Unsupervised Feature Learning Using Recurrent Neural Nets for Segmenting Hyperspectral Images (Supplementary material)

Lukasz Tulczyjew, Michal Kawulok, Jakub Nalepa
jnalepa@ieee.org

1 Gentle introduction to recurrent neural networks

RNNs are the artificial neural nets specialized in processing sequential data, and are able to handle variable-length sequences [1]. RNNs contain the building blocks called *cells*, and each cell has its hidden state $\boldsymbol{h}$, input $\boldsymbol{x}$, and learnable input-to-hidden and hidden-to-hidden weights and biases ($\boldsymbol{W}_{ih}$ and $\boldsymbol{W}_{hh}$, and $\boldsymbol{b}_{ih}$ and $\boldsymbol{b}_{hh}$). We have:

$$\boldsymbol{h}_t = \varphi(\boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hh} + \boldsymbol{W}_{ih}\boldsymbol{x}_t + \boldsymbol{b}_{ih}), \tag{1}$$

and $\boldsymbol{x}_t \in \mathbb{R}^n$, $\boldsymbol{h}_{t-1} \in \mathbb{R}^m$, $\boldsymbol{W}_{ih} \in \mathbb{R}^{m\times n}$, $\boldsymbol{W}_{hh} \in \mathbb{R}^{m\times m}$, $\boldsymbol{b}_{ih} \in \mathbb{R}^m$, $\boldsymbol{b}_{hh} \in \mathbb{R}^m$, φ is the activation function, n is the input size, m is the hidden-state size, t is the current step, and i and h correspond to the input and hidden layers [2].

To deal with the vanishing and exploding gradient problems, the gated mechanisms have been proposed in the literature, with the long short-term memory (LSTM) [3] and gated recurrent unit (GRU) [4] cells being the current research mainstream. In these techniques, the remembering capacity of the standard recurrent cells has been substantially improved using the components called *gates*.

LSTM cells[1] encompass three gates which control the information flow: the *input* gate is aimed at finding the relevant signal which will be absorbed into the hidden context, the *forget* gate determines how much previous information should propagate (or be excluded) for the current step, and the *output* gate is used for the prediction (Figure 1) [6]. These cells are equipped with a *cell state* ($\boldsymbol{c}$) to store long-term dependencies within the sequence. The state contains a recurrent connection to itself and paired with the outer self-loop of the whole unit which allows for maintaining temporal information more effectively. The *forget gate* ($\boldsymbol{f}_t$) decides which information should be retained in the new state by applying the sigmoid activation σ on the previous hidden state and the current input:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{if}\boldsymbol{x}_t + \boldsymbol{b}_{if} + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hi}), \tag{2}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ are the input of the current step t, and the hidden state of the previous step $(t-1)$. It outputs a number between 0 and 1 and applies it to the cell state—1 means that the

[1] Although there are several variants of LSTMs, the term LSTM cell denotes LSTM with three cells: the input, output, and forget ones [5].

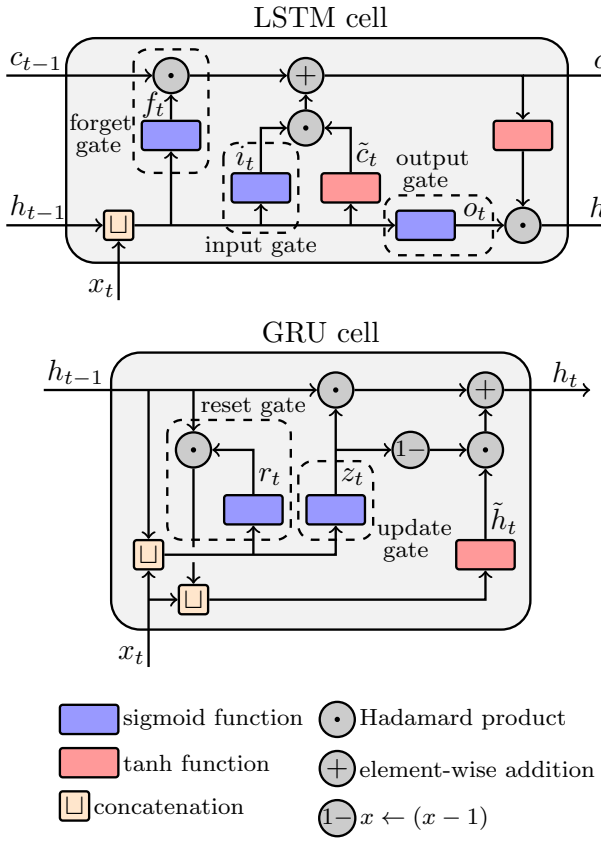

Figure 1: The LSTM and GRU cells.

signal should be fully retained, whereas 0 that is should be forgotten. The cell state of the step t is:

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t, \tag{3}$$

where $\odot$ is the *Hadamard product*, $\boldsymbol{i}_t$ is the vector of sigmoidal activations from the *input gate*, and $\tilde{\boldsymbol{c}}_t$ is the internal candidate for the new state. The $\boldsymbol{i}_t$ and $\tilde{\boldsymbol{c}}_t$ activations are:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{ii}\boldsymbol{x}_t + \boldsymbol{b}_{ii} + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hi}), \tag{4}$$

and

$$\tilde{\boldsymbol{c}}_t = \tanh(\boldsymbol{W}_{i\tilde{c}}\boldsymbol{x}_t + \boldsymbol{b}_{i\tilde{c}} + \boldsymbol{W}_{h\tilde{c}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{h\tilde{c}}). \tag{5}$$

Besides the scaling operation with $\boldsymbol{f}_t$, the cell state of the step $(t-1)$ is added to the combination of the new candidate activations and the input gate (Eq. 3). It allows us to update the previous information with the current context gathered from $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$. The hidden state becomes:

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \tag{6}$$

where $\boldsymbol{o}_t$ is the current *output gate* given as:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{io}\boldsymbol{x}_t + \boldsymbol{b}_{io} + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{ho}). \tag{7}$$

The GRU cells (Figure 1) contain less learnable parameters, thus provide a lower-complexity alternative to LSTM [7]. They are equipped with two types of gates—the *update* and *reset* ones, with the former used to determine how much of the past information (from the previous step) should be retained and taken forward to the next step, whereas the latter are exploited to decide how much of the past information should be forgotten (hence, how to add the new information to the memory). Such cells allow for increasing the rate of the model training while achieving results equivalent to LSTM [8], which is critical in the case of recurrent models [9] . The *reset* GRU gate is:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_{ir}\boldsymbol{x}_t + \boldsymbol{b}_{ir} + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hr}), \tag{8}$$

where $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ are the input at the step t and the hidden state of the step $(t-1)$. The same inputs go to the *update gate* which is a combination of the forget and input gates:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_{iz}\boldsymbol{x}_t + \boldsymbol{b}_{iz} + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{hz}). \tag{9}$$

Furthermore, the candidate vector can be computed by combining the input and the scaled previous hidden state:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_{i\tilde{h}}\boldsymbol{x}_t + \boldsymbol{b}_{i\tilde{h}} + \boldsymbol{r}_t \odot (\boldsymbol{W}_{h\tilde{h}}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{h\tilde{h}})). \tag{10}$$

The current hidden state finally becomes:

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}, \tag{11}$$

and the state is taken as the output of the cell (as already mentioned, the GRU cells do not have the output gate).

2 Details of the hyperspectral benchmarks

In this section, we report the numbers of examples in each class in the Salinas Valley (Table 1), Indian Pines (Table 2), Pavia University (Table 3), and Houston (Table 4) datasets.

3 Details of the deep models investigated in the letter

In this section, we gather the numbers of parameters alongside the floating point operations (FLOPs) of the models investigated in the letter (Table 5). For the details of 1D-CNN architectures for all datasets, see `https://gitlab.com/jnalepa/rnns_for_hsi`.

Table 1: The number of examples from each Salinas Valley class.

Class	Description	Examples#
1	Broccoli green weeds 1	2,009
2	Broccoli green weeds 2	3,726
3	Fallow	1,976
4	Fallow rough plow	1,394
5	Fallow smooth	2,678
6	Stubble	3,959
7	Celery	3,579
8	Grapes untrained	11,271
9	Soil vineyard green weeds	6,203
10	Corn senescent green weeds	3,278
11	Lettuce romaine 4 week	1,068
12	Lettuce romaine 5 week	1,927
13	Lettuce romaine 6 week	916
14	Lettuce romaine 7 week	1,070
15	Vineyard untrained	7,268
16	Vineyard vertical trellis	1,807
—	Total	54,129

Table 2: The number of examples from each Indian Pines class.

Class	Description	Examples#
1	Alfalfa	46
2	Corn-notill	1,428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2,455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1,265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93
—	Total	10,249

4 Qualitative analysis

In this section, we render the example visualizations of the segmentations elaborated over Salinas Valley, Indian Pines, and Pavia University (Figure 2), alongside the example segmentations obtained using our methods (with the Gaussian mixture model clustering) applied for original and reduced Houston data (Figure 3).

Table 3: The number of examples from each Pavia University class.

Class	Description	Examples#
1	Asphalt	6,631
2	Meadows	18,649
3	Gravel	2,099
4	Trees	3,064
5	Painted metal sheets	1,345
6	Bare soil	5,029
7	Bitumen	1,330
8	Self-blocking bricks	3,682
9	Shadows	947
—	Total	42,776

Table 4: The number of examples from each Houston class.

Class	Description	Examples#
1	Healthy grass	39,196
2	Stressed grass	130,008
3	Artificial turf	2,736
4	Evergreen trees	54,322
5	Deciduous trees	20,172
6	Bare earth	18,064
7	Water	1,064
8	Residental buildings	158,995
9	Non-residental buildings	894,769
10	Roads	183,283
11	Sidewalks	136,035
12	Crosswalks	6,059
13	Major thoroughfares	185,438
14	Highways	39,438
15	Railways	27,748
16	Paved parking lots	45,932
17	Unpaved parking lots	587
18	Cars	26,289
19	Trains	21,479
20	Stadium seats	27,296
—	Total	2,016,920

5 Detailed experimental results obtained using 1D-CNN

In this section, we gather the detailed experimental results (the averages and standard deviations) obtained using 1D-CNN over all investigated datasets and averaged over 30 independent runs. For the 1D-CNN experimental settings, see the main body of the letter.

Table 5: The number of parameters (Params) alongside the floating point operations reported in mega FLOPs (MFLOPs) for all datasets and configurations (full and reduced hyperspectral sets). Note that the metrics vary across the full and reduced scenarios, e.g. due to the architectural changes that reflect different data dimensionality (see e.g., the decoding part of our RNN-based feature extractor in Fig. 1 in the main body of the letter). In this table, we gather the models that were investigated in Section III.B of the letter (the specific GRU and LSTM architectures were selected in the sensitivity analysis reported in Section III.A).

Dataset→	SV		IP		PU		Houston	
Algorithm	Params ($\cdot 10^5$)	MFLOPs	Params ($\cdot 10^5$)	MFLOPs	Params ($\cdot 10^5$)	MFLOPs	Params ($\cdot 10^5$)	MFLOPs
1D-CNN(Full)	103.10	20.62	101.05	20.21	50.87	10.17	5.95	1.19
1D-CNN(Reduced)	10.94	2.19	10.94	2.19	10.94	2.19	2.61	0.52
VAE(Full)	52.55	10.50	51.53	10.29	26.67	5.33	13.09	2.61
VAE(Reduced)	6.69	1.33	6.69	1.33	6.69	1.33	6.69	1.33
3D-CAE(Full)	2.52	204.40	2.49	200.30	1.71	101.13	1.29	46.94
3D-CAE(Reduced)	1.09	21.38	1.09	21.38	1.09	21.38	1.09	21.38
GRU(Full)	13.59	7.67	13.33	7.52	7.10	3.91	3.69	1.93
GRU(Reduced)	2.09	1.00	2.09	1.00	2.09	1.00	2.09	1.00
LSTM(Full)	13.57	7.09	13.32	6.95	7.09	3.62	3.68	1.79
LSTM(Reduced)	2.07	0.93	2.07	0.93	2.07	0.93	2.07	0.93

Table 6: The per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Salinas Valley and averaged across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AA	OA	Kappa
Full	0.993	0.995	0.963	0.994	0.964	0.997	0.996	0.697	0.987	0.914	0.966	0.999	0.988	0.964	0.728	0.988	0.946	0.887	0.875
PCA	0.998	0.968	0.684	0.929	0.874	0.931	0.963	0.633	0.945	0.892	0.897	0.776	0.929	0.922	0.637	0.99	0.873	0.82	0.802
ICA	0.997	0.99	0.977	0.988	0.967	0.997	0.996	0.782	0.98	0.955	0.987	0.978	0.978	0.974	0.708	0.99	0.953	0.904	0.893
S-MSI	0.979	0.99	0.981	0.993	0.953	0.997	0.995	0.74	0.982	0.913	0.983	0.998	0.983	0.946	0.67	0.984	0.943	0.887	0.874

Table 7: Standard deviation of the per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Salinas Valley across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AA	OA	Kappa
Full	0.006	0.007	0.054	0.005	0.026	0.003	0.001	0.153	0.012	0.028	0.028	0.004	0.007	0.032	0.106	0.003	0.01	0.022	0.024
PCA	0.003	0.11	0.426	0.246	0.294	0.249	0.179	0.283	0.14	0.242	0.252	0.346	0.184	0.11	0.299	0.004	0.163	0.166	0.178
ICA	0.004	0.015	0.023	0.017	0.023	0.002	0.003	0.084	0.035	0.017	0.012	0.025	0.023	0.022	0.09	0.006	0.006	0.009	0.01
S-MSI	0.014	0.014	0.029	0.006	0.035	0.002	0.001	0.135	0.012	0.023	0.013	0.003	0.011	0.018	0.134	0.008	0.005	0.014	0.015

6 Beyond the basic clustering: clustering by fast search and find of density peaks

In this section, we show the flexibility of our unsupervised segmentation technique and experimentally prove that we can exploit any clustering algorithm once the latent representation is elaborated.

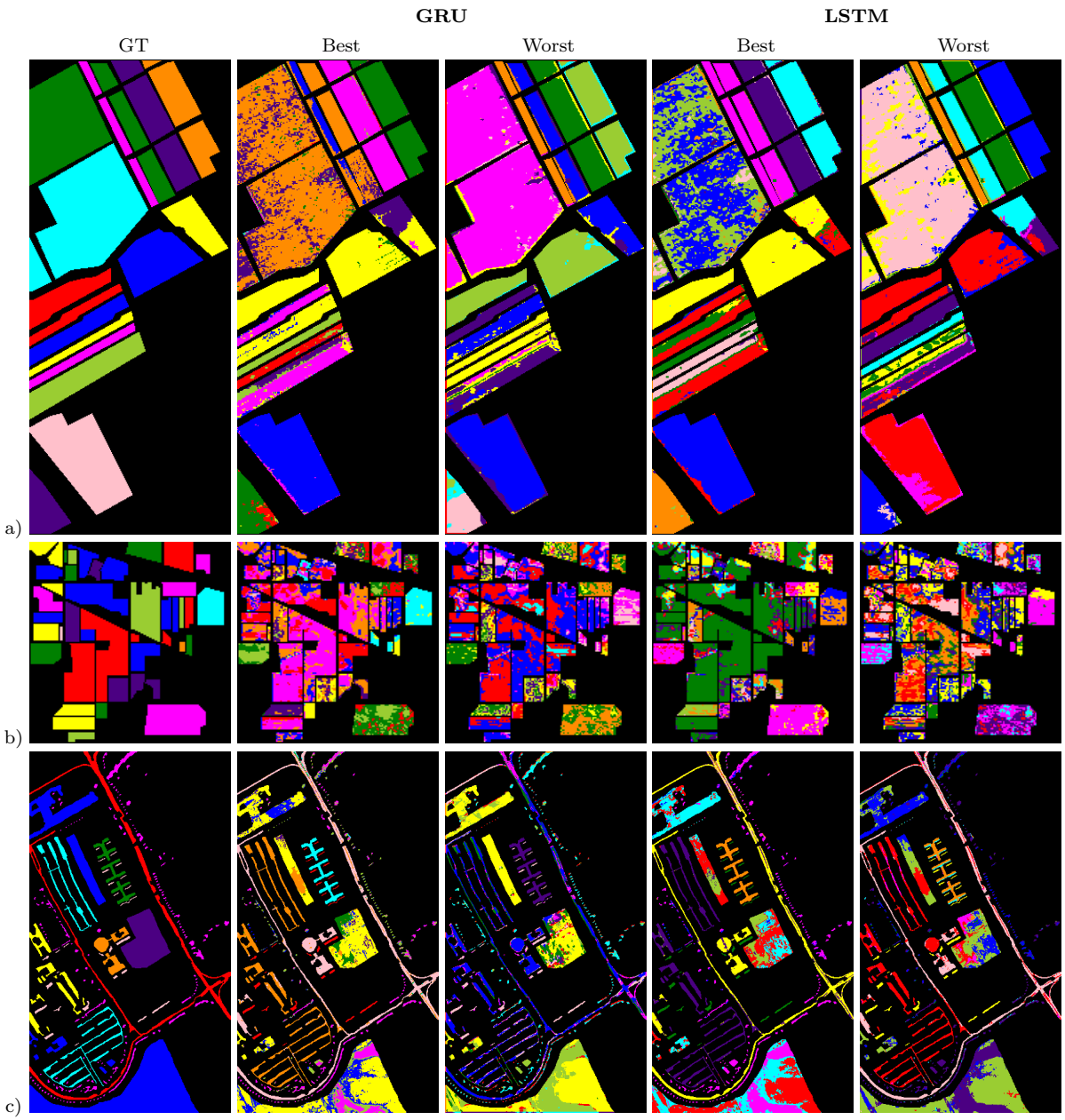

Figure 2: Segmentation of a) SV, b) IP, and c) PU obtained using the best and the worst variants of GRU and LSTM (see NMI in Table I in the main body of the letter), alongside the ground truth (GT). The same-color pixels (within a single segmentation, as the colors across different segmentations are not correlated) belong to the same class—we perform *unsupervised* segmentation, and the classes are unknown.

Here, we utilize clustering by fast search and find of density peaks [10]. It builds upon the observation that cluster centers are characterized by a higher density than their neighbors, and by a large distance from points with higher densities. Also—due to its intrinsic characteristics—this algorithm is capable of finding clusters of any shape. Although it shows high accuracy, its memory complexity is very high, as it requires keeping a global distance matrix in memory [11], and it can perform poorly over complex datasets of high dimensionality [12]. To address these issues, we partition the hyperspectral images before clustering, and additionally apply principal component

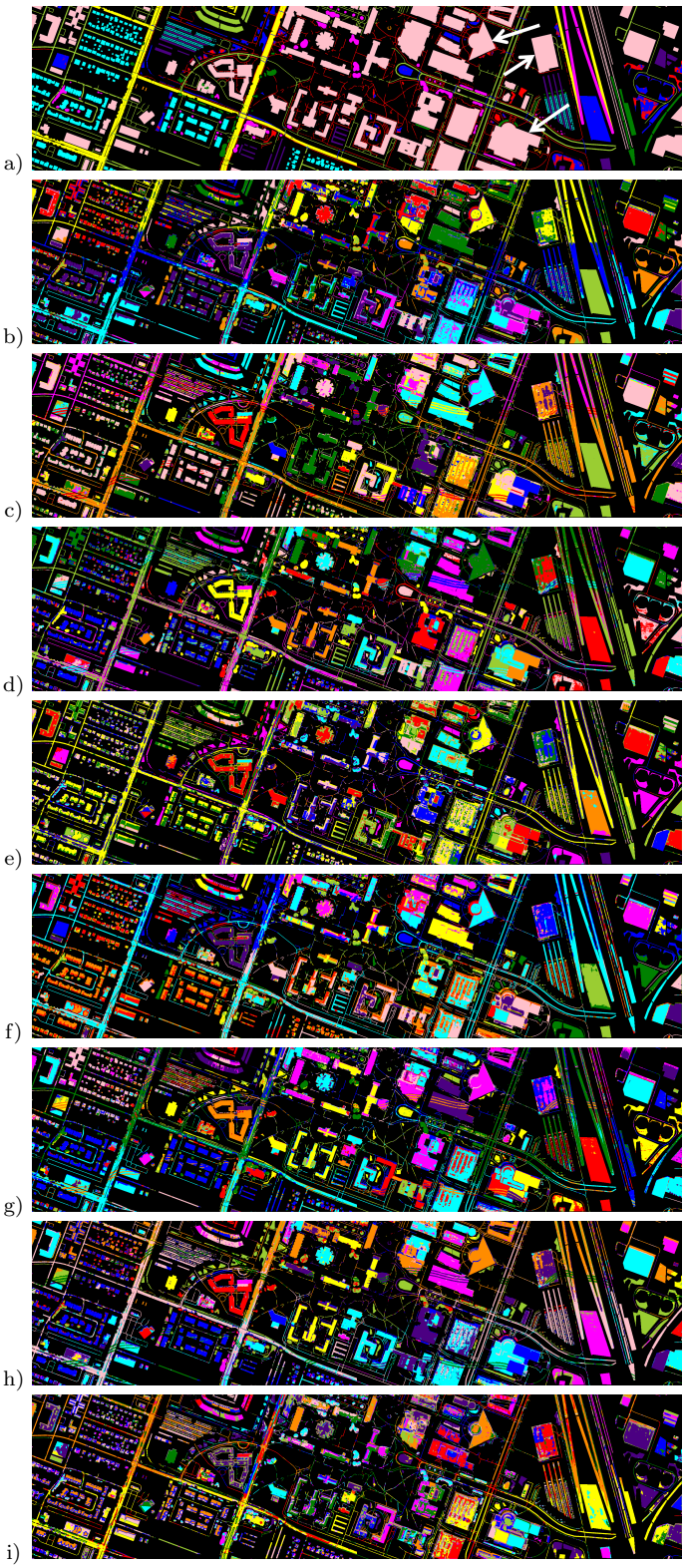

Figure 3: The Houston scene: a) the ground-truth segmentation, alongside the segmentations obtained using b) GRU, c) GRU (PCA), d) GRU (ICA), e) GRU (S-MSI), f) LSTM, g) LSTM (PCA), h) LSTM (ICA), and i) LSTM (S-MSI). The same-color pixels (within a single segmentation, as the colors across different segmentations are not correlated) belong to the same class—we perform *unsupervised* segmentation, and the classes are unknown. Note that our algorithms allowed us to extract more fine-grained details of some objects, e.g., the class of non-residential buildings annotated in light pink in the ground truth (see examples indicated with the white arrows).

Table 8: The per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Indian Pines and averaged across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AA	OA	Kappa
Full	0.775	0.654	0.762	0.847	0.902	0.963	0.894	0.98	0.872	0.756	0.653	0.877	0.987	0.952	0.411	0.962	0.828	0.777	0.749
PCA	0.92	0.591	0.719	0.666	0.9	0.9	0.768	0.909	0.87	0.445	0.563	0.774	0.99	0.924	0.359	0.954	0.766	0.691	0.655
ICA	0.898	0.598	0.65	0.75	0.941	0.963	0.893	0.973	0.827	0.695	0.618	0.773	0.983	0.913	0.434	0.946	0.803	0.736	0.702
S-MSI	0.79	0.698	0.72	0.827	0.928	0.963	0.835	0.988	0.873	0.824	0.664	0.881	0.989	0.946	0.406	0.981	0.832	0.79	0.762

Table 9: Standard deviation of the per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Indian Pines across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	AA	OA	Kappa
Full	0.185	0.171	0.101	0.102	0.093	0.033	0.069	0.022	0.114	0.144	0.147	0.079	0.023	0.041	0.146	0.025	0.049	0.063	0.068
PCA	0.102	0.198	0.16	0.267	0.149	0.127	0.267	0.18	0.204	0.233	0.199	0.197	0.014	0.181	0.157	0.029	0.11	0.121	0.129
ICA	0.044	0.127	0.132	0.105	0.041	0.023	0.06	0.042	0.156	0.141	0.108	0.103	0.028	0.067	0.105	0.025	0.031	0.04	0.044
S-MSI	0.196	0.132	0.135	0.121	0.055	0.028	0.122	0.012	0.194	0.084	0.132	0.069	0.014	0.048	0.105	0.019	0.042	0.037	0.041

Table 10: The per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Pavia University and averaged across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	AA	OA	Kappa
Full	0.87	0.861	0.797	0.964	0.997	0.889	0.872	0.796	0.999	0.894	0.872	0.835
PCA	0.275	0.386	0.841	0.873	0.588	0.252	0.236	0.219	0.394	0.451	0.398	0.326
ICA	0.665	0.693	0.753	0.945	0.976	0.686	0.701	0.568	0.947	0.771	0.713	0.645
S-MSI	0.882	0.803	0.798	0.947	0.997	0.873	0.849	0.73	0.999	0.875	0.839	0.796

Table 11: Standard deviation of the per-class classification accuracies, average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Pavia University across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	AA	OA	Kappa
Full	0.087	0.094	0.085	0.027	0.002	0.095	0.138	0.151	0.001	0.042	0.055	0.067
PCA	0.356	0.415	0.281	0.258	0.377	0.395	0.383	0.342	0.483	0.274	0.31	0.315
ICA	0.23	0.199	0.113	0.048	0.072	0.253	0.228	0.207	0.18	0.096	0.124	0.135
S-MSI	0.072	0.137	0.107	0.033	0.002	0.093	0.178	0.216	0.001	0.065	0.083	0.099

analysis (PCA) to extract two principal components from either original dataset (PCA), or from our RNN-based compressed representations. Since we do PCA over the latent representation, we skipped the variant in which we train our RNN extractors over the initially reduced space of 25 principal components (as it was done for other clustering approaches, reported in the main body of this letter). We split Salinas Valley into 4 non-overlapping patches of 128×217 size, Indian

Table 12: The per-class classification accuracies obtained using 1D-CNN for Houston and averaged across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
Full	0.934	0.899	0.99	0.962	0.94	0.985	0.993	0.792	0.732	0.433	0.597	0.696	0.555	0.905	0.982	0.938	1	0.894	0.955	0.978
PCA	0.793	0.737	0.72	0.872	0.76	0.918	0.996	0.39	0.542	0.401	0.511	0.65	0.331	0.702	0.893	0.809	1	0.739	0.899	0.838
ICA	0.872	0.775	0.994	0.884	0.847	0.945	0.997	0.65	0.638	0.327	0.461	0.473	0.436	0.684	0.886	0.861	0.972	0.818	0.785	0.906
S-MSI	0.941	0.88	0.992	0.964	0.911	0.985	0.992	0.769	0.707	0.392	0.592	0.688	0.512	0.878	0.988	0.895	0.999	0.88	0.938	0.968

Table 13: The average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Houston and averaged across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	AA	OA	Kappa
Full	0.859	0.728	0.667
PCA	0.728	0.555	0.483
ICA	0.759	0.62	0.545
S-MSI	0.841	0.705	0.637

Table 14: Standard deviation of the per-class classification accuracies obtained using 1D-CNN for Houston across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12	C13	C14	C15	C16	C17	C18	C19	C20
Full	0.027	0.03	0.006	0.019	0.019	0.009	0.006	0.027	0.049	0.064	0.048	0.071	0.06	0.029	0.007	0.019	0	0.025	0.024	0.01
PCA	0.23	0.226	0.414	0.099	0.209	0.124	0.005	0.312	0.154	0.154	0.132	0.1	0.174	0.271	0.13	0.16	0	0.202	0.067	0.176
ICA	0.092	0.109	0.006	0.101	0.092	0.069	0.005	0.177	0.1	0.147	0.122	0.202	0.135	0.101	0.096	0.062	0.044	0.078	0.136	0.103
S-MSI	0.03	0.033	0.009	0.015	0.023	0.011	0.009	0.059	0.043	0.052	0.046	0.055	0.08	0.027	0.009	0.033	0.003	0.026	0.032	0.013

Table 15: Standard deviation of the average accuracy (AA), overall accuracy (OA), and the kappa score obtained using 1D-CNN for Houston across all 30 independent runs. The measures are reported for *all labeled pixels in the scene* to make them comparable with unsupervised segmentation, in which the entire scene undergoes segmentation.

Alg.	AA	OA	Kappa
Full	0.014	0.026	0.027
PCA	0.09	0.114	0.109
ICA	0.025	0.053	0.054
S-MSI	0.014	0.029	0.028

Pines: 2 patches of 145×73 size, Pavia University: 8 patches of 152×170 size, and Houston: 144 patches of 200×200 size.

In Table 18, we gather the NMI and ARS values, together with the clustering time (in seconds), averaged across all patches for all investigated configurations, and the segmentations of selected patches are collected in Figures 4–7 for Salinas Valley, Indian Pines, Pavia University, and Houston,

Table 16: The average NMI and ARS scores obtained using 1D-CNN for all datasets and averaged across all 30 independent runs.

Alg.	SV(NMI)	SV(ARS)	IP(NMI)	IP(ARS)	PU(NMI)	PU(ARS)	Houston(NMI)	Houston(ARS)
Full	0.885	0.725	0.705	0.586	0.786	0.771	0.586	0.569
PCA	0.86	0.685	0.64	0.493	0.36	0.215	0.471	0.354
ICA	0.873	0.723	0.641	0.502	0.616	0.556	0.484	0.441
S-MSI	0.88	0.723	0.718	0.609	0.784	0.759	0.563	0.546

Table 17: Standard deviation of NMI and ARS scores obtained using 1D-CNN for all datasets and averaged across all 30 independent runs.

Alg.	SV(NMI)	SV(ARS)	IP(NMI)	IP(ARS)	PU(NMI)	PU(ARS)	Houston(NMI)	Houston(ARS)
Full	0.011	0.029	0.043	0.077	0.034	0.075	0.022	0.051
PCA	0.063	0.113	0.06	0.098	0.131	0.164	0.063	0.136
ICA	0.092	0.118	0.023	0.049	0.082	0.154	0.038	0.099
S-MSI	0.007	0.023	0.028	0.049	0.033	0.074	0.021	0.052

respectively. This experiment showed that our RNNs can be straightforwardly combined with other clustering techniques, and deliver high-quality segmentation that significantly outperforms PCA for the majority of cases. It also proves that our segmentation pipeline is built in a fully *plug-and-play* manner, and allows for deploying new, perhaps more efficient clustering techniques independently from the underlying feature extractor.

Table 18: The NMI and ARS metrics, alongside the execution time (in seconds) averaged across all patches in Salinas Valley (SV), Indian Pines (IP), Pavia University (PU), and Houston. The best results are boldfaced.

Dataset→	SV			IP			PU			Houston		
Algorithm	NMI	ARS	Time	NMI	ARS	Time	NMI	ARS	Time	NMI	ARS	Time
PCA	0.814	0.750	53.004	**0.552**	**0.376**	7.336	0.644	0.559	42.455	0.293	0.305	66.499
GRU	0.806	0.746	59.310	0.438	0.312	11.297	0.701	0.634	53.669	**0.364**	**0.291**	65.637
GRU(ICA)	0.636	0.583	57.605	0.334	0.212	10.602	0.277	0.166	52.355	0.309	0.211	70.400
GRU(S-MSI)	0.830	0.762	57.327	0.509	0.338	10.255	**0.733**	**0.670**	52.964	**0.364**	**0.291**	68.806
LSTM	0.794	0.747	61.355	0.452	0.372	11.121	0.722	0.644	52.547	**0.364**	0.285	67.950
LSTM(ICA)	0.793	**0.806**	56.347	0.390	0.292	10.436	0.392	0.271	52.592	0.301	0.191	68.497
LSTM(S-MSI)	**0.831**	0.761	57.916	0.504	0.354	10.267	0.727	0.661	52.089	0.363	0.289	70.115

References

[1] X. Jia, M. Wang, A. Khandelwal, A. Karpatne, and V. Kumar, "Recurrent generative networks for multi-resolution satellite data: An application in cropland monitoring," in *Proc. IJCAI*, 2019, pp. 2628–2634.

[2] G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding." in *Interspeech*, 2013, pp.

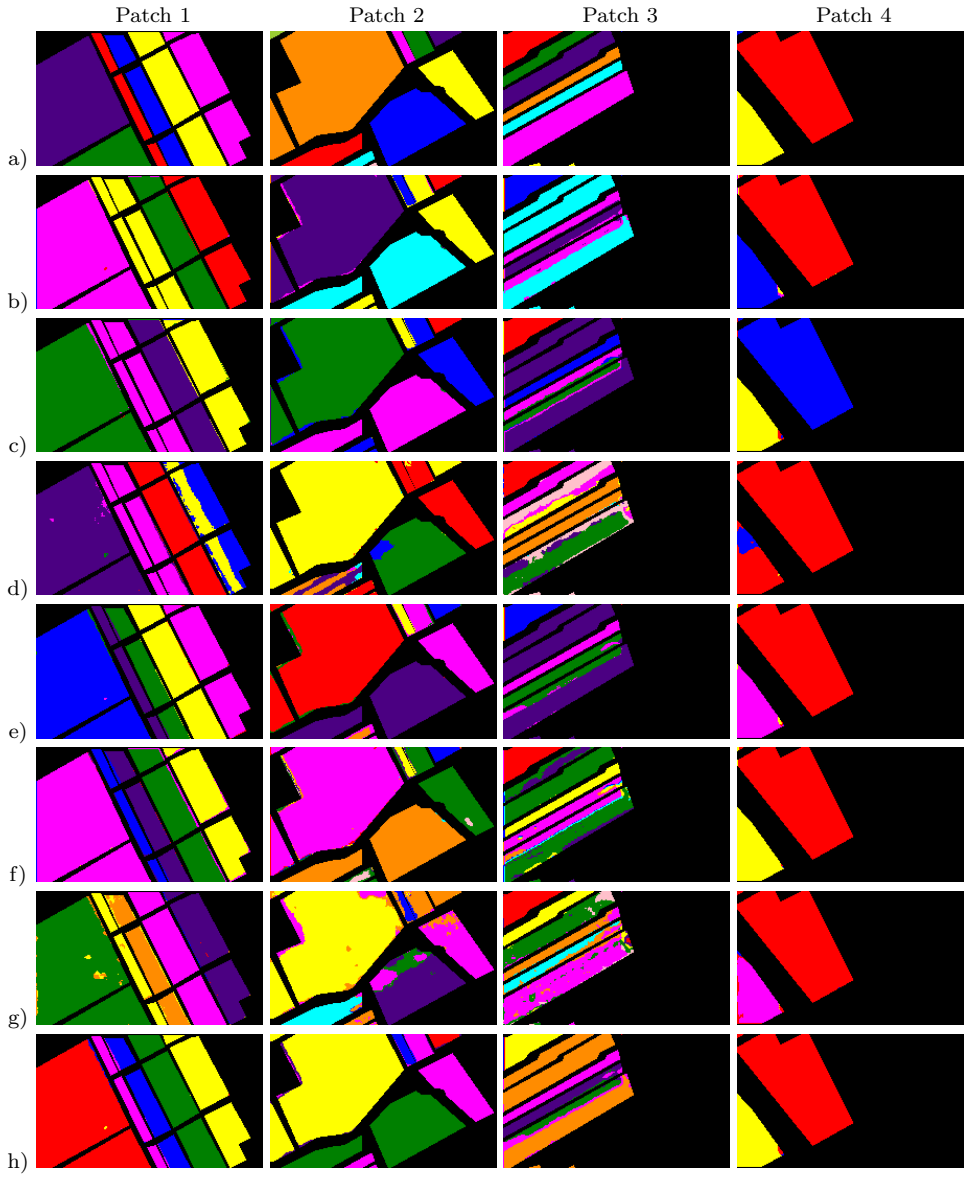

Figure 4: Segmentation of the Salinas Valley patches: a) ground truth, b) PCA, c) GRU, d) GRU(ICA), e) GRU(S-MSI), f) LSTM, g) LSTM(ICA), h) LSTM(S-MSI).

3771–3775.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014, pp. 1724–1734.

[5] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, July 2019.

Figure 5: Segmentation of the Indian Pines patches: a) ground truth, b) PCA, c) GRU, d) GRU(ICA), e) GRU(S-MSI), f) LSTM, g) LSTM(ICA), h) LSTM(S-MSI).

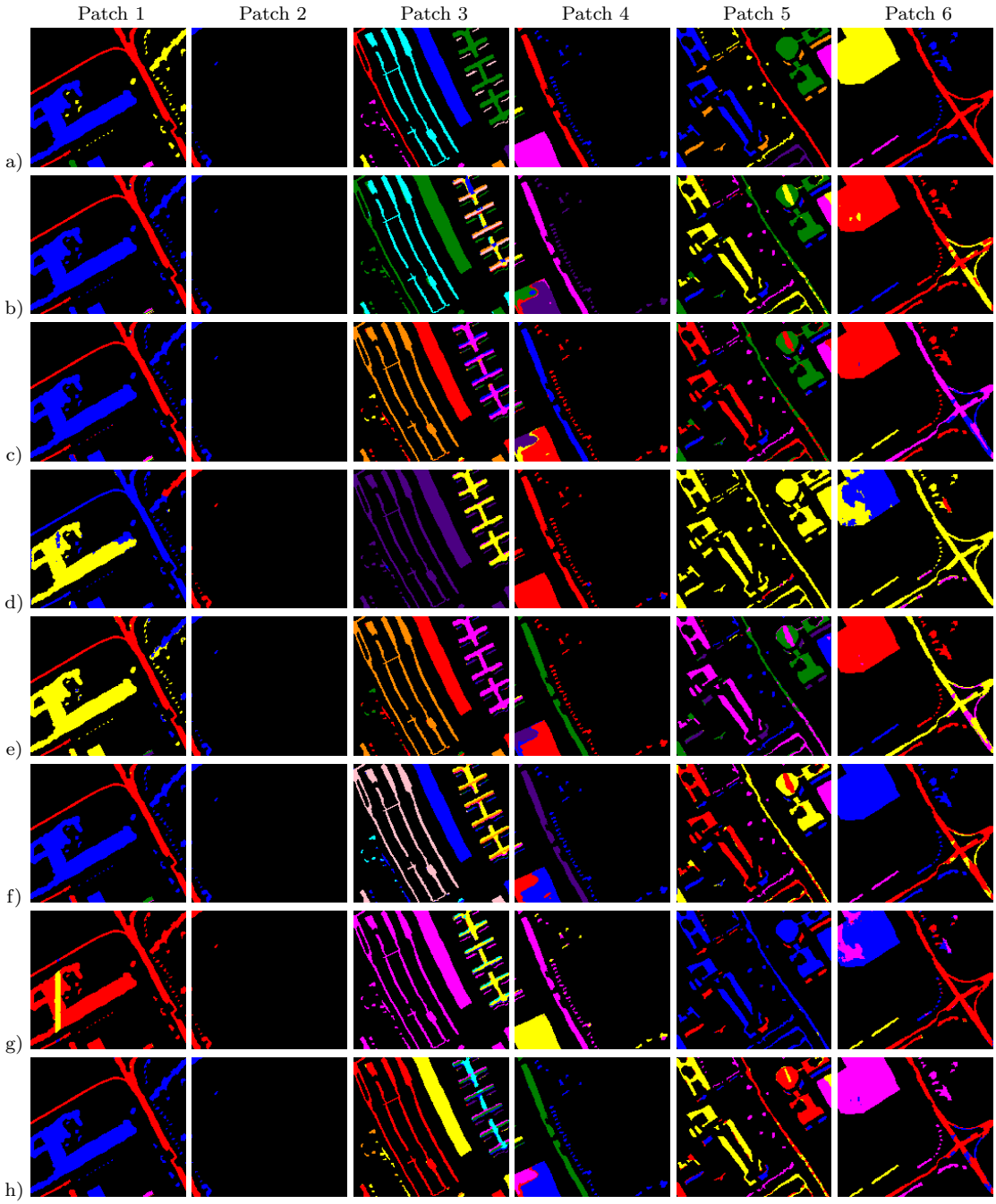

Figure 6: Segmentation of the selected Pavia University patches: a) ground truth, b) PCA, c) GRU, d) GRU(ICA), e) GRU(S-MSI), f) LSTM, g) LSTM(ICA), h) LSTM(S-MSI).

[6] Z. Li, D. He, F. Tian, W. Chen, T. Qin, L. Wang, and T. Liu, "Towards binary-valued gates for robust LSTM training," *CoRR*, vol. abs/1806.02988, 2018. [Online]. Available: http://arxiv.org/abs/1806.02988

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[8] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network

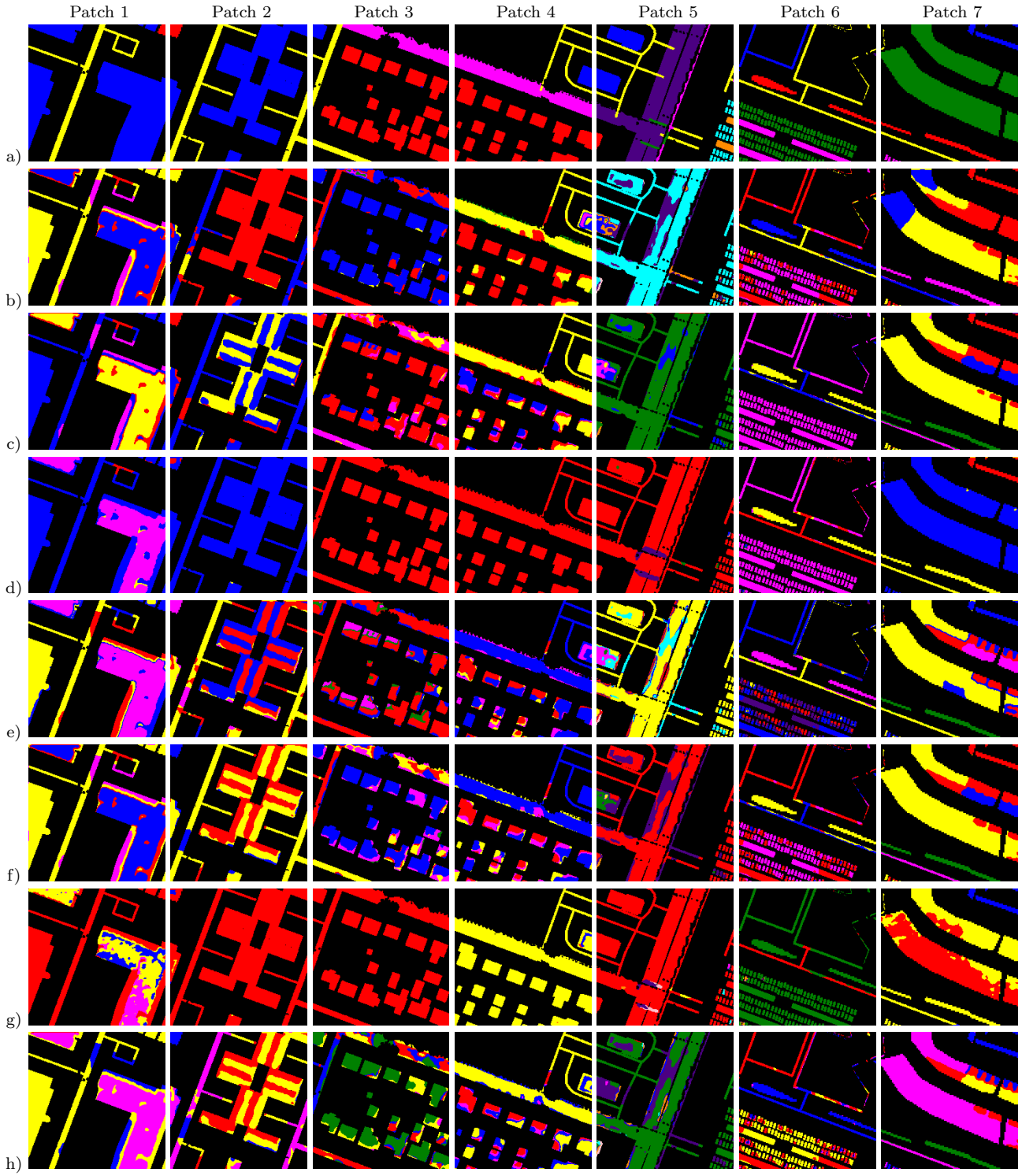

Figure 7: Segmentation of the selected Houston patches: a) ground truth, b) PCA, c) GRU, d) GRU(ICA), e) GRU(S-MSI), f) LSTM, g) LSTM(ICA), h) LSTM(S-MSI).

architectures," in *Proc. ICML*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 2342–2350.

[9] G. Zhong, G. Yue, and X. Ling, "Recurrent attention unit," *CoRR*, vol. abs/1810.12754,

2018. [Online]. Available: http://arxiv.org/abs/1810.12754

[10] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[11] J. Chengheng and L. Yongmei, "Parallel clustering by fast search and find of density peaks," in *Proc. ICALIP*, 2016, pp. 563–567.

[12] R. Liu, H. Wang, and X. Yu, "Shared-nearest-neighbor-based clustering by fast search and find of density peaks," *Inf. Sci.*, vol. 450, no. C, p. 200–226, Jun. 2018.