Adaptive Rectangular Convolution for Remote Sensing Pansharpening

Xueyang Wang, Zhixin Zheng, Jiandong Shao, Yule Duan, Liang-Jian Deng*
University of Electronic Science and Technology of China, Chengdu, China
xywang_uestc@std.uestc.edu.cn; zhengzx@std.uestc.edu.cn; shamy@std.uestc.edu.cn
yule.duan@outlook.com; liangjian.deng@uestc.edu.cn

Abstract

Recent advancements in convolutional neural network (CNN)-based techniques for remote sensing pansharpening have markedly enhanced image quality. However, conventional convolutional modules in these methods have two critical drawbacks. First, the sampling positions in convolution operations are confined to a fixed square window. Second, the number of sampling points is preset and remains unchanged. Given the diverse object sizes in remote sensing images, these rigid parameters lead to suboptimal feature extraction. To overcome these limitations, we introduce an innovative convolutional module, Adaptive Rectangular Convolution (ARConv). ARConv adaptively learns both the height and width of the convolutional kernel and dynamically adjusts the number of sampling points based on the learned scale. This approach enables ARConv to effectively capture scale-specific features of various objects within an image, optimizing kernel sizes and sampling locations. Additionally, we propose ARNet, a network architecture in which ARConv is the primary convolutional module. Extensive evaluations across multiple datasets reveal the superiority of our method in enhancing pansharpening performance over previous techniques. Ablation studies and visualization further confirm the efficacy of ARConv. The source code can be available at https://github.com/WangXueyang-uestc/ARConv.

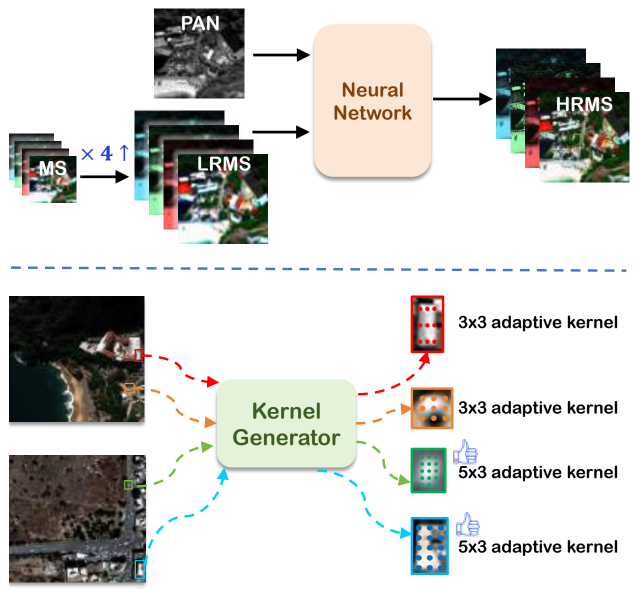

Figure 1. Top row: The flowchart of remote sensing pansharpening via a DL-based approach. Bottom row: An example of our Adaptive Rectangular Convolution (ARConv), boasting two distinct advantages: 1) its convolution kernels can adaptively modify sampling positions in accordance with object sizes; 2) the quantity of sampling points is dynamically determined across various feature maps, for instance, achieving a 5×3 adaptive rectangular convolution, which, to our knowledge, is the first attempt.

1. Introduction

High-quality remote sensing imagery plays a pivotal role in various domains, including military and agriculture. However, existing technologies are only capable of capturing low-resolution multispectral images (LRMS) and high-resolution panchromatic images (PAN). While LRMS imagery preserves rich spectral signatures, it is compromized by limited spatial resolution. Conversely, PAN imagery exhibits superior spatial granularity but lacks multispectral discriminability due to its grayscale representation. As illustrated in Fig. 1, pansharpening addresses this dichotomy through synergistic fusion to synthesize high-resolution multispectral (HRMS) imagery. Many pansharpening methods have been proposed [20], among which traditional methods are divided into Component Substitution (CS) [4, 28], Multi-Resolution Analysis (MRA) [30, 32], and variational optimization (VO) [11, 26]. Recent paradigm shifts toward deep learning have catalyzed extensive deployment of convolutional neural networks (CNNs) in this domain. Different from traditional methods, the input feature of PAN and LRMS images is extracted mainly through convolutional kernels. However, standard convo-

*Corresponding author.

lution has two major drawbacks. First, its sampling positions are fixed within a square window of a determined size, which restricts its ability to deform, thereby preventing it from adaptively finding the sampling locations. Second, the number of sampling points of the convolutional kernel is predetermined, making it challenging to adaptively capture features at different scales. In remote sensing images, the scale differences between different objects can be significant, such as small cars and large buildings, which standard convolutions are not adept at capturing, leading to inefficient feature extraction.

In recent years, many innovative convolutional methods have been proposed. Spatial adaptive convolution methods, such as PAC [25], DDF [40], LAGConv [16], and CANConv [9], can adaptively generate different convolution kernel parameters based on various spatial locations, enabling them to accommodate different spatial regions. However, these methods have yet to fully consider the rich scale information present in remote sensing images. Shape-adaptive convolutions, such as Deformable Convolution [5, 41], can adaptively adjust the position of each sampling point by learning offsets to extract features of objects with different shapes. Although this provides significant flexibility, the number of learnable parameters increases quadratically with the kernel size, making it difficult to achieve convergence on small datasets. Furthermore, it cannot adjust the number of sampling points based on the shape of the convolution kernel, which further limits its performance. Multi-scale convolutions, such as pyramid convolution [10], can extract information at different scales within the same feature map. *However, the size of their convolution kernels is predetermined, while the features in the image may exhibit different patterns and structures across scales. This can lead to imprecise feature fusion between scales, potentially affecting the model's overall performance.*

Based on the above analysis, we propose the Adaptive Rectangular Convolution (ARConv), which can not only adaptively adjust the sampling positions but also the number of sampling points, as shown in Fig. 1. The former is achieved by learning only two parameters: the height and width of the convolution kernel, without incurring additional computational burden as the kernel size increases. The latter selects an appropriate number of sampling points based on the average level of the learned height and width. Moreover, we introduce affine transformation to ARConv, which brings spatial adaptability. All of this enables our module to effectively extract features from objects of varying sizes in the feature map. The main contributions of this paper are summarized as follows:

1. ARConv is proposed as a module that can adaptively adjust the sampling positions and change the number of sampling points, enabling it to effectively capture scale-specific features of various objects in remote sensing images. Based on ARConv and the U-net architecture [23, 35], ARNet is introduced.
2. The relationship between the learned height and width of the convolution kernel and the actual object sizes is explored through heatmap visualizations. A certain level of correlation is observed, which validates the effectiveness of the proposed method.
3. The effectiveness of ARConv is validated by comparing it with various pansharpening methods across multiple datasets. The results demonstrate that ARConv achieves outstanding performance.

2. Related Works

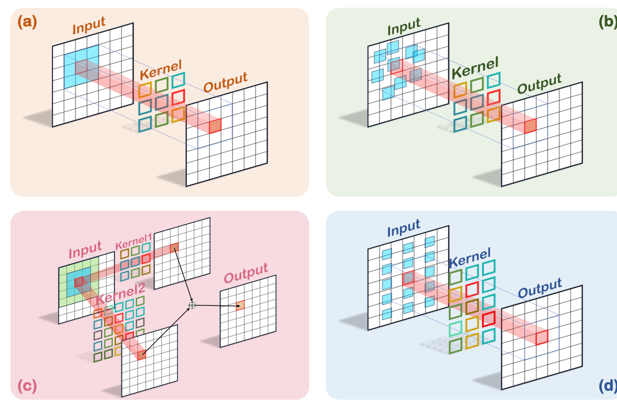

Figure 2. Diagrams illustrating the working principles of four types of convolutional kernels. (a) Standard Convolution. (b) Deformable Convolution [5, 41]. (c) Multi-scale Convolution [10, 18]. (d) Our proposed Convolution (ARConv).

2.1. Adaptive Convolution

Standard convolution, with its fixed shape and size, exhibits limited flexibility in handling geometric transformations, posing challenges in adapting to the varying scales and shapes of objects commonly encountered in visual tasks. Deformable Convolution [5, 41] was the pioneering approach to address this limitation by learning an offset matrix that adjusts the sampling positions at each pixel, as visualized in Fig. 2. This advancement enabled the convolutional kernel to deform in an unsupervised manner for the first time. Building on the concept of deformable convolution, Dynamic Snake Convolution [22] specifically optimizes feature extraction for tubular structures by employing carefully designed loss constraints to guide the deformation of convolutional kernels. Scale-adaptive Convolution [39] extends this flexibility by allowing the convolutional kernel to learn scaling ratios, dynamically modifying the receptive field to better capture features at different scales.

The aforementioned convolutional paradigms face a flexibility-efficiency trade-off: excessive deformability increases computational load with dense sampling points, while rigid structures fail to adapt to irregular objects. Moreover, fixed sampling point counts restrict dynamic adjustments to learned kernel geometries.

2.2. Multi-scale Convolution

Multi-scale convolution enhances the analysis of input data by utilizing convolutional kernels of different sizes, which facilitates the extraction of feature information across different scales. In comparison, standard convolution is restricted to capturing features at a single scale. Pyramidal Convolution (PyConv) [10] addresses this limitation by employing a hierarchical structure within each layer, utilizing a pyramid of convolutional kernels of diverse scales to process the input feature map comprehensively, as described in Fig. 2. To enhance computational efficiency and reduce the overall parameter count, the depth of each kernel—defined by the number of channels participating in the convolution operations is adaptively adjusted based on the pyramid level. Selective Kernel Networks [18] further refine this approach by incorporating a soft attention mechanism that dynamically selects the most relevant feature maps generated from multi-scale convolutions, thereby increasing the network's adaptability to variations in spatial resolution. However, these convolutional modules still cannot adaptively adjust the sampling positions and the number of sampling points of the convolution kernel based on the sizes of various objects in the feature map.

2.3. Motivation

Remote sensing images exhibit considerable variety in context, with objects differing significantly in size. Using convolutional kernels of varying sizes is more effective for extracting features from different regions compared to using fixed-size kernels. Traditional shape-adaptive convolutions can modify sampling positions to align with object shapes but cannot adjust the number of sampling points based on the shape of the kernel. Additionally, some deformable strategies require learning many parameters, leading to higher computational costs. While multi-scale convolutions can capture features at various scales within the same feature map, their kernel sizes remain fixed thus cannot adaptively adjust sampling positions based on the feature map's content. To overcome these limitations, we introduce Adaptive Rectangular Convolution (ARConv), a novel module that *treats the height and width of the convolutional kernel as learnable parameters.* This allows the shape of the kernel to adjust dynamically based on the size of different objects. With sampling points evenly distributed within a rectangular deformable region, *ARConv can flexibly modify sampling positions and adjust the number of points according to the average size of the learned kernels in each feature map.* Unlike conventional deformable convolutions [5], *our approach requires learning only two parameters, minimizing computational overhead as the number of sampling points increases.* To further enhance adaptability, we apply an affine transformation to the kernel's output, improving spatial flexibility.

3. Methods

This section details the design of ARConv and ARNet. The implementation of ARConv follows four steps: (1) Learning the height and width feature maps of the convolutional kernel. (2) Selecting the number of convolutional kernel sampling points. (3) Generating the sampling map. (4) The implementation of the convolution. In ARNet, the standard convolution layers of U-Net [23, 35] are replaced with ARConv modules to more effectively capture rich scale information for the pansharpening task. The overall architecture of ARConv is presented in Fig. 3.

3.1. Adaptive Rectangular Convolution

3.1.1. Learning the Height and Width of Convolution

The learning process can be mathematically formulated as:

$$\mathbf{y}_i = f_{\theta_i}(\mathbf{X}), \quad i \in \{1, 2\}, \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{H\times W\times C_{in}}$ represents the input feature map. H and W denote the height and width of the feature map, respectively, and C_{in} denotes the number of input channels. Besides, $f_{\theta_i}(\cdot)$ corresponds to two subnets responsible for predicting the height and width of the convolutional kernel, each consists of two components: a shared feature extractor and distinct height-width learners with θ_i representing the associated parameters. The output feature maps are denoted as $\mathbf{y}_i \in \mathbb{R}^{H\times W\times 1}$, where $\mathbf{y}_1$ is the height feature map and $\mathbf{y}_2$ is the width feature map, which are referred to as $\mathbf{h}$ and $\mathbf{w}$ in the Fig. 3, respectively. The final layer of the height-width learner is a Sigmoid function, where $\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$. So, $\mathbf{y}_i \in (0, 1)$, which only represent relative magnitudes thus cannot directly correspond to the height and width of the convolutional kernel, we adopt the following method to constrain their range of values.

$$\mathbf{y}_i = a_i \cdot \mathbf{y}_i + b_i, \quad i \in \{1, 2\}, \tag{2}$$

where a_i and b_i are modulation factors that constrain the range of height and width. Thus, the height of the convolutional kernel is constrained within the range $(b_1, a_1 + b_1)$, the width within $(b_2, a_2 + b_2)$.

The height and width feature maps are fed into the second part, which will be detailed later, where the number of sampling points for the convolution kernel, $k_h \cdot k_w$, is selected. Each feature map of height and width is then duplicated $k_h \cdot k_w$ times. Subsequently, a meshgrid operation is applied that aims to evenly distribute the sampling points in the kernel to generate the scaling matrix $Z_{ij} \in \mathbb{R}^{k_h\times k_w}$ at each location of the pixels (i, j).

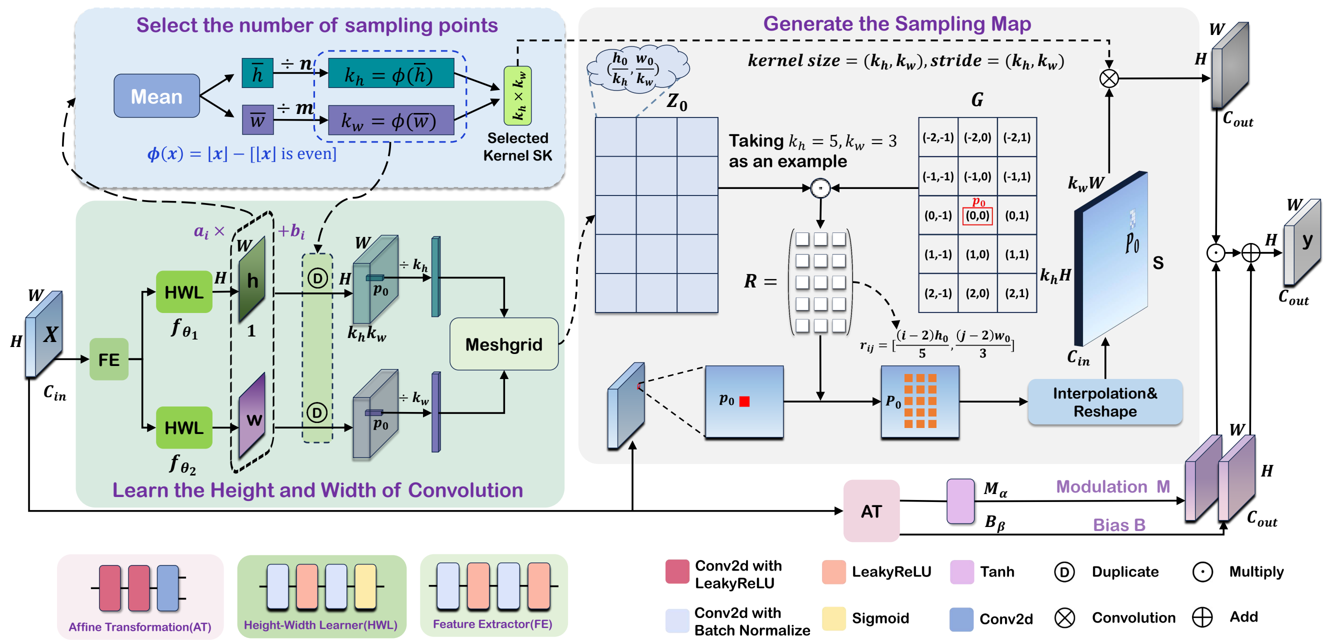

Figure 3. Overview of the ARConv architecture. (1)*Learning the kernel's height and width:* The model dynamically predicts convolutional kernel height and width through feature-driven parameter estimation. (2)*Selecting the number of sampling points:* Optimal kernel size is derived by aggregating spatial statistics from the predicted height and width dimensions. (3)*Generating the sampling map:* The sampling positions are determined using learnable offset parameters, with p_0 serving as the spatial anchor point. At last, the adaptive convolution is applied on the generated sampling map, enhanced by affine transformation.

3.1.2. Selecting the Number of Sampling Points

First, we calculate the mean of all the values in $\mathbf{y}_1$ and $\mathbf{y}_2$ to obtain the average level of the learned height and width. Then, the number of sampling points in the convolutional kernel in vertical and horizontal directions are derived from $k_h = \phi(\lfloor \frac{\bar{\mathbf{y}}_1}{n} \rfloor), k_w = \phi(\lfloor \frac{\bar{\mathbf{y}}_2}{m} \rfloor)$, where $\lfloor x \rfloor$ represents the floor of x, m and n denote the modulation coefficients that map the height and width of the convolutional kernel to the number of sampling points, The function $\phi(\cdot)$ can be expressed as follows:

$$\phi(x) = x - [x \text{ is even}], \tag{3}$$

Here, $[\cdot]$ denotes the Iverson bracket. Given a fixed height and width of the convolutional kernel, the larger the values of m and n, the fewer the sampling points and the sparser their distribution. From Eq. (3), we only select convolutional kernels with an odd number of sampling points. When $\lfloor \frac{\bar{\mathbf{y}}_1}{n} \rfloor$ or $\lfloor \frac{\bar{\mathbf{y}}_2}{m} \rfloor$ is even, we choose the nearest odd number that is smaller than the even number. Finally, the number of sampling points is:

$$N = k_h \cdot k_w. \tag{4}$$

3.1.3. Generating the Sampling Map

In standard convolution, the process involves sampling from the input feature map $\mathbf{X}$ using a regular grid $\mathbf{G}$, followed by a weighted summation of these sampled values with weights $\mathbf{w}$. For example,

$$\mathbf{G} = \{(-1,-1), (-1,0), \cdots, (1,0), (1,1)\}, \tag{5}$$

corresponds to a kernel covering a 3×3 region on the input map with no gaps between sampled points.

Formally, the standard convolution operation for one position $\mathbf{p}_0$ can be expressed as,

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{g}_n \in \mathbf{G}} \mathbf{w}(\mathbf{g}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{g}_n), \tag{6}$$

where $\mathbf{y}$ is the output feature map, $\mathbf{w}$ denotes the parameters of covolutional kernel and $\mathbf{g}_n$ represents the offsets of grid $\mathbf{G}$ relative to position $\mathbf{p}_0$.

For ARConv, we use $\mathbf{G} \in \mathbb{R}^{k_h \times k_w}$ to denote the offset matrix of the standard convolution with kernel size $k_h \times k_w$, which is shared across all pixels. The element at the i-th row and j-th column of $\mathbf{G}$, denoted by g_{ij}, is defined as:

$$g_{ij} = \left(\frac{2i - k_h - 1}{2}, \frac{2j - k_w - 1}{2} \right). \tag{7}$$

Next, we define $\mathbf{Z}_0 \in \mathbb{R}^{k_h \times k_w}$ as the scale matrix at position $\mathbf{p}_0$, which is computed in the first part. The element at the i-th row and j-th column of $\mathbf{Z}_0$, denoted by z_{ij}, is

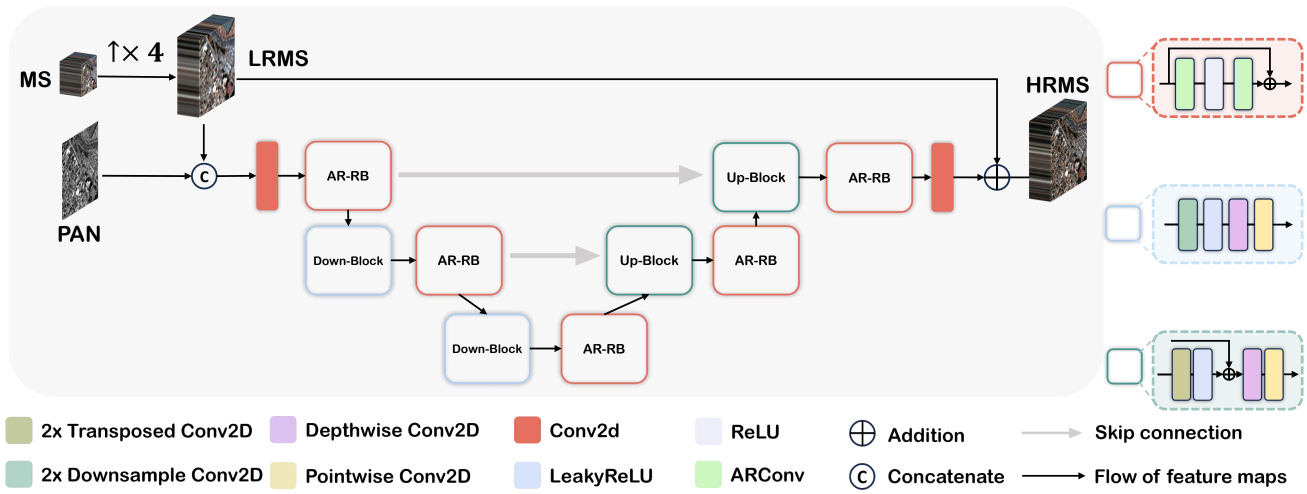

Figure 4. Overall architecture of ARNet. ARNet replaces the standard convolution in U-Net's Resblock with ARConv to create AR-Resblock. The model has down-sampling blocks to extract high-level features and up-sampling blocks to restore spatial resolution with transposed convolutions. Skip connections help transfer detailed spatial information.

given by:

$$z_{ij} = \left(\frac{h_0}{k_h}, \frac{w_0}{k_w}\right), \tag{8}$$

where h_0 and w_0 represent the height and width of the learned convolutional kernel at position $\mathbf{p}_0$, respectively. The offset matrix of ARConv at position $\mathbf{p}_0$, denoted by $\mathbf{R}$, is then computed as:

$$\mathbf{R} = \mathbf{Z}_0 \odot \mathbf{G}, \tag{9}$$

where $\odot$ denotes element-wise multiplication. The element at the i-th row and j-th column of $\mathbf{R}$, denoted by r_{ij}, is given by:

$$r_{ij} = \left(\frac{(2i - k_h - 1)h_0}{2k_h}, \frac{(2j - k_w - 1)w_0}{2k_w}\right). \tag{10}$$

It is evident that, in most cases, the sampling points do not coincide with the center of the grid points. Therefore, interpolation is required to estimate their pixel values. In this context, we employ bilinear interpolation, and its mathematical formulation is as follows:

$$\mathbf{t}(x, y) = \mathbf{w}_x^{\mathrm{T}} \mathbf{T} \mathbf{w}_y, \tag{11}$$

where $\mathbf{t}(x, y)$ represents the pixel value at the coordinates (x, y).

$$\mathbf{T} = \begin{pmatrix} \mathbf{t}(x_0, y_0) & \mathbf{t}(x_0, y_1) \\ \mathbf{t}(x_1, y_0) & \mathbf{t}(x_1, y_1) \end{pmatrix}, \tag{12}$$

where $(x_0, y_0), (x_0, y_1), (x_1, y_0), (x_1, y_1)$ are the coordinates of the four grid points closest to (x, y).

$$\mathbf{w}_x = \begin{pmatrix} 1 - w_x \\ w_x \end{pmatrix}, \mathbf{w}_y = \begin{pmatrix} 1 - w_y \\ w_y \end{pmatrix}, \tag{13}$$

where $w_x = \frac{x - x_0}{x_1 - x_0}, w_y = \frac{y - y_0}{y_1 - y_0}$. they represent the normalized interpolation weights in the x-direction and y-direction, respectively.

In summary, the convolution operation we proposed can be mathematically expressed as:

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{r}_n \in \mathbf{R}} \mathbf{w}(\mathbf{r}_n) \cdot \mathbf{t}(\mathbf{p}_0 + \mathbf{r}_n), \tag{14}$$

where $\mathbf{y}(\mathbf{p}_0)$ refers to the pixel value at position $\mathbf{p}_0$ in the output feature map $\mathbf{y}$, $\mathbf{w}$ denotes the parameters of convolutional kernel, $\mathbf{r}_n$ enumerates the elements in $\mathbf{R}$, $\mathbf{t}(\mathbf{p}_0 + \mathbf{r}_n)$ calculates the pixel value at position $\mathbf{p}_0 + \mathbf{r}_n$.

Whether using standard convolution or our method, each pixel in an image corresponds to a sampling window during the convolution operation. In standard convolution, the sampling points are all located at the grid centers, and the sampling window simply slides across the image with a fixed stride. However, in ARConv, the size of the sampling window vary for each pixel, making the traditional approach unsuitable. In practice, rather than generating a unique convolutional kernel for every pixel, we use an equivalent approach. We adopt an expansion technique, extracting the values at the locations of the sampling points corresponding to the sampling window at each pixel and assembling them into a new grid $\mathbf{P}_0$, which replaces the original pixel $\mathbf{p}_0$, where $\mathbf{P}_0 \in \mathbb{R}^{k_h \times k_w \times C_{in}}$, $\mathbf{p}_0 \in \mathbb{R}^{1 \times 1 \times C_{in}}$. After completing the expansion for each pixel, we obtain the final sampling map $\mathbf{S}$, which belongs to $\mathbb{R}^{(k_h H) \times (k_w W) \times C_{in}}$.

3.1.4. The Implementation of the Convolution

In this part, we apply a convolution to $\mathbf{S}$ for feature extraction, using a kernel size and stride both set to (k_h, k_w). To introduce spatial adaptability, we apply an affine transformation to the output feature map. We use two sub-networks,

Table 1. Performance benchmarking on the WV3 dataset was conducted using 20 reduced-resolution and 20 full-resolution samples. The top-performing results are highlighted in bold, while the second-best are indicated with an underline.

Methods	Reduced-Resolution Metrics			Full-Resolution Metrics		
	SAM↓	ERGAS↓	Q8↑	$D_\lambda \downarrow$	$D_s \downarrow$	HQNR↑
EXP [1]	5.800 ± 1.881	7.155 ± 1.878	0.627 ± 0.092	0.0232 ± 0.0066	0.0813 ± 0.0318	0.897 ± 0.036
MTF-GLP-FS [31]	5.316 ± 1.766	4.700 ± 1.597	0.833 ± 0.092	0.0197 ± 0.0078	0.0630 ± 0.0289	0.919 ± 0.035
TV [21]	5.692 ± 1.808	4.856 ± 1.434	0.795 ± 0.120	0.0234 ± 0.0061	0.0393 ± 0.0227	0.938 ± 0.027
BSDS-PC [29]	5.429 ± 1.823	4.698 ± 1.617	0.829 ± 0.097	0.0625 ± 0.0235	0.0730 ± 0.0356	0.870 ± 0.053
CVPR2019 [12]	5.207 ± 1.574	5.484 ± 1.505	0.764 ± 0.088	0.0297 ± 0.0059	0.0410 ± 0.0136	0.931 ± 0.0183
LRTCFPan [37]	4.737 ± 1.412	4.315 ± 1.442	0.846 ± 0.091	0.0176 ± 0.0066	0.0528 ± 0.0258	0.931 ± 0.031
PNN [19]	3.680 ± 0.763	2.682 ± 0.648	0.893 ± 0.092	0.0213 ± 0.0080	0.0428 ± 0.0147	0.937 ± 0.021
PanNet [38]	3.616 ± 0.766	2.666 ± 0.689	0.891 ± 0.093	0.0165 ± 0.0074	0.0470 ± 0.0213	0.937 ± 0.027
DiCNN [15]	3.593 ± 0.762	2.673 ± 0.663	0.900 ± 0.087	0.0362 ± 0.0111	0.0462 ± 0.0175	0.920 ± 0.026
FusionNet [6]	3.325 ± 0.698	2.467 ± 0.645	0.904 ± 0.090	0.0239 ± 0.0090	0.0364 ± 0.0137	0.941 ± 0.020
DCFNet [36]	3.038 ± 0.585	2.165 ± 0.499	0.913 ± 0.087	0.0187 ± 0.0072	0.0337 ± 0.0054	0.948 ± 0.012
LAGConv [16]	3.104 ± 0.559	2.300 ± 0.613	0.910 ± 0.091	0.0368 ± 0.0148	0.0418 ± 0.0152	0.923 ± 0.025
HMPNet [27]	3.063 ± 0.577	2.229 ± 0.545	0.916 ± 0.087	0.0184 ± 0.0073	0.0530 ± 0.0555	0.930 ± 0.011
CMT [24]	2.994 ± 0.607	2.214 ± 0.516	0.917 ± 0.085	0.0207 ± 0.0082	0.0370 ± 0.0078	0.943 ± 0.014
CANNet [9]	2.930 ± 0.593	2.158 ± 0.515	0.920 ± 0.084	0.0196 ± 0.0083	0.0301 ± 0.0074	0.951 ± 0.013
Proposed	**2.885 ± 0.590**	**2.139 ± 0.528**	**0.921 ± 0.083**	**0.0146 ± 0.0059**	**0.0279 ± 0.0068**	**0.958 ± 0.010**

$\mathbf{M}_\alpha$ and $\mathbf{B}_\beta$, to predict the matrices $\mathbf{M}$ and $\mathbf{B}$ for the affine transformation, with α and β as the parameters of these networks. The final output feature map is given by:

$$\mathbf{y} = \mathbf{SK} \otimes \mathbf{S} \odot \mathbf{M} \oplus \mathbf{B}, \tag{15}$$

where $\mathbf{y} \in \mathbb{R}^{H \times W \times C_{out}}$ is the output feature map. $\mathbf{SK} \in \mathbb{R}^{C_{in} \times k_h \times k_w \times C_{out}}$ is the parameter of the selected convolutional kernel, $\otimes$ represents the convolution operation, $\odot$ represents element-wise multiplication and $\oplus$ represents elements-wise plus.

3.2. ARNet Architecture

This section details the construction of ARNet which is shown at Fig. 4. Our network draws inspiration from the U-net architecture [23, 35], a well-known model in image segmentation that uses an encoder-decoder structure with skip connections to retain spatial information. In ARNet, we replace the standard convolutional layers in ResBlock [14] with our ARConv. The data flow proceeds as follows: First, the MS image is upsampled to match the resolution of the PAN image, generating the LRMS image. Next, the PAN and LRMS images are concatenated along the channel dimension and input into the network. ARNet involves a series of downsampling and upsampling steps, with ARConv layers at different depths adapting to find the optimal parameters for feature extraction at various scales. Finally, the learned details are injected back into the LRMS image [6, 15], refining it and producing the final output image with enhanced resolution and detail.

4. Experiments

4.1. Datasets, Metrics and Training Details

We evaluate the effectiveness of our method on several datasets, including 8-band data captured by the WorldView3 (WV3) sensor, as well as 4-band data captured by the QuickBird (QB) and the GaoFen-2 (GF2) sensors. Although we use a supervised learning approach, ground truth data is not directly available, so we apply Wald's protocol [7, 34] to construct our dataset. All three datasets are accessible from a public repository [8]. For test sets with different resolutions, we use different evaluation metrics. Specifically, we employ SAM [3], ERGAS [33], and Q8 [13] to assess the performance of ARNet on the reduced-resolution dataset, and D_s, D_λ, and HQNR [2] to evaluate its performance on the full-resolution dataset. During training, we employ the l_1 loss function along with the Adam optimizer [17], using a batch size of 16. Given that our method involves selecting convolutional kernels based on the learned height and width—an approach that can complicate convergence—we designate the initial 100 epochs as an exploratory phase. During this phase, we allow the model to explore different configurations. After these 100 epochs, we randomly select a combination of convolutional kernels from the 16 batches based on the results obtained and then fix this selection for the remainder of the training. *Further details on the dataset and training procedure are provided in the supplementary material Sec. 7.1 and 7.2.*

4.2. Results

The outstanding performance of ARNet has been thoroughly demonstrated through a comprehensive evaluation on the WV3, QB, and GF2 benchmark datasets. Tab. 1 to 3 provide a detailed comparison of ARNet against various state-of-the-art techniques, including traditional methods, general deep learning methods, and specialized convolution-based deep learning approaches similar to proposed work, such as LAGConv [16] and CANConv [9], *more details can be found in supplementary material Sec. 7.3*. The results clearly indicate that ARNet consistently delivers high-quality performance across different

Table 2. Performance benchmarking on the QB dataset using 20 reduced-resolution samples. Best in bold; second best underlined.

Methods	SAM↓	ERGAS↓	Q4↑
EXP [1]	8.435±1.925	11.819±1.905	0.584±0.075
TV [21]	7.565±1.535	7.781±0.699	0.820±0.090
MTF-GLP-FS [31]	7.793±1.816	7.374±0.724	0.835±0.088
BDSD-PC [29]	8.089±1.980	7.515±0.800	0.831±0.090
CVPR19 [12]	7.998±1.820	9.359±1.268	0.737±0.087
LRTCFPan [37]	7.187±1.711	6.928±0.812	0.855±0.087
PNN [19]	5.205±0.963	4.472±0.373	0.918±0.094
PanNet [38]	5.791±1.184	5.863±0.888	0.885±0.092
DiCNN [15]	5.380±1.027	5.135±0.488	0.904±0.094
FusionNet [6]	4.923±0.908	4.159±0.321	0.925±0.090
DCFNet [36]	4.512±0.773	3.809±0.336	0.934±0.087
LAGConv [16]	4.547±0.830	3.826±0.420	0.934±0.088
HMPNet [27]	4.617±0.404	**3.404±0.478**	0.936±0.102
CMT [24]	4.535±0.822	3.744±0.321	0.935±0.086
CANNet [9]	4.507±0.835	3.652±0.327	0.937±0.083
Proposed	**4.430±0.811**	3.633±0.327	**0.939±0.081**

Table 3. Performance benchmarking on the GF2 dataset using 20 reduced-resolution samples. Best in bold; second best underlined.

Methods	SAM↓	ERGAS↓	Q4↑
EXP [1]	1.820±0.403	2.366±0.554	0.812±0.051
TV [21]	1.918±0.398	1.745±0.405	0.905±0.027
MTF-GLP-FS [31]	1.655±0.385	1.589±0.395	0.897±0.035
BDSD-PC [29]	1.681±0.360	1.667±0.445	0.892±0.035
CVPR19 [12]	1.598±0.353	1.877±0.448	0.886±0.028
LRTCFPan [37]	1.315±0.283	1.301±0.313	0.932±0.033
PNN [19]	1.048±0.226	1.057±0.236	0.960±0.010
PanNet [38]	0.997±0.212	0.919±0.191	0.967±0.010
DiCNN [15]	1.053±0.231	1.081±0.254	0.959±0.010
FusionNet [6]	0.974±0.212	0.988±0.222	0.964±0.009
DCFNet [36]	0.872±0.169	0.784±0.146	0.974±0.009
LAGConv [16]	0.786±0.148	0.687±0.113	0.981±0.008
HMPNet [27]	0.803±0.141	**0.564±0.099**	0.981±0.020
CMT [24]	0.753±0.138	0.648±0.109	0.982±0.007
CANNet [9]	0.707±0.148	0.630±0.128	**0.983±0.006**
Proposed	**0.698±0.149**	0.626±0.127	0.983±0.007

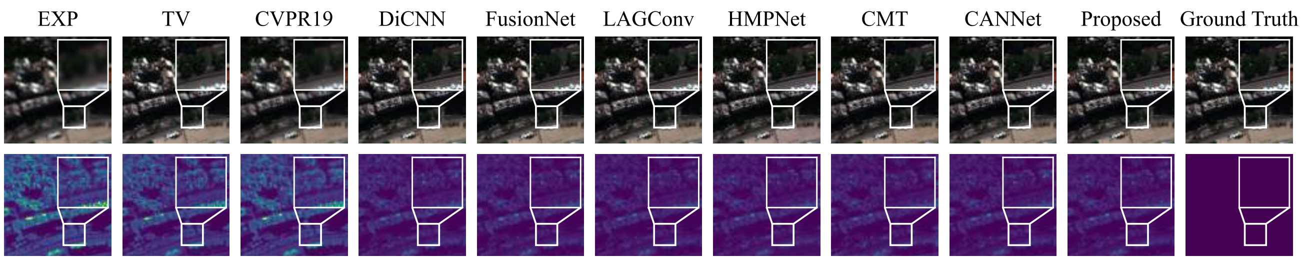

Figure 5. Qualitative comparison of benchmark methods on WV3 reduced-resolution dataset. Top: RGB outputs; Bottom: residuals vs. ground truth. *See Suppl. Sec. 7.5 for details.*

datasets, showcasing remarkable robustness. Moreover, visual assessments reveal that the images generated by ARNet are the closest to the ground truth, illustrating the ability of our convolutional approach to effectively adapt to varying object sizes and extract features at appropriate scales. *For further details on the benchmark tests and visual examples, please refer to the supplementary material Sec. 7.5.*

Table 4. Ablation study on WV3 reduced-res dataset: HWA (height and width adaptation), NSPA (sampling points adaptation), AT (affine transformation).

Methods	SAM↓	ERGAS↓	Q8↑
(a) w/o HWA	2.925±0.593	2.171±0.557	0.920±0.085
(b) w/o NSPA	2.911±0.603	2.152±0.565	0.921±0.083
(c) w/o AT	3.020±0.614	2.269±0.562	0.916±0.085
Proposed	2.885±0.590	2.139±0.528	0.921±0.083

4.3. Ablation Study

To assess the impact of different components in ARConv, we conducted ablation experiments by selectively removing certain modules: (a) without height and width adaptation, (b) without the number of sampling points adaptation, and (c) without affine transformation. The results are shown in Tab. 4. The performance drop in (a) and (b) highlights the effectiveness of ARConv in adapting to different object sizes. In (c), the sharp decline indicates limited flexibility in our deformation strategy, which is effectively mitigated by introducing spatial adaptability through affine transformation. Notably, the computational cost of this transformation does not increase with kernel size.

Table 5. Performance of different convolution kernel height and width learning ranges on WV3 reduced-resolution dataset.

Methods	SAM↓	ERGAS↓	Q8↑
(a) 1-3	2.923±0.600	2.164±0.546	0.919±0.085
(b) 1-9	2.896±0.588	2.145±0.544	0.921±0.084
(c) 1-18	2.885±0.590	2.139±0.528	0.921±0.083
(d) 1-36	3.044±0.646	2.216±0.578	0.916±0.087
(e) 1-63	3.066±0.593	2.249±0.554	0.912±0.095

4.4. Discussion

Different Height and Width Learning Ranges: To assess the impact of different convolutional kernel heights and widths on ARNet's performance, we designed five sets of experiments with varying height and width ranges: (a) 1-3, (b) 1-9, (c) 1-18, (d) 1-36, and (e) 1-63. In (a), the kernel size is fixed at 3×3, while in (b) to (e), the maximum kernel size is 7×7. As shown in Tab. 5, ARNet's performance initially improves with an increasing height and width range but declines beyond an optimal setting in case (c). This pattern arises because a smaller range results in densely packed sampling points that capture excessive noise, while a larger

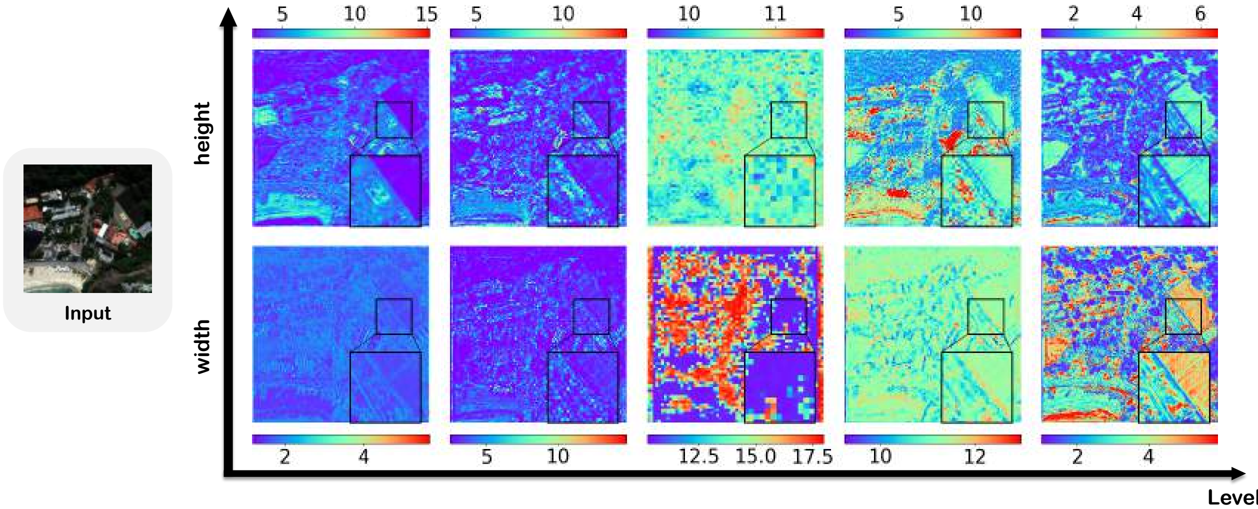

Figure 6. Heatmaps of the heights and widths learned at each pixel by convolutional kernels at different layers. The input image is a sample from the WV3 dataset. In the heatmaps, various colors represent different heights and widths captured by the convolutional kernels.

range spreads sampling points too sparsely, reducing the kernel's ability to capture fine details.

Table 6. Performance on the WV3 reduced-resolution dataset when replacing convolution kernels in other pansharpening methods with ARConv.

Methods	SAM↓	ERGAS↓	Q8↑
FusionNet [6]	3.325±0.698	2.467±0.645	0.904±0.090
AR-FusionNet	3.171±0.650	2.395±0.630	0.911±0.087
LAGNet [16]	3.104±0.559	2.300±0.613	0.910±0.091
AR-LAGNet	3.083±0.643	2.277±0.547	0.916±0.085
CANNet [9]	2.930±0.593	2.158±0.515	0.920±0.084
AR-CANNet	2.885±0.590	2.139±0.528	0.921±0.083

Replacing the Convolutional Modules in other Networks: We integrate ARConv as a plug-and-play module, replacing the original convolution layers in pansharpening networks such as FusionNet [6], LAGNet [16], and CANNet [9] to demonstrate ARConv's effectiveness. The results in Tab. 6 indicate that ARConv significantly enhances the performance of these networks. *Additional details on this experiment are provided in the supplementary materials Sec. 7.4.*

Convolutional Kernel Visualization: Fig. 6 shows the height and width feature maps learned by the convolutional kernels at different layers of ARNet. The overall heatmaps reveal the contours of various objects in the RGB image, especially in the outermost layers of the network. Although the intermediate layers appear disordered, they capture deeper semantic information, such as object sizes in the RGB image. For example, in the height heatmap of the fourth layer, the outline of a tilted building is faintly visible, with a thin blue line along the edges. This indicates that the learned heights of the convolutional kernels are smaller at the edges, reflecting the adaptation of the kernels to the building's dimensions. *Please refer to supplementary material in Sec. 7.6 for more visualizations.*

Table 7. Performance comparison between ARConv and DCNv2 on WV3 reduced-resolution.

Methods	SAM↓	ERGAS↓	Q8↑
Ours	2.881±0.590	2.149±0.531	0.921±0.084
DCNv2 [41]	3.151±0.679	2.425±0.656	0.915±0.083

Comparison with DCNv2: We removed the affine transformation from ARConv and adopted the same modulation method as DCNv2 [41]. Both models were trained for 600 epochs on the WV3 dataset. The results are shown in Tab. 7, where it is evident that our performance surpasses DCNv2. This may be because the deformation strategy in DCNv2 requires learning a larger number of parameters, which can hinder convergence in sharpening tasks.

5. Conclusion

In conclusion, we introduce an adaptive rectangular convolution module, ARConv, which dynamically learns height- and width-adaptive convolution kernels for each pixel based on the varying sizes of objects in the input image. By adjusting the number of sampling points according to the learned scale, ARConv overcomes the traditional limitations of fixed sampling shapes and point counts in convolution kernels. Integrated seamlessly into U-net as a plug-and-play module, ARConv forms ARNet, which has demonstrated outstanding performance across multiple datasets. Additionally, the visualization studies confirm that our convolutional kernels can effectively adjust their height and width based on the size and shape of objects, offering a novel solution for the pansharpening task.

6. Acknowledgement

This work was supported in part by the NSFC (12271083, 62273077), Sichuan Province's Science and Technology Empowerment for Disaster Prevention, Mitigation, and Relief Project (2025YFNH0001), and the Natural Science Foundation of Sichuan Province (2024NSFJQ0013).

References

[1] Bruno Aiazzi, Luciano Alparone, Stefano Baronti, and Andrea Garzelli. Context-driven fusion of high spatial and spectral resolution images based on oversampled multiresolution analysis. *IEEE Trans. Geosci. Remote. Sens.*, 40:2300–2312, 2002. 6, 7, 2, 3

[2] A. Arienzo, Gemine Vivone, Andrea Garzelli, Luciano Alparone, and Jocelyn Chanussot. Full-resolution quality assessment of pansharpening: Theoretical and hands-on approaches. *IEEE Geoscience and Remote Sensing Magazine*, 10:168–201, 2022. 6

[3] Joseph W. Boardman. Automating spectral unmixing of aviris data using convex geometry concepts. 1993. 6

[4] Jaewan Choi, Kiyun Yu, and Yongil Kim. A new adaptive component-substitution-based satellite image fusion by using partial replacement. *IEEE Transactions on Geoscience and Remote Sensing*, 49:295–309, 2011. 1

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 764–773, 2017. 2, 3

[6] Liang-Jian Deng, Gemine Vivone, Cheng Jin, and Jocelyn Chanussot. Detail injection-based deep convolutional neural networks for pansharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 59:6995–7010, 2021. 6, 7, 8, 2, 3

[7] Liang-Jian Deng, Gemine Vivone, Cheng Jin, and Jocelyn Chanussot. Detail injection-based deep convolutional neural networks for pansharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 59:6995–7010, 2021. 6

[8] Liang-Jian Deng, Gemine Vivone, Mercedes Eugenia Paoletti, Giuseppe Scarpa, Jiang He, Yongjun Zhang, Jocelyn Chanussot, and Antonio J. Plaza. Machine learning in pansharpening: A benchmark, from shallow to deep networks. *IEEE Geoscience and Remote Sensing Magazine*, 10:279–315, 2022. 6, 1

[9] Yule Duan, Xiao Wu, Haoyu Deng, and Liangjian Deng. Content-adaptive non-local convolution for remote sensing pansharpening. In *Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27738–27747, 2024. 2, 6, 7, 8, 3

[10] Ionut Cosmin Duta, Li Liu, Fan Zhu, and Ling Shao. Pyramidal convolution: Rethinking convolutional neural networks for visual recognition. *ArXiv*, abs/2006.11538, 2020. 2, 3

[11] Xueyang Fu, Zihuang Lin, Yue Huang, and Xinghao Ding. A variational pan-sharpening with local gradient constraints. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10257–10266, 2019. 1

[12] Xueyang Fu, Zihuang Lin, Yue Huang, and Xinghao Ding. A variational pan-sharpening with local gradient constraints. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10257–10266, 2019. 6, 7, 2, 3

[13] Andrea Garzelli and Filippo Nencini. Hypercomplex quality assessment of multi/hyperspectral images. *IEEE Geoscience and Remote Sensing Letters*, 6:662–665, 2009. 6

[14] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 6

[15] Lin He, Yizhou Rao, Jun Yu Li, Jocelyn Chanussot, Antonio J. Plaza, Jiawei Zhu, and Bo Li. Pansharpening via detail injection based convolutional neural networks. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12:1188–1204, 2018. 6, 7, 2, 3

[16] Zi-Rong Jin, Tian-Jing Zhang, Tai-Xiang Jiang, Gemine Vivone, and Liang-Jian Deng. Lagconv: Local-context adaptive convolution kernels with global harmonic bias for pansharpening. In *AAAI Conference on Artificial Intelligence*, 2022. 2, 6, 7, 8, 3

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. 6, 1

[18] Xiang Li, Wenhai Wang, Xiaolin Hu, and Jian Yang. Selective kernel networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 510–519, 2019. 2, 3

[19] Giuseppe Masi, Davide Cozzolino, Luisa Verdoliva, and Giuseppe Scarpa. Pansharpening by convolutional neural networks. *Remote. Sens.*, 8:594, 2016. 6, 7, 2, 3

[20] Xiangchao Meng, Huanfeng Shen, Huifang Li, Liangpei Zhang, and Randi Fu. Review of the pansharpening methods for remote sensing images based on the idea of meta-analysis: Practical discussion and challenges. *Inf. Fusion*, 46:102–113, 2019. 1

[21] Frosti Palsson, Johannes R. Sveinsson, and Magnus Orn Ulfarsson. A new pansharpening algorithm based on total variation. *IEEE Geoscience and Remote Sensing Letters*, 11:318–322, 2014. 6, 7, 2, 3

[22] Yaolei Qi, Yuting He, Xiaoming Qi, Yuanyuan Zhang, and Guanyu Yang. Dynamic snake convolution based on topological geometric constraints for tubular structure segmentation. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6047–6056, 2023. 2

[23] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015. 2, 3, 6, 1

[24] Wenjie Shu, Hong-Xia Dou, Rui Wen, Xiao Wu, and Liang-Jian Deng. Cmt: Cross modulation transformer with hybrid loss for pansharpening. *IEEE Geoscience and Remote Sensing Letters*, 21:1–5, 2024. 6, 7, 2, 3

[25] Hang Su, V. Jampani, Deqing Sun, Orazio Gallo, Erik G. Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11158–11167, 2019. 2

[26] Xin Tian, Yuerong Chen, Changcai Yang, and Jiayi Ma. Variational pansharpening by exploiting cartoon-texture similarities. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–16, 2021. 1

[27] Xin Tian, Kun Li, Wei Zhang, Zhongyuan Wang, and Jiayi Ma. Interpretable model-driven deep network for hyperspectral, multispectral, and panchromatic image fusion. *IEEE Transactions on Neural Networks and Learning Systems*, 35:14382–14395, 2023. 6, 7, 2, 3

[28] Gemine Vivone. Robust band-dependent spatial-detail approaches for panchromatic sharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 57:6421–6433, 2019. 1

[29] Gemine Vivone. Robust band-dependent spatial-detail approaches for panchromatic sharpening. *IEEE Transactions on Geoscience and Remote Sensing*, 57:6421–6433, 2019. 6, 7, 2, 3

[30] Gemine Vivone, Rocco Restaino, and Jocelyn Chanussot. Full scale regression-based injection coefficients for panchromatic sharpening. *IEEE Transactions on Image Processing*, 27:3418–3431, 2018. 1

[31] Gemine Vivone, Rocco Restaino, and Jocelyn Chanussot. Full scale regression-based injection coefficients for panchromatic sharpening. *IEEE Transactions on Image Processing*, 27:3418–3431, 2018. 6, 7, 2, 3

[32] Gemine Vivone, Rocco Restaino, Mauro Dalla Mura, Giorgio Licciardi, and Jocelyn Chanussot. Contrast and error-based fusion schemes for multispectral image pansharpening. *IEEE Geoscience and Remote Sensing Letters*, 11:930–934, 2014. 1

[33] Lucien Wald. Data fusion. definitions and architectures - fusion of images of different spatial resolutions. 2002. 6

[34] Lucien Wald, Thierry Ranchin, and Marc Mangolini. Fusion of satellite images of different spatial resolutions: Assessing the quality of resulting images. *Photogrammetric Engineering and Remote Sensing*, 63:691–699, 1997. 6

[35] Yudong Wang, Liang-Jian Deng, Tian-Jing Zhang, and Xiao Wu. Ssconv: Explicit spectral-to-spatial convolution for pansharpening. *Proceedings of the 29th ACM International Conference on Multimedia*, 2021. 2, 3, 6, 1

[36] Xiao Wu, Tingzhu Huang, Liang-Jian Deng, and Tian-Jing Zhang. Dynamic cross feature fusion for remote sensing pansharpening. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14667–14676, 2021. 6, 7, 2, 3

[37] Zhong-Cheng Wu, Ting-Zhu Huang, Liang-Jian Deng, Jie Huang, Jocelyn Chanussot, and Gemine Vivone. Lrtcfpan: Low-rank tensor completion based framework for pansharpening. *IEEE Transactions on Image Processing*, 32:1640–1655, 2023. 6, 7, 2, 3

[38] Junfeng Yang, Xueyang Fu, Yuwen Hu, Yue Huang, Xinghao Ding, and John Paisley. Pannet: A deep network architecture for pan-sharpening. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1753–1761, 2017. 6, 7, 2, 3

[39] Rui Zhang, Sheng Tang, Yongdong Zhang, Jintao Li, and Shuicheng Yan. Scale-adaptive convolutions for scene parsing. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2050–2058, 2017. 2

[40] Jingkai Zhou, V. Jampani, Zhixiong Pi, Qiong Liu, and Ming-Hsuan Yang. Decoupled dynamic filter networks. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6643–6652, 2021. 2

[41] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9300–9308, 2018. 2, 8