example Hitachi LTD uses SR technology to transform standard definition TV (SDTV) to high-definition television (HDTV) and Apple Inc. SR-based optical image stabilization patient to modify the motion outlier caused by multiple moving objects. SR techniques have been be employed in our phone, computer and TV right now.

Medical diagnosis: Medical imaging provides intuitive information about the human body structure. However, resolution limitations always degrade the value of medical images in the diagnosis. The goal is to increase the resolution of medical images while preserving the true imaging. Therefore, continuous and multi-view medical images can be easily acquired and operated.

Satellite image: The initial SR idea was inspired by the demand to improve the resolution of Landsat satellite sensing images. Due to the limitation of CCD hardware manufacture, multi-temporal or multi-view images for the same area bring SR techniques into the satellite image enhancement application.

2.7 Summary

The chapter introduces the work related to image Super-Resolution and provides the background of proposed methods. Firstly, the existing image Super-Resolution methods are divided into three types: interpolation, optimization, and learning methods. The related methods are reviewed and their limita-

tions in solving the Super-Resolution problem are analyzed. Secondly, several fundamental theories are introduced and discussed to show how these theories are used to deal with Super-Resolution. Finally, the applications of the image SR techniques are introduced.

Chapter 3

Bi-Dense Network for Image Super-Resolution

In this chapter, we will discuss how to improve reconstruction accuracy by improving the network structure. To begin with, the convolutional neural network structures are explored. Then the details of structure design are illustrated in the method section. The experimental results are also presented in the following.

3.1 Introduction

Learning-based methods especially using deep neural networks (Dong et al. 2014) have proven effective for image SR recently. However, as CNNs become increasingly deeper, skip connection was proposed to solve the gradient vanishing problem, such as VDSR (Kim et al. 2016*a*) and DRCN (Kim et al. 2016*b*) which embed ResNet (He et al. 2016) structure. Although residual networks have proven to facilitate gradient flow in training, performance is limited by constraining feature reuse of several layers, rather than all of the layers in a block.

In order to maximize the use of feature layers in the block, Dense Convolutional Network (DenseNet) (Huang et al. 2017) was proposed to archive

higher performance than ResNet. In contrast to ResNets, DenseNet has two modifications. First, instead of summation, DenseNet concatenates layers to preserve features for subsequent reuse. Second, instead of connecting two layers in the residual block, DenseNet connects all the layers in the block. This modification helps DenseNet to achieve better performance with fewer parameters than ResNet. This result indicates that feature learned at the early layers of the network matters to the task of SR, Therefore, by making more use of information in the feature layers, performance can be boosted. Inspired by the success of DenseNet in image classification, Tong et al. (Tong et al. 2017) proposed SRDenseNet for image SR. They removed the pooling layers of DenseNet to make it more suitable for image SR. Then they added skip connections between the blocks to mitigate the training of SRDenseNet. Zhang et al. (Zhang, Tian, Kong, Zhong & Fu 2018) also introduced the dense blocks in RDN. Compared with SRDenseNet, RDN uses the larger growth rates and deep supervision to improve the performance. However, all of these methods only reuse the local feature layers in the dense block, and pass block information to one neighbour block for feature learning. Haris et al. (Haris et al. 2018) proposed DBPN which is constructed by iterative up- and down- sampling blocks. All the early upsampling blocks are concatenated as the input for the next downsampling block, or all the early downsampling blocks are concatenated as the input for the next upsampling block. Therefore, each block information can't be reused by all the other blocks, which restricts the block information flow during propagation.

In order to make better use of block information, we propose a deep bi-dense network (DBDN) to enhance block information flowing by introducing a novel inter-block dense connection to the network. DBDN is built by intra-dense blocks and an inter-block dense net. Each layer connects to all the other layers in the intra-dense block and the output of the intra-dense block is the compressed concatenation of each layer in the block. Then, we use an inter-block dense net to connect these blocks. So each intra-dense block propagates its own local features to all successors. We reconstruct the SR image through

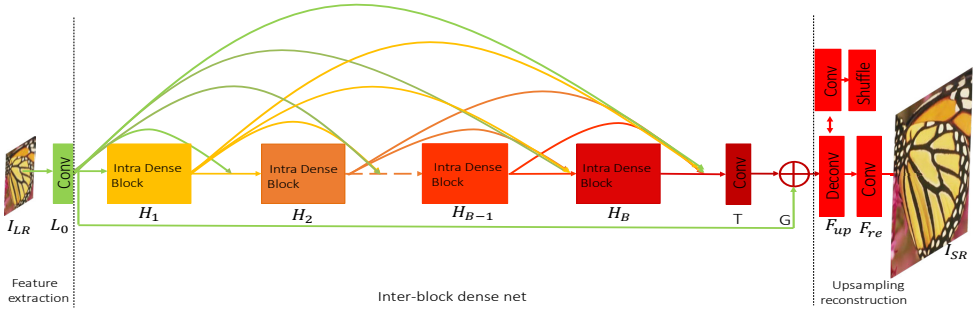

Figure 3.1: Network structure

concatenating all the blocks' outputs, forming a direct supervision pattern. For this reason, each block has direct access to the gradient from the loss function and ground truth. Due to the bi-dense architecture in the network, our DBDN outperforms the state-of-art methods on the benchmark datasets.

In summary, our work provides the following contributions:

• We propose a novel model called DBDN for image SR tasks. The model does not only reuse local feature layers in the dense block, but also reuses the block information in the network to archive excellent performance with moderate parameter numbers.

• We propose a compact intra-dense block where each layer is connected to every other layer to learn local features. Then, in order to preserve feature information and keep the model compact, we use the compressed concatenation of the output of all layers in the block as the output.

• We Introduce an inter-block dense net for high-level feature learning. Since the features learned in the early blocks of the network matter to the task of SR, we use inter-block dense connection to allow all of the early block information to be reused to learn the later block features.

3.2 Deep Bi-Dense Network

3.2.1 Network Structure

The network is used to estimate a HR image I_{SR} from a given LR image I_{LR} , which is downscaled from the corresponding original HR image I_{HR}. Our proposed network structure is outlined in Figure 3.1. The network can be divided into several sub-network structures: feature extraction layers for learning the low-level features, inter-block dense net for learning the high-level features, and upsampling reconstruction layers to learn upsampling features and produce the target HR image. Here, $Conv(f, n)$ denotes one convolutional layer, where f is the filter size and n is the number of filters.

We use one convolutional layer to extract the low-level features L^0 from the input images I_{LR}:

$$L^0 = Conv(3, n_r)(I_{LR}) \tag{3.1}$$

Here the feature extraction layer produces n_r channel features for further feature learning and global skip connection.

Following the initial feature extraction layer is the inter-block dense net. The inter-block dense net is the main component to learn features for SR, where intra-dense blocks are densely connected to learn high-level features. Then we further use the compression layer to compress the output features and skip connection between the inter-block dense net input and output to mitigate the training of network. The output of the inter-block dense net G can be obtained by

$$G = H_{Inter}(L^0) \tag{3.2}$$

Here H_{Inter} denotes the function of the inter-block dense net. More details about the inter-block dense net will be given in the next subsection.

For upsampling layers, there are two different types of upsampling sub-networks, which are illustrated in Figure 3.1. One is called the deconvolution (Schulter et al. 2015) layer and an inverse operation of a convolution layer, which can learn diverse upsampling filters that work jointly for predicting the

HR images. The other one is called the sub-pixel convolutional layer (Shi, Caballero, Huszár, Totz, Aitken, Bishop, Rueckert & Wang 2016), where the I_{SR} image is achieved by the period shuffling features produced by the previous convolution layer. In order to have a fair comparison with EDSR (Lim et al. 2017) and RDN (Zhang, Tian, Kong, Zhong & Fu 2018), we propose the baseline models with deconvolution layer for upsampling as DBDN and the model with sub-pixel upsampling layer as DBDN plus (DBDN+). Finally the target HR image is formulated as:

$$I_{SR} = Conv(3,3)(F_{up}(G)) \tag{3.3}$$

Here, F_{up} denotes the upsampling layer and the reconstruction layer is a three-channel output convolutional layer.

3.2.2 Inter-Block Dense Net

Now we present more details about our proposed inter-block dense net. As shown in Figure 3.1, Our inter-block dense net consists of inter-block dense connections, a global compression layer and a global skip connection.

Inter-Block Dense Connection

In order to enhance the block features and gradient flow, we densely connect the intra dense blocks to further reuse the blocks' information. The input for each block is the concatenation of all preceding blocks and the output of each block passes on to all the subsequent blocks. Supposing we have B intra dense blocks. The input of the B_{th} intra-dense block can be formulated as:

$$L_B = [H_1, H_2, ..., H_{B-1}] \tag{3.4}$$

Here H_b denotes the output of the b_{th} intra-dense block. More details about the intra-dense block will be shown in next subsection. And the output of the densely connected B intra-dense blocks H can be formulated as:

$$H = [H_1, H_2, ..., H_B] \tag{3.5}$$

Our inter-block dense connection makes more use of the feature layers and archives better SR performance compared with RDN and SRDenseNet. RDN and SRDenseNet use skip connections between blocks to pass only one early block information to the block, but our inter-block dense connection preserves and passes on all the early blocks to the block. We also demonstrate the effectiveness of inter-block dense connection in experiment section, and the results indicate that inter-block dense connection is crucial for image SR.

Global Compression Layer

After producing high level features, we compress the concatenation of all block features into n_r channel features with one compression layer. We use one 1×1 convolutional layer as the compression layer.

$$T = Conv(1, n_r)(H) \tag{3.6}$$

Here T denotes the output of the global compression layer.

Global Skip Connection

We add the global skip connection between the input L^0 and the output T of inter-block dense net.

$$G = T + L^0 \tag{3.7}$$

Here, G denotes the output of global skip connection. Since the input and output features are highly correlated in SR. Adding a global skip connection that bypasses the input features with an identity function can help gradient flow and mitigate the training of network.

3.2.3 Intra-Dense Blocks

The intra-dense block is presented in Figure 3.2, which contains input compression layer, densely connected layers, output compression layer and the skip connection. Here, let $conv_b^i(f, n)$ be the i_{th} convolutional layer in the b_{th} intra-dense block, where f is the filter size and n is the number of filters.

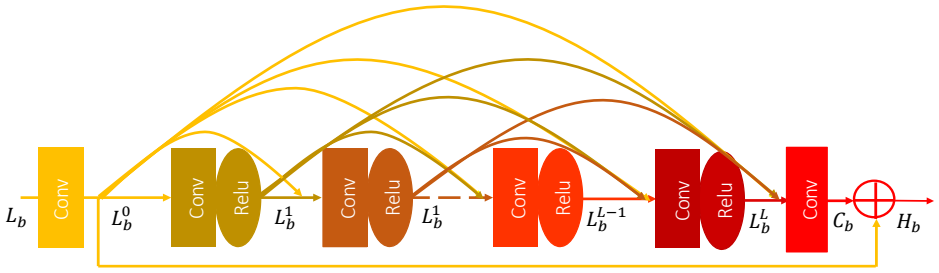

Figure 3.2: Intra dense blocks

Input Compression Layer

Since the input of the b_{th} intra-dense block L_b is the concatenation of all the preceding blocks, as the block number increases, the dimension of L_b becomes larger. In order to keep the model compact, the input compression layer is used to reduce the feature dimension into n_r before entering the dense layers.

We denote the result of the input compression layer in the b_{th} intra-dense block as L_b^0 and it can be formulated as:

$$L_b^0 = conv_b^0(1, n_r)(L_b) \tag{3.8}$$

Densely Connected Layers

After reducing the feature vector dimension, a set of densely connected layers are used to learn the high-level features. The i_{th} layer receives the features of all the preceding layers as input. We assume the densely connected layers have L layers and each convolution layer is followed by an ReLU function. Let L_b^i be the i_{th} layer output in the b_{th} intra dense block. The layer can then be expressed as:

$$L_b^i = max(0, conv_b^i(3, n_g)([L_b^0, L_b^1, ..., L_b^{i-1}])) \tag{3.9}$$

Here $[L_b^0, L_b^1, ..., L_b^{i-1}]$ refers to the concatenation of the features produced in all preceding layers of the i_{th} layer in the b_{th} intra dense block. Since each

layer generates n_g dimension features and passes the n_g dimension features to all the subsequent layers, the output of dense connected layers will have $n_r + n_g \times L$ features.

Output Compression Layer

In order to make the model more compact, we need to reduce feature dimension after densely connected layers. Inspired by the bottle neck structure, we add one convolutional layer $conv_b(1, n_r)$ as the output compression layer after the densely connected layers to compress the feature dimension from $n_r + n_g \times L$ to n_r.

$$C_b = conv_b^{L+1}(1, n_r)([L_b^0, L_b^1, ..., L_b^L]) \tag{3.10}$$

Here C_b denotes the features of output compression layer in the b_{th} intra-dense block. The reason for adding compression layer is that as model gets deeper and n_g becomes larger, the deep densely connected layers without a compression layer are more likely to be overfitting and will consume a large number of parameters without a better performance than the model with the compression layer. In the experiment section, we discuss the importance of compression layer in our model.

Local Skip Connection

Local skip connection is added between the input and output compression layer to help gradient flow during the training of network. Therefore the output of intra-block dense blocks H_b is:

$$H_b = C_b + L_b^0 \tag{3.11}$$

It should be noted that our intra dense block is different from the dense blocks used in DenseNet, SRDenseNet, and RDN. Our intra-dense block is more suitable for SR tasks, by removing BN module to keep the image details and adding compression layer to allow the wider and deeper network design. Because of our inter-block dense connection, we add an input compression

layer in the intra-dense block to develop the network which is different from the residual dense block in RDN.

3.3 Experiment

3.3.1 Implementation and Training Details

In the proposed networks, the convolutional layers in the dense connected blocks, transition layer and reconstruction layer are 3×3 filter size convolutional layers with one padding and one striding. The feature vector of the extraction layer, transition layer and deconvolutional layer is $n_r = 64$ dimension. The base model structure has 16 intra-dense blocks and each block has 8 dense layers. Here we set the feature dimension in each dense connected layer as $n_g = n_r$. In the upsampling sub-network of DBDN, for $2\times$ augmentation, we use a 6×6 deconvolutional layer with two striding and two padding. Then, for $3\times$ augmentation, we use a 9×9 deconvolutional layer with three striding and three padding. Finally, for $4\times$ augmentation, we use two successive 6×6 deconvolutional layers with two striding and two padding. In the upsampling sub-network of DBDN+, we use a $Conv(3, a^2 * n_r)$ convolutional layer followed by a pixel shuffle layer for $a\times$ augmentation, (a=2,3,4). We use the same training datasets as EDSR, the images from Flicker. To generate the LR image, we downscale the HR images using the bicubic interpolation with scale factors of $2\times, 3\times$ and $4\times$. In training batch, we use a batch size of 16 HR patches with the size of 96×96 as the targets and the corresponding LR patches with the size corresponding to the scale factors as inputs. We randomly augment the patches by flipping and rotating before training. To keep the image details, instead of transforming the RGB patches into a YCbCr space and only training the Y-channel image information, we use the 3-channel image information from the RGB for training. The entire network is optimized by Adam (Kingma & Ba 2014) with L1 loss by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate is initially set to 10^{-4} and halved at every 2×10^5 minibatch updates for 10^6 total minibatch updates.

Table 3.1: Public benchmark test results and Manga109 results (PSNR(dB)/SSIM). Red indicates the best performance and blue indicates the second best.

Datasets	Scale	Bicubic	VDSR	LapSRN	DRRN	SRDensenet	EDSR	RDN	DBPN	DBDN(ours)	DBDN+(ours)
Set5	2×	33.66/0.9929	37.53/0.9587	37.52/0.9591	37.74/0.9591	-/-	38.11/0.9601	38.24/0.9614	38.09/0.9600	**38.30/0.9617**	**38.35/0.9618**
	3×	30.39/0.8682	33.66/0.9124	33.82/0.9227	34.03/0.9244	-/-	34.65/0.9282	34.71/0.9296	-/-	**34.76/0.9299**	**34.83/0.9303**
	4×	28.42/0.8104	31.35/0.8838	31.54/0.8855	31.68/0.8888	32.02/0.8934	32.46/0.8968	32.47/0.8990	32.47/0.8980	**32.54/0.8991**	**32.70/0.9006**
Set14	2×	30.24/0.8688	33.03/0.9124	33.08/0.9130	33.23/0.9136	-/-	33.92/0.9195	34.01/0.9212	33.85/0.9190	**34.20/0.9224**	**34.34/0.9239**
	3×	27.55/0.7742	29.28/0.8209	29.77/0.8314	29.96/0.8349	-/-	30.52/0.8462	30.57/0.8468	-/-	**30.63/0.8478**	**30.75/0.8495**
	4×	26.00/0.7027	28.01/0.7674	28.19/0.7720	28.21/0.7721	28.50/0.7782	28.80/0.7876	28.81/0.7871	28.82/0.7860	**28.89/0.7890**	**29.00/0.7908**
BSD100	2×	29.56/0.8431	31.90/0.8960	31.80/0.8950	32.05/0.8973	-/-	32.32/0.9013	32.34/0.9017	32.27/0.9000	**32.39/0.9022**	**32.45/0.9028**
	3×	27.21/0.7385	28.82/0.7976	28.82/0.7973	28.95/0.8004	-/-	29.25/0.8093	29.26/0.8093	-/-	**29.31/0.8104**	**29.37/0.8112**
	4×	25.96/0.6675	27.29/0.7251	27.32/0.7280	27.38/0.7284	27.53/0.7337	27.71/0.7420	27.72/0.7418	27.72/0.7400	**27.76/0.7426**	**27.84/0.7446**
Urban100	2×	26.88/0.8403	30.76/0.8946	30.41/0.9101	31.23/0.9188	-/-	32.93/0.9351	32.84/0.9347	32.51/0.9321	**32.98/0.9364**	**33.36/0.9389**
	3×	24.46/0.7349	26.24/0.7989	27.14/0.8272	27.53/0.8378	-/-	28.80/0.8653	28.79/0.8655	-/-	**28.96/0.8682**	**29.17/0.8715**
	4×	23.14/0.6577	25.18/0.7524	25.21/0.7553	25.44/0.7638	26.05/0.7819	26.64/0.8033	26.61/0.8028	26.38/0.7945	**26.70/0.8050**	**27.00/0.8117**
Manga109	2×	30.80/0.9339	37.16/0.9739	37.27/0.9740	37.60/0.9736	-/-	38.96/0.9769	39.18/0.9780	38.89/0.9775	**39.46/0.9788**	**39.65/0.9793**
	3×	26.95/0.8556	31.48/0.9317	32.19/0.9334	32.42/0.9359	-/-	34.17/0.9473	34.13/0.9484	-/-	**34.46/0.9498**	**34.80/0.9512**
	4×	24.89/0.7866	27.82/0.8856	29.09/0.8893	29.18/0.8914	27.83/0.8782	31.11/0.9148	31.00/0.9151	30.91/0.9137	**31.23/0.9169**	**31.68/0.9198**

3.3.2 Comparison with State-of-Art Models

To confirm the ability of the proposed network, we performed several experiments and analysis. We compared our network with seven state-of-the-art image SR methods: VDSR (Dong et al. 2014), LapSRN (Schulter et al. 2015), DRRN (He et al. 2016), SRDenseNet (Tong et al. 2017), EDSR (Lim et al. 2017), and RDN (Zhang, Tian, Kong, Zhong & Fu 2018), DBPN (Haris et al. 2018). We carry out the test experiments using 5 datasets: Set5 (Bevilacqua et al. 2012), Set14 (Zeyde, Elad & Protter 2010), BSD100 (Timofte et al. 2014), Urban100 (Huang, Singh & Ahuja 2015) and Manga109 (Matsui, Ito, Aramaki, Fujimoto, Ogawa, Yamasaki & Aizawa 2017). Set5, Set14 and BSD100 are about nature scenes. Urban100 contains urban building structures and Manga109 is a dataset of Japanese manga. Table 3.1 shows the quantitative results comparisons for $2\times, 3\times, 4\times$ SR. For comparison, we measure PSNR and SSIM on the Y-channel and ignore the same amount of pixels as scales from the border. Note that the higher PSNR and SSIM values indicate the better quality. Our methods qualitatively outperform other CNN models with all scale factors in PSNR and SSIM. Compared to the light version CNN networks, DBDN outperforms them more than 1 dB in PSNR; Compared with the recent heavy version CNN networks, DBDN excels EDSR, RDN and DBPN about 0.1 dB in PSNR. Specifically, for the Man-

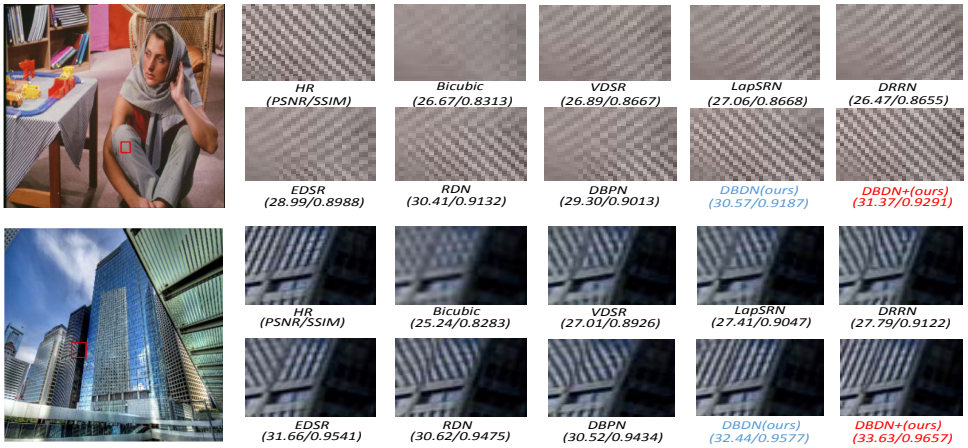

Figure 3.3: Qualitative comparison of our model with other works on $\times 2$ super-resolution. Red indicates the best performance and blue indicates the second best.

ga109 test dataset, our models exhibit significant improvements compared with the other state-of-art methods. Furthermore, DBDN plus surpasses DBDN about 0.1 dB.

We also provide visual comparison results as qualitative comparisons. Figure 3.3 shows the visual comparisons on the $2\times$ scale. For image 'barbara', all our methods can recover sharper and clearer pattern that are subjectively closer to the ground truth, while most of the compared methods generate blurred or biased cloth pattern. Similarly, for image 'img061' in the Urban100 dataset, all our methods can accurately recover the building structures. However, all the compared methods produce biased building lines. Figure 3.4 illustrates the qualitative analysis on the $4\times$ scale. Our methods suppress the blurring artifacts, recover patterns closer to the ground truths and exhibit better-looking SR outputs compared with the previous methods. This comparison demonstrates the effectiveness of our methods for image SR tasks.

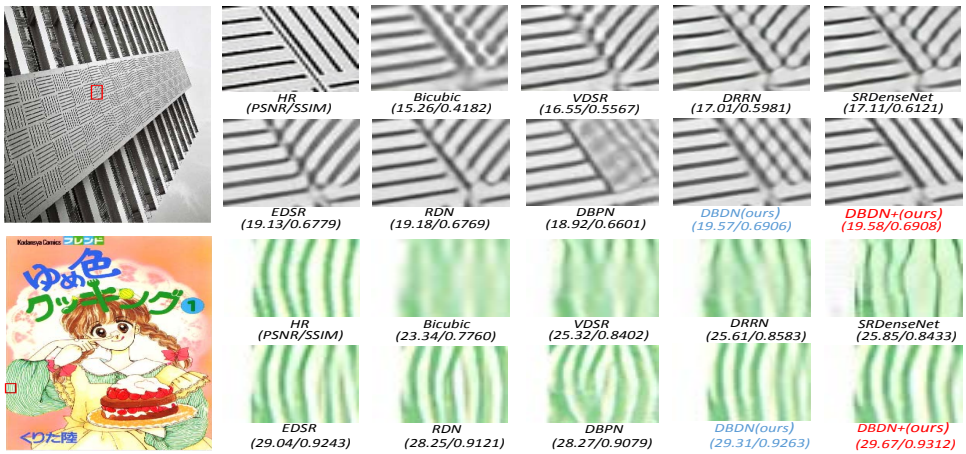

Figure 3.4: Qualitative comparison of our model with other works on $\times 4$ super-resolution. Red indicates the best performance and blue indicates the second best.

3.3.3 Model Analysis

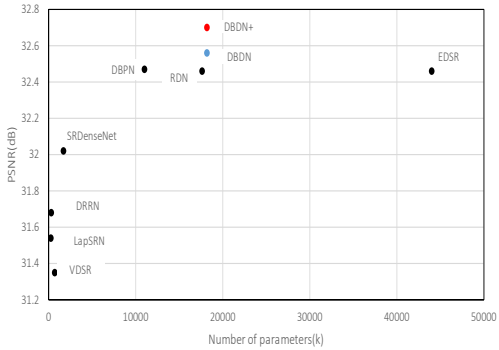

Figure 3.5: Performance vs number of parameters. The results are evaluated with Set5 for $4\times$ enlargement. Red indicates the best performance and blue indicates the second best.

Number of Parameters

To demonstrate the compactness of our model, we compare the model performance and network parameters of our model with the existing deep networks for image SR in Figure 3.5. Our model shows the trade-off between the parameter demands and performance. Since VDSR, DRRN, LapSRN and SRDenseNet are all light version networks, they all visibly concede the performance for the model parameter numbers. Although DBDN has more parameters than DBPN (Haris et al. 2018), DBDN has about 0.1dB higher performance than DBPN on Set5 for 4× enlargement. Compared with EDSR (Lim et al. 2017), which is one of the previous best performances, our DBDN network achieves the higher performance with about 58% fewer parameters. Compared with RDN (Zhang, Tian, Kong, Zhong & Fu 2018), DBDN achieves about 0.1 dB higher performance on Set5 for 4× enlargement with the same parameter numbers as RDN. Moreover, our DBDN+ outperforms all the other methods by a large margin with the same parameter numbers as DBDN.

Ablation Investigation

In this section, we will analyze the effects of different network modules on the model performance. Since skip connection has been discussed in many deep learning methods, we focus on the effects of the compression layer and the inter-block dense connection in our model. To demonstrate the effectiveness of the compression layer, we create a network without compression layer between the blocks. In order to form a fair comparison with the baseline model, the depth and number of parameters are kept the same for both methods. Therefore we set the model to have 128 convolutional layers in the block, each layer with 16 filters of the size of 3×3 and only one dense block to construct the total network. We denote this model as DBDN-W/O-Comp. To demonstrate the effectiveness of the inter-block dense connection, we chained the intra-dense blocks into line to construct a DBDN-W/O-Inter for learning the mapping function between the LR and HR image. The parameter setting of

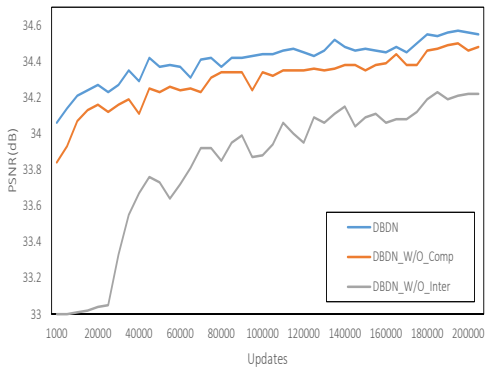

Figure 3.6: Discussion about transition layer, skip connection and inter-block dense connectivity in DBDN. The results are evaluated with Set5 for 3× enlargement in 200 epochs

DBDN-W/O-Inter is also the same as DBDN. We visualize the convergence process of these models. As we can see from Figure 3.6, The models without the compression layers and the inter-block dense connection suffer performance drop in the training, even when the models have the same condition of parameters . Especially for the inter-block dense connection, the best performance of DBDN on Set5 for 3× augmentation in 200 epoches is 0.6 dB higher than DBDN-W/O-Inter. Therefore, the inter-block dense connection is crucial for image SR performance.

3.4 Conclusion

We have proposed a DBDN for single image SR. Unlike the previous methods that only reuse several feature layers in a local dense block by using a dense connection, our proposed network extends previous intra-block dense connection approaches by including inter-block dense connections. The bi-dense connection structure helps the gradient and feature flows between the layers to archive better performance. The proposed method outperforms the state-of-art methods by a considerable margin on five standard benchmark

datasets in terms of PSNR and SSIM. The noticeable improvement can also visually be found in the reconstruction results. We also demonstrate that the different modules in our network improve the performance at different levels, and the inter-block dense connection has key contribution to our outstanding performance.

Chapter 4

Memory-Optimized Dense Network for Image Super-Resolution

In this chapter, we will discuss how to reduce memory consumption for running image super-resolution model. To begin with, the memory optimization methods are introduced. Then we present the network structure and storage mechanism for memory-efficient design. The corresponding experimental results and memory analysis are evaluated finally.

4.1 Introduction

Recently, a new dense architecture (Tong et al. 2017) was introduced in image super-resolution and achieved great success in terms of both reconstruction accuracy and computational performance. It is due to each layer is connected to all the other layers in the dense blocks, rather than only connected to one early layers in residual blocks. These connections promote feature reuse that early-layer features can be utilized by all other layers. The characteristics of dense architecture make it a very good fit for image super-resolution as they naturally induce skip connections. Tong (Tong et al. 2017) proposed

SRDenseNet, using dense network for image super-resolution by removing pooling players and transition layers. Zhang et al. (Zhang, Tian, Kong, Zhong & Fu 2018) also introduces dense blocks in RDN. Compared to SRDenseNet (Tong et al. 2017), RDN (Zhang, Tian, Kong, Zhong & Fu 2018) uses larger growth rates to construct a wider network for further improving the performance. But we find directly applying dense block in image SR will generate a large number of network parameters, this is because each layer in dense block uses all the previous feature maps as input. As a result, the number of parameters increase quadratically with network depth and the construction of deep network is commonly limited by GPU memory.

To reduce the training-time memory, many researchers (Chen et al. 2016) find that training a deep convolutional network, a large proportion of memory is used to store the intermediate outputs and backward gradients. Chen (Chen et al. 2015) implemented a 1000 layer memory-efficient ResNet (He et al. 2016) model using one GPU by dropping the intermediate results into the share memory allocations during training. ResNet model is construced by composite Conv-BN-Relu layers, so the algorithm only keeps the result of convolution, but drops the result of batch normalization and activation function within each composite layer. The dropped results are recomputed before each composite layer back-propagation. Plesis et al (Pleiss et al. 2017), observed the feature reuse mechanism of DenseNet causes large memory consumption. The intermediate feature maps in DenseNet, such as Batch Normalization and Concatenation, are responsible for the most of the memory consumption and can be recomputed in the share memory allocations before back-propagation. However, these models are only designed for image classification, which is not suitable for image super-resolution tasks. Furthermore, deep learning methods for image super-resolution require more GPU memory than image classification since the size of feature maps will not decrease as they are propagated to the network end. Therefore it is very important to have memory optimized training algorithms for image SR. In order to efficiently use GPU memory to train network, we implement a memory opti-

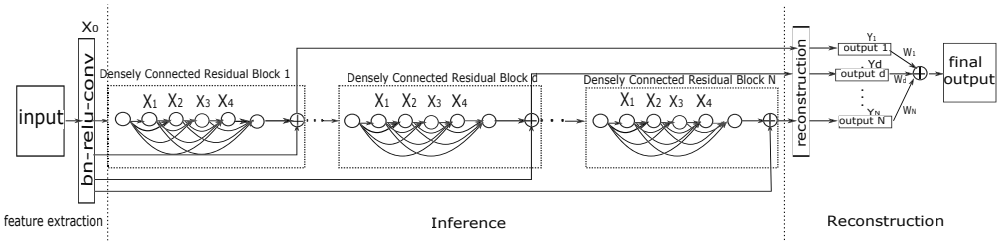

Figure 4.1: Our network structure with densely connected residual block and deep supervision.

mized deep dense network for image super-resolution. On the one hand, we rationally design layer connections in the blocks to reduce redundant features learning and minimize the model size. On the other hand, we adopt share memory allocation strategy to store the concatenated features and batch normalization intermediate feature maps to reduce the training memory cost of network. The proposed network reaches a promising performance on the benchmark datasets with less training-time memory.

4.2 Methods

In this section, we first describe the network structure, then introduce our densely connected residual block, and shared memory allocation for memory optimized deep dense network.

4.2.1 Network Structure

Our network configuration is outlined in Figure 4.1. The model consists of three sub-network structures: feature extraction, Inference, and reconstruction. Feature extraction sub-architecture is used to extract feature maps from the LR images and comprise a set of feature maps into a high dimensional vector. Then inference is aimed to expand the high dimensional vector into multiple channels vectors and construct a deep network structure for the task. Finally, the reconstruction sub-architecture is to generate the output

image.

Feature Extraction Sub-Architecture

It extracts patches from the interpolated input image X. The interpolated images are upscaled by bicubic interpolation as the low-resolution images. In our feature extraction sub-architecture, we implement a composite function $H_0()$ of three consecutive operations: batch normalization (BN) (Ioffe & Szegedy 2015), rectified liner unit (ReLU) (Nair & Hinton 2010) and convolutional filters (Conv) to extract features and represent them in the form of vectors. Considering a single low-resolution image (interpolated image) as input X, the feature extraction sub-architecture output as X_0, and X_0 is formulated as below:

$$X_0 = H_0(X) \tag{1}$$

Inference Sub-Architecture

The inference sub-architecture is the main part to learn the mapping from the LR features to HR features. In the inference sub-architecture, we use our densely connected residual blocks to learn features and stack them into a chained network. Details about densely connected residual blocks will be given in the following subsection. Figure 4.1 illustrates the layout of the dense connectivity in our inference sub-architecture. Denote the inference sub-architecture contains N densely connected residual blocks, and the function of i_{th} densely connected residual block as $D_i()$. Due to inference sub-architecture is equivalent to the composition of the chained densely connected residual blocks, we formulate the output of inference sub-architecture Y^N as:

$$Y^N = D_N(D_{N-1}(...(D_1(X_0))...)) \tag{2}$$

Reconstruction Sub-architecture

The reconstruction layer is a 1-channel convolutional layer. Each densely connected residual block will generate their own reconstructed images. We

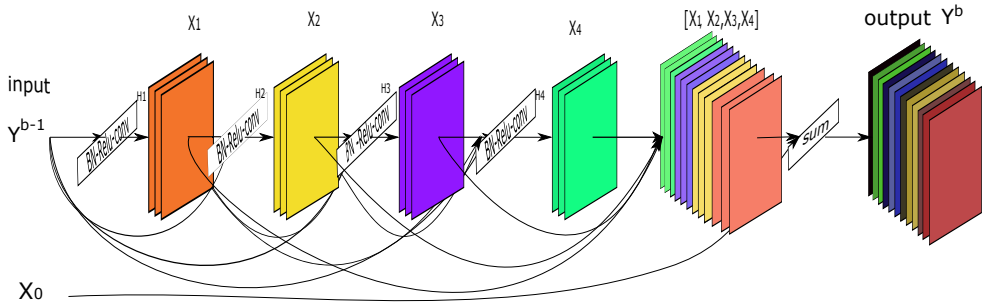

Figure 4.2: Densely connected residual block

then sum the averaged reconstructed images as the output of reconstruction sub-architecture. Denote the output of reconstructing the i_{th} densely connected residual block as Y_i

$$Y_i = f_{res}(D_i(D_{i-1}(...(D_1(X_0))...))) \tag{5}$$

Here, $D_i()$ and $f_{res}()$ are the functions of the i_{th} densely connected residual block and reconstruction. Denoting $\widehat{Y}$ as the reconstruction image, which is produced by adding up the averaged reconstructed images with the input X. We formulate the reconstruction image as below:

$$\widehat{Y} = \frac{Y_1 + .. + Y_i + .. + Y_N}{N} + X \tag{6}$$

Here we have N densely connected residual blocks to construct the network.

4.2.2 Densely Connected Residual Block

Figure 4.2 illustrates the layout of the dense connectivity in densely connected residual block. The ℓ_{th} layer in the block receives the feature channels of all preceding layers $X_0, ..., X_{\ell-1}$ as input.

$$X_\ell = H_\ell([X_0, X_1, ..., X_{\ell-1}]) \tag{3}$$

where [] refers to the concatenation function, and H_ℓ is the ℓ_{th} layer composite function.

To reduce computation redundancy and enhance feature reuse, we first keep the extracted feature X_0 and concatenated channel layers $[X_1, ..., X_4]$ the same channel size. Then sum them up to get the input for next densely connected residual block. This skip connection between X_0 and the output of the block uses the idea of residual learning. Denote Y^i as the output of i_{th} densely connected residual block. We formulate the i_{th} densely connected residual block output as below:

$$Y^i = [X_1^i, ..., X_4^i] + X_0 \tag{4}$$

Our densely residual block has two advantages: (1) it delves the network potential through feature reuse and helps the gradient back propagation with the direct connection structure. (2) replacing transition layers with summation largely reduces the network parameters and keeps the input feature information throughout the layers.

4.2.3 Shared Memory Allocation

In the dense network, the intermediate features are mostly generated from concatenation and batch normalization. So we create shared memory allocations for the output of concatenation and batch normalization.

The network is constructed by composite layers Concat-BN-ReLU-Conv as shown in Figure 4.3(a). The normal plain dense network keeps these intermediate features allocated in different GPU memory for use during back-propagation. As we can see in Figure 4.3(b), the back propagation is triggered by the last convolution layer forward propagation. Then the back propagation is processed in the reverse order of forward propagation to calculate the layer parameters, input, and output gradients and update the layer parameters depending on each layer input.

We use shared memory allocations for each concatenation and batch normalization. Instead of allocating new memory for concatenating the existing features, we copy pre-processed features into one shared memory storage to concatenate these features into one tensor as the input for next layer. For

batch normalization, we first calculate the mean and variation value of the input features, and then put the calculated batch normalization result into the shared memory allocation. As shown in Figure 4.3(c), the shared memory allocations for concatenation and BN are used by all dense layers, so data in the storages is changing. Therefore, after the last convolutional layer forward, the concatenation and BN features are recomputed to restore the last composite layer corresponding intermediate features. After the storages storing the right data, back propagation for the last composite layer is used in the regular way. Then after the last composite layer back-propagation is done, the former layer intermediate features will be recomputed to trigger the back-propagation of the composite layer. We also analyze the memory efficient property of using shared memory allocation during training in the experiment section, shared memory allocation largely reduces the memory demands for training the deep network in image SR.

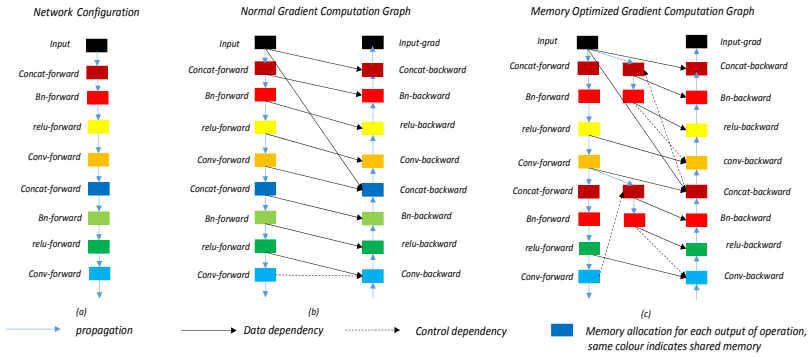

Figure 4.3: Computation graph of memory optimized network

4.2.4 Computation graph of memory optimized network

To illustrate the memory optimization function, we use a computation graph to demonstrate the network operation, which consists of the operational n-

odes such as concat, BN, ReLU and edges such as propagation, data dependency and control dependency. Fig 4.3 is the computation graph of two-layer proposed memory optimized network. Fig 4.3 (a) is the simplified forward propagation, which omits the weight nodes and gradients of the weights. The input data propagates forwardly through the two composite layers Concat-BN-ReLU-Conv and each layer will allocate new memory for storing the immediate outputs. Once the network configuration is given, the backward propagation will calculate gradients for refining the weights. Fig 4.3 (b) shows the normal back-propagation steps, and each operational node depends on the data from the output memory then calculate and store the gradients into new storages. Therefore, a great proportion of the memory is used to store the intermediate outputs and gradients. To solve this issue, memory optimized gradient computation graph are designed as shown in Fig 4.3 (c). Intermediate results are memory consuming and calculation cheap, which can be recycled and used for another operational nodes. In our design, Concat and BN operational nodes share memory across different composite layers, which have the same colors. Therefore, during each composite layer's back propagation, Concat-forward and BN-forward needs to restore the right output data into the shared memory allocation. Then the Concat-backward and BN-backward gradients will be calculated and stored into the shared gradient memory allocation. Memory optimization happens when the recycled allocations are used by other composite layers.

4.3 Experiment

In this section, we evaluate the performance of our model on several datasets. A description of the datasets is first provided, followed by the introduction of the implementation details. Then we analyze the depth and memory consumption of the model. After that, comparisons with state-of-art results are presented.

4.3.1 Datasets

The training dataset contains 291 images, where 91 images are from Yang et al (Yang et al. 2010), and other 200 images are from Berkeley Segmentation Dataset (Martin, Fowlkes, Tal & Malik 2001). For testing, we use four datasets 'Set5' (Bevilacqua et al. 2012), 'Set14' (Zeyde et al. 2010), 'BSD100' (Timofte et al. 2014) and 'Urban100' (Huang et al. 2015), which contains 5, 14 ,100 and 100 images respectively.

4.3.2 Implementation Details

Before the model setting, we first augment the training dataset the 291 images into 2328 images with flipping horizontally and rotating the training images by 90^o, 180^o, 270^o, which provide 7 additional augmented images of each original image. In addition to that, the training data were processed with multiple scale ($\times 2$, $\times 3$ and $\times 4$) augmentation by following previous works (Kim et al. 2016*a*, Dong et al. 2014).

We use 25 densely connected residual blocks to construct the training network. Each block contains 4 convolutional layers with 34 filters of the size 3×3. Training images are split into 31×31 patches with stride 21, so the receptive field is 31×31. We set the momentum parameter to 0.9 and weight decay to 0.0001 and 64 patches are used as a mini-batch for stochastic gradient descent.

The learning rate is initially set to 0.1 and then decreased half every 10 epochs. Since the initial learning rate is large in our design, we use the adjustable gradient clipping (Kim et al. 2016*a*) to avoid exploding gradients. We clip the gradients to $[-\frac{\theta}{\gamma}, \frac{\theta}{\gamma}]$, where γ denotes the current learning rate and gradient clipping parameter $\theta = 0.01$. The adjustable gradient clipping makes the convergence procedure fast, our 25 blocks network implemented with caffe takes about 4 days to train with one Titan X GPU.

Table 4.1: Public benchmark test results (PSNR(dB)/SSIM). Red indicates the best performance and blue indicates the second best.

Datasets	Scale	Bicubic	SRCNN	VDSR	DRCN	DRRN	MODN(ours)
Set5	2×	33.66/0.9929	36.66/0.9542	37.53/0.9587	37.63/0.9588	**37.74/0.9591**	**37.74/0.9593**
	3×	30.39/0.8682	32.75/0.9090	33.66/0.9213	33.82/0.9226	**34.03/0.9244**	**34.07/0.9248**
	4×	28.42/0.8104	30.48/0.8628	31.35/0.8838	31.53/0.8854	**31.68/0.8888**	**31.72/0.8889**
Set14	2×	30.24/0.8688	32.42/0.9063	33.03/0.9124	33.04/0.9118	**33.23/0.9136**	**33.25/0.9141**
	3×	27.55/0.7742	29.28/0.8209	29.77/0.8314	29.76/0.8311	**29.96/0.8349**	**29.97/0.8349**
	4×	26.00/0.7027	27.49/0.7503	28.01/0.7674	28.02/0.7670	**28.18/0.7720**	**28.25/0.7721**
BSD100	2×	29.56/0.8431	31.36/0.8879	31.90/0.8960	31.85/0.8942	**32.05/0.8973**	**32.06/0.8978**
	3×	27.21/0.7385	28.41/0.7863	28.82/0.7976	28.80/0.7963	**28.95/0.8004**	**28.96/0.8011**
	4×	25.96/0.6675	26.90/0.7101	27.29/0.7251	27.23/0.7233	**27.38/0.7284**	**27.42/0.7286**
Urban100	2×	26.88/0.8403	29.50/0.8946	30.76/0.9140	30.75/0.9133	**31.23/0.9188**	**31.27/0.9191**
	3×	24.46/0.7349	26.24/0.7989	27.14/0.8276	27.15/0.8276	**27.53/0.8378**	**27.55/0.8384**
	4×	23.14/0.6577	24.52/0.7221	25.18/0.7510	25.14/0.7510	**25.44/0.7638**	**25.52/0.7649**

4.3.3 Comparisons With State-of-Art Methods

In this section, we provide quantitative and qualitative comparisons. Since the training datasets also influence performance, We use SRCNN (Dong et al. 2014), VDSR (Kim et al. 2016*a*), DRCN (Kim et al. 2016*b*) and DRRN (Tai et al. n.d.) as the compared methods, which use the same training data as ours. In deploying trained models with test datasets, the luminance components of the image are applied for model deploying and the color components are applied with bicubic interpolation, which is the same as the previous experiments. We also crop the image to the same size as (Kim et al. 2016*b*, Kim et al. 2016*a*, Dong et al. 2014) methods.

Table 4.1 illustrates a summary of quantitative evaluation on several datasets. Generally, our model outperforms all the other methods in both PSNR and Structural SIMilarity (SSIM) (Wang, Bovik, Sheikh & Simoncelli 2004). In terms of PSNR, our model achieves an average 0.2 dB improvement over VDSR and DRCN. In comparison with DRRN, our model demonstrates an average 0.03 dB increase of PSNR than DRRN. Figure 4.4 shows qualitative comparisons among SRCNN, VDSR, DRRN and ours, we can see our network outperforms all the methods in the experimental PSNR and SSIM index in the most cases. For qualitative comparisons among SRCNN, VD-

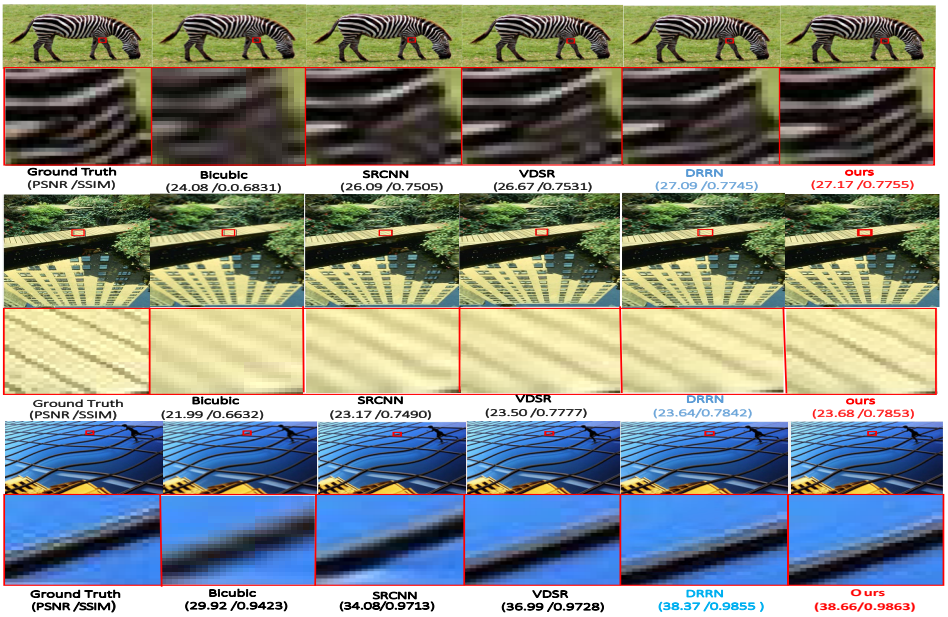

Figure 4.4: Qualitative comparison.(1)In the first row image "img014"(Set14 with scale factor $\times 4$), the zebra texture is more detailed in our method.(2)In the second row shows image"img021"(B100 with scale factor $\times 3$), the black line is clearer in our method (3) In the third row image "img082" (Urban100 with scale factor $\times 2$), we can see that is two line in ours.

SR, DRRN and ours, our network obviously produces patterns with sharper and cleaner edges compared with SRCNN and VDSR. In comparison with DRRN, our performance is slightly better. However, The computation complexity using our method is 20 percent less than DRRN. For 25 blocks, Our network has 11 billion FLOPs, while DRRN contains 14 billion FLOPs. This is because only 34 feature channels used in our convolutional layer, while the convolutional layer in DRRN (Tai et al. n.d.) has 128 feature channels.

4.3.4 Analysis

Depth Analysis

To analyze the influence of the network depth on the performance, we construct two networks with 9 and 25 blocks respectively and keep other parameters the same. Note that higher PSNR and SSIM values indicate better quality. As shown in Table 4.2, the model with 25 blocks (MODN-25B) demonstrates better reconstruction accuracy, which outperforms the model with 9 blocks (MODN-9B) more than 0.04 dB in PSNR and 0.001 in SSIM. This is mainly because the network with more parameters can extract more hierarchical features, which contributes to better performance. The depth analysis also indicates that our model allows deeper network for higher performance, so it is necessary to make it memory optimized for deeper structures.

Table 4.2: The depth analysis of network

Datasets	Scale	MODN-9B	MODN-25B
Set5	2×	37.67/0.9590	37.74/0.9593
	3×	33.95/0.9234	34.07/0.9248
	4×	31.56/0.8867	31.72/0.8889
Set14	2×	33.21/0.9134	33.25/0.9141
	3×	29.91/0.8329	29.97/0.8349
	4×	28.18/0.7702	28.25/0.7721
BSD100	2×	32.02/0.8970	32.06/0.8978
	3×	28.93/0.7993	28.96/0.8011
	4×	27.32/0.7265	27.42/0.7286
Urban100	2×	31.04/0.9167	31.27/0.9191
	3×	27.39/0.8333	27.55/0.8384
	4×	25.36/0.7577	25.52/0.7649

Memory Analysis

To investigate the memory efficient property of the model, we compare the training-time memory consumption of the models with shared memory allocation and the models without shared memory allocation. For fair comparison, we use one 31×31 patch as input for training these models. As shown in Table 4.3, for model with 9 blocks depth, the memory optimized

implementation (MODN-9B) consumes about 87% less memory than the 9 blocks model without shared memory allocation (Plain-9B). For model with 25 blocks depth, the memory optimized implementation (MODN-25B) consumes 88% less memory than the 25 blocks model without shared memory allocation (Plain-25B). This is because MODN only stores convolution feature maps and network parameters in the memory. However, the plain model also needs to store concatenated and normalized features in the memory, in addition to the convolution feature maps and network parameters. Therefore, generally the shared memory allocation reduces the memory consumption of the plain dense model more than 80%. In other words, our MODN can achieve better performance by adding more blocks with one GPU than the plain dense model.

Table 4.3: The memory analysis of network

Model	Blocks	Memory(GB)
MODN	$9B$	0.2
	$25B$	0.8
Plain	$9B$	1.6
	$25B$	6.3

4.4 Conclusion

In this work, we propose a memory optimized deep dense network for image super-resolution. First, we reduce the storage redundancy in the network training by carefully designing the skip connection and the dense connection. Then, we construct share memory allocation for training the network to reduce the training-time memory. The memory optimized model requires less memory than normal deep dense model. We have demonstrated that our network outperforms the some state-of-art methods on the benchmark datasets with relatively less computation complexity.

Chapter 5

One Deep Convolutional Network for Any-Scale Image Super-Resolution

The existing deep SR methods train a separate model for each input image scale, which are inefficient and impractical for generating SR of any scale factor. In this chapter, we propose a novel Any-Scale Deep Network (ASDN), which requires very few training scales to achieve one unified network for arbitrary-scale SR.

5.1 Introduction

As SR problems are highly ill-posed, many methods are based on image context to learn the mapping function between LR and HR scale patches, which need the image information of all the interested scales (e.g. dictionary learning (Yang et al. 2010), self-example learning (Zhu et al. 2014), random forest (Schulter et al. 2015), and convolutional neural network (Dong et al. 2014)). However, it is common for some image applications to zoom images into any required sizes to observe the details. Therefore, one model for any-scale image SR is needed. But it is inefficient to prepare training samples of

all these scales and impractical to optimize a network with all these scales. Thus, it is important to use a small amount of training scales to effectively generate the any scale SR model.

However, the most existing deep learning SR methods are trained with the scales to be tested. Some of the previous works require scale-specific networks trained independently for several integer scales (e.g. SRCNN (Dong et al. 2014), EDSR (Lim et al. 2017), RDN (Zhang, Tian, Kong, Zhong & Fu 2018) and so on). As mutual relationship among different scales was discovered in VDSR (Kim et al. 2016*a*), many models reconstruct HR images of multiple upscaling factors in one single model, such as MDSR (Lim et al. 2017) and DRCN (Kim et al. 2016*b*). However, these methods are designed for the fixed integer scales. Until very recently, Meta-SR (Hu et al. 2019) discussed about SR problems of arbitrary scale factor which train a dynamic meta-upscaling module to magnify images with dense decimal scales. However, Meta-SR (Hu et al. 2019) needs to train with all the required testing scales for any-scale SR. This is because that Meta-SR can only archieve SR results based on the trained scales. To have a full any scale SR model, Meta-SR will need to be trained with countless samples which might be impractical. In general, all the existing SR methods trained with all the scales of interest, require too much training samples and make it hard to solve any-scale SR with a single model.

To overcome previously mentioned limitations, we propose Laplacian Frequency Representation for SR of the small continuous scales $r\epsilon(1, 2]$, and Recursive Deployment for SR of the larger scales R out of the range. As we find a large decimal ratio R can be expressed as an integer power of ratios r in a small range. Therefore, if there is a network that can handle SR at the small continuous scales r, SR of the large decimal ratio R can be achieved by gradually upscaling the coarse HR image and recursively deploying it through the network multiple times with the small decimal ratios r. In this way, We only need to train the network with scales related to the small continuous scale range. To minimize the number of training scales,

we set the small continuous ratios $r\epsilon(1, 2]$. But how to represent the SR of continuous scales with the discrete scale SR values? Inspired by Laplacian Pyramid (Burt & Adelson 1987), which reconstructs HR images with the preserved residual information between two Laplacian pyramid levels. We propose Laplacian Frequency Representation to interpolate the residual image of each scale factor by the two neighbouring Laplacian pyramid levels. Thus, instead of directly training a network with each scale in the range, the continuous scale SR results can be represented by the several Laplacian frequency levels. It only entails the few training scales for the Laplacian frequency levels optimization, which dramatically reduces the update period and accelerates the network convergence compared with training all the scales of interest for any-scale SR.

To construct the SR network of small continuous scales r, we propose our network as Any-Scale Deep Network (ASDN), which mainly constructs by two kinds of branches, the Feature Mapping Branch (FMB) and Image Reconstruction Branches (IRB). The FMB is designed to learn the feature mapping function between LR and HR images and IRBs are proposed to parallel reconstruct multiple Laplacian frequency levels from the learnt HR features. Therefore, for SR of each scale r in the continuous range, the two IRBs corresponding to the neighbouring Laplacian frequency levels to the scale r will be enabled and the HR image is the interpolated results by the two reconstructed images from the IRB; For any scale factor R, the HR images can be achieved by recursively deployed from ASDN with small ratios r.

We present extensive comparisons on both fixed integer scales and any decimal scales on the commonly used benchmarks. For SR of fixed scales, we further fine-tune ASDN by inserting some transposed convolutional layers as the fixed-scale deep network (FSDN) for the reference in the comparison with the existing fixed-scale SR methods. Our ASDN is superior to the most existing deep methods, even without being trained with the specific-scale data samples and FSDN reaches the state-of-art fixed-scale SR performance.

For SR of any scale factor, we retrain many previous network structures (Lim et al. 2017), (Zhang, Tian, Kong, Zhong & Fu 2018), (Kim et al. 2016*a*) with our any-scale SR method into any-scale SR categories for comparison. Our ASDN is effective for SR of any desired scale and specifically achieve the state-of-art performance on scales within the small range $(1, 2]$.

5.2 Any-Scale SR Framework

Our any-scale HR image generation method is based on the idea that any upscale decimal ratio R can be expressed as an integer N power of decimal ratios r in a small range. To use the minimum training samples, we define the small decimal ratios $r \in (1, 2]$. For SR of each upscaling ratio R, The HR image of upscale ratio R can be achieved by recursively upscaled with a small ratio r and deployed N times. Therefore, we need to implement the network, which can handle continuous scale range $(1, 2]$ and determine the recursion times N and the small ratio value r at each recursion. The details of any-scale SR method including Laplacian Frequency Representation and Recursive Deployment and the structure of our proposed ASDN and its fixed-scale version FSDN are described in this section.

5.2.1 Any-Scale SR Method

There are two steps in the any-scale SR method: Laplacian Frequency Representation and Recursive Deployment. The proposed Laplacian Frequency Representation method is to generate HR images of decimal scales in a continuous scale range and Recursive Deployment is to define the recursion times N and the small ratio r at each recursion for any-scale SR prediction.

Laplacian Frequency Representation

To generate SR of decimal scales in a continuous ratio range, the intuitive method is to train the network with randomly dense scales in the range.

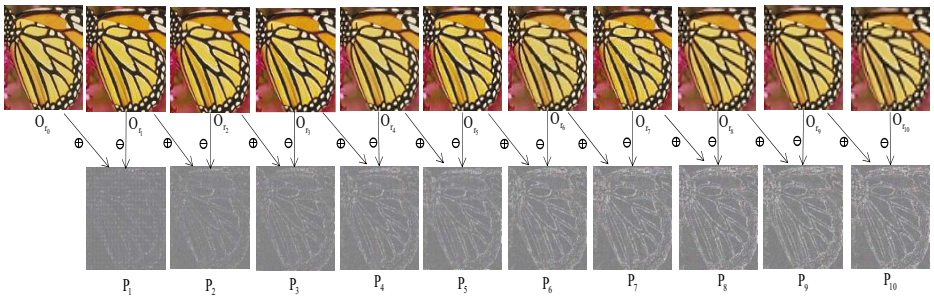

Figure 5.1: The Laplacian Frequency Representation has 11 Laplacian frequency levels $(O_{r_0}, ..., O_{r_{10}})$, with 10 phase in the scale range of $(1, 2]$ $(P_1, ..., P_{10})$. Each phase represents the difference between the successive Laplacian frequency levels.

However, we find this method consumes a large amount of training scale samples, and requires a long training period to optimize the network. Therefore, we want to accurately predict SR of continuous decimal scales with only training few scales in the range.

Inspired by Laplacian Pyramid (Burt & Adelson 1987) to preserve the residual information by subtracting two Laplacian pyramid levels, in this paper, the SR of fractional scales in the range is represented based on their two neighbouring Laplacian frequency levels. As shown in Figure 5.1, our proposed Laplacian Frequency Representation has L Laplacian frequency levels, and each frequency level l is tasked with learning the HR images O_{r_l} for the scale $r_l = \frac{l}{L-1} + 1, l = 0, ..., L-1$, which is trained with the corresponding scale SR samples. For the SR of the other decimal scales in the range of $(1, 2]$, the HR images of that scale is the weighted interpolation of its two neighbouring Laplacian frequency levels denoted as $O_{r_{i-1}}$ and O_{r_i}, the residual information between $O_{r_{i-1}}$ and O_{r_i} as Laplacian frequency phase P_i and $i = 1, ..., L-1$.

$$P_i = O_{r_{i-1}} - O_{r_i} \tag{5.1}$$

For a given scale factor r in this continuous range, the Laplacian frequency

represented HR images O_r can be defined as

$$O_r = O_{r_i} + w_r * P_i \tag{5.2}$$

Here the phase number $i = \lceil (L-1) * (r-1) \rceil$ and w_r is the weight parameter of the edge information for the r scale SR. We define the weight parameter according to distance proportion of the scale r to the O_{r_i} in the phase P_i.

$$w_r = (L-1) * (r_i - r) \tag{5.3}$$

The further evaluation of the accuracy of Laplacian Frequency Representation and the density influence of Laplacian frequency levels are discussed in the experiment section. We find Laplacian frequency represented SR results are similar to the directly learned results and the performance is stable as the Laplacian frequency levels become denser. Therefore, Laplacian Frequency Representation is efficient to generate HR images of continuous decimal scales in the range, with only training several scales, which requires less training data storage and optimization time for generating network that can handle continuous scale range $(1, 2]$.

Recursive Deployment

For SR of any upsampling ratio R in the larger range, it is impossible to train SR samples of all the scales to learn the mapping function of these scales. To minimize the training sample demands, the learned mapping function for SR in the scale range of $(1, 2]$ are reused in the SR predictions of the scale R. Therefore, the HR images of R can be generated by gradually upscaling and recursively deploying through the mapping network with small decimal ratios $r \in (1, 2]$.

we express the R as an integer N power of small decimal ratios $r \in (1, 2]$. The interger N denotes the recursion times for the deployment and the small ratio r is the upsampling ratio at each recursion. To determine the best solution of N and r for any scale R, several comparison experiments are performed in experiment section. As we observed, SR with the larger upscale

ratio r at the early recursions and the smaller recursive deployment times N has better performance than other N and r solutions.

Therefore, for any scale factor R, the recursive times N

$$N = \lceil (\log_2 R) \rceil \tag{5.4}$$

The upscale ratio r_n at each recursion n can be defined as

$$r_n = \begin{cases} 2 & \text{if } n \leq N-1 \\ \frac{R}{2^{N-1}} & \text{if } n = N \end{cases} \tag{5.5}$$

Based on the defined N and r solution for R, if the recursion time N is 1, the HR images of $R = r$ are directly deployed by the network. In other situations, the coarse HR images from the previous recursion are bicubic upscaled with the small ratio r as the input LR images at the current recursion. For better SR performance, at the early $N-1$ recursions, the small ratio $r_n = 2$, and at the N_{th} recursion, $r_n = \frac{R}{2^{N-1}}$.

5.2.2 Any-scale SR deep network

The network for multiple Laplacian frequency levels are constructed based on the parallel multi-scale framework (Lim et al. 2017). The model takes an interpolated LR images as input and predicts at the l_{th} frequency level for r_l scale SR. In our design, the network consists of $L = 11$ Laplacian frequency levels for SR in the scale range $(1, 2]$. The main framework for each frequency level is a global residual network which constructing by two branches: Feature Mapping Branch (FMB) and Image Reconstruction Branch (IRB).

Feature Mapping Branch

In the consideration of the recovered image quality and highlight high-frequency context imformation, the bi-dense structure (Wang, Shen & Zhang 2018) and channel attention modules (Zhang, Li, Li, Wang, Zhong & Fu 2018) are combined in the feature mapping branch (FMB) design as depicted in Figure

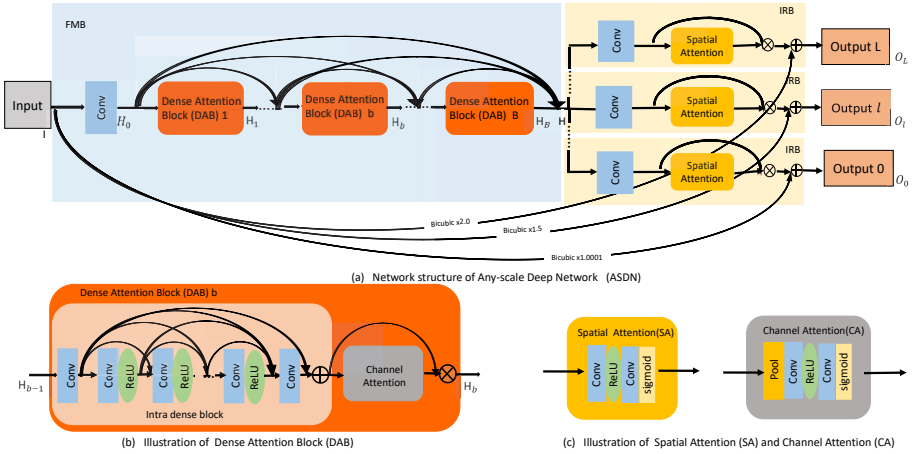

Figure 5.2: (a) The overall architecture of the proposed ASDN network, multiple image reconstruction branches (IRBs) parallel allocate after feature mapping branch (FMB). The FMB adopts the bi-dense structure from DBDN and the spatial attention in IRB is the same spatial attention module from CSFM. (b) The dense attention block (DAB) in a, which combines the Intra dense block from DBDN and channel attention module from RCAN. (c) The illustration of the adopted spatial attention (SA) and channel attention (CA)modules from CSFM and RCAN.

5.2. We keep the high-level architecture design of bi-dense network (Wang et al. 2018) (see Figure 5.2(a)) and replace the original intra dense block with the proposed Dense Attention Block (DAB). Based on the observation that discriminative learning of high-frequency information across feature channels can boost feature learning performance (Zhang, Li, Li, Wang, Zhong & Fu 2018), the proposed DAB employs channel attention module (see Figure 5.2(c)) after concatenated feature channels in the original intra dense block (Wang et al. 2018). Therefore, the high-frequency information of the concatenated channel features are highlighted before preceding into the next block and thus allow the network to focus on more useful channels to improve reconstruction performance.

Image Reconstruction Branch

At each Laplacian frequency level, the locations of the tiny textures need to be recovered are different. These tiny texture regions usually contain high-frequency information, while the smooth areas have more low-frequency information. Therefore, to recover high-frequency details for image SR of different scales, it is helpful to mask out the discriminative high-frequency locations with spatial attention mechanism (Liu et al. 2018) in IRB. As shown in Figure 5.2(b), the learned high-level features are firstly restored into image space by a 3 channel convolutional layer at each Laplacian frequency level. Then the restored image goes into the spatial attention (SA) unit in Figure 5.2(c), which is based on the CSFM (Hu et al. 2018) to mask out the adaptive high-frequency information in the HR images of different scales. To preserve the smooth areas information and concentrate on training high-frequency information, the input interpolated LR images are added with the network output by identity skip connection (SC) to generate HR images. The image reconstruction branch for the each Laplacian frequency level has the same structure and allocates in parallel after the FMB, which is randomly selected at each update for training the 11 Laplacian frequency levels.

5.2.3 Any-scale HR image reconstruction

As mentioned above, we have constructed an Any-scale SR deep network which can effectively restore the HR images of small upscale ratio r within the range of $(1, 2]$ and proposed the Recursive Deployment method to efficiently recursively reconstruct HR images of scales R larger than the range $(1, 2]$. To generate an any-scale HR image of R, we can firstly determine the small upscaling ratio r at each recursion and recursion times N based on equation 5.4 and 5.5. According to the idea that any upscale decimal ratio R can be expressed as an integer N power of decimal ratios r in a small range, the HR image of any-scale R can be generated by recursively deployed N times through ASDN with small ratio r.

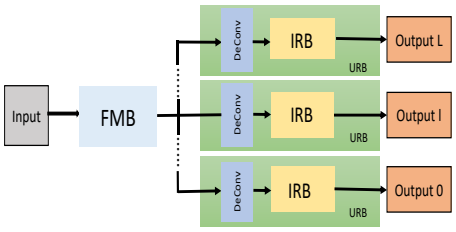

Figure 5.3: The architecture of fixed-scale deep network (FSDN) for SR of multiple integer scales. FMB is the Feature Mapping Branch structure in ASDN and the Upscaling Reconstruction Branches (URB) parallel construct behind FMB for multi-scale SR, which are built by a deconvolutional layer followed with Image Reconstruction Branch (IRB).

5.2.4 Fixed-Scale Deep Network

For some practical applications where only require SR of specific scales, our ASDN can be fine-tuned to a fixed-scale network (FSDN) to further improve the reconstruction accuracy for the scales of interest by training the corresponding LR and HR training scale samples. In our design, the LR images are downscaled from original HR image on the specific scales with bicubic interpolation. FSDN shares the same network structure as ASDN, except the specific scale deconvolutional layer is inserted at the front of each IRB, which follows the common multi-scale single upscaling SR networks (Lim et al. 2017). As shown in Figure 5.3, the upscaling reconstruction branch (URB) is consisted of a transposed layer and IRB, and multi-scale URBs are parallel-arranged after FMB. At each training epoch, the network will select the specific-scale training samples and URB to optimize FSDN.

5.3 Experiments

In this section, we describe the implementation details of our models, including both model hyper-parameters, training and testing details. Then we compare the proposed any-scale network and the fine-tuned fixed-scale model

with several state-of-art SR methods on both fixed and any scale benchmark datasets including the quantitative, qualitative comparisons and any-scale comparisons. The effectiveness evaluation of the proposed any-scale method and the contribution study of different components in the proposed any-scale deep network are also provided in the paper.

5.3.1 Implementation Details

Network settings In the proposed ASDN, all convolutional layers have 64 filters and 3×3 kernel size except the layers in IRB for restoring images and the convolutional layers in CA and SA units. The layers for image restoration have 3 filters and all the convolutional layers in CA and SA units are 1×1 kernel size, which adopt the same setting as CSFM (Hu et al. 2018). Meanwhile, the 3×3 kernel size convolutional layer zero-pads the boundaries before applying convolution to keep the size of all feature maps the same as the input of each level. ASDN and FSDN share the same FMB structure, where 16 DAB are densely connected and each DAB has 8 dense layers. But in the URB of FSDN, the deconvolutional layer settings follow single upsampling networks (Lim et al. 2017) to upscale feature mappings with the corresponding scales.

Training details The original training images are from DIV2K dataset and Flicker dataset (Agustsson & Timofte 2017). The input LR images for ASDN are bicubic interpolated from the training images with 11 decimal ratios r, which are evenly-distributed in the range of $(1, 2]$. In each training batch, 16 randomly augmented RGB patches with the size of 48×48 are extracted from LR images as the input, and the LR images are randomly selected from one scale training samples among the total 11 scales training data. Here the data augmentation includes horizontal flips and 90 degree rotations are adopted on each patch. At each update of training the ASDN, the FMB and the corresponding scale IRB are enabled and updated with the scale related training batch. To fine-tune the FSDN, the input LR images are downscaled by the scale factor among $2\times, 3\times, 4\times$, and $8\times$. In

the training batch, a batch of 96×96 size patches is used as the targets and the corresponding scale LR RGB patches to optimize the specific scale modules. In general, ASDN and FSDN are all built with the platform Torch and optimized by Adam (Kingma & Ba 2014) with L1 (Shalev-Shwartz & Tewari 2011) loss by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate is initially set to 10^{-4} and halved at every 2×10^5 minibatch updates for 10^6 total minibatch updates.

Testing details Our proposed networks are tested on five widely-used benchmark datasets for image SR: Set5 (Bevilacqua et al. 2012), Set14 (Zeyde et al. 2010), BSD100 (Timofte et al. 2014), Urban100 (Huang et al. 2015) and Manga109 (Matsui et al. 2017). To test any-scale network (ASDN) for SR of a random scale s, the testing images are first downscaled with the scale factor s as the LR images. If the scale s is not larger than 2, the LR images are upsampled with scale s and forwarded into the ASDN with the two enabled neighbouring Laplacian frequency levels of the scale s. HR images are predicted by interpolating these two levels based on Eq. 5.2. While if the scale s is larger than 2, the testing recursion times are based on $N = \lceil (\log_2 R) \rceil$. The previous recursion outputs are upscaled and deployed through ASDN with $r(n)$ as the Eq. 5.5 at each recursion n, except the initial recursion which uses the LR images as input. To test fixed-scale network (FSDN), the testing input images are downscaled by the 7fixed scales s and deployed into the FSDN with the scale corresponding modules are enabled to yield the testing output.

5.3.2 Comparison with State-of-arts

To confirm the ability of the proposed methods, We first compare with state-of-art SR algorithms for qualitative and quantitative analysis on the normal fixed scales $2\times, 3\times, 4\times, 8\times$, which includes dictionary-based methods (A+ (Timofte et al. 2014), RFL (Schulter et al. 2015)), self-similarity based method (SelfExr (Huang et al. 2015)), predefined upsampling methods (SRCNN (Dong et al. 2014), VDSR (Kim et al. 2016*a*), DRRN (He et al. 2016),

Table 5.1: Quantitative evaluation of state-of-the-art SR algorithms. We report the average PSNR/SSIM for 2×, 3×, 4× and 8× SR. **Black** indicates the best performance

scale	Algorithms	Set5		Set14		BSD100		Urban100		Manga109	
		PSNR	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**	**PSNR**	**SSIM**
2×	Bicubic	33.64	0.929	30.22	0.868	29.55	0.842	26.66	0.841	30.84	0.935
	A+ (Timofte et al. 2014)	36.54	0.954	32.28	0.905	31.22	0.886	28.87	0.896	35.53	0.966
	RFL (Schulter et al. 2015)	36.59	0.954	32.28	0.905	31.18	0.885	29.55	0.898	35.12	0.967
	SelfExSR (Huang et al. 2015)	36.60	0.955	32.24	0.904	31.20	0.887	29.55	0.898	35.82	0.969
	SRCNN (Dong et al. 2014)	36.65	0.954	32.29	0.903	31.36	0.888	29.52	0.895	35.72	0.968
	VDSR (Kim et al. 2016*a*)	37.53	0.958	32.97	0.913	31.90	0.896	30.77	0.914	37.16	0.974
	DRRN (He et al. 2016)	37.74	0.959	33.23	0.914	32.05	0.897	31.23	0.919	37.52	0.976
	LapSRN (Lai et al. 2017*a*)	37.52	0.959	33.08	0.913	31.80	0.895	30.41	0.910	37.27	0.974
	MemNet (Tai, Yang, Liu & Xu 2017)	37.78	0.959	33.28	0.914	32.08	0.898	31.33	0.919	37.72	0.974
	SRMDNF (Zhang, Zuo & Zhang 2018)	37.79	0.960	33.32	0.916	32.05	0.899	31.33	0.920	38.07	0.976
	EDSR (Lim et al. 2017)	38.11	0.960	33.92	0.920	32.32	0.901	32.93	0.935	39.10	0.976
	RDN (Zhang, Tian, Kong, Zhong & Fu 2018)	38.24	0.961	34.01	0.921	32.34	0.902	32.96	0.936	39.19	0.978
	DBPN (Haris et al. 2018)	38.09	0.961	33.85	0.920	32.27	0.900	32.55	0.932	38.89	0.978
	RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018)	38.27	0.961	34.12	0.922	32.41	0.903	**33.34**	**0.938**	39.44	0.979
	ASDN(ours)	38.12	0.961	33.82	0.919	32.30	0.901	32.47	0.931	39.16	0.978
	FSDN(ours)	**38.27**	**0.961**	**34.18**	**0.923**	**32.41**	**0.903**	33.13	0.937	**39.49**	**0.979**
3×	Bicubic	30.39	0.867	27.53	0.774	27.20	0.738	24.47	0.737	26.99	0.859
	A+ (Timofte et al. 2014)	32.59	0.909	29.13	0.819	28.30	0.784	26.03	0.797	29.93	0.912
	RFL (Schulter et al. 2015)	32.47	0.906	29.07	0.818	28.23	0.782	25.88	0.792	29.61	0.905
	SelfExSR (Huang et al. 2015)	32.66	0.910	29.18	0.821	28.30	0.786	26.45	0.810	27.57	0.905
	SRCNN (Dong et al. 2014)	32.75	0.909	29.30	0.822	28.41	0.786	26.25	0.801	30.59	0.914
	VDSR (Kim et al. 2016*a*)	33.66	0.921	29.77	0.831	28.82	0.798	27.41	0.830	32.01	0.934
	DRRN (He et al. 2016)	34.03	0.924	29.96	0.835	28.95	0.800	27.53	0.764	32.42	0.939
	LapSRN (Lai et al. 2017*a*)	33.82	0.922	29.87	0.832	28.82	0.798	27.07	0.828	32.21	0.935
	MemNet (Tai et al. 2017)	34.09	0.925	30.00	0.835	28.96	0.800	27.57	0.839	32.51	0.937
	SRMDNF (Zhang, Zuo & Zhang 2018)	34.12	0.925	30.04	0.838	28.97	0.802	27.57	0.839	33.00	0.940
	EDSR (Lim et al. 2017)	34.65	0.928	30.52	0.846	29.25	0.809	28.80	0.865	34.17	0.948
	RDN (Zhang, Tian, Kong, Zhong & Fu 2018)	34.71	0.929	30.57	0.847	29.26	0.809	28.80	0.865	34.13	0.948
	RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018)	34.74	0.930	**30.65**	0.848	29.32	0.811	**29.09**	**0.870**	34.44	0.949
	ASDN(ours)	34.48	0.928	30.35	0.843	29.18	0.808	28.45	0.858	33.87	0.947
	FSDN(ours)	**34.75**	**0.930**	30.63	**0.848**	**29.33**	**0.811**	28.98	0.868	**34.53**	**0.950**
4×	Bicubic	28.42	0.810	26.10	0.704	25.96	0.669	23.15	0.660	24.92	0.789
	A+ (Timofte et al. 2014)	30.30	0.859	27.43	0.752	26.82	0.710	24.34	0.720	27.02	0.850
	RFL (Schulter et al. 2015)	30.17	0.855	27.24	0.747	26.76	0.711	24.20	0.712	26.80	0.841
	SelfExSR (Huang et al. 2015)	30.34	0.862	27.41	0.753	26.84	0.710	24.83	0.740	27.83	0.866
	SRCNN (Dong et al. 2014)	30.49	0.862	27.61	0.754	26.91	0.712	24.53	0.724	27.66	0.858
	VDSR (Kim et al. 2016*a*)	31.35	0.882	28.03	0.770	27.32	0.730	25.18	0.750	28.82	0.886
	DRRN (He et al. 2016)	31.68	0.889	28.21	0.772	27.38	0.728	25.44	0.764	29.18	0.891
	MemNet (Tai et al. 2017)	31.74	0.889	28.26	0.772	27.40	0.728	25.50	0.763	29.42	0.894
	SRMDNF (Zhang, Zuo & Zhang 2018)	31.96	0.892	28.35	0.778	27.49	0.734	25.68	0.773	30.09	0.902
	LapSRN (Lai et al. 2017*a*)	31.54	0.885	28.19	0.772	27.32	0.727	25.21	0.756	29.46	0.890
	EDSR (Lim et al. 2017)	32.46	0.896	28.80	0.788	27.71	0.742	26.64	0.803	31.02	0.915
	RDN (Zhang, Tian, Kong, Zhong & Fu 2018)	32.47	0.899	28.81	0.787	27.72	0.742	26.61	0.803	31.00	0.915
	DBPN (Haris et al. 2018)	32.42	0.898	28.76	0.786	27.68	0.740	26.38	0.796	30.91	0.914
	RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018)	32.63	0.900	28.87	0.789	27.77	0.744	**26.82**	**0.809**	31.22	0.917
	ASDN(ours)	32.27	0.896	28.66	0.784	27.65	0.740	26.27	0.792	30.91	0.913
	FSDN(ours)	**32.63**	**0.900**	**28.89**	**0.789**	**27.79**	**0.744**	26.79	0.807	**31.44**	**0.919**
8×	Bicubic	24.40	0.658	23.10	0.566	23.97	0.548	20.74	0.516	21.47	0.650
	A+ (Timofte et al. 2014)	25.52	0.692	23.98	0.597	24.20	0.568	21.37	0.545	22.39	0.680
	RFL (Schulter et al. 2015)	25.36	0.677	23.88	0.588	24.13	0.562	21.27	0.535	22.27	0.668
	SelfExSR (Huang et al. 2015)	25.52	0.704	24.02	0.603	24.18	0.568	21.81	0.576	22.99	0.718
	SRCNN (Dong et al. 2014)	25.33	0.690	23.76	0.591	24.13	0.566	21.29	0.544	22.46	0.695
	VDSR (Kim et al. 2016*a*)	25.93	0.724	24.26	0.614	24.49	0.583	21.70	0.571	23.16	0.725
	LapSRN (Lai et al. 2017*a*)	26.15	0.738	24.35	0.620	24.54	0.586	21.81	0.581	23.39	0.735
	MemNet (Tai et al. 2017)	26.16	0.741	24.38	0.619	24.58	0.584	21.89	0.583	23.56	0.738
	SRMDNF (Zhang, Zuo & Zhang 2018)	37.79	0.960	33.32	0.916	32.05	0.899	31.33	0.920	38.07	0.976
	EDSR (Lim et al. 2017)	26.96	0.776	24.91	0.642	24.81	0.599	22.51	0.622	24.69	0.784
	DBPN (Haris et al. 2018)	27.21	0.784	25.13	0.648	24.88	0.601	22.73	0.631	25.14	0.799
	RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018)	27.31	0.788	25.23	0.651	24.98	**0.606**	**23.00**	**0.645**	25.24	0.803
	ASDN(ours)	27.02	0.776	24.99	0.641	24.82	0.600	22.57	0.620	24.73	0.748
	FSDN(ours)	**27.33**	**0.789**	**25.24**	**0.651**	**24.98**	0.604	22.90	0.638	**25.24**	**0.803**

MemNet (Tai et al. 2017) and SRMDNF (Zhang, Zuo & Zhang 2018)), and single upsampling methods (RDN (Zhang, Tian, Kong, Zhong & Fu 2018), Lapsrn (Lai et al. 2017*a*), EDSR (Lim et al. 2017), RCAN (Zhang, Li, Li, Wang, Zhong & Fu 2018)).

Quantitative Comparison

We compare the performance of our any-scale SR networks with the state-of-art on the five challenging dataset benchmarks. To evaluate image SR performance, PSNR (Hore & Ziou 2010) and SSIM (Wang et al. 2004) quality metrics are used for quantitative evaluation in this paper. PSNR is defined by the mean square error between the ground truth and the reconstructed image. SSIM is a method for measuring the similarity between two images by luminance, contrast, and structure comparisons. Note that the higher PSNR and SSIM values indicate the better quality. Following previous works, we measure PSNR and SSIM on the Y-channel, and ignore the same amount of pixels as the scales from the boarder.

Table 5.1 shows quantitative comparisons for $\times 2, 3, 4, 8$ SR. For fair comparisons with the recent single upsampling networks, we fine-tune the ASDN with the fixed $\times 2, 3, 4, 8$ scale SR samples as FSDN for reference. It is obvious that FSDN has better performance than all the state-of-art methods, except RCAN on some datasets. Compared to the recent published DBPN and RDN, FSDN outperforms about 0.05 dB. Although on Urban100, which is consisted of straight line building structure images, RCAN has better performance than FSDN due to the more channel attentions across the network, and more sensible to the sharp edges in the image reconstruction. On other datasets, FSDN reconstruction accuracy is comparable to RCAN. This indicates the network, which is the same main framework as ASDN is effective to learn mapping functions for SR tasks.

Due to the strong ability of the framework, our ASDN performs favorably against the existing methods especially compared to the predefined upsampling methods. Noted that ASDN does not use any $3\times, 4\times, 8\times$ SR samples

for training but still generates comparable results as EDSR. There are mainly two reasons for ASDN drops behind some upsampling models. First, these upsampling models are trained with fixed-scale SR samples, and customized for the $2\times, 3\times, 4\times, 8\times$ scales deployments, but ASDN is trained with scales in $(1, 2]$. Second, the upsampling layers (Schulter et al. 2015) can improve the reconstruction performance, as shown in our experiment, FSDN (the upsampling version of ASDN) has more than $0.1dB$ PSNR compared to ASDN on scale $2\times$. However, some of the upsampling layers can only apply for SR of the integer scales (Schulter et al. 2015), such as transposed layers. Although, Meta-upsampling (Hu et al. 2019) layer can upscale images with decimal scales, these scale factors need to be trained before deployment. Therefore, we compromise some reconstruction accuracy for the continuous scale SR using the predefined upsampling structure, which only requires to be trained with several representative scales and our ASDN is still very profound on the normal fixed scales compared with the existing predefined upsampling deep methods.

Any-Scale Comparison

In this section, in order to evaluate the efficiency of our ASDN for any upscale ratio SR, we firstly compare ASDN with other methods. The Bicubic interpolation method is adopted as the reference, and some deep learning networks frameworks (EDSR, RDN, VDSR) are retrained with the proposed any-scale SR method and the same training data as our ASDN for any-scale SR comparison, denoted as EDSR-Conv, RDN-Conv and VDSR-Conv. Meta-EDSR and Meta-RDN (Hu et al. 2019) are dynamic meta-upsampling models which are trained with scale factors from $\times 1$ to $\times 4$ at the stride of 0.1.

The experimental results are shown in Table 5.2, which use the PSNR value for comparison. The first row shows the PSNR value on SR of 9 scales from $\times 1.1$ to $\times 1.9$, which are trained with these models and it is obvious that our ASDN reaches the state-of-art performance. The second row

Table 5.2: Results of any-scale SR on different methods tested on BSD100. The first row shows the results of Laplacian frequency levels and the second column row demonstrates SR performance on randomly selected scales and boundary condition. **Black** indicates the best performance

Method \ Scale	**X1.1**	**X1.2**	**X1.3**	**X1.4**	**X1.5**	**X1.6**	**X1.7**	**X1.8**	**X1.9**
Bicubic	36.56	35.01	33.84	32.93	32.14	31.49	30.90	30.38	29.97
VDSR-Conv	42.13	39.52	37.88	36.53	35.42	34.50	33.72	33.03	32.41
EDSR-Conv	42.92	40.11	38.33	36.93	35.79	34.85	34.06	33.38	32.75
RDN-Conv	42.86	40.04	38.25	36.86	35.72	34.78	33.99	33.29	32.67
Meta-EDSR (Hu et al. 2019)	42.72	39.92	38.16	36.84	35.78	34.83	34.06	33.36	32.78
Meta-RDN (Hu et al. 2019)	42.82	**40.40**	38.28	36.95	35.86	34.90	34.13	33.45	32.86
ASDN(ours)	**43.05**	40.24	**38.42**	**37.02**	**35.87**	**34.92**	**34.14**	**33.46**	**32.86**
Method \ Scale	**X2.0**	**X2.8**	**X4.0**	**X5.7**	**X8.0**	**X11.3**	**X16.0**	**X22.6**	**X32.0**
Bicubic	29.55	27.53	25.96	24.96	23.67	22.65	21.73	20.73	19.90
VDSR-Conv	31.89	29.23	27.25	25.56	24.58	23.49	22.47	21.39	20.38
EDSR-Conv	32.23	29.54	27.58	26.01	24.78	23.65	22.63	21.55	20.53
RDN-Conv	32.07	29.47	27.51	25.94	24.72	23.60	22.58	21.50	20.51
Meta-EDSR (Hu et al. 2019)	32.26	29.61	27.67	-	-	-	-	-	-
Meta-RDN (Hu et al. 2019)	**32.35**	**29.67**	**27.75**	-	-	-	-	-	-
ASDN(ours)	32.30	29.63	27.65	**26.07**	**24.85**	**23.70**	**22.66**	**21.59**	**20.55**

illustrates ASDN efficiency on the scales not trained before and evaluates the effective scale range of our proposed any-scale SR network. For SR of scales out of the range, ASDN is comparable to Meta-EDSR, but slightly drops behind Meta-RDN. This is due to ASDN is the recursively deployed results and Meta-RDN is customized with these scales. Although the recursively deployed SR results have slight dropback as the directly deployed results, recursive deployment can still effectively generate SR of scales not trained before. Through this way, ASDN only needs 11 training scales for any-scale SR, but Meta-EDSR entails 40 scale training sets and only works for SR of the trained scales. Furthermore, ASDN always keeps a gap between Bicubic interpolated results on BSD100 as the testing scale goes larger. Although the gap decreases gradually as the scale factor becomes larger. The gap is about 3dB within scale range $(2, 4]$, about 2dB within scale range $(4, 8]$ and about 1dB for scales larger than $\times 8$. We find the scale boundary condition is limited by the testing HR image size. When the testing scale factor is very

large, the downscaled LR image may be single pixel level which is too small for further downsampling and precise HR prediction.

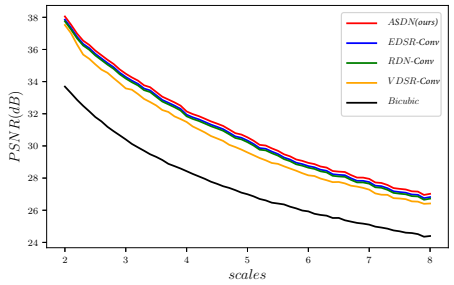

Figure 5.4: PSNR comparison of ASDN with other works within continuous scale range $(\times 2, \times 8]$ on Set5

Figure 5.4 shows the any-scale SR results on a continuous scale range. We test our any-scale network performance with random decimal scales distributed in the commonly used range of $\times 2$ to $\times 8$ on Set5 and plot out the results into the line. It is proved that ASDN and the models trained with our any-scale SR method can effectively reconstruct HR images of continuous upscale ratios. Our ASDN outperforms all the other methods, which is generally 0.15 dB better than EDSR-Conv, outperforms VDSR-Conv by 0.6 dB and keeps the robust more than 3dB PSNR deference from Bicubic method in the continuous scale range. The result demonstrates our ASDN can effectively reconstruct HR images of continuous upscale ratios and our any-scale training method is flexible to many deep CNN networks.

Qualitative Comparison

We show visual comparisons on the testing datasets for $2\times$, $4\times$ and $8\times$ SR. For $2\times$ enlargement of Set14 in Figure 5.5, FSDN suppresses the bias artifacts and recovers the cloth pattern and text closer to the ground truths than all the other methods. Meanwhile ASDN tends to construct less biased images than other methods. For $4\times$ enlargement of the parallel straight lines

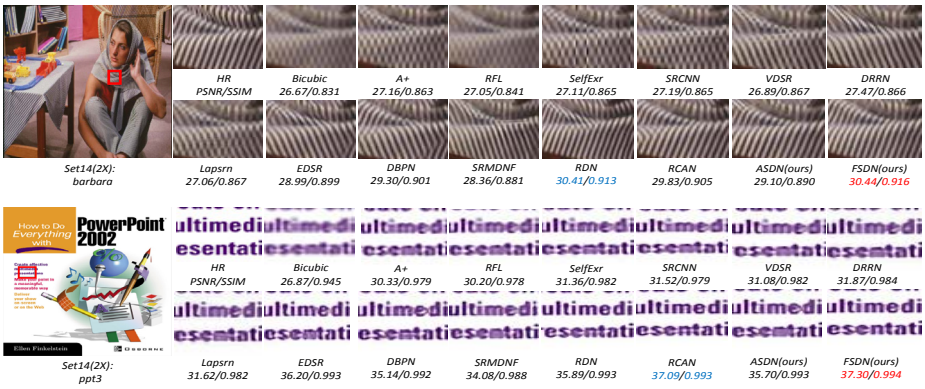

Figure 5.5: Qualitative comparison of our models with other works on $\times 2$ super-resolution Red indicates the best performance, blue indicates the second best

in Figure 5.6, Our methods generates the clearer building line, while other methods suffer the blurring artifacts. RCAN tends to generate misleading strong edges due to the more channel attention structure, but our ASDN and FSDN generates soft patterns more close to the ground truth. The reconstruction performance on $8\times$ SR is further analyzed in Figure 5.7. FSDN restores the sharper characters than the compared networks and ASDN is able to recover more accurate textures from the distorted LR image than many other fixed-scale methods.

5.3.3 Study of Any-Scale Methods

We study the effects of Laplacian Frequency Representation and Recursive Deployment of the any-scale SR methods.

Laplacian Frequency Representation

To evaluate the accuracy of the Laplacian Frequency Representation for continuous scale SR. We compare the reconstruction results of the Laplacian Frequency Representation with the direct deployed HR images of 100 scales

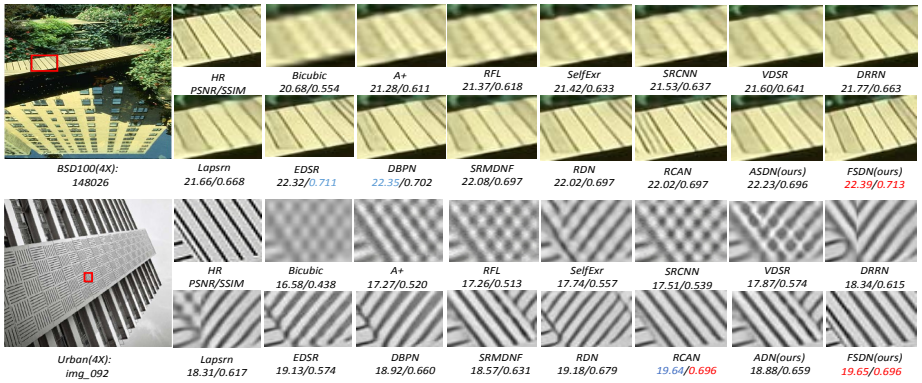

Figure 5.6: Qualitative comparison of our models with other works on $\times 4$ super-resolution Red indicates the best performance, blue indicates the second best

Figure 5.7: Qualitative comparison of our models with other works on $\times 8$ super-resolution Red indicates the best performance, blue indicates the second best

in the range $(1, 2]$. We first modify EDSR, RDN and ASDN frameworks into the single predefined upsampling networks and train them with these 100 scales SR samples as EDSR-100, RDN-100 and ASDN-100 to generate HR images of Set5 on the 100 scales. Then we reconstruct the single redefined upsampling EDSR-100 and RDN-100 with 11 parallel IRBs as EDSR-Conv and RDN-Conv, which as suggested in Sec. 5.3.2 trained with the same method and data as ASDN. As shown in Figure 5.8, It is obvious that the

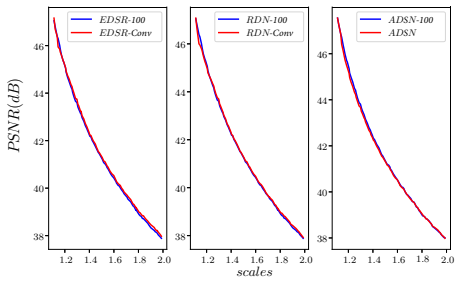

Figure 5.8: PSNR of Laplacian frequency represented and specific scale trained HR images of scales in $(1, 2]$ on Set5

Laplacian frequency represented HR images have similar quality to the direct deployed HR images.

To analyze the influence of the Laplacian frequency level density on the SR performance, we train ASDN on $5, 9, 17$ evenly distributed upscale decimal ratios in $(1, 2]$ with DIV2K, which separates the Laplacian Frequency Representation into $4, 8$ and 16 phases and names ASDN-4, ASDN-8, and ASDN-16 separately. Figure 5.9 demonstrates the performance of the three versions of ASDN with scales in $(1, 2]$. In order to make the difference more obvious, we choose some scale ranges in $(1, 2]$. It illustrates that ASDN-4 drops behind ASDN-8 and ASDN-16 commonly, and ASDN-8 and ASDN-16 almost overlap. The results show the Laplacian frequency level density influences SR performance. In some extent, the model trained with more dense scales achieves better performance, but it saturates beyond a certain point, such as 10 phases. Due to this reason, we can generate HR images of any decimal scale in the range of $(1, 2]$ by the several Laplacian frequency levels in $(1, 2]$.

Recursive Deployment

In order to investigate the effects of recursive deployment for HR images of larger decimal scales. We mainly demonstrate the comparison of recursive

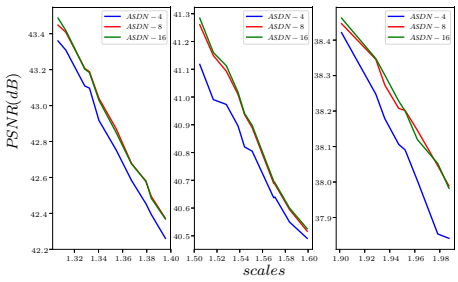

Figure 5.9: PSNR of various ASDNs trained with different Laplacian frequency level densities

deployment and direct deployment on scales $\times 2, \times 3, \times 4$ We trained VDSR-Conv, EDSR-Conv, RDN-Conv and ASDN with 11 evenly distributed upscale decimal ratios in $(1, 2]$ as the recursive models and the HR images are twice upscaled with the upscale ratios $\times\sqrt{2}$, $\times\sqrt{3}$, $\times\sqrt{4}$. To form the fair comparisons, we trained VDSR-Conv, EDSR-Conv, RDN-Conv and ASDN with $\times 2, \times 3, \times 4$ SR images as the direct deployment models. Table 5.3 illustrates the PSNR of recursive deployment and direct deployment. It is obvious that recursive deployment generally leads to the SR performance decline compared to the direct deployment. But the difference between recursive deployment and direct deployment goes down as the scale goes up. Since the decline is still in an acceptable range and goes gentle as the upscale ratios up, we adopt recursive deployment for SR in higher upscale ratio ranges.

Methods	Direct deployment			Recursive deployment		
	×2	×3	×4	×2	×3	×4
VDSR-Conv	37.57	33.77	31.56	36.86	33.70	31.50
EDSR-Conv	38.04	34.45	32.29	37.18	34.32	32.26
RDN-Conv	38.05	34.46	32.31	37.27	34.38	32.23
Ours	38.12	34.52	32.28	37.35	34.43	32.27

Table 5.3: PSNR of the recursive deployment and direct deployment on SR for $\times 2, \times 3, \times 4$

To determine the best solution of recursive times N and upscale ratios

r for recursive deployment. We also explore various combinations of N and r to deploy any-scale HR images with different strategies. Table 5.4 illustrates the performance of the HR images deployed by different strategies with ASDN on Set5. It is obvious that the larger upscale ratio r combined with the smaller recursive time N will contribute to a better performance. Furthermore, choosing the larger upscale ratios in the early recursions can produce the better results than using the smaller scales. In these reasons, we recommend to choose $N = \lceil(\log_2 R)\rceil$ with the largest upscale ratios $r = 2$ at the early $N - 1_{th}$ recursions for large scale SR.

Scale(R)	**Recursion(N)**	**UpscaleRatio(r)**	**PSNR**
2×	1	2.000	**38.12**
	2	1.414, 1.414	37.35
		1.333, 1.500	37.03
		1.500, 1.333	37.54
3×	2	1.732, 1.732	34.43
		1.500, 2.000	34.19
		2.000, 1.500	**34.48**
	3	1.442, 1.442, 1.442	33.18
4×	2	2.000, 2.000	**32.27**
	3	1.587, 1.587, 1.587	31.96
		1.800, 1.800, 1.234	32.16
		1.800, 1.234, 1.800	31.86
		2.000, 1.800, 1.100	32.24

Table 5.4: PSNR of recursive deployment and direct deployment on SR for $\times 2, \times 3, \times 4$. **Black** indicates the best performance

5.3.4 Abviation Study

In this section, we evaluate the influence of different network modules, such as channel attention (CA) in FMB, spatial attention (SA) in IRB and skip connection (SC) between input and output. To demonstrate the effect of CA in the proposed network structure, we remove the CA from the FMB. In Table 5.5, we can see when CA is removed, the PSNR value on Set5 ($\times 2$) is relatively low compared to the model having CA. To investigate the effect of SA, we remove the SA from the ASDN to compare with the network with SA. SA can improve performance by 0.02 dB or 0.01 dB with or without CA in

the models. We further investigate the contribution of SC to the network by comparing the models with or without SC. Adding global skip connections between the network input and output generally improves 0.04 dB on Set5. Generally combining attention modules and skip connection into the network design, helps the residual high-frequency information reconstruction.

Module	Different combination of CA, SA and SC							
CA	×	×	×	√	√	√	×	√
SA	×	×	√	×	√	×	√	√
SC	×	√	×	×	×	√	√	√
PSNR	37.92	37.96	37.93	37.95	37.97	37.99	37.97	**38.01**

Table 5.5: Investigate of channel attention (CA), spatial attention (SA) and skip connection (SC). **Black** indicates the best performance

5.4 Conclusion

In this paper, we propose an any-scale deep network (ASDN) to generate HR images of any scale with one unified network by adopting our proposed any-scale SR method, including Laplacian Frequency Representation for SR of small continuous scale ranges and Recursive Deployment for larger scale SR. Our method helps to reduce the demands of training scale samples for any-scale SR, which accelerate the network convergence. The extensive comparisons show our ASDN is superior to the most state-of-art methods on both fixed-scale and any-scale benchmarks.

Chapter 6

Summary

In summary, this thesis presents several techniques to address the image super-resolution problem.

In chapter 3, we present a bi-dense structure for image SR networks, which is the extension of deep dense based SR method. Firstly, we propose the compact intra-dense block where each layer is connected to all the other layers to learn local features. In order to keep the model compact, each block output and input are compressed by the convolutional neural layer. Then inter-block dense net is designed to connect these intra-dense blocks for creating the SR model. Since the features learned in the early blocks of the network matter to the task of SR, the inter-block dense connection allows all of the early block information to be reused for learning features in the later block.

In chapter 4, memory optimization of dense network is introduced into image SR to solve memory hungry and computation quadratically growth of dense models. To reduce the memory consumption, we propose a novel memory-optimized dense model which replaces the concatenation with summation and adopt shared memory allocation strategy to store the concatenated features and Batch Normalization intermediate feature maps to reduce the training memory cost of network.

In chapter 5, we present an Any-Scale Deep Network (ASDN) to generate

HR images of arbitrary scales. Firstly, the HR images of the continuous scales are predicted by Laplacian Frequency Representation, which are based on the weighted interpolation of their two neighbouring Laplacian frequency levels, which efficiently reduces the training scale demands for learning the continuous scale SR. Then, Recursive Deployment is used for generating the HR images of larger upsampling ratio. As we have the mapping function between the interpolated LR images and HR images of small scales, the HR images of the larger scales can be gradually upsampled and recursively deployed. It extends one model to any-scale SR with only few training scales.

Each chapter (i.e. from Chapter 3 to Chapter 5) of this thesis is supported by at one published or submitted papers[1] listed in **List of Publications**. Therefore, what we have done and propose in this thesis is of great significance to image super-resolution problem. In the future, we might extend the SR algorithms into a real-time application.

[1]The papers of chapter 3 and 4 are published, and the paper of chapter 5 is still under review

Bibliography

Agustsson, E. & Timofte, R. (2017), Ntire 2017 challenge on single image super-resolution: Dataset and study, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops', Vol. 3, p. 2.

Babacan, S. D., Molina, R. & Katsaggelos, A. K. (2008), Total variation super resolution using a variational approach, *in* '2008 15th IEEE International Conference on Image Processing', IEEE, pp. 641–644.

Bevilacqua, M., Roumy, A., Guillemot, C. & Alberi-Morel, M. L. (2012), 'Low-complexity single-image super-resolution based on nonnegative neighbor embedding'.

Burt, P. J. & Adelson, E. H. (1987), The laplacian pyramid as a compact image code, *in* 'Readings in Computer Vision', Elsevier, pp. 671–679.

Chang, H., Yeung, D.-Y. & Xiong, Y. (2004), Super-resolution through neighbor embedding, *in* 'Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.', Vol. 1, IEEE, pp. I–I.

Chen, M.-J., Huang, C.-H. & Lee, W.-L. (2005), 'A fast edge-oriented algorithm for image interpolation', *Image and Vision Computing* **23**(9), 791–798.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C. & Zhang, Z. (2015), 'Mxnet: A flexible and efficient machine

learning library for heterogeneous distributed systems', *arXiv preprint arXiv:1512.01274* .

Chen, T., Xu, B., Zhang, C. & Guestrin, C. (2016), 'Training deep nets with sublinear memory cost', *arXiv preprint arXiv:1604.06174* .

Dai, S., Han, M., Wu, Y. & Gong, Y. (2007), Bilateral back-projection for single image super resolution, *in* '2007 IEEE International Conference on Multimedia and Expo', IEEE, pp. 1039–1042.

Dong, C., Loy, C. C., He, K. & Tang, X. (2014), Learning a deep convolutional network for image super-resolution, *in* 'European Conference on Computer Vision', Springer, pp. 184–199.

Haris, M., Shakhnarovich, G. & Ukita, N. (2018), Deep backprojection networks for super-resolution, *in* 'Conference on Computer Vision and Pattern Recognition'.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Hore, A. & Ziou, D. (2010), Image quality metrics: Psnr vs. ssim, *in* 'Pattern recognition (icpr), 2010 20th international conference on', IEEE, pp. 2366–2369.

Hu, X., Mu, H., Zhang, X., Wang, Z., Sun, J. & Tan, T. (2019), 'Meta-sr: A magnification-arbitrary network for super-resolution', *arXiv preprint arxiv:1903.00875v1* .

Hu, Y., Li, J., Huang, Y. & Gao, X. (2018), 'Channel-wise and spatial feature modulation network for single image super-resolution', *arXiv preprint arXiv:1809.11130* .

Huang, G., Liu, Z., Van Der Maaten, L. & Weinberger, K. Q. (2017), Densely connected convolutional networks, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4700–4708.

Huang, G., Sun, Y., Liu, Z., Sedra, D. & Weinberger, K. Q. (2016), Deep networks with stochastic depth, *in* 'European Conference on Computer Vision', Springer, pp. 646–661.

Huang, J.-B., Singh, A. & Ahuja, N. (2015), Single image super-resolution from transformed self-exemplars, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 5197–5206.

Hung, K.-W. & Siu, W.-C. (2011), Fast video interpolation/upsampling using linear motion model, *in* '2011 18th IEEE International Conference on Image Processing', IEEE, pp. 1341–1344.

Hung, K.-W. & Siu, W.-C. (2012*a*), 'Fast image interpolation using the bilateral filter', *IET Image Processing* **6**(7), 877–890.

Hung, K.-W. & Siu, W.-C. (2012*b*), 'Robust soft-decision interpolation using weighted least squares', *IEEE Transactions on Image Processing* **21**(3), 1061–1069.

Ioffe, S. & Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International Conference on Machine Learning', pp. 448–456.

Irani, D. G. S. B. M. (2009), Super-resolution from a single image, *in* 'Proceedings of the IEEE International Conference on Computer Vision, Kyoto, Japan', pp. 349–356.

Karras, T., Aila, T., Laine, S. & Lehtinen, J. (2017), 'Progressive growing of gans for improved quality, stability, and variation', *arXiv preprint arXiv:1710.10196* .

Keys, R. (1981), 'Cubic convolution interpolation for digital image processing', *IEEE transactions on acoustics, speech, and signal processing* **29**(6), 1153–1160.

Kim, J., Kwon Lee, J. & Mu Lee, K. (2016*a*), Accurate image super-resolution using very deep convolutional networks, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1646–1654.

Kim, J., Kwon Lee, J. & Mu Lee, K. (2016*b*), Deeply-recursive convolutional network for image super-resolution, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1637–1645.

Kingma, D. P. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Lai, W.-S., Huang, J.-B., Ahuja, N. & Yang, M.-H. (2017*a*), Deep laplacian pyramid networks for fast and accurate superresolution, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', Vol. 2, p. 5.

Lai, W.-S., Huang, J.-B., Ahuja, N. & Yang, M.-H. (2017*b*), 'Fast and accurate image super-resolution with deep laplacian pyramid networks', *arXiv:1710.01992* .

LeCun, Y., Bengio, Y. & Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436.

Li, X. & Orchard, M. T. (2001), 'New edge-directed interpolation', *IEEE transactions on image processing* **10**(10), 1521–1527.

Lim, B., Son, S., Kim, H., Nah, S. & Lee, K. M. (2017), Enhanced deep residual networks for single image super-resolution, *in* 'The IEEE conference on computer vision and pattern recognition (CVPR) workshops', Vol. 1, p. 4.

Liu, Y., Wang, Y., Li, N., Cheng, X., Zhang, Y., Huang, Y. & Lu, G. (2018), 'An attention-based approach for single image super resolution', *arXiv preprint arXiv:1807.06779* .

Martin, D., Fowlkes, C., Tal, D. & Malik, J. (2001), A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, *in* 'Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on', Vol. 2, IEEE, pp. 416–423.

Matsui, Y., Ito, K., Aramaki, Y., Fujimoto, A., Ogawa, T., Yamasaki, T. & Aizawa, K. (2017), 'Sketch-based manga retrieval using manga109 dataset', *Multimedia Tools and Applications* **76**(20), 21811–21838.

Nair, V. & Hinton, G. E. (2010), Rectified linear units improve restricted boltzmann machines, *in* 'Proceedings of the 27th international conference on machine learning (ICML-10)', pp. 807–814.

Peleg, T. & Elad, M. (2014), 'A statistical prediction model based on sparse representations for single image super-resolution', *IEEE transactions on image processing* **23**(6), 2569–2582.

Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L. & Weinberger, K. Q. (2017), 'Memory-efficient implementation of densenets', *arXiv preprint arXiv:1707.06990* .

Ren, Z., He, C. & Zhang, Q. (2013), 'Fractional order total variation regularization for image super-resolution', *Signal Processing* **93**(9), 2408–2421.

Schulter, S., Leistner, C. & Bischof, H. (2015), Fast and accurate image upscaling with super-resolution forests, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 3791–3799.

Shalev-Shwartz, S. & Tewari, A. (2011), 'Stochastic methods for l1-regularized loss minimization', *Journal of Machine Learning Research* **12**(Jun), 1865–1892.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D. & Wang, Z. (2016), Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 1874–1883.

Sun, J., Xu, Z. & Shum, H.-Y. (2008), Image super-resolution using gradient profile prior, *in* '2008 IEEE Conference on Computer Vision and Pattern Recognition', IEEE, pp. 1–8.

Tai, Y.-W., Liu, S., Brown, M. S. & Lin, S. (2010), Super resolution using edge prior and single image detail synthesis, *in* '2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition', IEEE, pp. 2400–2407.

Tai, Y., Yang, J. & Liu, X. (n.d.), 'Image super-resolution via deep recursive residual network'.

Tai, Y., Yang, J., Liu, X. & Xu, C. (2017), Memnet: A persistent memory network for image restoration, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 4539–4547.

Timofte, R., De Smet, V. & Van Gool, L. (2014), A+: Adjusted anchored neighborhood regression for fast super-resolution, *in* 'Asian Conference on Computer Vision', Springer, pp. 111–126.

Tipping, M. E. & Bishop, C. M. (2003), Bayesian image super-resolution, *in* 'Advances in neural information processing systems', pp. 1303–1310.

Tong, T., Li, G., Liu, X. & Gao, Q. (2017), Image super-resolution using dense skip connections, *in* 'Computer Vision (ICCV), 2017 IEEE International Conference on', IEEE, pp. 4809–4817.

Wang, Y., Shen, J. & Zhang, J. (2018), 'Deep bi-dense networks for image super-resolution', *arXiv preprint arXiv:1810.04873* .

Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. (2004), 'Image quality assessment: from error visibility to structural similarity', *IEEE transactions on image processing* **13**(4), 600–612.

William, J. H., Venkateswaran, N., Narayanan, S. & Ramachandran, S. (2016), 'An example-based super-resolution algorithm for selfie images', *The Scientific World Journal* **2016**.

Xian, Y. & Tian, Y. (2016), 'Single image super-resolution via internal gradient similarity', *Journal of Visual Communication and Image Representation* **35**, 91–102.

Yang, J., Wright, J., Huang, T. S. & Ma, Y. (2010), 'Image super-resolution via sparse representation', *IEEE transactions on image processing* **19**(11), 2861–2873.

Yang, S., Kim, Y. & Jeong, J. (2008), 'Fine edge-preserving technique for display devices', *IEEE Transactions on Consumer Electronics* **54**(4), 1761–1769.

Zeyde, R., Elad, M. & Protter, M. (2010), 'On single image scale-up using sparse-representations', *International conference on curves and surfaces* pp. 711–730.

Zhang, K., Gao, X., Li, J. & Xia, H. (2016), 'Single image super-resolution using regularization of non-local steering kernel regression', *Signal Processing* **123**, 53–63.

Zhang, K., Zuo, W. & Zhang, L. (2018), Learning a single convolutional super-resolution network for multiple degradations, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', Vol. 6.

Zhang, L. & Wu, X. (2006), 'An edge-guided image interpolation algorithm via directional filtering and data fusion', *IEEE transactions on Image Processing* **15**(8), 2226–2238.

Zhang, Y., Li, K., Li, K., Wang, L., Zhong, B. & Fu, Y. (2018), 'Image super-resolution using very deep residual channel attention networks', *arXiv preprint arXiv:1807.02758* .

Zhang, Y., Tian, Y., Kong, Y., Zhong, B. & Fu, Y. (2018), Residual dense network for image super-resolution, *in* 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)'.

Zhu, Z., Guo, F., Yu, H. & Chen, C. (2014), 'Fast single image super-resolution via self-example learning and sparse representation', *IEEE Transactions on Multimedia* **16**(8), 2178–2190.