SpanConv: A New Convolution via Spanning Kernel Space for Lightweight Pansharpening

Zhi-Xuan Chen[1] , **Cheng Jin**[1] , **Tian-Jing Zhang**[1] , **Xiao Wu**[1] and **Liang-Jian Deng**[1*]

[1]University of Electronic Science and Technology of China, Chengdu, 611731

{zhixuan.chen, cheng.jin}@std.uestc.edu.cn, zhangtianjinguestc@163.com, wxwsx1997@gmail.com, liangjian.deng@uestc.edu.cn

In this supplementary, We begin with a description of the particular methods required to generate SpanKernel. Following that, additional specifics about the training's implementation are offered. Additionally, we present a visual comparison using the WorldView-3 and QuickBird datasets and illustrate the potential to generalize using the WorldView-2 datasets. Finally, a discussion of function loss and ablation studies is conducted.

1 Generation detail of SpanKernel

The detailed procedure is listed as follows:

1. Set two independent kernels, dubbed as navigated kernel and respectively denoted as $\bar{K}_1 \in \mathbb{R}^{k\times k}$ and $\bar{K}_2 \in \mathbb{R}^{k\times k}$, where the two kernels are learned by the network training.
2. For each navigated kernel $\bar{K}_i,\ i = 1, 2$, learn C_{in} coefficients to extend the $\bar{K}_i$ to C_{in} channels for the convolution.
3. Span the two set of navigated kernels with learned coefficients to generate the final SpanKernel, *i.e.,*

$$F^{(j)} = \lambda_1^{(j)} \cdot \bar{K}_1 + \lambda_2^{(j)} \cdot \bar{K}_2,\ j = 1, \cdots, C_{in}, \quad (1)$$

where $F^{(j)} \in \mathbb{R}^{k\times k}$ denotes the j-th extended kernel, and $\lambda_i^{(j)},\ i = 1, 2$ represents the coefficients of navigated kernels.

2 Training Details

To ensure a fair comparison, all deep learning models are implemented and tested on the same datasets using Python 3.8.5 and their respective training frameworks, i.e., PyTorch 1.10 [Paszke *et al.*, 2019] and TensorFlow 1.14 [Abadi *et al.*,] on a desktop computer running Ubuntu 20.04 and two NVIDIA GeForce GTX 2080 GPUs. In summary, DiCNN, PanNet, LPPN, and FusionNet are trained using the TensorFlow framework, whereas the remaining networks use the PyTorch framework. For LightNet, we set the kernel size as 3×3 and 800 epochs for network training. The learning rate is initially set to 0.0025 and is multiplied by 0.75 per 120 epochs. For training, the Adam optimizer and the ℓ_1 loss function are used with a batch size of 8, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The training hyper-parameters are empirically calculated.

*Contact Author

3 Visual Comparison

This section compares the state-of-the-art approaches to our proposed LightNet visually. We begin with the WorldView-3 and QuickBird datasets and then evaluate the generalization performance of a trained WorldView-3 model on WorldView-2 datasets. It is worth noting that we select the model with the best performance among the saved ones, as the model from the latest epoch may not perform optimally.

3.1 Reduced Resolution Assessment

We test the compared methods on a large dataset of 130 test cases extracted from both WorldView-3 and QuickBird data. Among them, we depict a typical result for each dataset in Figs. 2, 3. As seen in the building area at the lower left region of WorldView-3 data and the parking lot area at the right middle region, BT-H, BDSD-PC, and MTF-GLP-FS all exhibit distinct blurring flaws. Additionally, the other deep learning-based approaches, such as DRPNN, MSDCNN, BDPN, DiCNN, and PNN, exhibit modest spectral distortion. The remaining deep learning-based algorithms achieve comparable visual performance. Additionally, we provide residual maps between the methodologies described above and photographs of Ground Truth (GT) to facilitate visual comparison. As can be observed, our method achieves similar results to state-of-the-art methods while requiring much less computing parameters.

3.2 Full Resolution Assessment

We also exhibit the outcomes of full-resolution test in Fig. 1, which are generated on WorldView-3.As illustrated in the pictures, conventional methods provide a more spatially accurate representation, whose boundaries are distinct, and spectral distortion is minimal. For DL-based approaches, BDPN method with rich number of parameters performs well. With fewer parameters, the Proposed LightNet achieves comparable outcomes to the aforementioned networks.

3.3 Generalization Ability

For generalization ability, we utilize WorldView-2 datasets, whose spectral number is identical to the WorldView-3 datasets. The datasets can be downloaded from the same link as the WorldView-3 datasets. From the outcomes, we can observe that BDPN, DRPNN, MSDCNN methods have spectral distortions, especially surrounding the road and building in

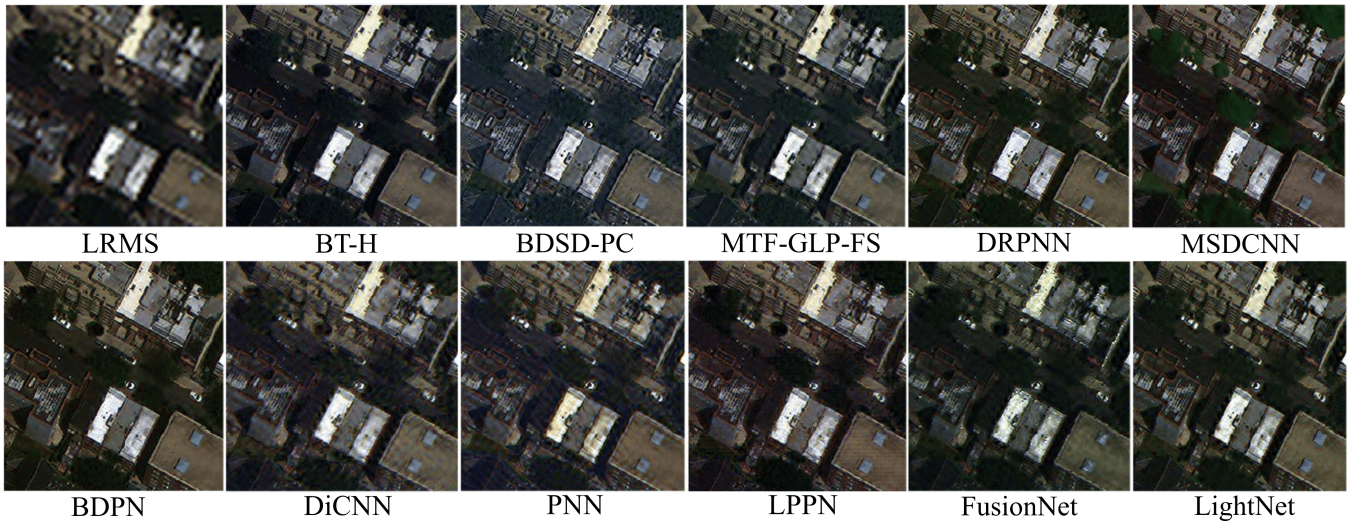

Figure 1: The visual comparisons on a full resolution WorldView-3 case (depicted bands: 1, 3 and 5). The fusion results are listed by means of BT-H, BDSD-PC, MTF-GLP-FS, BDPN, DiCNN, PNN, LPPN, FusionNet and proposed LightNet.

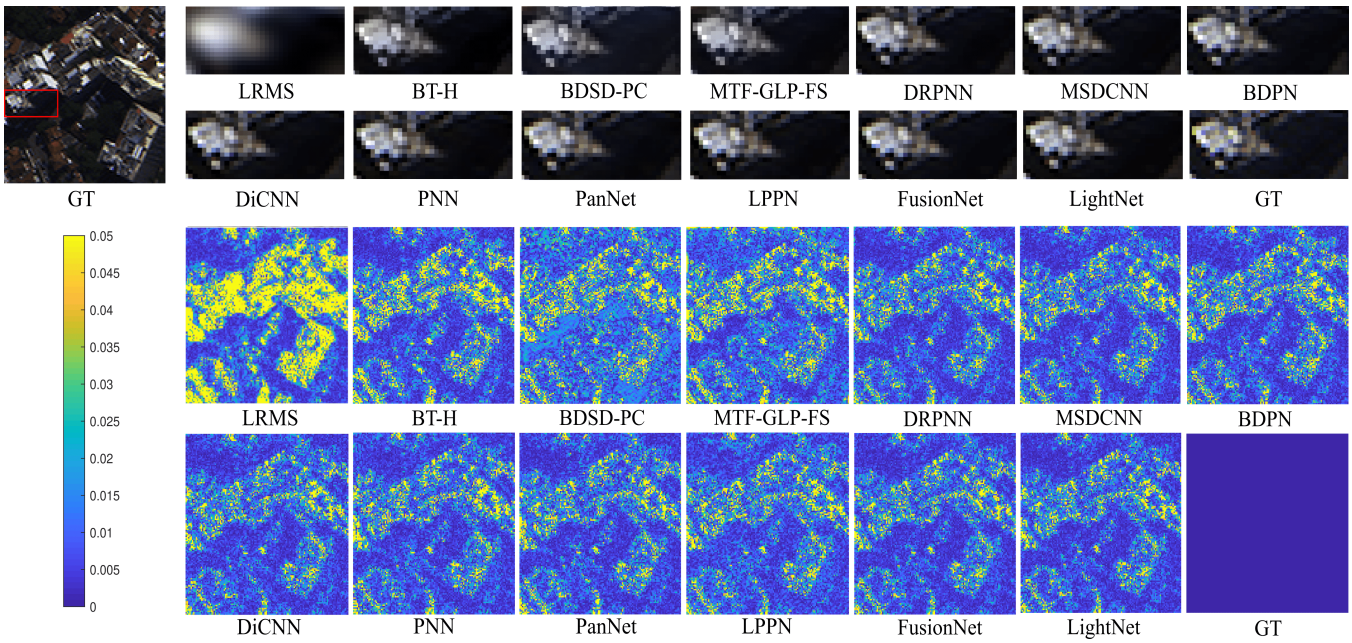

Figure 2: The visual comparisons on a reduced resolution WorldView-3 case (depicted bands: 1, 3 and 5). First two rows: The fusion results by means of BT-H, BDSD-PC, MTF-GLP-FS, BDPN, DiCNN, PNN, PanNet, LPPN, FusionNet and proposed LightNet. Third and fourth rows: The corresponding residual maps using the GT image as reference. To aid the visual inspection, we display the residual maps obtained by the analysis of the second spectral band.

the upper left region. DiCNN, PNN, and LPPN have various blurring effects. At last, our method holds the competitive results, for it has a closer appearance at the GT image.

4 Discussion on Loss Function

Additionally, we investigate the effect of various loss functions on model training. Tab. 1 loss displays the results. As a consequence, the model trained with the ℓ_2 loss function performs slightly worse than models trained with other loss functions. While the performance of ℓ_1 and SSIM is comparable, we opt for the more often used ℓ_1 function as our training loss function.

5 Ablation Study

In this section, we conduct the ablation study on the LightNet. The experiments in this section are all tested in the reduced resolution WorldView-3 testing data.

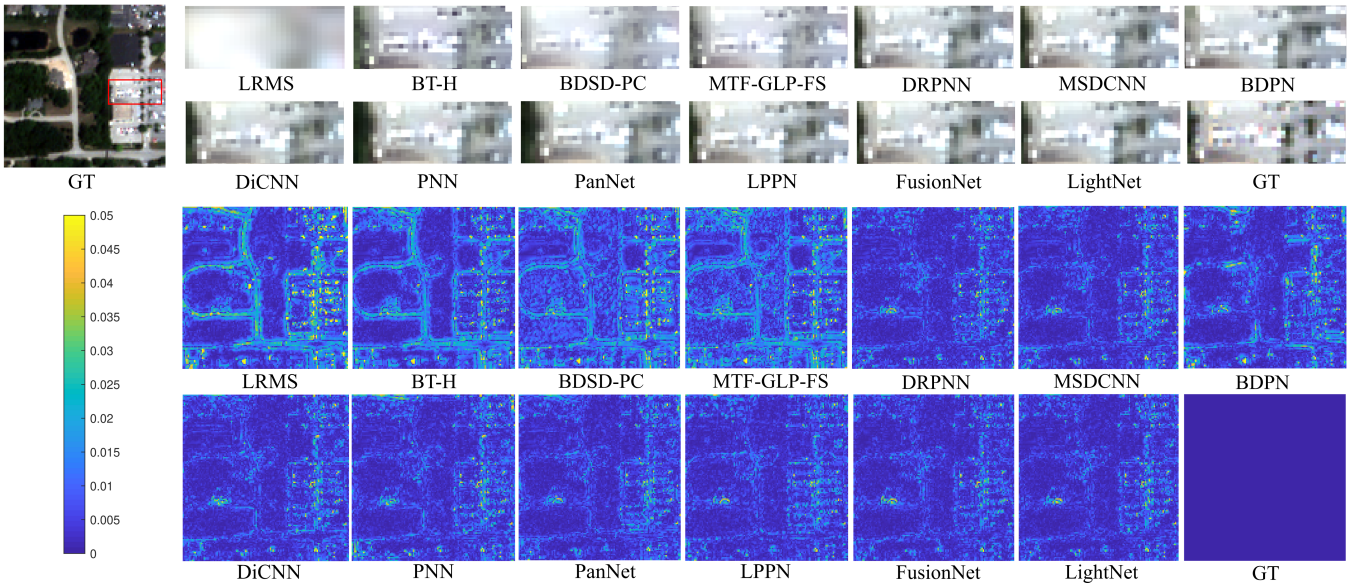

Figure 3: The visual comparisons on a reduced resolution QuickBird case (depicted bands: 1, 2 and 3). First two rows: The fusion results by means of BT-H, BDSD-PC, MTF-GLP-FS, BDPN, DiCNN, PNN, PanNet, LPPN, FusionNet and proposed LightNet. Third and fourth rows: The corresponding residual maps using the GT image as reference. To aid the visual inspection, we display the residual maps obtained by the analysis of the first spectral band.

Table 1: The comparison of LightNet trained with different loss functions.

Loss Function	Q8	SAM	ERGAS	Params
ℓ_1 Loss	0.919±0.053	4.897±0.792	**2.901±0.554**	**16K**
ℓ_2 Loss	0.910±0.062	4.980±0.789	2.947±0.538	**16K**
SSIM	**0.921±0.052**	**4.867±0.791**	2.917±0.573	**16K**

5.1 Block Number

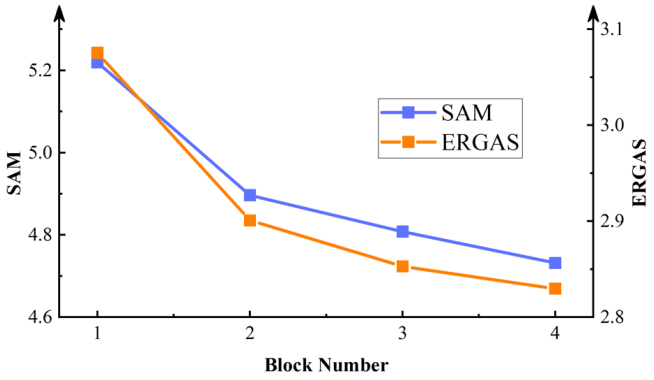

Figure 5: The performance of the LightNet with increasing block number in the reduced resolution WV3 testing data.

We study the effect of the block number in the extraction belly part to the LightNet. The results are reported in Fig. 5. According to the results, the performance of the LightNet continues to improve with more extraction belly blocks, and the improvement of the LightNet from 1 block to 2 blocks is prominent. Obviously, the more blocks means larger parameters. Considering the effect and parameters, we choose 2 blocks as our default architecture.

5.2 The Ratio of the LightNet

We also research the different ratio on the performance of the LightNet. The results are displayed in Fig. 6. The "1x" indicates that the channel count should be increased by a factor of 1 in comparison to the intended original network, and other cases can be analogized. As the network depth increases, the performance of the LightNet continues to improve. When the ratio factor is increased from 0.5 to 1, the performance of LightNet significantly increases and eventually stabilizes. For this result, we set the channel number as proposed network.

5.3 Replacement of the SpanConv

For proving the effectiveness of the SpanConv in the LightNet, we replace the SpanConv in different positions of the LightNet with standard convolution. The results are displayed in Tab. 2. The first column indicates the location of the replacement, whereas "**none**" refers to our original architecture. When the SpanConv in the fusion head portion is replaced, the performance is comparable to the original design. This implies that the conventional convolution in the shallow layers must be represented by SpanConv in general. However, when the reconstruction tail section is replaced, LightNet's performance degrades. We believe this is because standard convolution hardly learns highly linearly related kernels. For detail, please refer to Sec. 4.5 of the main text, in which we discuss the condition of the correlation between kernels becomes prominent in the reconstruction tail part of the LightNet. SpanKernel benefits from representing standard kernels. Standard convolution, on the other hand, is

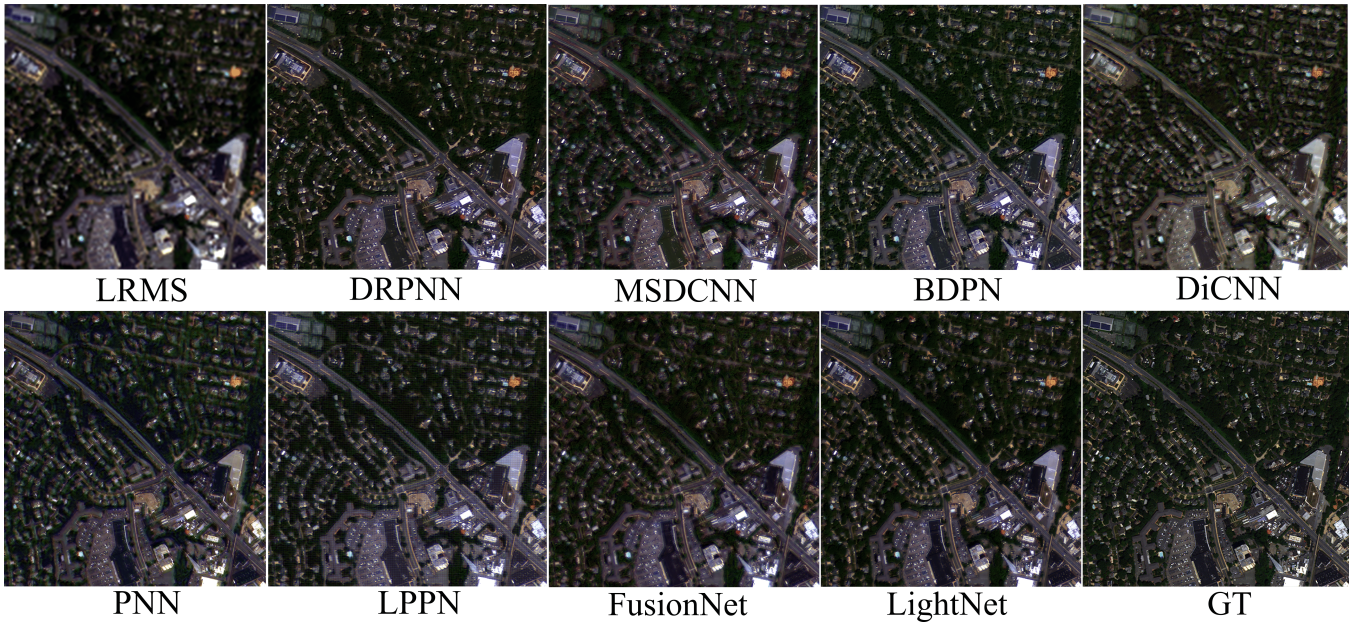

Figure 4: The visual comparisons on a reduced resolution WorldView-2 case (depicted bands: 1, 3 and 5). The fusion results are listed by means of BDPN, DiCNN, PNN, LPPN, FusionNet and proposed LightNet.

unable to properly train a collection of strongly linearly connected kernels when restricted to big parameters, resulting in poor performance.

Table 2: The comparison of replacing the SpanConv in the different position of the LightNet with standard convolution.

Replace Position	Q8	SAM	ERGAS	Params
head+belly+tail	0.918±0.054	5.006±0.810	2.924±0.539	52K
head	**0.919±0.053**	**4.859±0.763**	2.909±0.553	21K
belly	**0.919±0.053**	4.908±0.787	2.918±0.556	42K
tail	0.914±0.059	5.135±0.805	3.002±0.521	21K
none	**0.919±0.053**	4.897±0.792	**2.901±0.554**	**16K**

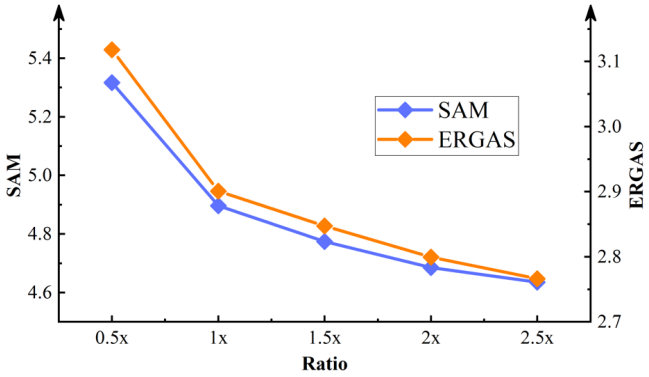

Figure 6: The performance of the LightNet with different ratio in the reduced resolution WV3 testing data.

References

[Abadi *et al.*,] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from *tensorflow.org*, 2015.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS) 32*, pages 8024–8035. Curran Associates, Inc., 2019.