Fast Kernel-Space Diffusion for Remote Sensing Pansharpening

Supplementary Material

Abstract

In this supplementary material, we first provide a more detailed explanation of specific components of our proposed method that could not be thoroughly described in the main paper, including the network we use in our main experiments and further information on latent encoders. We then elaborate on the experimental settings and implementation details. Finally, we present additional experimental results, including generalization assessment on WorldView-2 dataset, further visualizations on the WorldView-3, GaoFen-2, and QuickBird datasets, as well as ablation studies on more factors.

6. Methods Explanation

6.1. Pansharpening Network

The baseline network we build is demonstrated in Fig. 7. The panchromatic (PAN) image $\mathbf{P} \in \mathbb{R}^{H\times W\times 1}$ is first duplicated along the channel dimension to match the number of channels in the low-resolution multispectral (LRMS) image $\mathbf{M} \in \mathbb{R}^{H\times W\times C}$. The duplicated PAN image is then subtracted by the LRMS image, and the resulting difference is fed into the network. After processing, the network output is added to the input LRMS image to produce the reconstructed high-resolution multispectral image $\mathbf{H} \in \mathbb{R}^{H\times W\times C}$, denoted as $\mathbf{H}_1$ or $\mathbf{H}_2$ as defined in the main text. To reduce spatial resolution and increase channel depth, downsampling layers are inserted between successive encoder blocks in the network. Conversely, in the decoder, upsampling operations with a scale factor of 2 are applied between adjacent layers to increase spatial resolution and reduce the number of channels. The skip connections adopt a concatenation-based strategy, similar to that of the standard U-Net [40]. Each block in the backbone corresponds to a ResBlock [14]. Within each block, only the 3×3 convolution layer is reparameterized using the proposed KSDiff method. Given the multi-scale nature of the network, the latent representation is appropriately rescaled before being injected into each convolution layer. Specifically, shallower layers, which operate on higher spatial resolutions, use latent representations with larger scales, whereas deeper layers employ downsampled versions of the generated latent representation. Detailed parameter settings are provided in Sec. 7.

6.2. PLFE$_2$

As shown in Fig. 8 (a), the structure of PLFE_2 is a simplified, halved version of PLFE_1, which is presented in Fig. 3 in the main text. All core components, including the cross-attention mechanism and the Fusion-Gate module, remain identical to those in PLFE_1. The role of PLFE_2 differs from PLFE_1. In PLFE_1, the module encodes the ground-truth , PAN, LRMS images to obtain a latent representation $\mathbf{z}_0 \in \mathbb{R}^{N\times C_z}$. In contrast, PLFE_2 produces a conditioning vector $\mathbf{c} \in \mathbb{R}^{N\times C_z}$ only with PAN and LRMS images as inputs, which is injected into the denoising network to guide the reverse diffusion process for latent reconstruction. The resulting latent representation is denoted as $\hat{\mathbf{z}}_0 \in \mathbb{R}^{N\times C_z}$, which is then fed into the kernel generator.

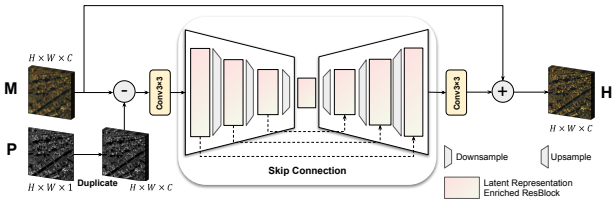

Figure 7. Baseline network.

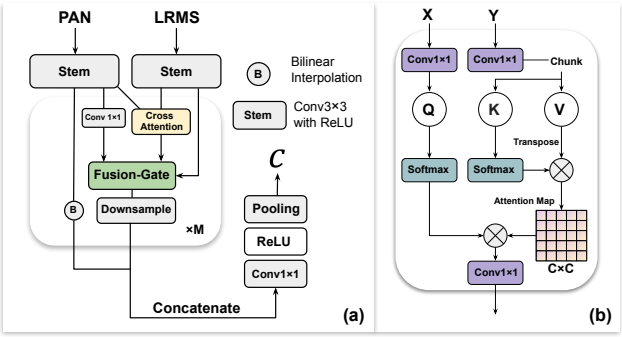

Figure 8. (a) PLFE_2. (b) The cross attention in latent encoders.

7. Details on Experiments

7.1. Datasets

We conducted experiments using datasets derived from WorldView-3 (WV3), QuickBird (QB), and GaoFen-2 (GF2) satellite imagery. These datasets consist of image patches cropped from full remote sensing scenes and are partitioned into training and testing subsets. The WV3 dataset contains four images from two geographic locations, Rio and Tripoli, captured in different seasons: Rio in January, Rio in May, Tripoli in March, and Tripoli in August. The right quarter of the Rio in May and Tripoli in August images was cropped to form the test set. The remaining areas of all images were

Table 7. Summary of compared pansharpening methods.

Method	Category	Year	Introduction
BDSD-PC [51]	CS	2019	Discusses the constraints of the band-dependent spatial-detail (BDSD) approach in images with over four spectral bands.
MTF-GLP-FS [53]	MRA	2018	An iterative algorithm for estimating injection coefficients at full resolution in regression-based pansharpening.
BT-H [1]	MRA	2006	A multiresolution pansharpening framework using a generalized Laplacian pyramid (GLP) with an MTF-adjusted filter to enhance spatial details while preserving spectral information.
PNN [33]	ML	2016	The first deep CNN architectures designed specifically for pansharpening.
DiCNN [15]	ML	2019	Introduces a detail-injection mechanism to enhance spatial detail learning in CNNs.
MSDCNN [58]	ML	2017	Employs multiscale and densely connected convolutional layers to extract both spatial and spectral features.
FusionNet [8]	ML	2020	A residual learning-based network that emphasizes detail preservation for fusion tasks.
CTINN [72]	ML	2022	Use a customized transformer architecture and an information-lossless invertible neural module to model long-range dependencies.
LAGConv [20]	ML	2022	Employ local-context adaptive convolution kernels and global harmonic bias to enhance image fusion.
MMNet [73]	ML	2023	A model-driven deep unfolding network with memory-augmentation.
DCFNet [60]	ML	2021	A dynamic cross feature fusion network for pansharpening that integrates multi-scale branches and contextualized feature transfers.
PanMamba [16]	ML	2025	A state-space model based method for pansharpening.
PanDiff [34]	Diffusion-based ML	2023	Employs a diffusion model to learn HRMS-IMS difference map distributions, guided by PAN/LRMS images with a modal intercalibration module.
PLRDiff [41]	Diffusion-based ML	2024	Proposes a low-rank diffusion model for pansharpening, combining a pre-trained diffusion model for image structure capture with a Bayesian approach for spectral preservation.

used for training. This configuration avoids any spatial overlap and incorporates seasonal variation, thereby preventing both spatial and temporal leakage. For QB and GF2 datasets, each consists of a single image (QB Indianapolis and GF2 Guangzhou, respectively). The rightmost 1,000 pixels were cropped to form the test sets, while the rest of the image was used for training. Since the cropped test areas do not overlap with the training regions, spatial independence is maintained. Temporal leakage is not applicable in these cases, as only single-date imagery is available.

The training data comprise simulated downsampling-based PAN/LRMS/GT triplets, with dimensions of $64 \times 64 \times 1$, $64 \times 64 \times C$, and $64 \times 64 \times C$, respectively. Specifically, the WV3 training set includes approximately 10,000 eight-channel samples ($C = 8$), while the GF2 and QB training sets contain around 20,000 and 17,000 four-channel samples ($C = 4$), respectively. For evaluation, the reduced-resolution test set for each satellite consists of 20 simulated PAN/LRMS/GT triplets representing diverse land cover types, with dimensions of $256 \times 256 \times 1$, $256 \times 256 \times C$, and $256 \times 256 \times C$, respectively. Additionally, a full-resolution test set is provided, comprising 20 original PAN/LRMS image pairs with spatial sizes of 512×512. All datasets and preprocessing procedures were obtained from the PanCollection repository [9].

7.2. Training Details

The proposed KSDiff framework is implemented using PyTorch 2.1.0 and Python 3.10. All experiments are conducted on a Linux system (Ubuntu 22.04) equipped with a single NVIDIA GeForce RTX 4090 GPU and CUDA version 12.1. We adopt the AdamW optimizer [31] with a learning rate of 0.8×10^{-4} and a weight decay of 1×10^{-4} for both the pre-training and diffusion training stages. The denoising diffusion process follows a cosine noise schedule [36], without learning the variance as proposed in [36]. The total number of diffusion timesteps is set to 500 across all datasets. An exponential moving average (EMA) with a decay rate of 0.995 is employed. For the pre-training stage, the number of training iterations is set to 60k for all datasets (WV3, GF2, and QB). In the diffusion model training stage, the iteration counts are set to 50k, 70k, and 10k for WV3, GF2, and QB, respectively. The batch size during pre-training is fixed at 64 for all datasets. For diffusion model training, batch sizes are set to 32 for WV3, and 16 for both GF2 and QB. For DDIM sampling [45], we use 25 sampling steps uniformly across the three datasets.

The denoising network consists of five MLP blocks utilizing GELU activation [17]. The conditioning of the diffusion model was implemented via feature concatenation [42]. The latent encoders PLFE_1 and PLFE_2 are designed with three pyramid stages. The latent representation $\mathbf{z}$ has dimensions $N = 16$ and $C_z = 128$. The pansharpening network employs a U-Net [40] architecture with four encoder and four decoder blocks. The encoder begins with 32 channels, which are scaled by factors of 1, 2, 2, and 4 across successive blocks. The dimension of the latent representation N is downsampled by the same factors before integration into kernel generation. For the kernel generator, the low-rank core tensor has dimensions (r_1, r_2, r_3, r_4) set to $(4, 4, 2, 2)$ for the first three blocks and $(8, 8, 2, 2)$ for the fourth block.

7.3. Compared Methods

Table 7 provides a brief overview of the pansharpening methods compared in the main text. We compare the proposed KSDiff with both traditional and recent machine learning (ML) approaches. Three traditional methods are considered: BDSD-PC [51], MTF-GLP-FS [53], and BT-H [1]. Nine machine learning methods without diffusion mechanisms are also included: PNN [33], DiCNN [15], MSDCNN [58], FusionNet [8], CTINN [72], LAGConv [20], MMNet [73], DCFNet [60], and PanMamba [16]. Additionally, we include two recent diffusion-based ML methods: PanDiff [34] and PLRDiff [41]. The runtimes of these methods (as reported in Tab. 1 in the main text) are evaluated and compared using the WV3 reduced-resolution dataset.

7.4. Ablation Study

Continuing from Tab. 4 in the main text, we provide the detailed implementation of our ablation studies. The experiment was implemented on the WV3 reduced-resolution dataset. To assess the impact of the latent diffusion prior, we trained the baseline U-Net independently for 100,000 iterations using an ℓ_1 loss function and a batch size of 64. For the ablation of our proposed PLFE module, we replaced the latent encoder with the one adopted in DiffIR [63]. In this setup, the PAN, LRMS, and GT images (for the pre-training stage), or the PAN and LRMS images (for the diffusion model training stage), were directly concatenated at the input of the encoder. The alternative encoder comprises six residual blocks. All ablation experiments were trained until full convergence. Other training configurations, including the optimizer, learning rate, and weight decay, were kept consistent with those used in the main experiments, as illustrated in Sec. 7.2.

7.5. Other Backbones

This section provides the implementation details of the experiments discussed in Sec. 4.4 in the main text, where we investigate the effect of replacing standard convolutional operations with our proposed KSDiff kernel generation pipeline in other backbone networks. The experiment was implemented on the WV3 reduced-resolution dataset. For the DiCNN model [15], which comprises a three-layer convolutional neural network with an inner channel dimension of 64, we modulated the middle convolution layer using the KSDiff mechanism. In the case of FusionNet [8], which contains four standard ResBlocks [14] with 32 inner channels each, we replaced the standard convolutional layers within the ResBlocks with their KSDiff-enhanced counterparts. For LAGNet [20], which already incorporates adaptive convolution kernels, we substituted the kernels in its adaptive convolution modules with diffusion-prior-enhanced versions generated by KSDiff. For all three models, the batch size was consistently set to 64 for both the pre-training and diffusion model training stages. The size of the latent representation $\mathbf{z}$ is $N = 16, C_z = 32$. The low-rank core tensor size (r_1, r_2, r_3, r_4) was fixed at $(4, 4, 2, 2)$, and the learning rate was set to 1×10^{-4}. All experiments were trained until full convergence.

7.6. Core Tensor Size

This section provides the detailed configuration of the experiments investigating the impact of low-rank core tensor sizes, as discussed in Sec. 4.4 in the main text. The experiment was implemented on the WV3 reduced-resolution dataset. The inner channel dimension of FusionNet [8] is fixed at 32, and all convolutional kernels have a size of $k = 3$, making it a suitable and efficient setting for controlled comparative experiments. All configurations presented in the main text

were trained for 100,000 iterations in both the pre-training stage and the diffusion model training stage, with a batch size of 64 and a learning rate of 1×10^{-4}. The experimental results highlight the advantageous properties of leveraging low-rank representations and the high-dimensional structure of 4D tensors. We leave the exploration of these properties across a broader range of tasks, architectures, and parameter settings as a direction for future work.

Table 8. Result on the GF2 full-resolution dataset. The best results are highlighted in bold and the second best results are underlined.

Method	GaoFen-2		
	D_λ ($\pm$ std)	D_s ($\pm$ std)	HQNR ($\pm$ std)
BDSD-PC [51]	0.0759±0.0067	0.1548±0.0063	0.7812±0.0091
MTF-GLP-FS [53]	0.0336±0.0029	0.1404±0.0062	0.8309±0.0075
BT-H [1]	0.0602±0.0043	0.1313±0.0043	0.8165±0.0068
PNN [33]	0.0367±0.0065	0.0943±0.0050	0.8726±0.0083
DiCNN [15]	0.0413±0.0029	0.0992±0.0029	0.8636±0.0037
MSDCNN [58]	0.0269±0.0029	0.0730±0.0021	0.9020±0.0029
FusionNet [8]	0.0400±0.0028	0.1013±0.0030	0.8628±0.0041
CTINN [72]	0.0586±0.0058	0.1996±0.0033	0.8381±0.0053
LAGConv [20]	0.0324±0.0029	0.0792±0.0030	0.8910±0.0046
MMNet [73]	0.0428±0.0067	0.1033±0.0029	0.8583±0.0060
DCFNet [60]	0.0234±0.0026	0.0659±0.0021	0.9122±0.0027
PanMamba [16]	**0.0231±0.0025**	**0.0572±0.0023**	0.9210±0.0028
PanDiff [34]	0.0265±0.0044	0.0729±0.0023	0.9025±0.0047
PLRDiff [41]	0.2804±0.0292	0.1413±0.0057	0.6164±0.0234
KSDiff (ours)	0.0233±0.0028	0.0588±0.0023	**0.9257±0.0024**
Ideal value	**0**	**0**	**1**

Table 9. Result on the QB full-resolution dataset. The best results are highlighted in bold and the second best results are underlined.

Method	QuickBird		
	D_λ ($\pm$ std)	D_s ($\pm$ std)	HQNR ($\pm$ std)
BDSD-PC [51]	0.1975±0.0075	0.1636±0.0108	0.6722±0.0129
MTF-GLP-FS [53]	0.0489±0.0033	0.1383±0.0053	0.8199±0.0076
BT-H [1]	0.2300±0.0161	0.1648±0.0037	0.6434±0.0144
PNN [33]	0.0569±0.0025	0.0624±0.0053	0.8844±0.0068
DiCNN [15]	0.0920±0.0032	0.1067±0.0047	0.8114±0.0069
MSDCNN [58]	0.0602±0.0034	0.0667±0.0065	0.8774±0.0087
FusionNet [8]	0.0586±0.0042	0.0522±0.0020	0.8922±0.0049
CTINN [72]	0.1738±0.0074	0.0731±0.0053	0.7663±0.0097
LAGConv [20]	0.0844±0.0053	0.0676±0.0030	0.8536±0.0040
MMNet [73]	0.0890±0.0114	0.0972±0.0085	0.8225±0.0071
DCFNet [60]	0.0454±0.0033	0.1239±0.0060	0.8360±0.0035
PanMamba [16]	0.0491±0.0028	0.0443±0.0033	0.9102±0.0029
PanDiff [34]	0.0587±0.0039	0.0642±0.0056	0.8813±0.0093
PLRDiff [41]	0.7775±0.0233	0.3038±0.0204	0.1615±0.0200
KSDiff (ours)	**0.0380±0.0027**	**0.0426±0.0029**	**0.9130±0.0026**
Ideal value	**0**	**0**	**1**

8. Additional Results

8.1. Main Results

Tab. 8 and Tab. 9 present the quantitative performance benchmarks on the full-resolution GF2 and QB datasets. The results indicate that the proposed KSDiff method exhibits strong generalization capabilities across different data domains. Fig. 9 to Fig. 14 provide qualitative comparisons of visual outputs generated by various methods on representative samples from the WV3, GF2, and QB datasets. For the reduced-resolution data, corresponding error maps between the predicted outputs and ground-truth references are also included. The visual and quantitative comparisons consistently demonstrate that KSDiff produces results that are highly consistent with the ground-truth. Furthermore, the method shows robust performance in scenarios lacking reference images, attributed to the diffusion model's powerful distribution estimation capability.

Table 10. Generalization of DL-based methods on WV2 dataset.

Method	WorldView-2			
	SAM ($\pm$ std)	ERGAS ($\pm$ std)	$Q2^n$ ($\pm$ std)	SCC ($\pm$ std)
PNN [33]	7.1158±0.3758	5.6152±0.2108	0.7619±0.0207	0.8782±0.0039
DiCNN [15]	6.9216±0.1766	6.2507±0.1285	0.7205±0.0167	0.8552±0.0065
MSDCNN [58]	6.0064±0.1425	4.7438±0.1104	0.8241±0.0179	0.8972±0.0024
FusionNet [8]	6.4257±0.1923	5.1363±0.1152	0.7961±0.0165	0.8746±0.0030
CTINN [72]	6.4103±0.1331	4.6435±0.0847	0.8172±0.0195	0.9147±0.0023
LAGConv [20]	6.9545±0.1059	5.3262±0.0712	0.8054±0.0187	0.9125±0.0023
MMNet [73]	6.6109±0.0717	5.2213±0.0477	0.8143±0.0177	0.9136±0.0045
DCFNet [60]	5.6194±0.1350	4.4887±0.0841	0.8292±0.0182	0.9154±0.0019
KSDiff (ours)	**5.1944±0.1197**	**4.1052±0.0796**	**0.8485±0.0185**	**0.9288±0.0017**
Ideal value	**0**	**0**	**1**	**1**

Table 11. Ablation study on the trade-off hyperparameter λ in the loss function of the diffusion model training stage.

λ	WorldView-3			
	SAM$\downarrow$ ($\pm$ std)	ERGAS$\downarrow$ ($\pm$ std)	Qn$\uparrow$ ($\pm$ std)	SCC$\uparrow$ ($\pm$ std)
0.1	2.8232±0.1167	2.0858±0.1021	0.9167±0.0186	0.9870±0.0010
0.2	2.8159±0.1155	2.0801±0.1006	0.9168±0.0188	0.9871±0.0009
1	2.8102±0.1147	2.0756±0.0973	0.9221±0.0183	0.9870±0.0010
5	2.8078±0.1147	2.0790±0.1015	0.9203±0.0185	0.9870±0.0010
10	2.8106±0.1149	2.0877±0.1016	0.9189±0.0190	0.9869±0.0010
Ideal value	**0**	**0**	**1**	**1**

8.2. Generalization

To assess the generalization capability of deep learning–based methods, we evaluated models trained on the WV3 dataset using 20 reduced resolutions from the WorldView-2 dataset. As shown in Tab. 10, the quantitative results reveal that the KSDiff method consistently outperforms others across all four evaluation metrics, demonstrating the strong generalization ability of our approach.

8.3. Effect of Hyperparameter λ

For the total loss design in the diffusion model training stage, $\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda \mathcal{L}_{\text{reg}}$, we initially set $\lambda = 1$, which was found to be empirically effective. Throughout our experiments, no issues such as gradient collapse were observed. To further examine the influence of λ, we conducted an ablation study in which all models were trained until full convergence. The results, summarized in Tab. 11, demonstrate that the overall performance remains relatively consistent across a range of λ values, indicating that our method is not particularly sensitive to this hyperparameter.

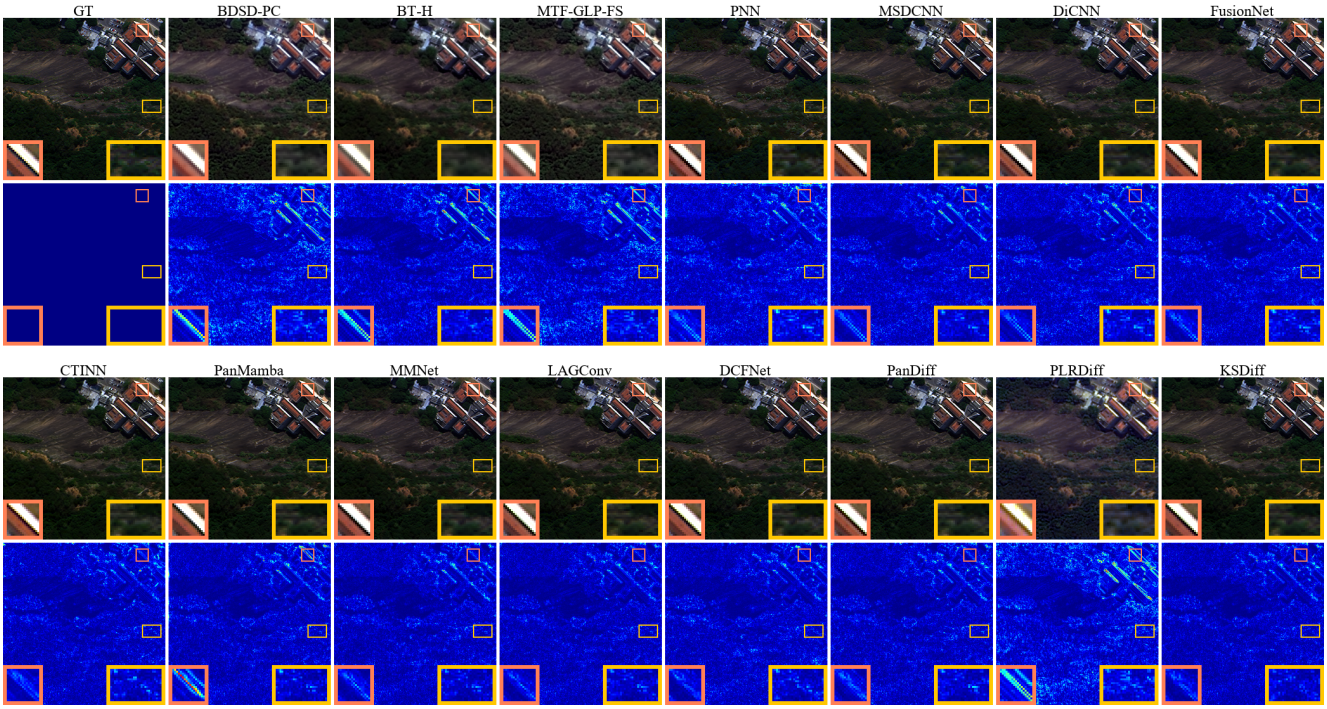

Figure 9. Comparison of qualitative results for representative methods on the WV3 reduced-resolution dataset.

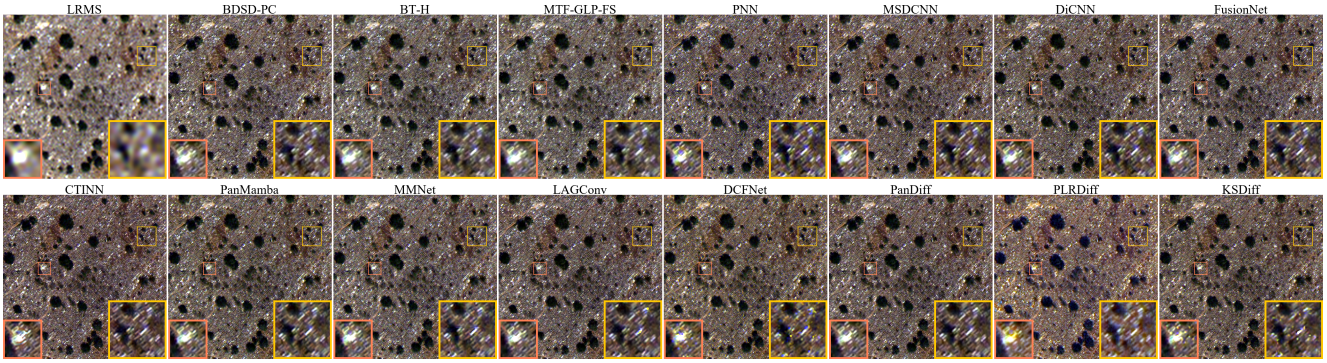

Figure 10. Comparison of qualitative results for representative methods on the WV3 full-resolution dataset.

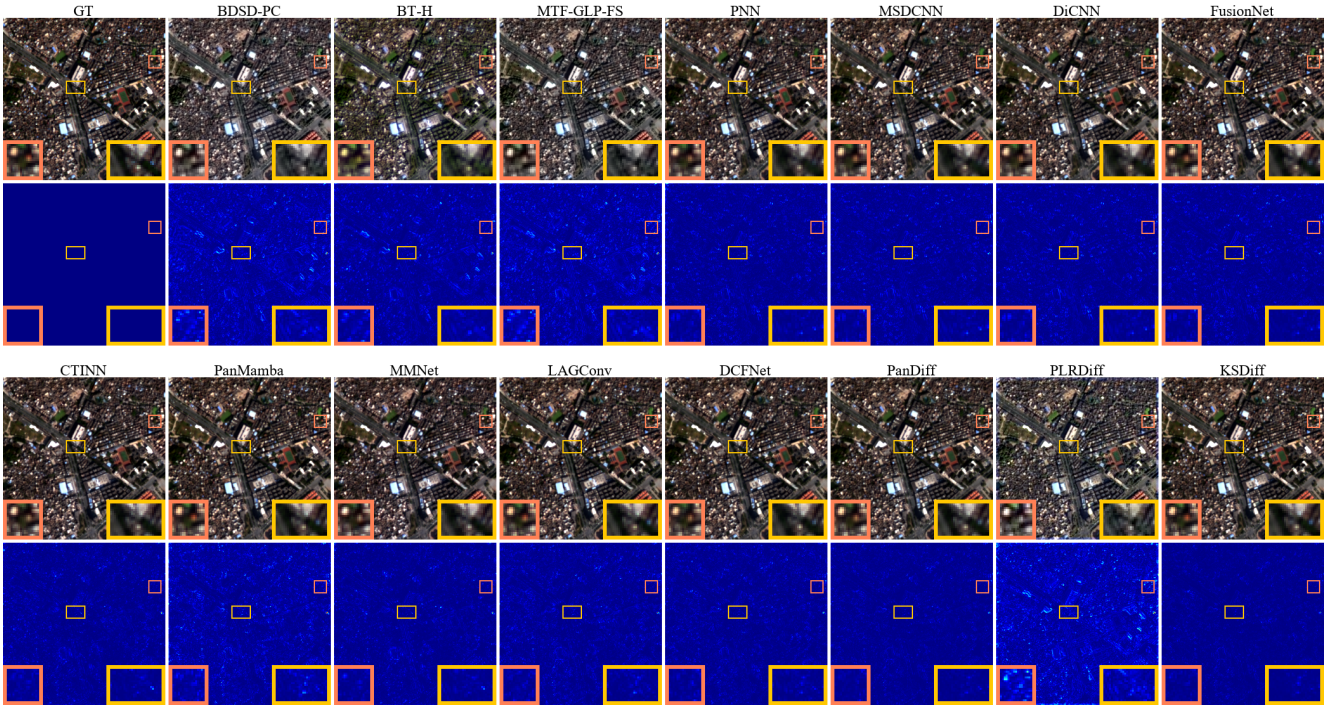

Figure 11. Comparison of qualitative results for representative methods on the GF2 reduced-resolution dataset.

Figure 12. Comparison of qualitative results for representative methods on the GF2 full-resolution dataset.

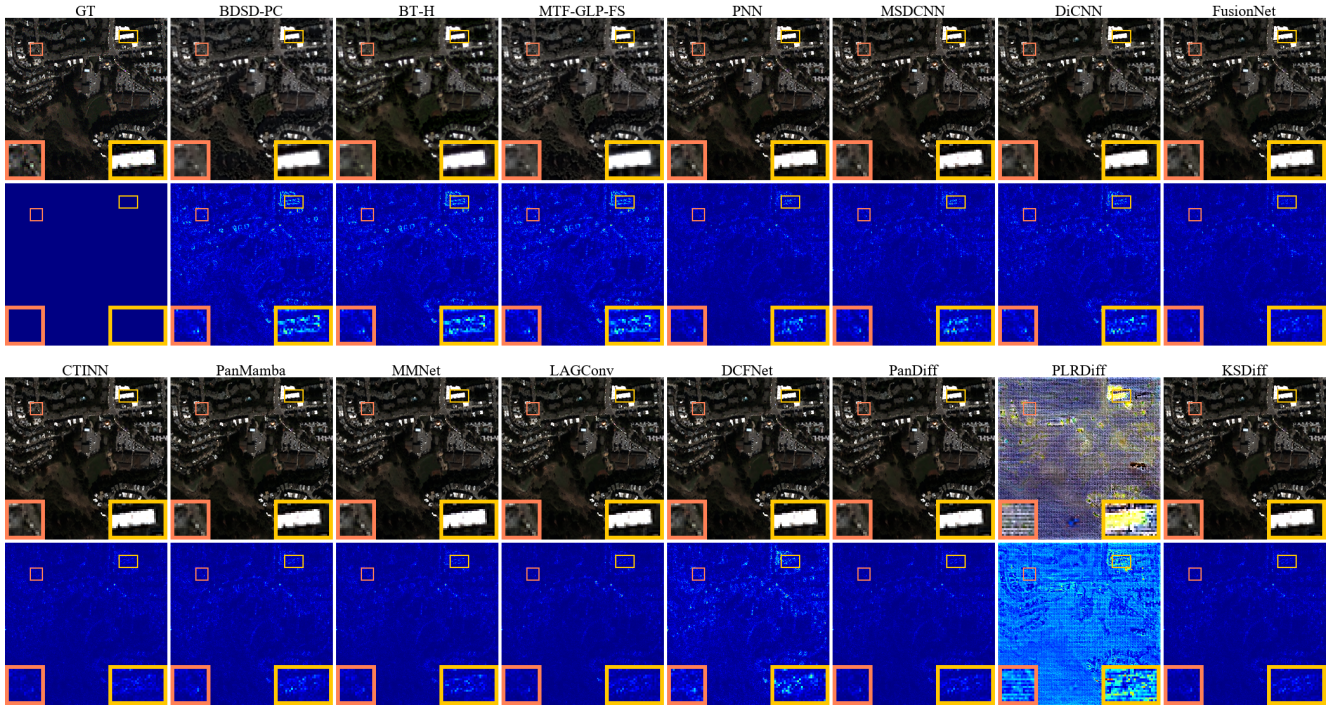

Figure 13. Comparison of qualitative results for representative methods on the QB reduced-resolution dataset.

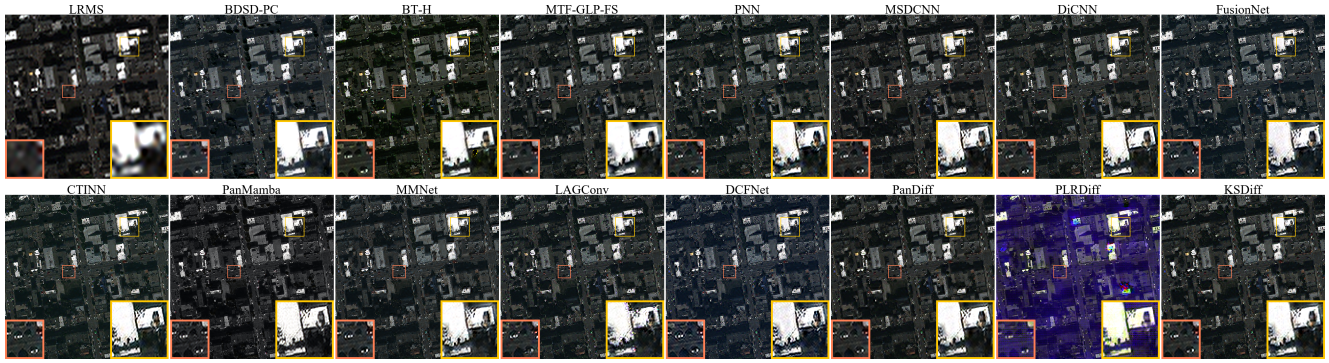

Figure 14. Comparison of qualitative results for representative methods on the QB full-resolution dataset.