Sparsity and patch for image restoration

UE COMPIM : Flipped classroom

Achraf JENZRI, Selman SEZGIN, Mohamed Nassim LAADHAR and Hamdi SGHIR

March 2023

Table of Contents

Hyperspectral Image (HSI)

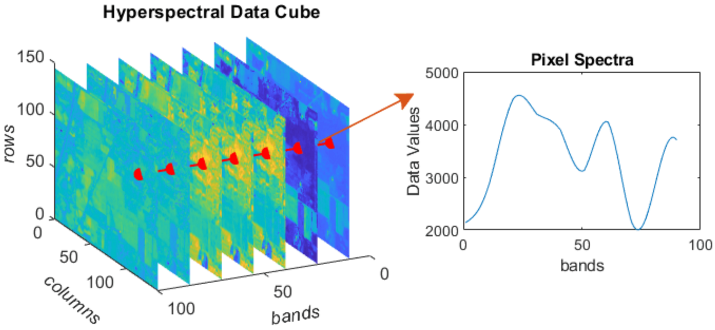


Figure: Hyperspectral Image (HSI)

Hyperspectral Image (HSI) Denoising

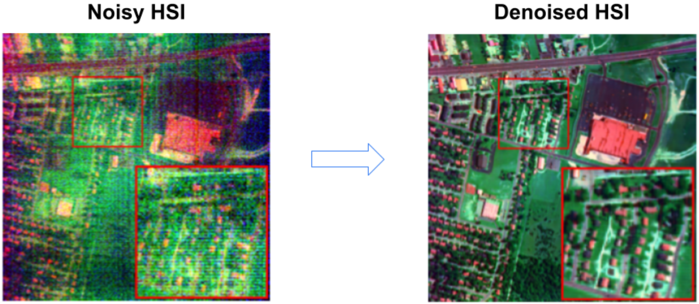


Figure: Example of HSI Denoising

Related Work on Hyperspectral Image Denoising

- Learning-free and low-rank approaches
- Sparse coding models
- Deep learning
- Hybrid approaches

Table of Contents

Sparse Coding

Sparse coding is a technique used in signal processing and machine learning for **representing data efficiently using a small number of non-zero coefficients**. The aim is to find a set of basis vectors ϕ_i such that we can represent an input vector X as a linear combination of these basis vectors:

$$X = \sum_{i=1}^{k} a_i \phi_i$$

Sparsity

Sparsity indicates that **most of the signal's coefficients are zero or close to zero** and it is a desirable property of the sparse coding representation because :

- it allows **a more efficient and effective representation of the input data**.
- it requires only **a small number of non-zero coefficients to accurately represent the underlying signal**, compared to a dense representation that requires many non-zero coefficients.
- it **improves the accuracy and quality** of the denoising process

Denoising Based on Spectral Dictionary Learning and Sparse Coding

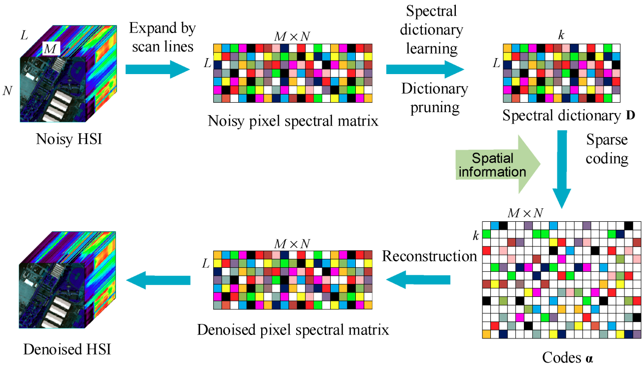


Figure: Image Denoising with Spectral Dictionary Learning and Sparse Coding

Image denoising with dictionary learning

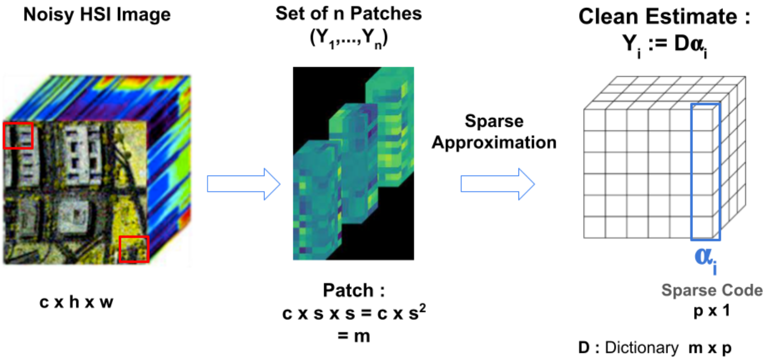


Figure: Sparce Approximation

Image denoising with dictionary learning

Let us consider y as noisy image in $R^{c \times h \times w}$ with c channels and two spatial dimensions. We denote by $y_1, .., y_n$ the n overlapping patches from y of size $c \times s \times s$, which we represent as vectors in R^m with m $= cs^2$. Assuming that a dictionary $D = [d_1, .., d_p] \in R^{m \times p}$, each patch y_i is processed by computing a sparse approximation:

$$\min_{\alpha_i} \frac{1}{2} \|y_i - D\alpha_i\|^2 + \lambda \|\alpha_i\|_1$$

- α_i : Sparse Code
- Each patch y_i admits a "clean" estimate $D\alpha_i$.
- $\|.\|_1$ l_1-norm; induce sparsity in the problem solution (l_0-penalty, which counts the number of non-zero elements)
- λ controls the amount of the regularization

Example of Image denoising with dictionary learning : PCA

- The learned dictionary is typically constructed using the **top principal components of the input image data**

- Norm 0 : the number of non-zero coefficient
  Exact sparsity in the solution
  BUT NP-hard problem
- Norm 1 : sum of the absolute values of the coefficients
  Sparse solution
  BUT sometimes not exact sparsity in the solution (high dimension)
- Norm 2 : penalizes the sum of the squared coefficients
  Smooth and continuous solution, less overfitting and algorithm stability
  BUT not effective for promoting sparsity

Differentiable programming for sparse coding : ISTA

ISTA is a proximal **gradient descent** method for solving the Lasso problem in Eq.(1) which consists of the following iterations:

$$\alpha_i^{(t+1)} = S_\lambda[\alpha_i^{(t)} + \mu D^T(y_i - D * \alpha_i^{(t)})]$$

- $\mu > 0$: step-size (Learning rate)
- $S_\lambda[u] = sign(u)max(|u| - \lambda, 0)$: the soft-thresholding operator
- T iterations
- α_i : Sparse code

Differentiable programming for sparse coding : LISTA

- **ISTA** : Iterative algorithm, consisting of a sequence of **linear transformations** followed by a **thresholding**. But **slow** to converge and may require **many iterations**

$$\alpha_i^{(t+1)} = S_\lambda[\alpha_i^{(t)} + \mu D^T(y_i - D * \alpha_i^{(t)})]$$

- **LISTA** : a set of **linear transformations** and a **learned thresholding**, which is optimized to **minimize loss function** $\mathcal{L}_\theta$ that measures the quality of the recovered signal.
- Incorporates learning into the algorithm to improve **speed convergence** and no need to compute $\nabla_\theta \mathcal{L}_\theta$.

Differentiable programming for sparse coding : LISTA

The LISTA algorithm is used to **train dictionaries** for supervised learning tasks, which considers the following iterations:

$$\alpha_i^{(t+1)} = S_\lambda[\alpha_i^{(t)} + C^T(y_i - D * \alpha_i^{(t)})]$$

- C Matrix of the same size as D
  $\rightarrow$ Improves the results quality, Faster Convergence
  $\rightarrow$ If C $= \mu D$, We recover ISTA
- Clean Estimate : $W\alpha_i^{(T)}$; W $\neq$ D to correct the potential bias due to l_1-minimization
- T : the number of LISTA steps

Differentiable programming for sparse coding : LISTA

- The reason for allowing a different dictionary W than D is to correct the potential bias due to l1-minimization.
- Finally, the denoised image $\hat{x}$ is reconstructed by averaging the patch estimates:

$$\hat{x} = \frac{1}{m} \sum_{i=1}^{n} R_i W \alpha_i^{(T)}$$

* R_i : a linear operator that places the patch x_i at position i in the image

Differentiable programming for sparse coding : LISTA

- The LISTA point of view enables us to learn the model parameters C, D, W in a supervised fashion.
- A typical loss, which we optimize by stochastic gradient descent, is then:

$$\min_{C,D,W,\lambda} \mathbb{E}[\|\hat{\mathbf{x}}(\mathbf{y}) - \mathbf{x}\|^2]$$

* Where (x, y) is a pair of clean/noisy images

Table of Contents

Architecture overview

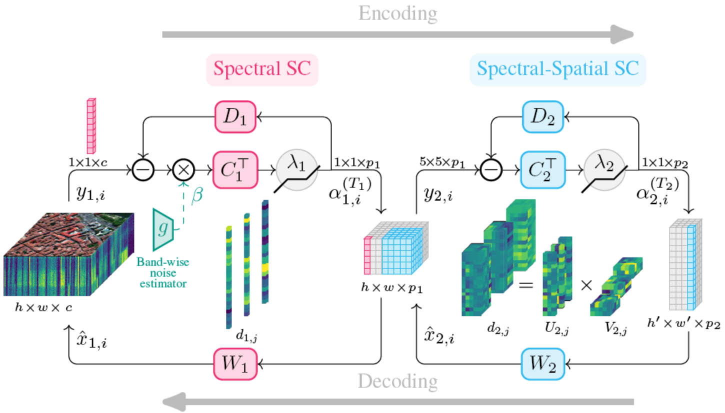


Figure: A trainable sparse coding model (T3SC)

The low-rank assumption

- Every element of dictionary $\mathbf{D} = [d_1, \ldots, d_p] \in \mathbb{R}^{cs^2 \times p}$ can be factorized in the following way:

$$d_j = vec\left(U_j \times V_j\right)$$

  with $U_j \in \mathbb{R}^{s^2 \times r}$ (spatial term) and $V_j \in \mathbb{R}^{r \times c}$ (spectral term).
- Without this decomposition, cs^2p parameters; with this decomposition: only $(s^2 + c)rp$ parameters.
- Less trainable parameters, faster learning step.

Multilayer extension

- First layer tuned to a specific sensor because of parameter c. But the second layer is generic.
- The T3SC model can be written as

$$\hat{\mathbf{x}}(\mathbf{y}) = \Psi^{dec} \circ \Psi^{enc}(\mathbf{y})$$

  where $\mathbf{y}$ noisy image and $\hat{\mathbf{x}}(\mathbf{y})$ its denoised version.
- This model can be generalized to several layers:

$$\hat{\mathbf{x}}(\mathbf{y}) = \Psi_1^{dec} \circ \cdots \circ \Psi_L^{dec} \circ \Psi_L^{enc} \circ \cdots \circ \Psi_1^{enc}(\mathbf{y})$$

Table of Contents

convolutional sparsity coding

$$\min_{\{\boldsymbol{\alpha}_i \in \mathbb{R}^p\}_{i=1,\ldots,n}} \frac{1}{2} \left\| \mathbf{y} - \frac{1}{m} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{D} \boldsymbol{\alpha}_i \right\|^2 + \lambda \sum_{i=1}^{n} \|\boldsymbol{\alpha}_i\|_1.$$

Figure: The Lasso Problem of a CSC model

Noise Adaptive Sparse Coding

The knowledge encapsulation is clear when it comes to knowing the noise applied in each band of an image .

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i\|^2 + \lambda \|\boldsymbol{\alpha}_i\|_1, \tag{1}$$

Figure: The Lasso Problem seen from a maximization a posteriori

$$\min_{\{\boldsymbol{\alpha}_i \in \mathbb{R}^p\}_{i=1,\ldots,n}} \frac{1}{2} \left\| \mathbf{y} - \frac{1}{m} \sum_{i=1}^{n} \mathbf{R}_i \mathbf{D} \boldsymbol{\alpha}_i \right\|^2 + \lambda \sum_{i=1}^{n} \|\boldsymbol{\alpha}_i\|_1. \tag{6}$$

Figure: The Lasso Problem of a CSC model

Here, the noise is i.i.d

Noise Adaptive Sparse Coding

If the noise is different from one band to another, we can adjust that by applying a normalization term β_j on each extracted band

$$\min_{\boldsymbol{\alpha}_i \in \mathbb{R}^p} \frac{1}{2} \sum_{j=1}^{c} \beta_j \|\mathbf{M}_j(\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i)\|^2 + \lambda \|\boldsymbol{\alpha}_i\|_1,$$

Figure: noise adaptation

Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

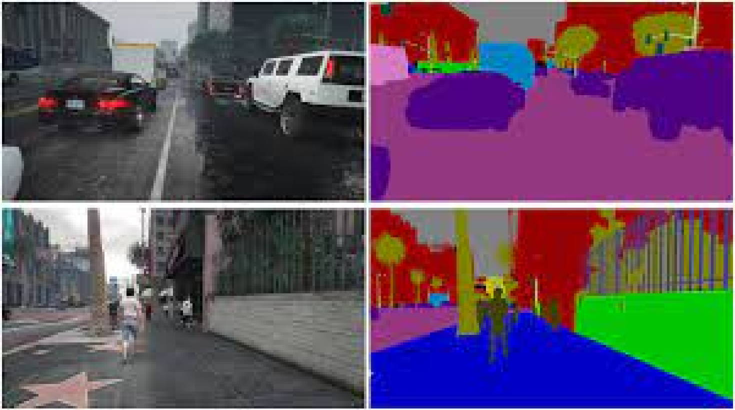


Figure: ground truth image

Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

RGB representation has three bands of a signe image

Figure: RGB image

Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

predicting pixel values given their context

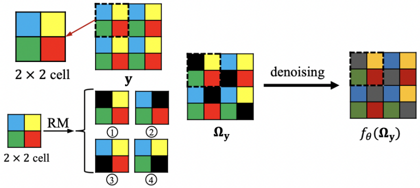


Figure: blind-spot denoising

Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

Hyperspectral imaging has quite large number of channels

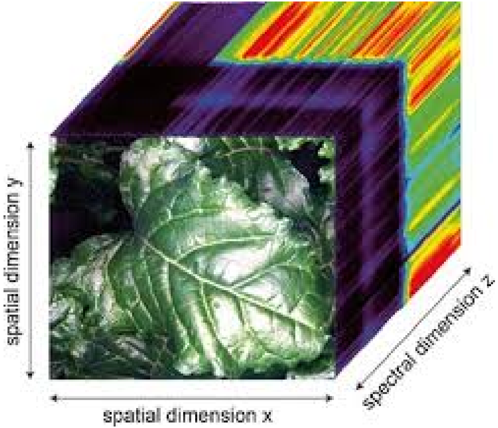
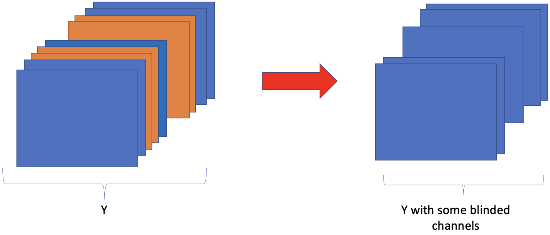


Figure: Hyperspectral image

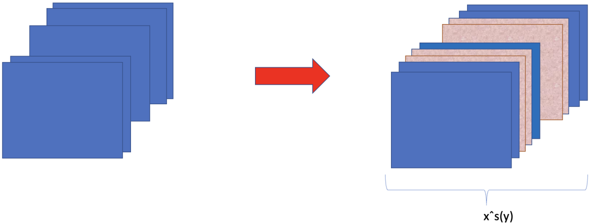
Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

Blind-Band denoising



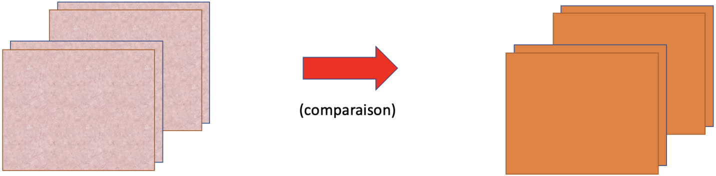
Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

Blind-Band denoising



Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

Blind-Band denoising



Self-Supervised Learning: Blind-Band denoising with No Ground Truth Data

Blind-Band denoising

$$\min_{\mathbf{C},\mathbf{D},\mathbf{W},\lambda} \mathbb{E}_{\mathbf{x},\mathbf{y},S}\left[\sum_{j \notin S} \|\mathbf{M}_j(\hat{\mathbf{x}}_S(\mathbf{y}) - \mathbf{y})\|^2\right]$$

Figure: blind-spot denoising

Table of Contents

Models used for testing

We choose to compare the presented approach of **T3SC** to :

- **Traditional methods :** BM3D BM4D GLF LLRT MG-Meet
- **Deep Learning Methods :** SMDS-Net QRNN3D

Datasets

We evaluate our approach on two datasets with significantly different properties :

- **ICVL:**consists of 204 images of size 1392×1300 with 31 bands (100 images for training, 50 for testing).
- **Washington DC Mall:** consists of a high-quality image of size 1280×307 with 191 bands.
  - **2 sub images for training** of size 600×307 and 480×307 respectively.
  - **One sub-image** of size 200×200 for testing.

Implementation procedure

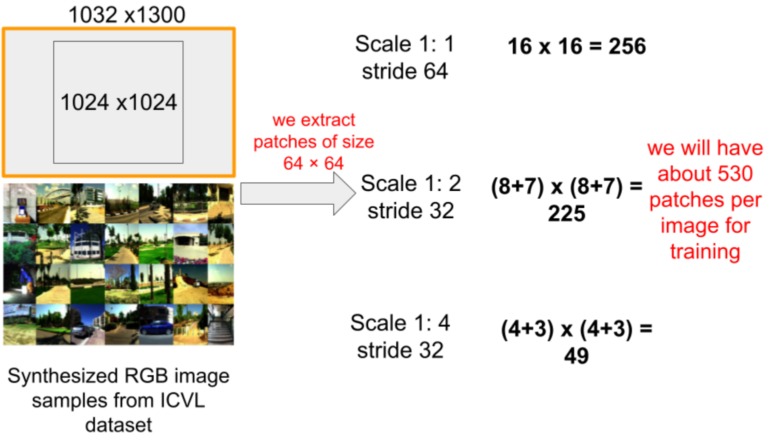


Figure: Training procedure for ICVL dataset

Implementation procedure

- **Normalization :** Before denoising, HSI images are normalized to [0, 1] using global min-max normalization.
- **Training & Evaluating the models:** We evaluate our model against different types of synthetic noise:
  - Gaussian noise with known variance σ^2.
  - Gaussian noise with unknown band-dependent variance.
  - Noise with spectrally correlated variance.
  - Stripes noise: from 33% of the bands, around 10-15% of their columns are affected by a value uniformly sampled in the interval $[-0.25, 0.25]$ which is added to them.
- **Metrics:**
  - Mean Peak Signal-to-Noise Ratio (MPSNR).
  - Mean Structural Similarity Index Measurement (MSSIM).

Quantitative results on synthetic noise

σ	Metrics	Noisy	BM3D	BM4D	GLF	LLRT	NGMeet	SMDS	QRNN3D	T3SC	T3SC-SSL
5	MPSNR	34.47	46.17	48.85	51.25	51.86	**52.74**	50.91	48.80	52.62	51.42
	MSSIM	0.7618	0.9843	0.9916	0.9949	0.9951	**0.9960**	0.9944	0.9918	0.9959	0.9952
25	MPSNR	21.44	37.86	39.89	43.16	43.43	44.74	42.83	44.20	**45.38**	44.73
	MSSIM	0.1548	0.9269	0.9510	0.9695	0.9746	0.9796	0.9700	0.9782	**0.9825**	0.9805
50	MPSNR	16.03	34.22	34.22	39.26	39.69	41.08	39.25	41.67	**42.16**	41.62
	MSSIM	0.0502	0.8654	0.8654	0.9197	0.9504	0.9602	0.9382	0.9655	**0.9677**	0.9646
100	MPSNR	10.85	30.43	32.47	34.79	36.39	37.55	35.64	37.19	**38.99**	38.50
	MSSIM	0.0144	0.7557	0.8155	0.7982	0.9182	0.9311	0.8815	0.9140	**0.9439**	0.9394
[0-15]	MPSNR	33.89	45.81	45.35	50.57	48.50	41.67	48.23	52.07	**53.31**	51.26
	MSSIM	0.6386	0.9767	0.9735	0.9948	0.9899	0.9078	0.9900	0.9957	**0.9967**	0.9955
[0-55]	MPSNR	23.36	39.06	38.43	44.22	41.13	32.94	41.76	47.13	**48.64**	46.82
	MSSIM	0.2601	0.9231	0.9074	0.9818	0.9580	0.7565	0.9620	0.9884	**0.9911**	0.9882
[0-95]	MPSNR	19.06	36.17	35.55	41.43	38.44	29.40	38.94	43.98	**46.30**	44.75
	MSSIM	0.1614	0.8760	0.8540	0.9674	0.9354	0.6609	0.9357	0.9753	**0.9859**	0.9822
Corr.	MPSNR	28.85	42.73	42.13	47.05	45.76	38.06	45.98	48.90	**49.89**	48.78
	MSSIM	0.4740	0.9599	0.9070	0.9881	0.9824	0.8536	0.9835	0.9911	**0.9923**	0.9911
Strip.	MPSNR	21.20	34.88	37.70	42.06	39.38	39.78	41.98	44.60	**44.74**	43.80
	MSSIM	0.1508	0.8641	0.9198	0.9628	0.9258	0.9333	0.9655	**0.9806**	0.9805	0.9773

Figure: Denoising performance on ICVL with various types of noise patterns.

Quantitative results on synthetic noise

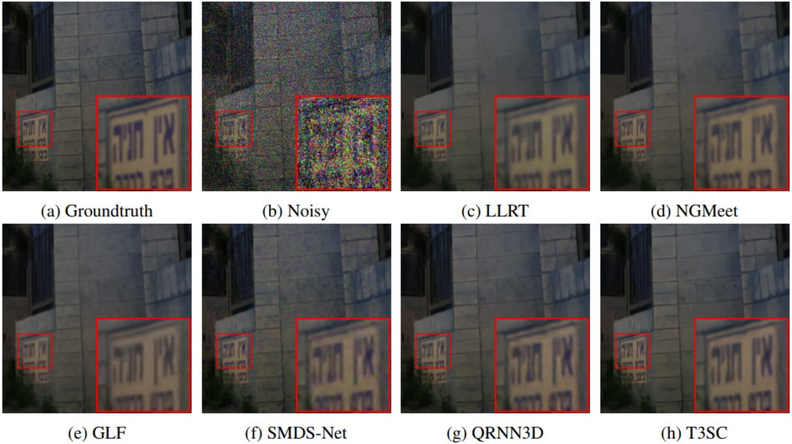


Figure: Denoising results with Gaussian noise $\sigma = 25$ on ICVL with bands 9, 15, 28.

Quantitative results on synthetic noise

	BM3D	BM4D	GLF	LLRT	NGMeet	SMDS	QRNN3D	T3SC	T3SC-SSL
Inference time (s)	1677	2382	5565	24384	2686	74.3	**3.6**	5.8	54.2

Figure: Inference time per image on ICVL with $\sigma = 50$

Results on real noise

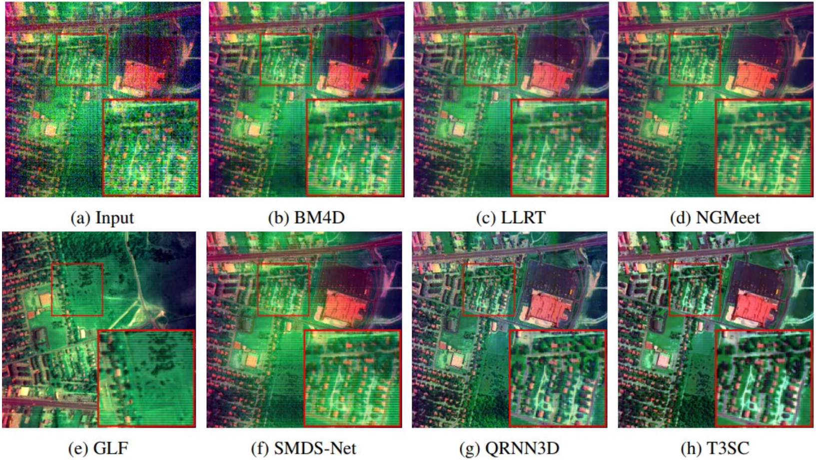


Figure: Visual result on a real HSI denoising experiment on Urban dataset with bands 1, 108, 208.

Table of Contents

Conclusion

- This supervised solution had achieved the goals of the state of the art.
- Limitation of the self-supervised method under more complex noise.
- The HSI is used in several applications : agriculture, natural disaster management planning, astronomy, archaeology, medicine the petroleum industry and military applications for surveillance.