APPF Project Report

Hyperspectral Image Super-Resolution Using Deep Learning

Writer: Yu Sun, Yunfan Fu

Date: 15th, May, 2024

Contents

1. Introduction

With the advancement of spectral imaging technology, hyperspectral imaging has experienced rapid growth in recent years. This imaging technique involves capturing and processing images at a multitude of wavelengths or spectral bands, offering rich data for various applications like environmental monitoring, agriculture, surveillance, and medical imaging.

However, the enhancement in spectral resolution often comes at the expense of spatial resolution. Super-resolution techniques emerge as a solution, aiming to improve the spatial resolution of images to discern finer details. In the realm of hyperspectral imaging, hyperspectral image super-resolution (HSI-SR) proves to be a valuable solution to address the challenge of low spatial resolution.

According to prevent research and practical experience, HSI-SR has demonstrated several advantages: enhancing spatial resolution to capture finer scene details, efficiently fusing hyperspectral imaging with other modalities, reducing data volume and enhancing the performance of downstream tasks. Therefore, the proposal and development of this project hold significant importance.

This report provides detailed description about our whole experiment process, from data preprocessing, model training and testing, to performance evaluation. Also, it gives detailed guidance about the structure of our code repository on GitHub, including the function of each folder and each file in the folder, by following the instruction in this report, we can make sure everyone can conduct the same experiment as us.

2. Data Processing

2.1 Processing Strategies

The original HSIs are stored as .hdr and .bil files, at the beginning we get 31 HSIs as the whole dataset, based on the splitting strategies provided in all the reviewed paper, we choose 20 HSIs as the training and validation dataset, and choose 11 HSIs as the test dataset.

All the HSIs' shape are 385 * 500 * 480, of which 480 is the number of bands, to get more samples for model training, we apply a random crop algorithm to crop 64 * 64 * 480 patches from the HSIs, the number of patches can be decided as a hyperparameter in the code, and we choose 80 in our experiments, which means we have overall 1600 64 * 64 * 480 samples for training and validation.

After the above 31 HSIs, we get extra 2 HSIs through the HSI camera, and apply them to final training and testing, these 2 images' id are 2024-04-26--17-25-10_round-0_cam-1_tray-1 and 2024-04-26--17-22-00_round-0_cam-1_tray-1, both HSIs' shape is 1971 * 500 * 480.

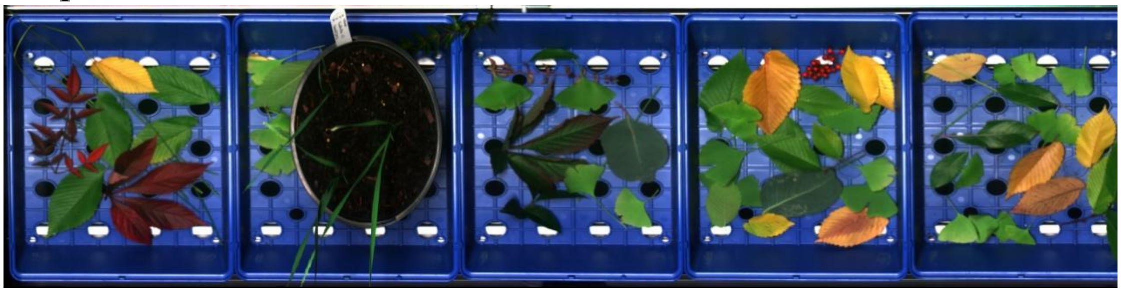
Figure.1 2024-04-26_17-25-10 HSI Visualization

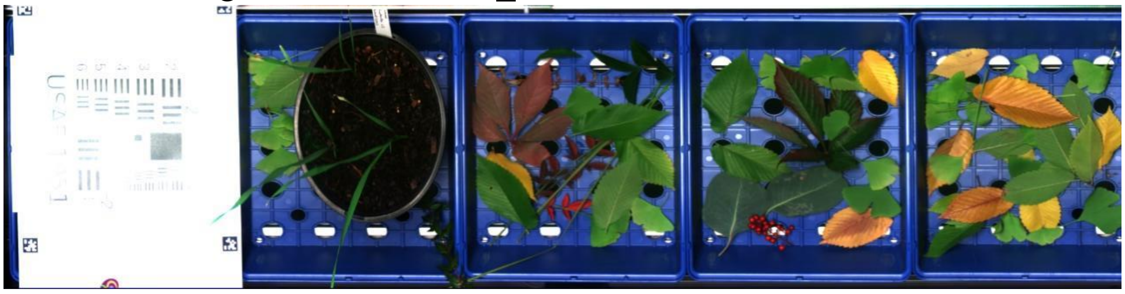
Figure. 2 2024-04-26_17-22-00 HSI Visualization

The HSI 2024-04-26_17-25-10 is applied to training and validation, we random crop 400 64 * 64 * 480 samples from it and finally we get 2000 samples for training and validation.

Continually, we select 1750 samples for training and 250 samples for validation. This means for each 80 patches from a single HSI, we use the former 70 for training and the rest for validation.

The HSI 2024-04-26_17-22-00 contains a USAF-51 chart, which is meaningful for visual performance comparison, we crop this image for 2 sides, and get 2 new HSIs of shape 385 * 500 * 480. Finally, we have 13 385 * 500 * 480 samples for model test.

2.2 Processing Codes

All the preprocessing codes are stored in ‘HSI Preprocessing’ folder in the GitHub repository, firstly, we use HSI2MAT.py to convert the original files to .mat files, and then we apply train_msd.m and test_msd.m to randomly crop the HSIs and store the data as the required mat format for specific models.

During the whole experiments, the MSDformer and EUNet follow the same preprocessing strategies and require same dataset format, therefore they could both use the train_msd.m and test_msd.m to preprocessing.

data_augment.m and modcrop.m are not used in the final experiments, but they might be useful for future improvement.

3. Experiments

3.1. Models

In this Project, we firstly train and test model with a very few amounts of data to check whether the code is complete and applicable. The first and second model we tried are SFCSR (Hyperspectral Image Super-Resolution Using Spectrum and Feature Context) and MCNet (Mixed 2D/3D Convolutional Network for Hyperspectral Image Super-Resolution), both models are created by same author. These two models are based on python 2.7, Pytorch 0.3.1 environment, which is a very old version. We successfully update the version to Pytorch 0.4.1, however, the codes still contain lots of errors and bugs, we spend over one week on these 2 models, and finally given up them. We also believe that old version models are not suitable for practical application.

Apart from the above 2 models, there still exists some models that don't meet the requirement of our project. These models have strict requirement for input's shape (must be square) and can not fit the complex situation in practical application. These two models include PDE-Net (Deep Posterior Distribution-based Embedding for Hyperspectral Image Super-resolution) and HIR-Diff (Unsupervised Hyperspectral Image Restoration Via Improved Diffusion Models).

Finally, we find 3 trainable models based on the brief training result, including MSDformer (Multi-scale Deformable Transformer for Hyperspectral Image Super-Resolution), EUNet (Efficient Unfolding Network with Disentangled Spatial-Spectral Representation for Hyperspectral Image Super-Resolution) and DCNN (Diffused Convolutional Neural Network for Hyperspectral Image Super-Resolution).

However, the DCNN model requires a very high GPU memory during the test, since it has three inputs (one input HSI, and two feature maps), it needs over 80 GB memory to conduct a super-resolution, which is not practical for real tasks. In conclusion, we finally get 2 trainable and testable models from all the above models.

3.2. Model Training and Testing

In the realm of model training, we meticulously generated a robust dataset comprising 1750 images for the training set and 250 images for the validation set, all derived from high-quality HSI imagery. We embarked on training two sophisticated models, MSDformer and EUNET, each undergoing a rigorous training regimen spanning 200 epochs.

For the MSDformer model, setting up the training environment involves configuring a detailed training script. This script requires specifying the path to the training data and adjusting critical parameters such as batch size, the number of GPUs utilized, and other vital hyperparameters. Additionally, to enhance the model's training efficacy, it is essential to fine-tune settings like the learning rate, the choice of optimizer, and whether to apply data augmentation techniques. The execution of the training process is initiated with the following command:

```
python main.py train --dataset_name 'appf' --n_blocks 4 --epochs 200 --batch_size 48 --model_title "MSDformer" --n_subs 8 --n_ovls 2 --n_feats 240 --n_scale 2 --gpus "0,1"
```

For the EUNET model, necessary parameters must be predefined in the option.py file. Once these settings are in place, training can be seamlessly initiated by executing the main.py script.

During the evaluation phase, we employ distinct test sets to gauge the performance of both models. This test set comprises 13 HSI images that were excluded from the training and validation phases to ensure unbiased assessment. We evaluate the models based on several metrics, including accuracy, recall rates, and F1 scores, to determine their generalization capabilities and real-world applicability. Furthermore, the models' outputs for specific images can be examined using a visualization tool, providing deeper insights into their operational performance.

3.3. Evaluation Metrics

For quantitative evaluation, we apply PIQE (Perception based Image Quality Evaluator) and LIQE (Language-Image Quality Evaluator) to evaluate the models' performance. Both methods are blind image quality evaluators for one single image and follow the rule of 'the lower, the better'.

In this project, we use them to evaluate: 1) the 770 * 1000 * 500 HSI after bicubic interpolation; 2) the 770 * 1000 * 480 HSI generated by MSDformer Model; 3) the 770 * 1000 * 480 HSI generated by EUNet Model. All HSI are based on 385 * 500 * 480 input HSI and the metric value is got with the RGB image of HSI.

For visual comparison, we firstly extract RGB images from the HSIs and compare their appearance. Since all the HSIs are caught on a same HSI camera, they share the same RGB bands (289, 157, 105), therefore it is easy to extract RGB images based on the experiment results.

All the evaluation codes are stored in 'Model Evaluation' folder in the GitHub repository. The LIQE.py and PIQE.py are used to evaluate the quality value of RGB images. Visual_v6.py and visual_v7.py are used to extract RGB images from .mat files, of which v6.py is written to deal with version-6 mat file and v7.py is written to deal with version-7.3 mat file.

4. Results

4.1 Model Performance

Both MSDformer and EUNet require 3 inputs for model training and validation: 1) 64 * 64 * 480 patch as the ground truth of input patch; 2) 32 * 32 * 480 input patch which is generated by down sampling ground truth; 3) 64 * 64 * 480 bicubic patch which is generated by up sampling the input patch with bicubic interpolation. The codes have been provided in folder 'HSI preprocessing'.

Therefore, it can be viewed that the way to implement bicubic method does influence the models' performance.

① implement bicubic with Python:

bicubic_band = zoom(band, scale_factor, order=3)

Bicubic method values:

BICUBIC	**PIQE**	**LIQE**	**LIQE_mix**
1	88.6911	1.1426	1.6003
2	88.0058	1.0498	1.5501
3	87.7435	1.0950	1.8280
4	86.0234	1.1290	1.6148
5	88.2220	1.1133	1.9948
6	88.5350	1.1791	1.7347
7	87.0768	1.1073	1.6246
8	88.8471	1.0899	1.8000
9	87.2498	1.1059	1.6292
10	87.7918	1.1038	1.8088
11	88.1262	1.1120	1.5510
12	88.7899	1.0709	1.3574
Average	87.9252	1.1082	1.6739

MSDformer model values:

MSDformer	**PIQE**	**LIQE**	**LIQE_mix**
1	87.0581	1.1089	1.3111
2	87.5362	1.0423	1.2469
3	86.2084	1.0933	1.4023
4	86.6348	1.0946	1.2188
5	87.0424	1.0937	1.5175

6	87.2231	1.1390	1.3724
7	86.8641	1.0945	1.2104
8	87.6502	1.0815	1.3904
9	86.9862	1.0830	1.2202
10	86.8855	1.0854	1.4446
11	87.6424	1.0898	1.2693
12	88.1664	1.0521	1.1121
Average	87.1581	1.0882	1.3097

EUNet model values:

EUNet	**PIQE**	**LIQE**	**LIQE_mix**
1	85.1254	1.1137	1.6323
2	85.1374	1.0392	1.4931
3	86.1422	1.0781	1.7602
4	85.7853	1.0938	1.5689
5	89.0804	1.1086	1.9773
6	84.6287	1.1554	1.7062
7	86.2785	1.0955	1.6224
8	88.0295	1.0802	1.8249
9	86.7773	1.0905	1.5712
10	89.2633	1.0959	1.8627
11	88.0739	1.1057	1.5820
12	88.2237	1.0646	1.2458
Average	86.8788	1.0934	1.6539

②implement bicubic with MatLab:

```
ms_bicubic = single(imresize(ms, upscale_factor));
```

Bicubic method values:

BICUBIC	**PIQE**	**LIQE**	**LIQE_mix**
1	88.6911	1.1426	1.6003
2	88.0058	1.0498	1.5501
3	87.7435	1.0950	1.8280
4	86.0234	1.1290	1.6148
5	88.2220	1.1133	1.9948
6	88.5350	1.1791	1.7347
7	87.0768	1.1073	1.6246
8	88.8471	1.0899	1.8000
9	87.2498	1.1059	1.6292

10	87.7918	1.1038	1.8088
11	88.1262	1.1120	1.5510
12	88.7899	1.0709	1.3574
Average	87.9252	1.1082	1.6739

MSDformer model values:

MSDformer	**PIQE**	**LIQE**	**LIQE_mix**
1	86.7261	1.1275	1.5010
2	86.3651	1.0390	1.4059
3	86.6069	1.0788	1.5356
4	85.2566	1.0919	1.3965
5	86.3768	1.1019	1.7613
6	87.3924	1.1403	1.5452
7	85.2529	1.0751	1.3574
8	86.6248	1.0876	1.5761
9	85.1581	1.0782	1.3908
10	86.4888	1.0956	1.6439
11	87.3618	1.0940	1.3819
12	87.6329	1.0619	1.2226
Average	86.4369	1.0892	1.4765

EUNet model values:

EUNet	**PIQE**	**LIQE**	**LIQE_mix**
1	85.1254	1.1137	1.6323
2	85.1374	1.0392	1.4931
3	86.1422	1.0781	1.7602
4	85.7853	1.0938	1.5689
5	89.0804	1.1086	1.9773
6	84.6287	1.1554	1.7062
7	86.2785	1.0955	1.6224
8	88.0295	1.0802	1.8249
9	86.7773	1.0905	1.5712
10	89.2633	1.0959	1.8627
11	88.0739	1.1057	1.5820
12	88.2237	1.0646	1.2458
Average	86.8788	1.0934	1.6539

4.2 Further Improvement

Based on 4.1 result, MSDformer's performance is better and we finally we select this model as our final tool to implement HSI super-resolution.

However, there exist an issue that the RGB image generated by bicubic interpolation has a very light line on the left side, which decrease the model's performance to some extent.

Through getting the extract value of pixels in the blue line, we find the pixel in the blue line has a negative pixel value, which lead to the abnormal result, therefore, we set all the negative value in the bicubic image as 0 and reconduct the same training and testing process based on MSDformer model.

Bicubic method values:

BICUBIC	PIQE	LIQE	LIQE_mix
1	88.0770	1.1070	1.5837
2	87.9071	1.0352	1.4986
3	87.1759	1.0729	1.7688
4	85.7081	1.0784	1.5780
5	87.7367	1.0933	1.9302
6	87.6935	1.1427	1.6863
7	87.1310	1.0809	1.6034
8	88.7880	1.0704	1.7860
9	87.1828	1.0795	1.6069
10	87.8623	1.0873	1.7893
11	87.6522	1.0879	1.5325
12	88.5455	1.0517	1.3021
Average	87.6217	1.0823	1.6388

MSDformer model values:

MSDformer	PIQE	LIQE	LIQE_mix
1	85.0756	1.1046	1.4376
2	83.6617	1.0326	1.3490
3	86.2138	1.0655	1.4937
4	84.8789	1.0667	1.3809
5	85.5945	1.0861	1.6764
6	86.2016	1.1129	1.4805
7	85.1775	1.0672	1.3363

8	85.0304	1.0651	1.4992
9	84.4864	1.0640	1.3476
10	85.7771	1.0825	1.6436
11	87.5729	1.0840	1.3395
12	86.7654	1.0472	1.1729
Average	85.5363	1.0732	1.4298

Through all the above performance comparison, we find that MSDformer model does enhance HSIs' information and details while enhancing the spatial resolution of input HSI by a factor of 2.

For the processing of output HSI, we firstly set all the nagativa value to 0 (hsi[hsi < 0] = 0), and then limit its value no larger than 1 (for band x, hsi[:, :, x] = hsi[:, :, x] / max(hsi[:, :, x])). If we need to change the output mat file back to hdr and bil, we could directly time the result with 4095, then the result will be limited in [0, 4095].

5. Conclusion

In conclusion, through data preprocessing, model training and testing, performance evaluation, we finally decide MSDformer as the best model for HSI super-resolution currently. Firstly, it can enhance the spatial resolution of input HSI by a factor of 2, which is our basic goal; secondly, it does improve the input HSI's details and information when compared with baseline model bicubic method through multiple quantitative and visual evalution.

Appendix

GitHub Links for All Codes:

- SFCSR: https://github.com/qianngli/SFCSR;
- MCNet: https://github.com/qianngli/MCNet;
- PDE-Net: https://github.com/jinnh/PDE-Net;
- HIR-Diff: https://github.com/LiPang/HIRDiff;
- MSDformer: https://github.com/Tomchenshi/MSDformer;
- EUNet: https://github.com/denghong-liu/EUNet;
- DCNN: https://github.com/LifeiBanana/DCNN;
- LIQE: https://github.com/zwx8981/LIQE

Other Links:

- PIQE: https://au.mathworks.com/help/images/ref/piqe.html;
- Bicubic Self-Implementation: https://medium.com/@amanrao032/image-upscaling-using-bicubic-interpolation-ddb37295df0.

References

1. Wang Q, Li Q, Li X. Hyperspectral image super resolution using spectrum and feature context[J]. IEEE Transactions on Industrial Electronics, 2020, 68(11): 11276-11285.
2. Li Q, Wang Q, Li X. Mixed 2D/3D convolutional network for hyperspectral image super-resolution[J]. Remote sensing, 2020, 12(10): 1660.
3. Hou J, Zhu Z, Hou J, et al. Deep posterior distribution-based embedding for hyperspectral image super-resolution[J]. IEEE Transactions on Image Processing, 2022, 31: 5720-5732.
4. Chen S, Zhang L, Zhang L. MSDformer: Multi-scale Deformable Transformer for Hyperspectral Image Super-Resolution[J]. IEEE Transactions on Geoscience and Remote Sensing, 2023.
5. Liu D, Li J, Yuan Q, et al. An efficient unfolding network with disentangled spatial-spectral representation for hyperspectral image super-resolution[J]. Information Fusion, 2023, 94: 92-111.
6. Jia S, Zhu S, Wang Z, et al. Diffused convolutional neural network for hyperspectral image super-resolution[J]. IEEE Transactions on Geoscience and Remote Sensing, 2023, 61: 1-15.
7. Zhang W, Zhai G, Wei Y, et al. Blind image quality assessment via vision-language correspondence: A multitask learning perspective[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023: 14071-14081.