Pixel-Aware Deep Function-Mixture Network for Spectral Super-Resolution

Lei Zhang,[1]* Zhiqiang Lang,[2]* Peng Wang,[3] Wei Wei,[2] Shengcai Liao,[1] Ling Shao,[1] Yanning Zhang[2]
[1]Inception Institute of Artificial Intelligence, United Arab Emirates
[2]School of Computer Science, Northwestern Polytechnical University, China
[3]School of Computing and Information Technology, University of Wollongong, Australia

Abstract

Spectral super-resolution (SSR) aims at generating a hyperspectral image (HSI) from a given RGB image. Recently, a promising direction is to learn a complicated mapping function from the RGB image to the HSI counterpart using a deep convolutional neural network. This essentially involves mapping the RGB context within a size-specific receptive field centered at each pixel to its spectrum in the HSI. The focus thereon is to appropriately determine the receptive field size and establish the mapping function from RGB context to the corresponding spectrum. Due to their differences in category or spatial position, pixels in HSIs often require different-sized receptive fields and distinct mapping functions. However, few efforts have been invested to explicitly exploit this prior.

To address this problem, we propose a pixel-aware deep function-mixture network for SSR, which is composed of a new class of modules, termed function-mixture (FM) blocks. Each FM block is equipped with some **basis functions**, i.e., parallel subnets of different-sized receptive fields. Besides, it incorporates an extra subnet as a **mixing function** to generate pixel-wise weights, and then linearly mixes the outputs of all basis functions with those generated weights. This enables us to pixel-wisely determine the receptive field size and the mapping function. Moreover, we stack several such FM blocks to further increase the flexibility of the network in learning the pixel-wise mapping. To encourage feature reuse, intermediate features generated by the FM blocks are fused in late stage, which proves to be effective for boosting the SSR performance. Experimental results on three benchmark HSI datasets demonstrate the superiority of the proposed method.

Introduction

Hyperspectral images (HSIs) that capture the reflectance of scenes with extremely high spectral resolution (Chakrabarti and Zickler 2011), often contain hundreds or thousands of spectral bands, where each pixel has a spectrum (Zhang et al. 2018b). Profiting from the abundant spectral information, HSIs have been widely applied to various tasks, e.g., classification (Akhtar and Mian 2018), detection (Manolakis and Shaw 2002) and tracking (Van Nguyen, Banerjee, and Chellappa 2010) etc. However, the expense of obtaining such spectral information is to increase the pixel size on the sensor, which inevitably limits the spatial resolution of HSIs (Arad and Ben-Shahar 2016). Thus, it is crucial to investigate how to generate high-spatial-resolution (HSR) HSIs.

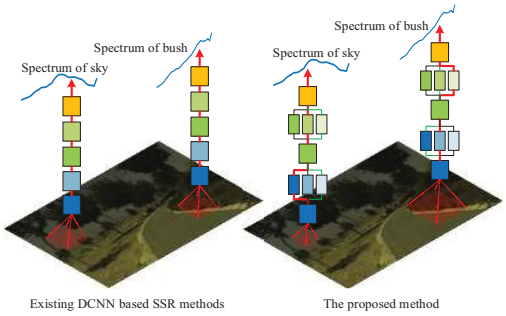


Figure 1: Existing DCNN based SSR methods and the proposed method. The former often take a fixed-sized receptive field and learn a universal mapping function for all pixels, while the latter can adaptively determine the receptive field size and the mapping function for each pixel.

Different from convnetioanl HSIs super-resolution (Mei et al. 2017; Zhang et al. 2018a) that directly improves the spatial resolution of a given HSI, spectral super-resolution (SSR) (Arad and Ben-Shahar 2016; Xiong et al. 2017) adopts an alternative way and attempts to produce an HSR HSI by increasing the spectral resolution of a given RGB image with satisfactory spatial resolution. Early SSR methods (Arad and Ben-Shahar 2016; Aeschbacher, Wu, and Timofte 2017; Jia et al. 2017) often formulate SSR as a linear inverse problem, and exploit the inherent low-level statistic of HSR HSIs as priors. However, due to the limited expressive capacity of their handcrafted prior models, these methods fail to well generalize to challenging cases. Recently, witnessing the great success of deep convolutional neural networks (DCNNs) in a wide range of tasks (Simonyan and Zisserman 2014; He et al. 2016; 2017), increasing efforts have been invested to learn a DCNN based

*The first two authors contributed equally to this work. The corresponding author is Wei Wei (email: weiweinwpu@nwpu.edu.cn)

mapping function to directly transform the RGB image into an HSI (Alvarez-Gila, Van De Weijer, and Garrote 2017; Arad and Ben-Shahar 2017; Shi et al. 2018; Fu et al. 2018). These methods essentially involve mapping the RGB context within a size-specific receptive field centered at each pixel to its spectrum in the HSI, as shown in Figure 1. The focus thereon is to appropriately determine the receptive field size and establish the mapping function from RGB context to the corresponding spectrum. Due to the difference in category or spatial position, pixels in HSIs often necessitate collecting different RGB information and adopting various recovery schemes for SSR. Therefore, to obtain an accurate DCNN based SSR approach, it is crucial to adaptively determine the receptive field size and the RGB-to-spectrum mapping function for each pixel. However, most existing DCNN based SSR methods treat all pixels in HSIs equally and learn a universal mapping function with a fixed-sized receptive field, as shown in Figure 1.

In this study, we present a pixel-aware deep function-mixture network for SSR, which is flexible to pixel-wisely determine the receptive field size and the mapping function. Specifically, we first develop a new module, termed the function-mixture (FM) block. Each FM block consists of some parallel DCNN based subnets, among which one is termed the *mixing function* and the remaining are termed *basis functions*. The basis functions take different-sized receptive fields and learn distinct mapping schemes; while the mixture function generates pixel-wise weights to linearly mix the outputs of the basis functions. In this way, the pixel-wise weights can determine a specific information flow for each pixel and consequently benefit the network to choose appropriate RGB context as well as the mapping function for spectrum recovery. Then, we stack several such FM blocks to further improve the flexibility of the network in learning the pixel-wise mapping. Furthermore, to encourage feature reuse, the intermediate features generated by the FM blocks are fused in late stage, which proves to be effective for boosting the SSR performance. Experimental evaluation on three benchmark HSI datasets shows the superiority of the proposed approach for SSR.

In summary, we mainly contribute in three aspects. ***i)*** We present an effective pixel-aware deep function-mixture network for SSR, which is flexible to learn the pixel-wise RGB-to-spectrum mapping. To our best knowledge, this is the first attempt to explore this in SSR. ***ii)*** We design a new FM module, which is flexible to plug in any modern DCNN architectures; ***iii)*** We demonstrate new state-of-the-art performance on three benchmark SSR datasets.

Related Work

We first review the existing approaches for SSR and then introduce some techniques related to this work.

Spectral Super-resolution Early methods mainly focus on exploiting appropriate image priors to regularize the linear inverse SSR problem. For example, Arad and Aeschbacher et. al (Arad and Ben-Shahar 2016; Aeschbacher, Wu, and Timofte 2017) investigated the sparsity of the latent HSI on a pre-trained over-complete spectral dictionary. Jia et. al (Jia et al. 2017) considered the manifold structure of HSIs in a low-dimensional space. Recently, most methods turn to learning a deep mapping function from the RGB image to an HSI. For example, Alvarez-Gila et al. (Alvarez-Gila, Van De Weijer, and Garrote 2017) implemented the mapping function using an U-Net architecture (Ronneberger, Fischer, and Brox 2015) and trained it based on both the mean-square-error (MSE) loss and the adversarial loss (Goodfellow et al. 2014). Shi et. al (Shi et al. 2018) developed a deep residual network consisting of residual blocks to learn the mapping function. Despite obtaining impressive performance for SSR, these methods are limited by learning a universal RGB-to-spectrum mapping function for all pixels in HSIs. This leaves space for learning more flexible and adaptive mapping function.

Receptive Field in DCNNs Receptive field is an important concept in the DCNN, which determines the sensing space of a convolutional neuron. There are many efforts dedicating to adjusting the size or shape of the receptive field (Yu and Koltun 2015; Wei et al. 2017; Dai et al. 2017) to meet the requirement of specific tasks at hand. Thereinto, dilated convolution (Yu and Koltun 2015) or kernel separation (Seif and Androutsos 2018) were often utilized to enlarge the receptive field. Recently, Wei et. al (Wei et al. 2017) changed the receptive field by inflating or shrinking the feature maps using two affine transformation layers. Dai et. al (Dai et al. 2017) proposed to adaptively determine the context within the receptive field by estimating the offsets of pixels to the central pixel using an additional convolution layer. In contrast, we take a totally different direction and learn the pixel-wise receptive field size by mixing some basis function with different receptive field sizes.

Multi-column Network Multi-column network (Cireşan, Meier, and Schmidhuber 2012) is a specicial type of network that feeds the input into several parallel DCNNs (i.e., columns), and then aggregates their outputs for final prediction. With the ability of using more context information, the multi-column network (MCNet) often shows better generalization capacity than that with only a single column in various tasks, e.g., classification (Cireşan, Meier, and Schmidhuber 2012), image processing (Agostinelli, Anderson, and Lee 2013), counting (Zhang et al. 2016) etc. Although we also adopt a similar multi-column structure in our module design, the proposed network is obviously different from these existing multi-column networks. First, MCNet employs a separation-and-aggregation architecture which processes the input with parallel columns and then aggregates the outputs of all columns for output. In contrast, we adopt a recursive separation-and-aggregation architecture by stacking multiple FM modules, each of which can be viewed as an enhanced multi-column module, as shown in Figure 1, 3. Second, when applied to SSR, MCNet still learns a universal mapping function and fails to flexibly handle each pixel in an explicit way. In contrast, the proposed FM block incorporates a mixing function to generate pixel-wise weights and mix the outputs of all basis functions. This enables to flexibly customize the pixel-wise mapping function. In addition, we fuse the intermediate feature generated by FM blocks in

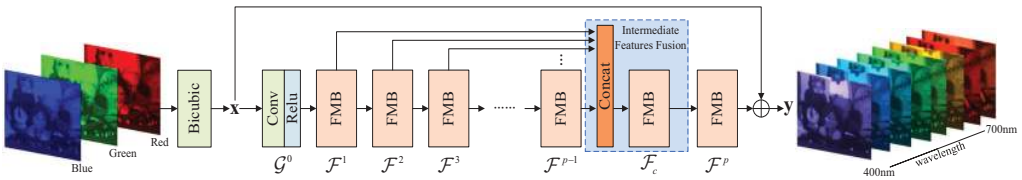


Figure 2: Architecture of the proposed pixel-aware deep function-mixture network. Bicubic denotes the bicubic interpolation in the spectral domain. FMB denotes the function-mixture block.

the network for feature reuse.

Proposed Network

In this section, we present the technical details of the proposed pixel-aware deep function-mixture network. As shown in Figure 2, the proposed network adopts a global residual architecture as (Kim, Kwon Lee, and Mu Lee 2016a). Its backbone consists of multiple FM blocks and fuses the intermediate features generated by previous FM block with skip connections. In the following, we will first introduce the basic FM block, and then discuss how to incorporate multiple FM blocks and the intermediate features fusion into the network for performance enhancement.

Function-mixture Block

The proposed network essentially establishes a mapping function from an RGB image to the HSI counterpart, and thus each FM block implies a mapping subfunction. In this study, we attempt to utilize the FM block to adaptively determine the receptive field size and the mapping function for each pixel, i.e., to obtain a pixel-dependent mapping subfunction. To this end, a direct solution is to introduce an additional hypernetwork (Ha, Dai, and Le 2016; Jia et al. 2016) to generate the subfunction parameters for each pixel. However, this will incur expensive computational complexity and great training difficulty (Ha, Dai, and Le 2016). To avoid this problem, we borrow the idea in function approximation (Cybenko 1989) and assume that all pixel-dependent subfunctions can be accurately approximated by mixing some *basis functions* with pixel-wise weights. Due to being shared by all subfunctions, these basis functions can be learned by DCNNs. While the pixel-wise mixing weights can be viewed as the pixel-wise attention (Sato and Lauro 2014), which also can be directly generated by a DCNN.

Following this idea, we construct the FM block with a separation-and-aggregation structure, as shown in Figure 3. First, a convolutional block, i.e. a convolutional layer followed by a Rectified Linear Unit (ReLu) (Nair and Hinton 2010), is utilized for initial feature representation. Then, the obtained features are fed into multiple parallel subnets. Thereinto, one subnet is utilized to generate the pixel-wise mixing weights. For simplicity, we term it the *mixing function*. While the remaining subnets represent the basis functions. Finally, the outputs of all basis functions are linearly mixed based on the generated pixel-wise weights. Let $\mathbf{x}^{u-1}$ denote the input for the u-th FM block $\mathcal{F}^u$ and n denote the number of basis functions in $\mathcal{F}^u$. The output $\mathbf{x}^u$ of $\mathcal{F}^u$ can

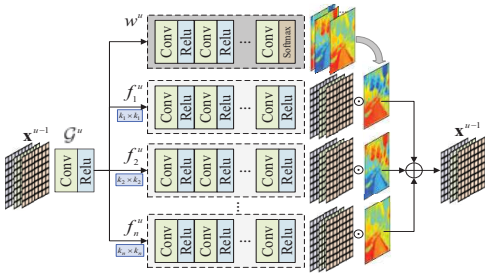


Figure 3: Architecture of the proposed function-mixture block where k_i $(i = 1, \cdots, n)$ denotes the convoluational filter size in the i-th basis function f_i^u.

be formulated as

$$\begin{aligned}\mathbf{x}^u &= \mathcal{F}^u(\mathbf{x}^{u-1}) = \sum\nolimits_{i=1}^{n} f_i^u(\bar{\mathbf{x}}^u, \theta_i^u) \odot w^u(\bar{\mathbf{x}}^u, \vartheta^u)[i] \\ \text{s.t.,}\ \bar{\mathbf{x}}^u &= \mathcal{G}^u(\mathbf{x}^{u-1}, \omega^u), \\ &\sum\nolimits_{i=1}^{n} w^u(\bar{\mathbf{x}}^u, \vartheta^u)[i] = \mathbf{1}, w^u(\bar{\mathbf{x}}^u, \vartheta^u) \succeq 0,\end{aligned} \tag{1}$$

where $f_i^u(\cdot, \theta_i^u)$ denotes the i-th basis function parameterized by θ_i^u and $w^u(\cdot, \vartheta^u)$ represents the mixing function parameterized by ϑ^u. When $f_i^u(\bar{\mathbf{x}}^u, \theta_i^u)$ is of size $c \times h \times w$ (i.e., channel $\times$ height $\times$ width), $w^u(\bar{\mathbf{x}}^u, \vartheta^u)$ is of size $n \times h \times w$, and $w^u(\bar{\mathbf{x}}^u, \vartheta^u)[i]$ represents the mixing weights of size $h \times w$ generated for all pixels corresponding to the i-th basis function. $\odot$ denotes the point product. $\bar{\mathbf{x}}^u$ denotes the features output by the convolutional block $\mathcal{G}^u(\cdot, \omega^u)$ in $\mathcal{F}^u$, and ω^u represents the convolutional filters. Inspired by (Everitt 2005), we also require the mixing weights to be non-negative and the summation across all basis functions is equal to 1, as shown in Eq. (1).

In this study, we implement the basis functions and the mixing function by stacking m consecutive convolutional blocks, as shown in Figure 3. Moreover, we equip these basis functions with different-sized convolutional filters to ensure they take different-sized receptive fields and learn distinct mapping schemes. For the mixing function, we introduce a Softmax unit at the end to comply with the constraints in Eq. (1). Apparently, with such a pixel-wise mixture architecture, the proposed FM block is able to determine the receptive field size and the mapping function for each pixel.

Multiple FM Blocks

As shown in Figure 2, in the proposed network, we first introduce an individual convolutional block, and then stack

multiple FM blocks for the intermediate feature representation and the ultimate output. For an input RGB image $\mathbf{x}$, the output of the network with p FM blocks can be given as

$$\begin{aligned} &\mathbf{y} = \mathbf{x} + \mathcal{F}^p\left(\mathcal{F}^{p-1}\left(\cdots \mathcal{F}^2\left(\mathcal{F}^1\left(\mathbf{x}^0\right)\right)\right)\right), \\ &\text{s.t.,} \quad \mathbf{x}^0 = \mathcal{G}^0\left(\mathbf{x}, \omega^0\right), \end{aligned} \tag{2}$$

where $\mathbf{y}$ denotes the generated HSI and $\mathbf{x}^0$ represents the output of the first convolutional block $\mathcal{G}^0(\cdot, \omega^0)$ parameterized by ω^0. It is worth noting that in this study we increase the spectral resolution of $\mathbf{x}$ to the same as that of $\mathbf{y}$ by the bilinear interpolation. In addition, $\mathcal{F}^1, \cdots, \mathcal{F}^{p-1}$ show the same architecture, while the output of $\mathcal{F}^p$ will be adjusted according to the number of spectral bands in $\mathbf{y}$.

By stacking multiple FM blocks, we can pixel-wisely adjust the receptive field size and the mapping function at multiple levels, thus increase the flexibility of the proposed network. In addition, considering that each FM block defines the mapping subfunction for each pixel, the ultimate mapping function obtained by stacking p FM blocks can be viewed as a composition function of p subfunctions. Since each subfunction is approximated by the mixture of n basis functions, the ultimate mapping function can be viewed as the mixture of n^p basis functions, which show much larger expressive capacity than a single FM block in pixel-wisely fitting a mapping function.

Intermediate Features Fusion

As previously mentioned, the FM blocks in the porposed network extract different levels of features from the input. Inspired by (Kim, Kwon Lee, and Mu Lee 2016b; Zhang et al. 2018c), to reuse these intermediate features for performance enhancement, we employ skip connections to aggregate the intermediate features generated by each FM block before the ultimate output block with a concatenation operation, as shown in Figure 2. To better utilize all of these features for pixel-wise representation, we introduce an extra FM block $\mathcal{F}_c$ to fuse the concatenation result. With such an intermediate feature fusion operation, the output of the proposed network can be reformulated as

$$\mathbf{y} = \mathbf{x} + \mathcal{F}^p\left(\mathcal{F}_c\left(\left[\mathcal{F}^{p-1}\left(\cdots \mathcal{F}^1\left(\mathbf{x}^0\right)\right), \cdots, \mathcal{F}^1\left(\mathbf{x}^0\right)\right]\right)\right) \tag{3}$$

Experiment

In this section, we will conduct extensive comparison experiments and carry out an ablation study to demonstrate the effectiveness of the proposed method in SSR.

Experimental Setting

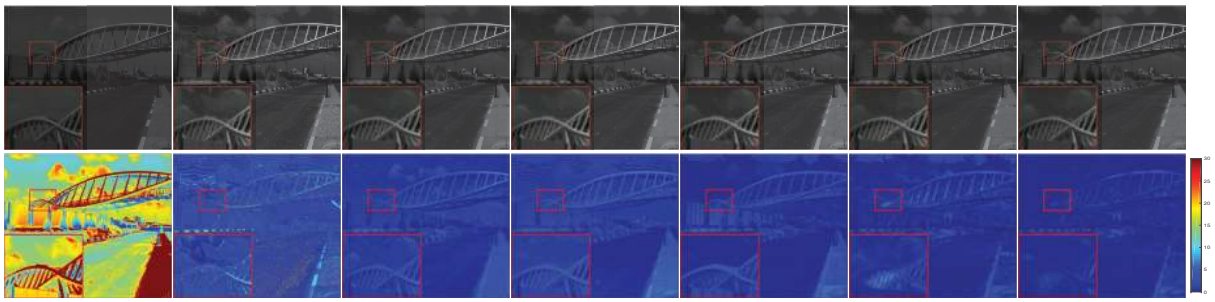
Datasets In this study, we adopt three benchmark HSI datasets, including NTIRE (Timofte et al. 2018), CAVE (Yasuma et al. 2010) and Harvard (Chakrabarti and Zickler 2011). In NTIRE dataset, there are 255 paired HSIs and RGB images which have the same spatial resolution, e.g., 1392×1300. Each HSI consists of 31 successive spectral bands ranging from $400nm$ to $700nm$ with a $10nm$ interval. CAVE dataset contains 32 HSIs. Similar to NTIRE, each HSI contains 31 spectral bands ranging from $400nm$ to $700nm$ with a $10nm$ interval but with the spatial resolution 512×512. Harvard dataset is another common benchmark for HSIs. It consists of 50 HSIs with spatial resolution 1392×1040. Each image contains 31 spectral bands captured from $420nm$ to $720nm$ with a $10nm$ interval. For the CAVE and Havard datasets, inspird by (Dong et al. 2016; Zhang et al. 2018a), we adopt the spectral response function of Nikon D700 camera to generate the corresponding RGB image for each HSI. In the following experiments, we randomly select 200 paired images from the NTIRE dataset as the training set and the remaining 55 paired images for testing. For the CAVE dataset, we randomly choose 22 paired images for training and the remaining 10 paired images for testing. While in the Harvard dataset, 30 paired images are randomly chosen as the training set and the remaining 20 paired images are utilized for testing.

Comparison Methods In this study, we compare the proposed method with 6 existing methods including the bilinear interpolation (BI) (Hou and Andrews 1978), Arad (Arad and Ben-Shahar 2016), Aitor (Alvarez-Gila, Van De Weijer, and Garrote 2017), HSCNN+ (Shi et al. 2018), deep convolution neural network (DCNN) and the multi-column network (MCNet). Among them, the BI utilizes the bilinear interpolation to increase the spectral resolution of the input RGB image. The Arad is a sparsity induced conventional SSR method. The Aitor and HSCNN+ are two recent DCNN based state-of-the-art SSR methods, especially HSCNN+ which is the winner in NTIRE 2018 spectral reconstruction task. The DCNN and MCNet are two baselines for the proposed method. The DCNN is a variant of the proposed method that is implemented by replacing each FM block in the proposed method with a convolutional block. For the MCNet, we implement it following the basic architecture in (Zhang et al. 2016) with convolutional blocks. Moreover, the column number is set as n and the convolutions in n columns are equipped with n kinds of different-sized filters, which is similar as the proposed method. We further control the depth of each column to make sure the model complexity of the MCNet is comparable to the proposed method. By doing this, the only difference between the MCNet and the proposed network is the network architecture. For fair comparison, all these DCNN based competitors and the spectral dictionary in the Arad (Arad and Ben-Shahar 2016) are retrained on the training set utilized in the experiments.

Evaluation Metrics To objectively evaluate the SSR performance of each method, we employ four standard metrics, including the root-mean-square error (RMSE), peak signal-to-noise ratio (PSNR), spectral angle sapper (SAM) and structural similarity index (SSIM). The RMSE and PSNR measure the numerical difference between two images. The SAM computes the average spectral angle between two spectra from two image at the same spatial position to indicate the spectral reconstruction accuracy. The SSIM is utilized to measure the spatial structure similarity between two images. In general, a larger PSNR or SSIM and a smaller RMSE or SAM indicate better performance.

Table 1: Numerical results of different methods on three benchmark SSR datasets. The best results are in bold.

Methods	NTIRE				CAVE				Harvard			
	RMSE	PSNR	SAM	SSIM	RMSE	PSNR	SAM	SSIM	RMSE	PSNR	SAM	SSIM
BI	15.41	25.73	15.30	0.8397	26.60	21.49	34.38	0.7382	30.86	19.44	39.04	0.5887
Arad	4.46	35.63	5.90	0.9082	10.09	28.96	19.54	0.8695	7.85	31.30	8.32	0.8490
Aitor	1.97	43.30	1.80	0.9907	6.80	32.53	17.50	0.8768	3.29	39.21	4.93	0.9671
HSCNN+	1.55	45.38	1.63	0.9931	4.97	35.66	8.73	0.9529	2.87	41.05	4.28	0.9741
DCNN	1.23	47.40	1.30	0.9939	5.77	34.09	11.35	0.9275	2.88	40.83	4.24	0.9724
MCNet	1.11	48.43	1.13	0.9951	4.84	35.92	8.98	0.9555	2.83	40.70	4.26	0.9689
Ours	**1.03**	**49.29**	**1.05**	**0.9955**	**4.54**	**36.33**	**7.07**	**0.9611**	**2.54**	**41.54**	**3.76**	**0.9796**



(a) BI (b) Arad (c) Aitor (d) HSCNN+ (e) DCNN (f) MCNet (g) Ours

Figure 4: Visual SSR results of the 31-th band and the reconstruction error maps of an example image from the NTIRE dataset for different methods. The reconstruction error is obtained by computing the mean-square error between two spectrum vectors from the super-resolution result and the ground truth at each pixel. Best view on the screen.

Implementation Details In the proposed method, we adopt 4 FM blocks (i.e., including $\mathcal{F}_c$ and p=3). Each block contains $n = 3$ basis functions. The basis functions and the mixing functions consist of m=2 convolutional blocks. Each convolutional block contains 64 filters. In each FM block, basis functions are equipped with 3 different-sized filters for convolution, i.e., 3×3, 7×7 and 11×11. While the filter size in all other convolutional blocks is fixed as 3×3.

In this study, we implement the proposed method on the Pytorch platform (Ketkar 2017) and train the network using the following model

$$\min_{\theta} \frac{1}{N} \sum\nolimits_{i=1}^{N} \|\mathbf{y}_i - f(\mathbf{x}_i, \theta)\|_1, \tag{4}$$

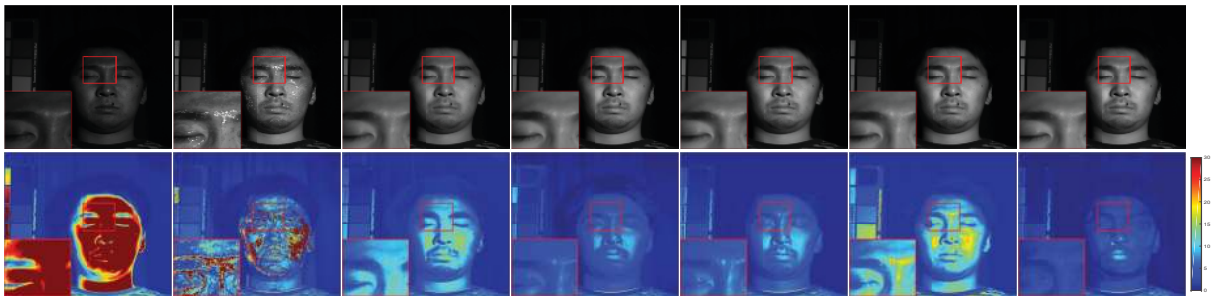
where $\{(\mathbf{y}_i, \mathbf{x}_i)\}$ denotes the i-th paired HSI and RGB image, respectively. N denotes the number of training pairs. f denotes the ultimate mapping function defined by the proposed network and θ represents all involved parameters. $\|\cdot\|_1$ represents the ℓ_1 norm based loss. In the training stage, we employ the Adam optimizer (Kingma and Ba 2014) with the weight decay 1e-6. The learning rate is initially set as 1e-4 and halved in every 20 epochs. The batch size is 128. We terminate the optimization at the 100-th epoch.

Performance Evaluation

Performance comparison Under the same experimental settings, we evaluate all those methods on the testing set from each benchmark dataset. Their numerical results are reported in Table 1. It can be seen that these DCNN based comparison methods often produce more accurate results than the interpolation or the sparsity induced SSR method. For example, on the NTIRE dataset, the RMSE of the Aitor and HSCNN+ are less than 2.0 while that of the BI and Arad are higher than 4.0. Nevertheless, the proposed method obviously outperforms these DCNN based competitors. For example, compared with the state-of-the-art HSCNN+, the proposed method reduces the RMSE by 0.52 and improves the PSNR by 3.19db on the NTIRE dataset. This profits from the ability of the proposed method in adaptively determining the receptive field size and the mapping function for each pixel. With such an ability, the proposed method is able to handle each pixel more flexibly. Moreover, since various mapping functions can be approximated by the mixture of the learned basis functions, the proposed method can better generalize to the unknown pixels.

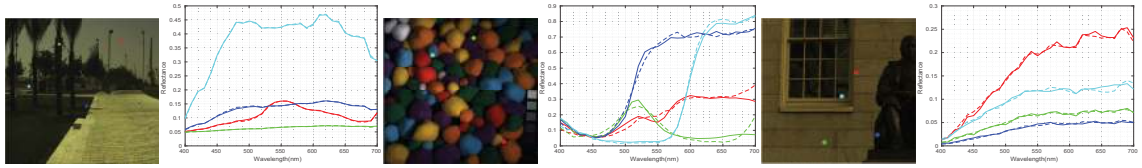
In addition, as shown in Table 1, the proposed method also performs better than two baselines, i.e., DCNN and MCNet. For example, on the NTIRE dataset, the PSNR obtained by the proposed method is higher than that of DCNN and MCNet by 1.89db and 0.86db. Since the only difference between the proposed method and DCNN is the discrepancy between the convolutional block and the proposed FM block, the superiority demonstrates that the proposed FM block is much powerful than the convolutional block for SSR. Similarly, the advantage of the proposed method over MCNet clarifies that the proposed network architecture is more effective than the multi-column architecture in SSR.

To further clarify the conclusions above, we visualize some SSR results of different methods in Figure 4 and Figure 5. As can be seen, the proposed method recovers more



(a) BI (b) Arad (c) Aitor (d) HSCNN+ (e) DCNN (f) MCNet (g) Ours

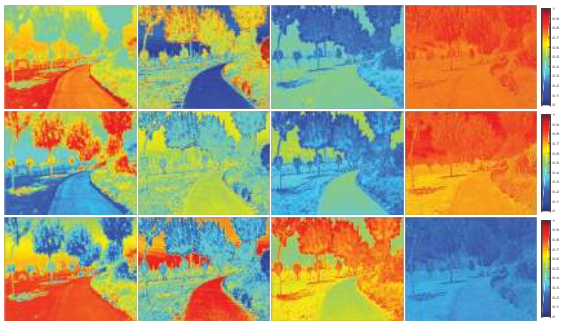
Figure 5: Visual SSR results of the 28-th band and the reconstruction error maps of an example image from the CAVE dataset for different methods. The reconstruction error is computed as Figure 4.



(a) NTIRE (b) Spectra (c) CAVE (d) Spectra (e) Havard (f) Spectra

Figure 6: Recovered spectra form the super-resolution results of the proposed method on three example images chosen from three datasets. In each image, we select four different positions and plot the curves of the recovered spectra (i.e., denoted by dash lines) and the corresponding ground truth spectra (i.e., denoted by solid lines).

details and produces smaller reconstruction error than other competitors. In addition, we sketch some spectrum curves recovered by the proposed method in Figure 6, where the produced spectra are very close to the ground truth.



(a) $\mathcal{F}^1$ (b) $\mathcal{F}^2$ (c) $\mathcal{F}_c$ (d) $\mathcal{F}^3$

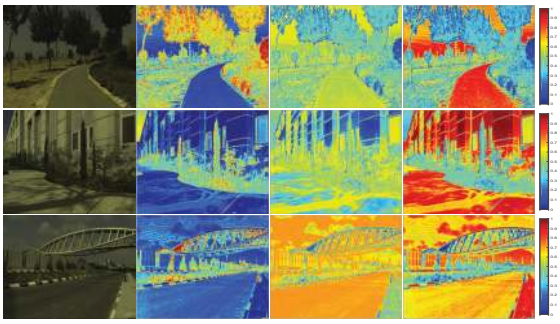
Figure 7: Pixel-wise weights generated by the mixing function in different FM blocks of the proposed network. Figures in each column show the weight maps for three basis functions (from top to bottom: f_1^u, f_2^u and f_3^u). For visualization convenience, we normalize each weight map into the range [0,1] using the inner maximum and minimum values.

Pixel-wise mixing weights In this study, we mix the outputs of the basis functions with pixel-wise weights to adaptively learn the pixel-wise mapping. To validate that the proposed method can effectively produce the pixel-wise weights as expected, we choose an example image from the NTIRE and visualize the produced pixel-wise weights in each FM block, as shown in Figure 7. We can find that, i) pixels from different categories or spatial positions are often given different weights. For example, in the second weight map generated by $\mathcal{F}^1$, the weights for the pixels from 'road' are obviously smaller than that for the pixels from 'tree'. ii) Pixels from the same category are pone to be given similar weights. For example, pixels from 'road' are given similar weights in each weight map in Figure 7 (a)(b). To further clarify these two observations, we visualize the weight maps of some other images generated by the FM block $\mathcal{F}^2$ in Figure 8, where similar phenomenon can be observed. iii) In the intermediate FM blocks (i.e., $\mathcal{F}^1$ and $\mathcal{F}^2$ in Figure 7), the high level block (e.g., $\mathcal{F}^2$) can distinguish finer difference between pixels than the low level block (e.g., $\mathcal{F}^1$), viz., only highly similar pixels will be assigned to similar weights. iv) Due to being forced to match the output, in the weight maps generated by the ultimate output block $\mathcal{F}^3$, the weight difference between pixels from various categories is not as obvious as that in previous FM block (e.g., $\mathcal{F}^1$ and $\mathcal{F}^1$), as shown in Figure 7(a)(b)(d).

According to the above observations, we can conclude that the proposed network can effectively generate the pixel-wise mixing weights and thus is able to pixel-wisely determine receptive field size and mapping function.

Ablation study

In this part, we carry out an ablation study on the NTIRE dataset to demonstrate the effect of the different ingredients, the number of basis functions and the number of FM blocks on the proposed network.



(a) Image (b) f_1^2 (c) f_2^2 (d) f_3^2

Figure 8: Pixel-wise weights generated by the mixing function in the FM block $\mathcal{F}^2$ on different images. In each row, figures from left to right denote the input RGB image and three generated weight maps corresponding to the basis functions f_1^2, f_2^2 and f_3^2. For visualization convenience, we normalize each weight map into the range [0,1] using the inner maximum and minimum values.

Table 2: Effect of pixel-wise mixture & intermediate feature fusion in the proposed network.

Methods	RMSE	PSNR	SAM	SSIM
Ours w/o mix	1.10	48.44	1.16	0.9950
Ours w/o fusion	1.05	48.97	1.09	0.9953
Ours	**1.03**	**49.29**	**1.05**	**0.9955**

Effect of Different Ingredients In the proposed network, there are two important ingredients, namely the pixel-wise mixture and the intermediate feature fusion. To demonstrate their effectiveness, we compare the proposed method with its two variants. One (i.e., 'Ours w/o mix') disables the pixel-wise mixture in the proposed network, which implies mixing the outputs of the basis functions with equal weights; while the other (i.e., 'Ours w/o fusion') disables the intermediate feature fusion, i.e., removing the skip connections as well as the FM block $\mathcal{F}_c$. The numerical results are reported in Table 2. It can be seen that the proposed method obviously outperforms these two variants. This demonstrate that both the pixel-wise mixture and the intermediate feature fusion are crucial for the proposed network.

Table 3: Effect of the number n of basis functions and the number p of FM blocks.

Methods	RMSE	PSNR	SAM	SSIM
Ours (n =1)	1.47	45.82	1.57	0.9913
Ours (n =2)	1.08	48.76	1.10	0.9952
Ours (n =3)	1.03	49.29	1.05	0.9955
Ours (n =5)	**0.98**	**49.87**	**1.00**	**0.9958**
Ours (p =2)	1.05	48.95	1.09	0.9954
Ours (p =3)	1.03	49.29	1.05	0.9955
Ours (p =4)	1.05	49.42	1.05	0.9954
Ours (p =6)	**1.00**	**49.59**	**1.02**	**0.9956**

Effect of the Number of Basis Functions In the above experiments, we fix the number of basis functions as $n = 3$ in each FM block. Intuitively, increasing n will enlarge the expressive capacity of the basis fictions and thus lead to better performance, vice versa. To validate this, we evaluate the proposed method on the NTIRE dataset using different n, i.e., n =1, 2, 3 and 5. The obtained numerical results are provided in Table 3. As can be seen, the reconstruction accuracy gradually increases as the number n of basis functions increases. When n =1, the proposed method degenerates to the convolutional blocks based network, which shows the lowest reconstruction accuracy in Table 3. When n increases to 5, the obtained RMSE is even lower than 1.0 and the PSNR is close to 50db. Since a larger n often incur higher computational complexity, we can make a balance between the accuracy and efficiency by tuning n to customize the proposed network for a specific device.

Effect of the Number of FM Blocks In addition to the number of basis functions, the model complexity of the proposed method also depends on the number p of the FM blocks. To demonstrate the effect of p, we evaluate the proposed method on the NTIRE dataset using different number of FM blocks, i.e., p=2,3,4 and 6. The obtained numerical results are reported in the second part of Table 3. Similar as the case of n, the performance of the proposed method can be gradually improved as the number p of FM blocks increases. We also find an interesting thing, increasing n may be more effective than increasing p in terms of boosting the performance of the proposed method.

Conclusion

In this study, to flexibly handle the pixels from different categories or spatial positions in HSIs, we present a pixel-aware deep function-mixture network for SSR, which is composed of multiple FM blocks. Each FM block consists of one mixing function and some basis functions, which are implemented as parallel DCNN based subnets. Thereinto, the basis functions take different sized receptive fields and learn distinct mapping schemes; while the mixing function generates the pixel-wise weights to linearly mix the outputs of all these basis functions. This enables to pixel-wisely determine the receptive field size and mapping function. Moreover, we stack several such FM block in the network to further increase its flexibility in learning the pixel-wise mapping. To boost the SSR performance, we also fuse the intermediate features generated by the FM blocks for feature reuse. With extensive experiments on three benchmark SSR datasets, the proposed method shows superior performance over several existing state-of-the-art competitors. It is worth noting that in the future the idea in this study also can be generalized to other tasks requiring pixel-wise modelling, e.g., semantic segmentation, colorization etc.

Acknowledgement This work was supported in part by the National Natural Science Foundation of China (No. 61671385, 61571354).

References

Aeschbacher, J.; Wu, J.; and Timofte, R. 2017. In defense of shallow learned spectral reconstruction from rgb images. In *ICCV*, 471–479.

Agostinelli, F.; Anderson, M. R.; and Lee, H. 2013. Adaptive multi-column deep neural networks with application to robust image denoising. In *NeurIPS*, 1493–1501.

Akhtar, N., and Mian, A. 2018. Nonparametric coupled bayesian dictionary and classifier learning for hyperspectral classification. *IEEE transactions on neural networks and learning systems* 29(9):4038–4050.

Alvarez-Gila, A.; Van De Weijer, J.; and Garrote, E. 2017. Adversarial networks for spatial context-aware spectral image reconstruction from rgb. In *ICCV*, 480–490.

Arad, B., and Ben-Shahar, O. 2016. Sparse recovery of hyperspectral signal from natural rgb images. In *ECCV*, 19–34. Springer.

Arad, B., and Ben-Shahar, O. 2017. Filter selection for hyperspectral estimation. In *CVPR*, 3153–3161.

Chakrabarti, A., and Zickler, T. 2011. Statistics of real-world hyperspectral images. In *CVPR*, 193–200. IEEE.

Cireşan, D.; Meier, U.; and Schmidhuber, J. 2012. Multi-column deep neural networks for image classification. *arXiv preprint arXiv:1202.2745*.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2(4):303–314.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *CVPR*, 764–773.

Dong, W.; Fu, F.; Shi, G.; Cao, X.; Wu, J.; Li, G.; and Li, X. 2016. Hyperspectral image super-resolution via non-negative structured sparse representation. *IEEE Transactions on Image Processing* 25(5):2337–2352.

Everitt, B. S. 2005. Finite mixture distributions. *Encyclopedia of statistics in behavioral science*.

Fu, Y.; Zhang, T.; Zheng, Y.; Zhang, D.; and Huang, H. 2018. Joint camera spectral sensitivity selection and hyperspectral image recovery. In *ECCV*, 788–804.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NeurIPS*, 2672–2680.

Ha, D.; Dai, A.; and Le, Q. V. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *ICCV*, 2961–2969.

Hou, H., and Andrews, H. 1978. Cubic splines for image interpolation and digital filtering. *IEEE Transactions on acoustics, speech, and signal processing* 26(6):508–517.

Jia, X.; De Brabandere, B.; Tuytelaars, T.; and Gool, L. V. 2016. Dynamic filter networks. In *NeurIPS*, 667–675.

Jia, Y.; Zheng, Y.; Gu, L.; Subpa-Asa, A.; Lam, A.; Sato, Y.; and Sato, I. 2017. From rgb to spectrum for natural scenes via manifold-based mapping. In *ICCV*, 4705–4713.

Ketkar, N. 2017. Introduction to pytorch. In *Deep learning with python*. Springer. 195–208.

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016a. Accurate image super-resolution using very deep convolutional networks. In *CVPR*, 1646–1654.

Kim, J.; Kwon Lee, J.; and Mu Lee, K. 2016b. Deeply-recursive convolutional network for image super-resolution. In *CVPR*, 1637–1645.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Manolakis, D., and Shaw, G. 2002. Detection algorithms for hyperspectral imaging applications. *IEEE signal processing magazine* 19(1):29–43.

Mei, S.; Yuan, X.; Ji, J.; Zhang, Y.; Wan, S.; and Du, Q. 2017. Hyperspectral image spatial super-resolution via 3d full convolutional neural network. *Remote Sensing* 9(11):1139.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Ronneberger, O.; Fischer, P.; and Brox, T. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 234–241. Springer.

Sato, K., and Lauro, R. D. 2014. Deep networks with internal selective attention through feedback connections. In *International Conference on Neural Information Processing Systems*.

Seif, G., and Androutsos, D. 2018. Large receptive field networks for high-scale image super-resolution. In *CVPR Workshops*, 763–772.

Shi, Z.; Chen, C.; Xiong, Z.; Liu, D.; and Wu, F. 2018. Hscnn+: Advanced cnn-based hyperspectral recovery from rgb images. In *CVPR Workshops*, 939–947.

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

Timofte, R.; Gu, S.; Wu, J.; and Van Gool, L. 2018. Ntire 2018 challenge on single image super-resolution: methods and results. In *CVPR Workshops*, 852–863.

Van Nguyen, H.; Banerjee, A.; and Chellappa, R. 2010. Tracking via object reflectance using a hyperspectral video camera. In *CVPR Workshops*, 44–51. IEEE.

Wei, Z.; Sun, Y.; Wang, J.; Lai, H.; and Liu, S. 2017. Learning adaptive receptive fields for deep image parsing network. In *CVPR*, 2434–2442.

Xiong, Z.; Shi, Z.; Li, H.; Wang, L.; Liu, D.; and Wu, F. 2017. Hscnn: Cnn-based hyperspectral image recovery from spectrally undersampled projections. In *ICCV*, 518–525.

Yasuma, F.; Mitsunaga, T.; Iso, D.; and Nayar, S. K. 2010. Generalized assorted pixel camera: postcapture control of resolution, dynamic range, and spectrum. *IEEE transactions on image processing* 19(9):2241–2253.

Yu, F., and Koltun, V. 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.

Zhang, Y.; Zhou, D.; Chen, S.; Gao, S.; and Ma, Y. 2016. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, 589–597.

Zhang, L.; Wei, W.; Bai, C.; Gao, Y.; and Zhang, Y. 2018a. Exploiting clustering manifold structure for hyperspectral imagery super-resolution. *IEEE Transactions on Image Processing* 27(12):5969–5982.

Zhang, L.; Wei, W.; Zhang, Y.; Shen, C.; van den Hengel, A.; and Shi, Q. 2018b. Cluster sparsity field: An internal hyperspectral imagery prior for reconstruction. *International Journal of Computer Vision* 126(8):797–821.

Zhang, Y.; Tian, Y.; Kong, Y.; Zhong, B.; and Fu, Y. 2018c. Residual dense network for image super-resolution. In *CVPR*, 2472–2481.