Content-Adaptive Non-Local Convolution for Remote Sensing Pansharpening

Supplementary Material

Abstract

The supplementary materials offer further insights into the CANConv method proposed in our paper. We delve into a thorough analysis of the clustering model within CANConv, offering additional details on experimental settings, encompassing datasets and training parameters. Furthermore, we introduce alternative methods used in benchmarking results and elaborate on the settings for discussion experiments. Lastly, we provide additional benchmarks on QB and GF2 full-resolution datasets and visual comparisons of results among benchmarked methods.

6. Analysis on KNN and K-Means

Many previous works[23, 29, 46] have used the K-Nearest Neighbors (KNN) model to capture similarity relationships in feature maps, while our method employs the K-Means clustering algorithm, which has significant differences between the two approaches.

K-Nearest Neighbors (KNN): In traditional machine learning, the KNN algorithm is commonly used for classification and regression tasks by determining the classification or regression value based on the values of the k-nearest neighbors to the sample to be predicted. Previous graph convolution methods used the KNN model to model similarity relationships between patches in images, requiring the computation of pairwise distances between all patches and finding the k most similar patches for each patch, incurring a large computational cost. These methods achieve information propagation through convolution layers by concatenating patches along the channel dimension, increasing the spatial dimensions of the feature map by a factor of k, and pre-trained weights cannot adapt to changes in k.

K-Means: This paper adopts a clustering approach to model similarity relationships between patches and selects the simple unsupervised K-Means algorithm to perform clustering, dividing samples into K clusters to maximize similarity within each cluster and minimize similarity between clusters. The typical usage of the K-Means algorithm in traditional machine learning involves iteratively computing K cluster centers on the training set and directly finding the cluster center closest to the sample during prediction. In deep learning for vision tasks, the dataset contains a vast number of patches, and the data distribution of the feature map changes with training epochs. To achieve maximum flexibility to adapt to different input data, *we choose to cluster all patches in a single image during both training and inference, recomputing cluster centers, rather than only comparing samples with those in the training set during inference.* As an unsupervised clustering algorithm, K-Means does not guarantee that the same content will be assigned the same cluster number in each image. For example, the ocean on image 1 may belong to cluster 3, while the ocean on image 2 may belong to cluster 6. This limitation prevents us from specifying convolution kernels based on cluster numbers in the PWAC module; instead, we generate convolution kernels adaptively based on the content of the clusters. The benefit of this approach is the *decoupling of learnable parameters from the value of K*. We don't need to store K sets of convolution kernel parameters, using only one set of parameters to generate different convolution kernels for all clusters, and allowing for changing the value of K at any time to adapt to varying inputs.

7. Backpropagation in Cluster Algorithm

Since K-Means is an unsupervised clustering algorithm, we have to carefully handle the gradients. Though K-Means is not differentiable, it will not block the backpropagation in the network, since its output $\mathbf{I}$ is only used for index-selecting $\mathbf{X}$ to get $\mathbf{c}_i$. It is still possible to estimate gradients of $\mathbf{X}$ directly from $\mathbf{c}_i$, while ignoring gradients from $\mathbf{I}$. The whole process can be written as

$$\frac{\partial \mathcal{L}}{\partial \mathbf{X}} = \frac{\partial \mathcal{L}}{\partial \mathbf{c}_i}\left(\frac{\partial \mathbf{c}_i}{\partial \mathbf{X}} + \underbrace{\frac{\partial \mathbf{c}_i}{\partial \mathbf{I}} \cdot \frac{\partial \mathbf{I}}{\partial \mathbf{X}}}_{\text{Ignored}}\right).$$

In practice, we compute gradients of $\mathbf{c}_i$ and $\mathbf{p}_{xy}$ in $\mathbf{X}$ using the following formulas:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \mathbf{c}_i} = &\left(\sum_{(x,y)\in S_i} \mathbf{p}_{xy}^\top \times \frac{\partial \mathcal{L}}{\partial \mathbf{Y}_{xy}}\right)\frac{\partial \mathbf{W}_i}{\partial \mathbf{c}_i} \\ &+ \left(\sum_{(x,y)\in S_i} \mathbf{p}_{xy}\right)\frac{\partial \mathbf{b}_i}{\partial \mathbf{c}_i},\end{aligned} \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{p}_{xy}} = \frac{\partial \mathcal{L}}{\partial \mathbf{Y}_{xy}} \times \mathbf{W}_{\mathbf{I}_{xy}}^\top + \frac{1}{|S_{\mathbf{I}_{xy}}|}\frac{\partial \mathcal{L}}{\partial \mathbf{c}_{\mathbf{I}_{xy}}}, \tag{9}$$

where $\mathcal{L}$ refers to the loss function. $\partial \mathbf{W}_i/\partial \mathbf{c}_i$, $\partial \mathbf{W}_i/\partial \mathbf{c}_i$ and the gradients of the learnable parameters in f_k, f_b and $\mathbf{W}$, can be easily calculated using automatic differentiation frameworks. Experimental results indicate that ignoring gradients related to clustering and index operations does not affect the convergence of the network.

Table 7. Introduction for pansharpening methods involved in the benchmark.

Method	Category	Year	Introduction
EXP [1]		2002	Simply upsamples the MS image.
MTF-GLP-FS [36]	MRA	2018	Estimates the injection coefficients at full resolution rather than reduced resolution.
TV [27]	VO	2013	Employs total variation as a regularization technique for addressing an ill-posed problem defined by a commonly utilized explicit model for image formation.
BDSD-PC [34]	CS	2018	Addresses the limitations of the band-dependent spatial-detail (BDSD) method in images with more than four spectral bands.
CVPR2019 [14]	VO	2019	Integrates a more precise spatial preservation strategy by considering local gradient constraints within distinct local patches and bands.
LRTCFPan [42]	VO	2023	Utilizes low-rank tensor completion (LRTC) as the foundation and incorporating various regularizers for enhanced performance.
PNN [25]	ML	2016	The first convolutional neural network (CNN) for pansharpening with three convolutional layers.
PanNet [44]	ML	2017	Deeper CNN for pansharpening.
DiCNN [17]	ML	2019	Introduces the detail injection procedure into pansharpening CNNs.
FusionNet [11]	ML	2021	Combines ML techniques with traditional fusion schemes like CS and MRA.
DCFNet [41]	ML	2021	Considers the connections of information between high-level semantics and low-level features through the incorporation of multiple parallel branches.
MMNet [43]	ML	2022	A model-driven deep unfolding network with memory-augmentation.
LAGConv [19]	ML	2022	Adaptive convolution with enhanced ability to leverage local information and preserve global harmony.
HMPNet [33]	ML	2023	An interpretable model-driven deep network tailored for the fusion of hyperspectral (HS), multispectral (MS), and panchromatic (PAN) images

8. Details on Experiments and Discussion

8.1. Datasets

We conducted experiments on data collected from the WorldView-3 (WV3), QuickBird (QB) and GaoFen-2 (GF2) satellites. The datasets consist of images cropped from entire remote sensing images, divided into training and testing sets. The training set comprises PAN/LRMS/GT image pairs obtained by downsampling simulation, with dimensions of 64×64, $16 \times 16 \times C$ and $64 \times 64 \times C$, respectively. The WV3 training set contains approximately 10,000 pairs of eight-channel images ($C = 8$), while the QB training set contains around 17,000 pairs of four-channel images ($C = 4$). GF2 training set has about 20,000 pairs of four-channel images ($C = 4$). The reduced-resolution testing set for each satellite consists of 20 downsampling simulated PAN/LRMS/GT image pairs with various representative land covers, with dimensions of 256×256, $64 \times 64 \times C$, and $256 \times 256 \times C$, respectively. The full-resolution test set includes 20 pairs of original PAN/LRMS images with dimensions of 512×512 and 128×128. Our datasets and data processing methods are downloaded from the PanCollection repository [12].

8.2. Training Details

When training CANNet on the WV3 dataset, we utilized the ℓ_1 loss function and Adam optimizer [20] with a batch size of 32. The initial learning rate was set at 10^{-3}, which was reduced to 10^{-4} after 250 epochs. The total duration of the training was 500 epochs. Regarding the network architecture, we set the number of channels in the hidden layers to 32, the number of clusters K during training was set to 32 and the threshold η was 0.005. To encourage stable clustering learning, we recalculated and updated the cluster indices every 10 epochs during training. For the QB dataset, we maintained a constant learning rate of 5×10^{-4} and only

Table 8. Result benchmark on the QB dataset with 20 full-resolution samples. **Bold**: best, underline: second best.

Method	$D_\lambda \downarrow$	$D_s \downarrow$	HQNR↑
EXP [1]	0.0436±0.0089	0.1502±0.0167	0.813±0.020
TV [27]	0.0465±0.0146	0.1500±0.0238	0.811±0.034
MTF-GLP-FS [36]	0.0550±0.0142	0.1009±0.0265	0.850±0.037
BDSD-PC [34]	0.1975±0.0334	0.1636±0.0483	0.672±0.058
CVPR19 [14]	0.0498±0.0119	0.0783±0.0170	0.876±0.023
LRTCFPan [42]	**0.0226±0.0117**	0.0705±0.0351	0.909±0.044
PNN [25]	0.0577±0.0110	0.0624±0.0239	0.884±0.030
PanNet [44]	0.0426±0.0112	0.1137±0.0323	0.849±0.039
DiCNN [17]	0.0947±0.0145	0.1067±0.0210	0.809±0.031
FusionNet [11]	0.0572±0.0182	0.0522±0.0088	0.894±0.021
DCFNet [41]	0.0469±0.0150	0.1239±0.0269	0.835±0.016
MMNet [43]	0.0768±0.0257	**0.0374±0.0201**	0.889±0.041
LAGConv [19]	0.0859±0.0237	0.0676±0.0136	0.852±0.018
HMPNet [33]	0.1838±0.0542	0.0793±0.0245	0.753±0.065
Proposed	0.0370±0.0129	0.0499±0.0092	**0.915±0.012**

Table 9. Result benchmark on the GF2 dataset with 20 full-resolution samples. **Bold**: best, underline: second best.

Method	$D_\lambda \downarrow$	$D_s \downarrow$	HQNR↑
EXP [1]	0.0180±0.0081	0.0957±0.0209	0.888±0.023
TV [27]	0.0346±0.0137	0.1429±0.0282	0.828±0.035
MTF-GLP-FS [36]	0.0553±0.0430	0.1118±0.0226	0.839±0.044
BDSD-PC [34]	0.0759±0.0301	0.1548±0.0280	0.781±0.041
CVPR19 [14]	0.0307±0.0127	**0.0622±0.0101**	0.909±0.017
LRTCFPan [42]	0.0325±0.0269	0.0896±0.0141	0.881±0.023
PNN [25]	0.0317±0.0286	0.0943±0.0224	0.877±0.036
PanNet [44]	**0.0179±0.0110**	0.0799±0.0178	0.904±0.020
DiCNN [17]	0.0369±0.0132	0.0992±0.0131	0.868±0.016
FusionNet [11]	0.0350±0.0124	0.1013±0.0134	0.867±0.018
DCFNet [41]	0.0240±0.0115	0.0659±0.0096	0.912±0.012
MMNet [43]	0.0443±0.0298	0.1033±0.0129	0.857±0.027
LAGConv [19]	0.0284±0.0130	0.0792±0.0136	0.895±0.020
HMPNet [33]	0.0819±0.0499	0.1146±0.0126	0.813±0.049
Proposed	0.0194±0.0101	0.0630±0.0094	**0.919±0.011**

trained for 200 epochs, while all other parameters were kept the same as in the WV3 dataset.

Here we provide details regarding performing K-Means in CANConv. We select initial cluster centers using the K-Means++[3] method. We initialize the centers separately for different samples at different layers, because they capture distinct features (As show in Fig. 7). We stop iterating when less than 1% of cluster assignments are changed. In practice, it typically takes 20-25 iterations to converge.

8.3. Compared Methods

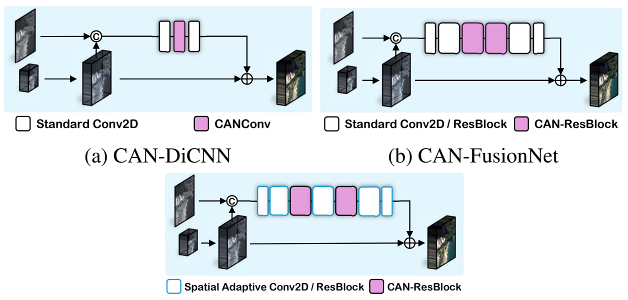
Tab. 7 provides a brief overview of pansharpening methods compared in the main text. We compare the proposed CANNet with both traditional and machine learning (ML) methods. We choose representative traditional methods from three categories including CS, VO and MRA. We also select classic and recent ML methods for benchmarking.

8.4. Replacing Standard Convolution

This section presents the details of the discussion experiment on replacing standard convolution. Fig. 8 shows which layers or blocks are replaced with their CANConv counterparts. In hyperparameter tuning, we increased the learning rate on CAN-DiCNN to 10^{-3} to foster faster convergence, and kept all other parameters the same as the original network. The number of clusters K was set to 32 and the threshold η was 0.005.

8.5. Additional Results

Tabs. 8 and 9 showcase performance benchmarks on the full-resolution QB and GF2 datasets. The HQNR metric [37] is an improvement upon the QNR metric. Combining assessments of both spatial and spectral consistency, HQNR provides a comprehensive reflection of the image-fusion effectiveness of different methods. It is considered one of the most important metrics on full-resolution datasets. In Figs. 9 to 16, we present visual output comparisons across various methods on sample images from the WV3, QB and GF2 datasets, including residuals between outputs and ground truth for reduced-resolution samples. The comparative analysis highlights that, overall, CANNet produces results closely aligned with the ground truth. Leveraging self-similarity information, CANNet excels in handling repetitive texture areas, surpassing the performance of previous methods, as shown in Fig. 15. Also, CANNet exhibits adaptive processing in detail-rich edge regions, resulting in more realistic and accurate outcomes.



□ Standard Conv2D
🟪 CANConv
□ Standard Conv2D / ResBlock
🟪 CAN-ResBlock
□ Spatial Adaptive Conv2D / ResBlock
🟪 CAN-ResBlock

(a) CAN-DiCNN (b) CAN-FusionNet

(c) CAN-LAGNet

Figure 8. Replacing standard convolution module with CANConv to leverage non-local self-similarity information. Highlighted modules are replaced with CANConv or CAN-ResBlock in the experiment.

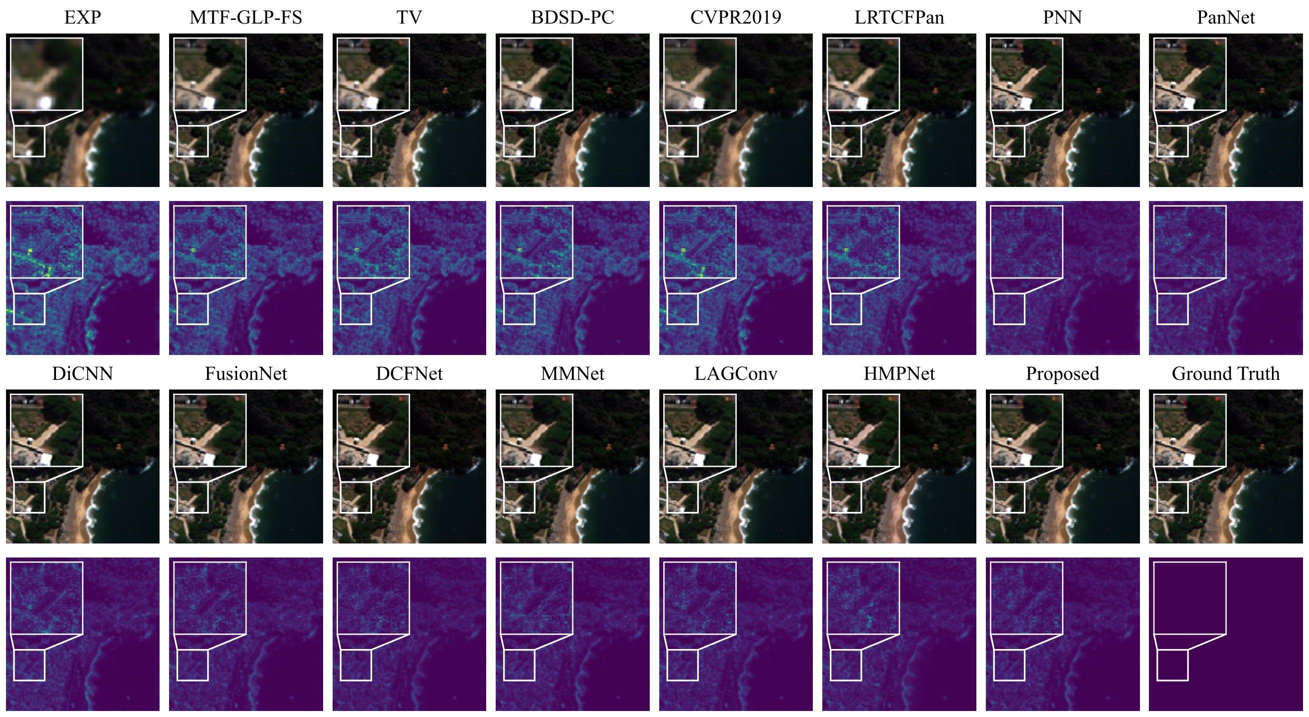


Figure 9. Qualitative result comparison between benchmarked methods on the sample image from WV3 reduced-resolution dataset. The first row presents RGB outputs, while the second row shows the residual compared to the ground truth.

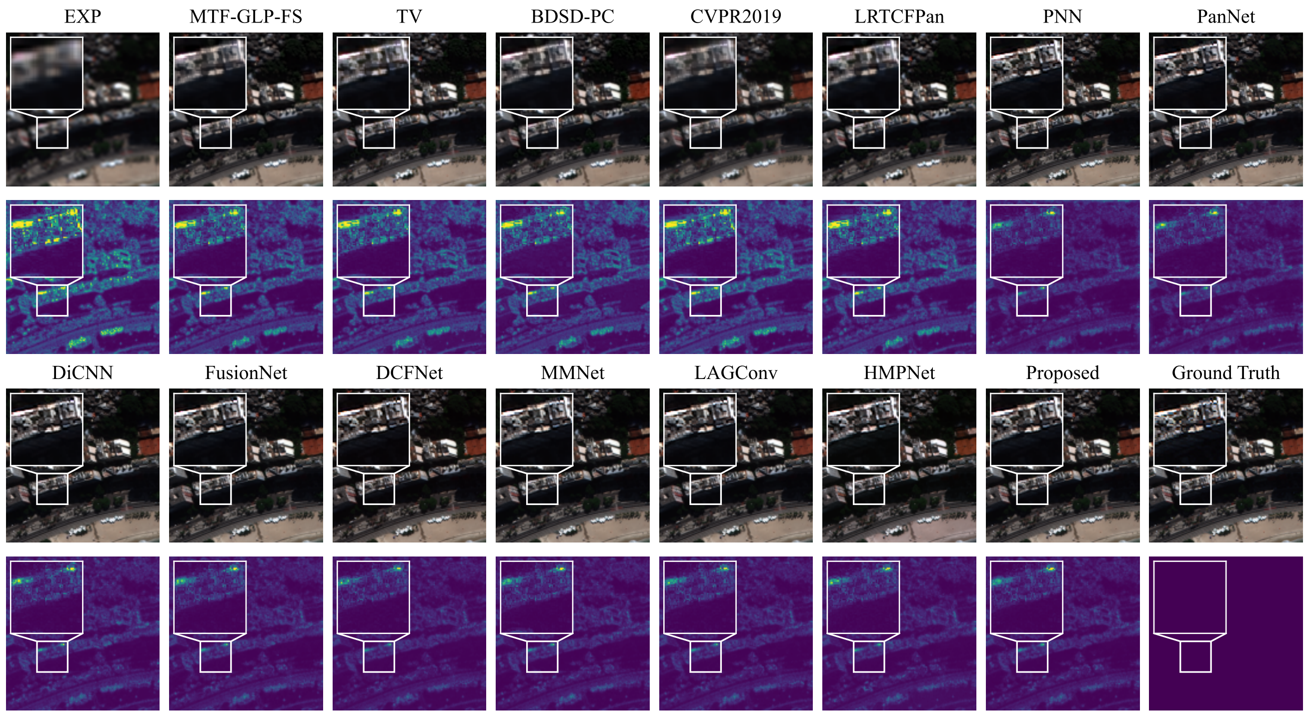


Figure 10. Qualitative result comparison between benchmarked methods on the sample image from WV3 reduced-resolution dataset.

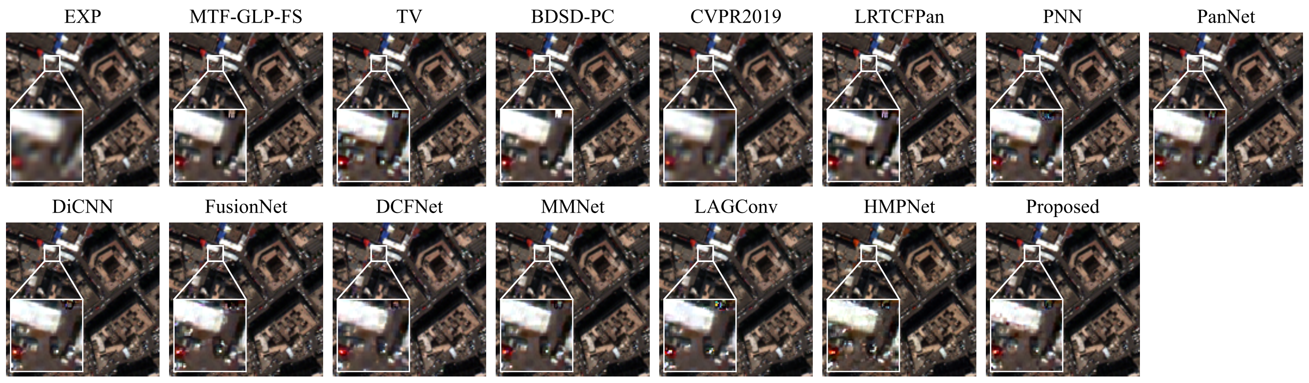


Figure 11. Qualitative result comparison between benchmarked methods on the sample image from the WV3 full-resolution dataset.

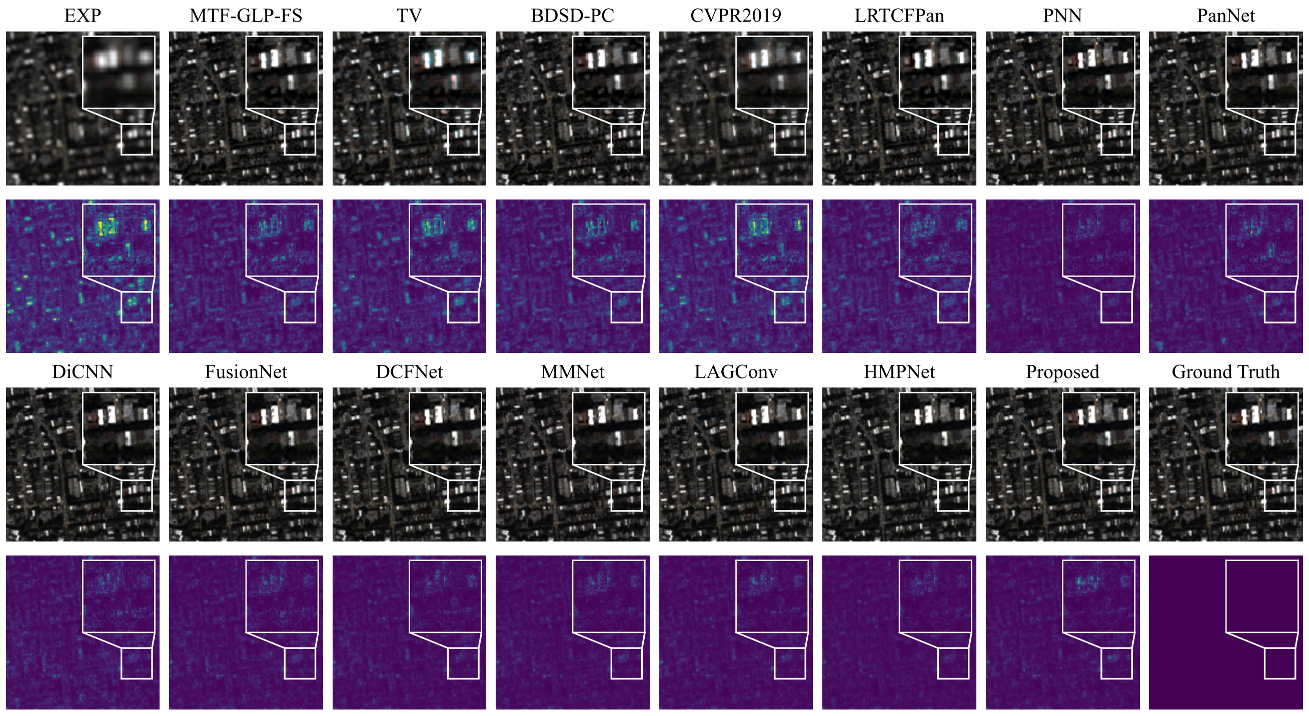


Figure 12. Qualitative result comparison between benchmarked methods on the sample image from the QB reduced-resolution dataset.

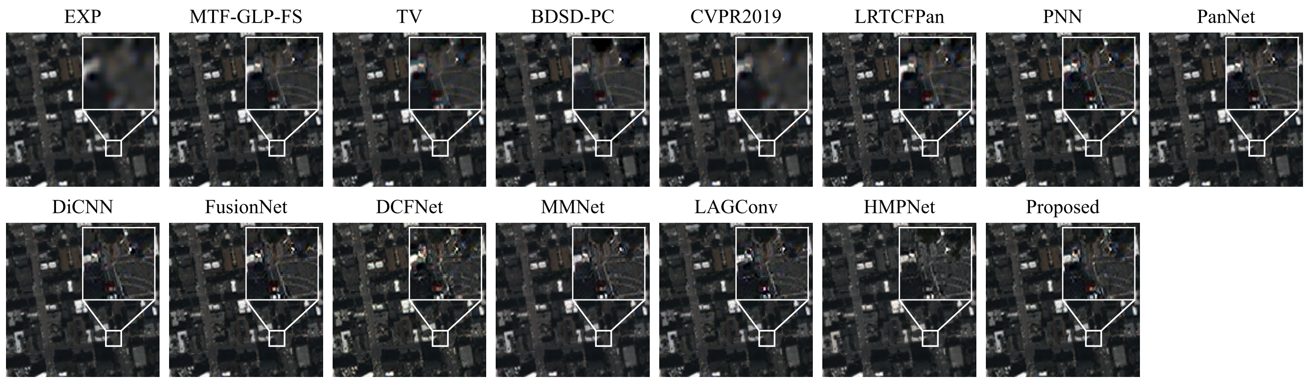


Figure 13. Qualitative result comparison between benchmarked methods on the sample image from the QB full-resolution dataset.

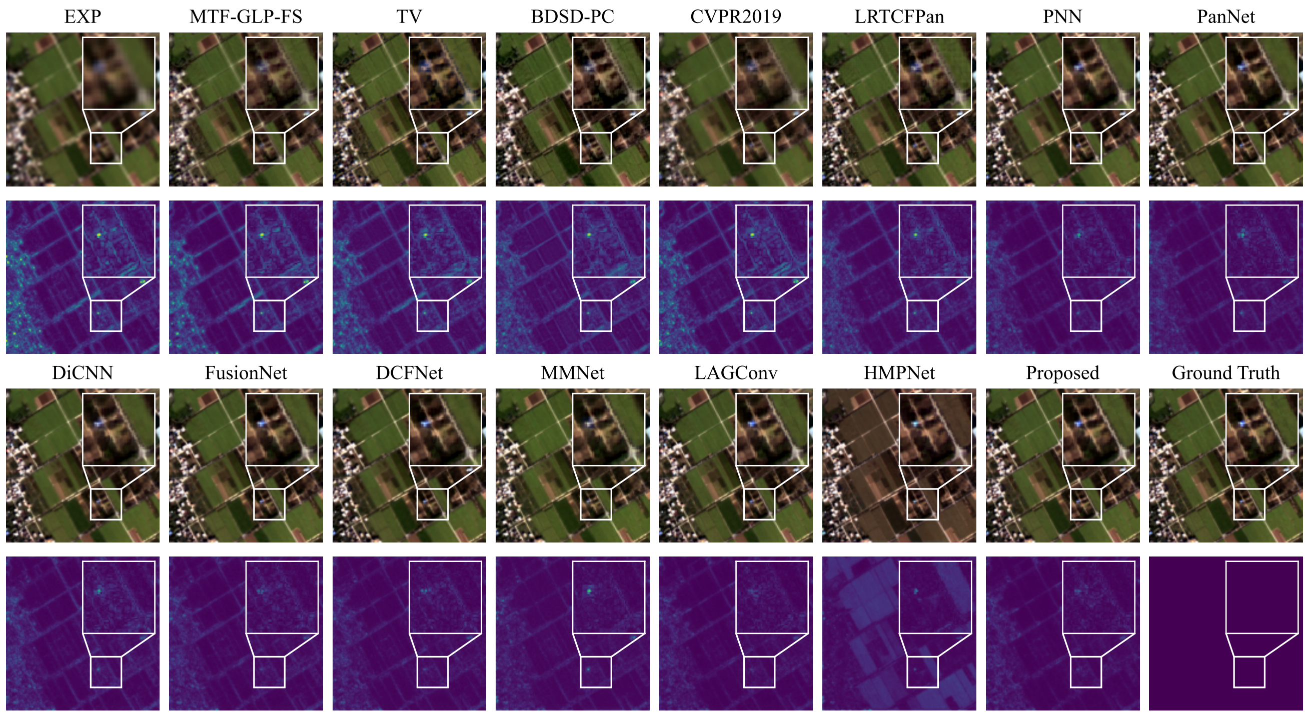


Figure 14. Qualitative result comparison between benchmarked methods on the sample image from the GF2 reduced-resolution dataset.

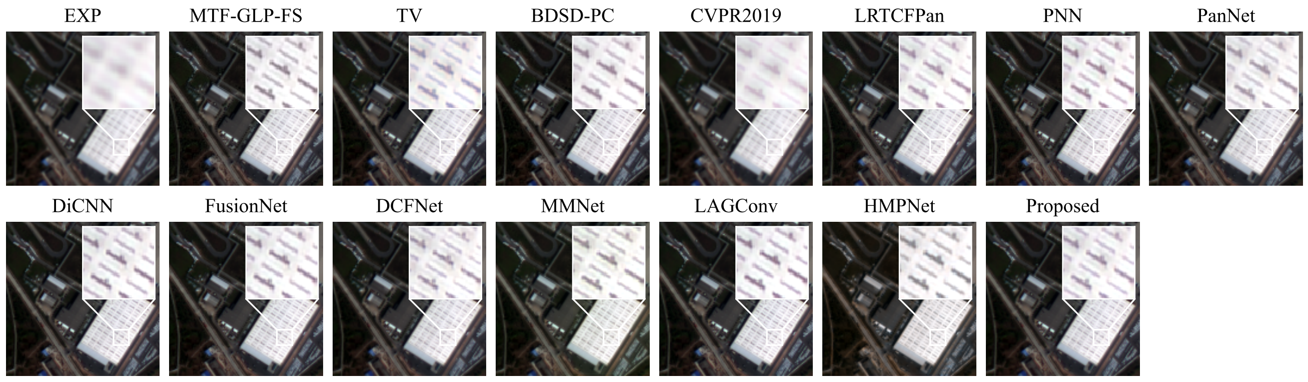


Figure 15. Qualitative result comparison between benchmarked methods on the sample image from the GF2 full-resolution dataset.

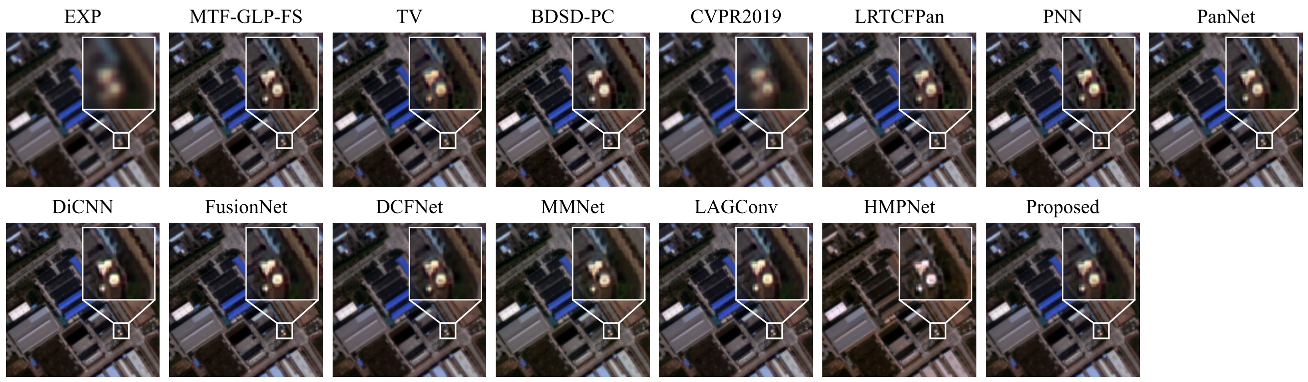


Figure 16. Qualitative result comparison between benchmarked methods on the sample image from the GF2 full-resolution dataset.