Modified Convolutional Neural Networks Architecture for Hyperspectral Image Classification (Extra-Convolutional Neural Networks)

Maissa HAMOUDA1 | Med Salim BOUHLEL2

1 Sousse University, ISITCom, SETIT Laboratory, GP1 Hammam Sousse, Sousse 4011, Tunisia
2 Sfax University, ENIS, SETIT Laboratory, Route Sokra km 4 - BP 1175, Sfax 3038, Tunisia

Correspondence
Maissa Hamouda, Sousse University, ISITCom, SETIT Laboratory, Tunisia.
Email: maissa_h@yahoo.fr

Funding information
Ministry of Higher Education and Scientific Research

Abstract
Classification of Hyperspectral Satellite Images (HSI) is a very important technology for object detection and cartography. Several problems can be detected, which make classification difficult (large size of the images, fusion between the classes, small amount of samples, etc.). Recently, several Convolutional Neural Networks (CNN-HSI) have been proposed for the classification of hyperspectral images. In this article, an improvement to CNN-HSI is proposed, aiming to reduce the number of erroneous pixels during classification (due to the limited number of samples). Thus, an extra-convolution technique (ExCNN) is proposed, where we add layers of global convolutions on the classified images, outgoing from classical CNN. The addition of 1 to 10 layers, on three real hyperspectral images, is tested. The results obtained are compared with other similar methods of the state of art, and show the effectiveness of the proposed method.

1 INTRODUCTION

Classification of satellite images is a very interesting technology for remote sensing, which can help in several areas such as cartography, security, ground control, and natural disasters [1–3]. Deep learning technologies are the ideal solutions for better classification of different types of images (satellite [4, 5], drones, medical [6], facial [7], etc.).

The classification of hyperspectral images by convolutional neural networks (CNN) has attracted the attention of researchers, by the perfect results obtained in recent years [8–10]. Thus, several factors can change the CNN-HSI results, either by the choice of parameters, the extraction method and the concatenation, or in the applied architecture [11, 12].

1.1 Type of extraction

There are several types of treatment; either extract spectral and spatial information at the same time then classify [1, 13, 14]; either extract separately the spatial information, then the spectral information, then classify and merge the results [15, 16]; either reduce the spectral information, extract the spatial information, then classify [17–20]; either select certain spectral bands only (the most important bands), then classify [21].

1.2 Classification in one or more dimensions

The approach in [22] uses the band-by-band extraction methodology, that is, classifies it in 2D, then merges the results. Other works use [23, 24] a 3D Network: The first network [23] builds convolutional layers in a low resolution space and extracts entities with spatial and spectral dimensions with an extended 3D core and a 3D deconvolution is used for the last layer, it is the enlargement of the desired size. The second network [24] consists of an unsupervised learning strategy for spatial and spectral entities proposed for HSI with a convolutional 3D auto-coder.

Finally, the application of different spectral reduction methods like smart feature extraction (SFE-CNN) [25], where is a probabilist reduction of the different spectral values of each spatial location of the image, to obtain at the end a single spatial plane (2D image), or deep-dual extraction (DCNN) [26], where first a CNN is used (1D) for the extraction of the spectral information, then the reduction of characteristics, and a second CNN...
(2D) for the extraction of the spatial information, then the classification of characteristics.

### 1.3 CNN architecture

The architecture of YANN in [27], which proposed the first architecture of CNN, with four phases, two convolutional, two fully connected, and ended with the Gaussian filter. Alex’s network [28] applied eight phases, five convolutional, and three fully connected. Where each convolutional phase is composed of three repetitive parts which are convolution, ReLU, and Pooling, successively. The architecture of [29], which is mainly composed of four phases, of which the first two convolutional, composed of convolution, ReLU, and Pooling, followed by two fully connected phases FC, and which is adapted specifically for the classification of hyperspectral images. The technical report of [30], which described a MATLAB library for CNN by applying a model consisting of several repetitive convolutional phases and a last fully connected phase. The approach in [31] is to use a triple architecture. CNN is built to extract the spatial and cascading spectral characteristics of double-scale spectral entities, from shallow layers to low layers. Spatial data and multilayer spectra are merged to learn complementary information between shallow layers with detailed information and deep layers with semantic information.

### 1.4 Adaptation of parameters

In order to automate all the parameters of the proposed network, consideration is given to the adaptive choice of the sizes of the convolution filters [32] based on the lot sizes adaptively chosen. The approach [33] consists in setting up a regularized CNN, in order to take charge of optimization and regularization in order to solve the problem of over-adaptation of limited packaging samples. Although many types of literature have been tested, they have been effectively filtered. The HSI is now part of the spectrum in the spatial context, the question of scale is not really exploited.

### 1.5 Using other types of information

The approach in [34] consists of an integrated CNN system with additional hacking functionality using semantic information from the HSI. The approach [35] is to install a multi-scale CNN convolution. However, the usual metric learning has been implemented. This similarity leads to an obvious redundancy of the model and, therefore, to show it on the descriptive capacity of deep metrics.

From the results collected from these different works, we noticed that: the application of data reduction may hinder classification accuracy; most of the old architectures were applied on simple images (not hyperspectral) where the number of bands is limited to 3, therefore it cannot generate good results on HSI images (which have more than 100 spectral bands); some architectures work well but there are still some noises in the pixels (erroneous pixels); the CNN architecture applied to a great influence on the results, so, as each layer has its role, while the layers of pooling allow accelerating the convergence of the results, the layers of convolutions help to bring together the pixels to group them. As the number of convolutional layers increases, the accuracy also increases. In order to improve the classification based on these definitions, we have proposed in this article a novel classification architecture by CNN.

In this article, we present a new ExCNN-HSI architecture, improvement of a previous CNN-HSI architecture [1]. This method is done by adding global convolutional layers at the end of processing (on the classified image). This is due to solve the problem of limited number of samples, and to correct unknown or erroneous pixels. This also allows the detection of anomalies.

### 2 PROPOSED APPROACH (ExCNN)

In this section we present the applied work steps. We use the same architecture applied in SFE-CNN [25] and DCNN [26]. Thus, our network is composed of three layers of convolutions and ReLU, two layers of subdivision (MaxPooling), two layers of fully connected, and one layer of softmax. The peculiarity of this work is the addition of several layers of convolution on the whole image, at the end of the treatment chain, after the softmax layer. These additional layers allow the disappearance of erroneous pixels, generated by the limited number of samples (Figure 1).

#### 2.1 Convolution processing

For $l$ layers, the neuron $n$ of the convolution layer $l$ is calculated by Equation (1).

$$y_i^n = f\left( \sum_{m \in V_{i,l}} y_{i-1}^m \otimes \eta_{m,n} + \beta_i^n \right).$$

(1)

The $f_l$ presents the function of activating the layer, where $y_i^n$ input data, and the $V_{i,l}$ presents the list of all groups in the layer $l-1$. Finally, $\eta_{m,n}$ is the convolution kernels of the neuron $m$ in the layer $l-1$ highlight a group $n$ in the layer $l$; and $\beta_i^n$ is the bias. Biases are the simplifying hypotheses formulated by a model to facilitate the learning of the target function. The size of the output sub-block $y_i^n$ is $(b_1 \times c_1 + 1) \times (b_1 \times c_1 + 1)$ pixels with $(c_1 \times c_1)$ pixels the size of the convolution filters $\eta_{m,n}$ and $(b_1 \times c_1)$ pixels the size of the sub-block of input information $y_{i-1}^n$.

#### 2.1.1 Rectified non-linear units

For every sub-block apply Equation (2):

$$f(x) = \max(0, x).$$
(2)
2.2 | MaxPooling processing

For each subset of data, calculate the maximum of \( r \times c \) pixels, which will be multiplied by an adjustable weight before being added to a bias term. The result passes through an activation function to produce an output for the subset of data \( r \times c \). The output matrix \( z^{l-1}_n \) is calculated by Equation (3).

\[
\begin{align*}
z^{l-1}_n(i, j) &= \max\{y^{l-1}_n(ri-1, cj-1), \ldots, y^{l-1}_n(ri, cj-1), \ldots, y^{l-1}_n(ri, cj)\} \quad (3)
\end{align*}
\]

Let \( y^{l-1}_n \) be the result of \( r \times c \) pixels of each subset of data. The neuron \( n \) and the subdivision layer \( l \) is calculated by Equation (4).

\[
\begin{align*}
y^{l}_n &= f\left(\sum_{m=1}^{N_l} y^{l-1}_m \eta^{l}_m + \beta^{l}_n\right) \quad (4)
\end{align*}
\]

And the size of the output characteristics map is \( h/ \times w/ \), with \( h/ = h^{-1}/r \) and \( w/ = w^{-1}/c \).

2.3 | Full connected

Let \( l \) the exit layer and \( N^l \) the number of output neurons. The output of the neuron \( n \) is calculated by (5). The parameter \( \beta^{l}_n \) is the bias associated with the neuron \( n \) of the layer \( L \). \( \eta^{l}_m \) is the weight of the characteristic map \( m \) from the last convolutional layer, to the neuron \( n \) in the output layer.

\[
\begin{align*}
y^{l}_n &= f\left(\sum_{m=1}^{N^l} \eta^{l}_m x^{l}_m + \beta^{l}_n\right) \quad (5)
\end{align*}
\]

2.4 | Softmax

Let \( y \) be a vector of the inputs, and \( i \) represents the output units; so let us apply (6) for \( i = 1, 2, \ldots, n \):

\[
\begin{align*}
y^{i}_n &= f\left(\frac{\phi^{i}}{\sum_{j=1}^{N^l} \phi^{j}}\right) \quad (6)
\end{align*}
\]

2.5 | Last convolution layers

Use the same Equation (1), instead of applying the convolution on a lot of pixel and reducing the size each time, here we apply the convolution on the whole image and we keep the same size \( H \times W \). We tested our approach on 10 additional convolution layers, that is, we arrive at convolution layer number 13 (counting the 3 layers at the beginning).

2.6 | Classification

Create a temporary vector containing the Euclidean distance between the values known in the database and the value we wanted to classify.

Let \( x \) be the active pixel, \( y \) the values brought by the base. The simplified Euclidean Equation (7) calculates the distance
between the two points.

\[ D_{x_c} = \sqrt{x^2 - y^2}. \]  

(7)

Take the smallest values obtained and project as shown in Equation (8) the class \( C_x \) corresponding to the pixel \( x \).

\[ C_x = f(\min_{i} D_{x_i}). \]  

(8)

The execution time of this step depends on the size of the recognition base that we have. In addition, our network does not require a pre-requisite knowledge of the number of class of the image, it discovers them as and when processing. By this function we come to the end of our network. The resulting image is a very well classified 2D image.

3 | RESULTS AND DISCUSSIONS

In order to analyze water pollution in the Gabes Tunisia area, we mainly tested our approaches on hyperspectral imagery detected by the HyperionEO1 sensor. Then we validated our contributions on other public scenes downloaded from the website [36] which are: Salinas in California and Indian Pines in Indiana.

3.1 | Datasets

The three datasets applied for the tests are:

- **IndianPines** are composed of 220 spectral bands, reduced to 200 by removing the water absorption region. It is captured by AVIRIS, in northwestern Indiana, from wavelengths between 0.4 and \( 2.5 \times 10^{-6} \) m. Their sizes are from 145 \( \times \) 145 pixels and are composed of 16 Classes, namely: alfalfa, two types of corn, three different grass pastures, oat hay, several types of soybeans, and wood.

- **GulfofGabes** are composed of 125 spectral bands. They were captured by HyperionEO−1, in the region of Gabès in Tunisia, the spatial resolution of 30 m. Their sizes are from 100 \( \times \) 100 pixels and are composed of 7 Class, namely: water, soil, hydrated soil, and vegetation species.

- **SalinasA** are composed of 204 spectral bands were captured by the sensor AVIRIS, in the Salinas Valley region of California. The size of Salinas is 86 \( \times \) 83 pixels. It is made of 6 Classes namely: green broccoli 1, corn, romaine lettuce 4, romaine lettuce 5, romaine lettuce 6, romaine lettuce 7.

3.2 | Validation and discussion

Before starting, we randomly set the sub-block and filter sizes, which are shown in the Table 1.
The architecture of the network is as follows: \textit{subset of data} \rightarrow Conv \rightarrow ReLU \rightarrow Pool \rightarrow Conv \rightarrow ReLU \rightarrow Pool \rightarrow Conv \rightarrow ReLU \rightarrow FC \rightarrow FC \rightarrow Softmax \rightarrow Conv \times 10 \rightarrow Class. For the first tests, we applied a state-of-the-art method. This method showed good results, but several erroneous pixels. Then we started adding additional convolution layers at the end of the chain, applied over the entire image and not just on the block. We mean by $N \times X$ (Network) the following architecture: \textit{subset of data} \rightarrow Conv \rightarrow ReLU \rightarrow Pool \rightarrow Conv \rightarrow ReLU \rightarrow Pool \rightarrow Conv \rightarrow ReLU \rightarrow FC \rightarrow FC \rightarrow Softmax \rightarrow Conv \times X \rightarrow Class. The first column (indicated $N$) is the state-of-the-art architecture where $X = 0$.

We present the results class by class and at the bottom of Tables 2–4, we present the Cohen's Kappa ($K$) (the effectiveness of classification with respect to the random assignment of values), overall accuracy (OA) (the number of correctly classified pixels divided by the total number of reference pixels), average precision (AA) (the average of each precision per class (sum of the precision for each class / number of classes)) values for the two datasets, and the time (seconds).

From the experiments, we notice that the proposed method is very efficient for the different number of extra layers. However, in some cases, we notice that the lesser of layers (e.g. 3 extra layers) produces better results than the addition of 10 layers.

### 4 | VARIATION IN THE NUMBER OF SAMPLES

In this section, we have varied the number of samples of each class, on the proposed method ($ExCNN, x = 10$) (Tables 5–7).

From the experiments, we notice that the proposed method is effective for cases where the number of samples is very low; and it is very effective for a number of samples greater than 50%.

### 5 | DISCUSSIONS

In this last section, we tried to adapt state-of-the-art approaches (Table 8):
### TABLE 5  Classification results for each novel addition of the number of samples (NS) for the Indian Pines datasets

| Class/method | NS—50% | NS—60% | NS—70% | NS—80% | NS—90% | NS—100% |
|--------------|--------|--------|--------|--------|--------|---------|
| Class 1      | 0.99596 | 0.9953 | 0.99582 | 0.99572 | 0.99544 | 0.99562 |
| Class 2      | 0.87119 | 0.87017 | 0.86645 | 0.86733 | 0.86392 | 0.86416 |
| Class 3      | 0.90426 | 0.90642 | 0.91096 | 0.91385 | 0.91792 | 0.92105 |
| Class 4      | 0.97814 | 0.97849 | 0.97781 | 0.97788 | 0.97781 | 0.97746 |
| Class 5      | 0.95526 | 0.95594 | 0.95464 | 0.95487 | 0.9542  | 0.95405 |
| Class 6      | 0.93108 | 0.93305 | 0.93181 | 0.93159 | 0.93121 | 0.93056 |
| Class 7      | 0.9972  | 0.99724 | 0.99738 | 0.99734 | 0.99734 | 0.99734 |
| Class 8      | 0.95614 | 0.95632 | 0.95488 | 0.95478 | 0.95472 | 0.95453 |
| Class 9      | 0.998   | 0.99838 | 0.9981  | 0.99815 | 0.99819 | 0.9981  |
| Class 10     | 0.90866 | 0.90884 | 0.90684 | 0.90691 | 0.9072  | 0.90754 |
| Class 11     | 0.78037 | 0.7782  | 0.77577 | 0.77281 | 0.7689  | 0.76647 |
| Class 12     | 0.94468 | 0.94466 | 0.94428 | 0.944   | 0.94412 | 0.94359 |
| Class 13     | 0.98134 | 0.9809  | 0.98009 | 0.9804  | 0.9805  | 0.9805  |
| Class 14     | 0.88422 | 0.88412 | 0.8827  | 0.88165 | 0.87979 | 0.87967 |
| Class 15     | 0.9638  | 0.96396 | 0.96348 | 0.96304 | 0.96363 | 0.96328 |
| Class 16     | 0.99108 | 0.99136 | 0.99116 | 0.99126 | 0.99139 | 0.99115 |
| K            | 0.5179  | 0.6201  | 0.7167  | 0.8092  | 0.9067  | 1       |
| AA           | 0.5404  | 0.6520  | 0.7255  | 0.8257  | 0.9192  | 1       |
| OA           | 65.6029 | 73.0321 | 79.8335 | 86.4495 | 93.3841 | 100     |

| Time (s)     | 1.4665e+03 | 1.4599e+03 | 1.6793e+03 | 754.3672 | 925.2853 | 1.1913e+03 |

### TABLE 6  Classification results for each novel addition of the number of samples (NS) for the Gulf Gabes datasets

| Class/method | NS—50% | NS—60% | NS—70% | NS—80% | NS—90% | NS—100% |
|--------------|--------|--------|--------|--------|--------|---------|
| Class 1      | 0.069012 | 0.099542 | 0.12868 | 0.15651 | 0.17433 | 0.18567 |
| Class 2      | 0.94342 | 0.94085 | 0.93787 | 0.937   | 0.93554 | 0.9334  |
| Class 3      | 0.89366 | 0.89177 | 0.8904  | 0.88543 | 0.88279 | 0.8794  |
| Class 4      | 0.88445 | 0.88088 | 0.88173 | 0.87706 | 0.87318 | 0.8694  |
| Class 5      | 0.8789  | 0.87474 | 0.87301 | 0.86879 | 0.86416 | 0.861   |
| Class 6      | 0.90876 | 0.90698 | 0.90268 | 0.90097 | 0.89877 | 0.8956  |
| Class 7      | 0.81366 | 0.81231 | 0.80471 | 0.80008 | 0.79497 | 0.7892  |
| K            | 0.6074  | 0.6975  | 0.7800  | 0.8674  | 0.9365  | 1       |
| AA           | 0.6983  | 0.7751  | 0.8394  | 0.9040  | 0.9561  | 1       |
| OA           | 78.2600 | 83.0600 | 87.5100 | 92.3500 | 96.2800 | 100     |

| Time (s)     | 508.4154 | 509.1863 | 506.4380 | 547.0039 | 366.4432 | 366.6878 |

- CNN with spectral characteristics reductions: (1) [25](CNN(x = 0) by smart spectral feature extraction and reduction), (2) [26](Dual-CNN(x = 0) for deep extraction and classification).
- CNN with adaptive parameters: (3) [22](CNN(x = 0) by adaptive kernels and batches, with mean-pooling layers), (4) [32](CNN(x = 0) by adaptive kernels, batches and filters), (5) [1](CNN(x = 0) by adaptive kernels and batches, with Max-Pooling layers).

All these methods previously mentioned were re-calculated with the same initial parameters of this current article (Table 1: windowSize = 15 × 15, KernelsNumber = 10, KernelSize = 3 × 3, and SubdivisionFilter = 3 × 3) and spectral reduction by the mean for some cases ((3) [22], (4) [32], and (5) [1]).

From the experiments, we notice that the proposed method is very efficient for the different data tests, and is better than the methods of the state of the art, in precision, and sometimes in calculation time.
### Table 7: Classification results for each novel addition of the number of samples (NS) for the SalinasA datasets

| Class/method | NS—50%       | NS—60%       | NS—70%       | NS—80%       | NS—90%       | NS—100%      |
|--------------|---------------|---------------|---------------|---------------|---------------|---------------|
| Class 1      | 0.87189       | 0.87372       | 0.87704       | 0.88122       | 0.88495       | 0.89045       |
| Class 2      | 0.65945       | 0.65129       | 0.64418       | 0.6388        | 0.63275       | 0.6236        |
| Class 3      | 0.83675       | 0.83669       | 0.83438       | 0.83029       | 0.82799       | 0.8274        |
| Class 4      | 0.61377       | 0.60411       | 0.5961        | 0.59367       | 0.58113       | 0.57271       |
| Class 5      | 0.82119       | 0.82267       | 0.81807       | 0.81647       | 0.81301       | 0.81115       |
| Class 6      | 0.7934        | 0.78571       | 0.78356       | 0.78141       | 0.7780        | 0.77613       |
| K            | 0.5202        | 0.6256        | 0.7254        | 0.8123        | 0.9053        | 0.9998        |
| AA           | 0.5890        | 0.6803        | 0.7645        | 0.8378        | 0.9153        | 0.9999        |
| OA           | 60.3530       | 69.0670       | 77.3186       | 84.5055       | 92.1827       | 99.9860       |
| Time (s)     | 362.8061      | 361.7014      | 365.1198      | 370.8783      | 364.3180      | 357.7860      |

### Table 8: Comparison with state-of-the-art methods

| Datasets/method | ECNN (x = 0) | (1) | (2) | (3) | (4) | (5) | Proposed ECNN (x = 10) |
|-----------------|--------------|-----|-----|-----|-----|-----|------------------------|
| Indian Pines    | 71.7337      | 81.0131 | 96.0428 | 94.5493 | 93.8454 | 96.7087 | 100                   |
| Gulf of Gabes   | 87.8700      | 87.8600 | 88.9700 | 97.0700 | 87.2800 | 97.3900 | 100                   |
| SalinasA        | 77.1224      | 90.6556 | 93.1634 | 91.8605 | 89.7450 | 94.8585 | 99.9860               |
| (a) OA (%)      |              |       |       |       |       |       |                        |

| Datasets/method | ECNN (x = 0) | (1) | (2) | (3) | (4) | (5) | Proposed ECNN (x = 10) |
|-----------------|--------------|-----|-----|-----|-----|-----|------------------------|
| Indian Pines    | 1.1384e+03   | 778.3728 | 2.2785e+03 | 1.8843e+05 | 1.3428e+05 | 1.3315e+05 | 1.1913e+03           |
| Gulf of Gabes   | 510.3969     | 511.2224 | 879.5588 | 4.4462e+04 | 3.4945e+04 | 3.7507e+04 | 366.6878             |
| SalinasA        | 356.0504     | 484.2463 | 710.5860 | 1.6972e+04 | 1.6805e+04 | 1.3528e+04 | 357.7860             |
| (b) Time (s)    |              |       |       |       |       |       |                        |

### 6 CONCLUSIONS

In recent years, satellite imagery has become very important. It is used to meet the needs of security, environmental protection, and many object detection applications, due to the large amount of information provided. However, the limited availability of samples may reduce the quality of classification in processing hyperspectral images. In recent years, CNN present an efficient solution for image classification. In this paper, we propose a semi-supervised classification framework by extending the CNN classification structure to predict all bad or unknown pixels. These methods consist of increasing the number of convolutional layers at the end of the network, organizing the classification at 10 levels (each time a layer is added and tested). To validate our proposed designs, we tested them on three different hyperspectral image datasets of SalinasA, Gulf of Gabes, and Indian Pines. Therefore, we noticed the increase in precision with each addition of a layer, which validates our proposed approaches.

### ACKNOWLEDGEMENT

This work was supported by the Ministry of Higher Education and Scientific Research of Tunisia.

### ORCID

Maissa HAMOUDA [https://orcid.org/0000-0001-8596-4797](https://orcid.org/0000-0001-8596-4797)

### REFERENCES

1. Hamouda, M., Ettabaa, K.S., Boughlel, M.S.: Adaptive batch extraction for hyperspectral image classification based on convolutional neural network. In: International Conference on Image and Signal Processing. Springer, pp. 310–318 (2018)
2. Khan, M.J., et al.: Automated forgery detection in multispectral document images using fuzzy clustering. In: 2018 13th IAPR International Workshop on Document Analysis Systems (DAS) IEEE (2018)
3. Khan, M.J., et al.: Automatic target detection in satellite images using deep learning J. Space Technol. 7(1), 44–49 (2017)
4. Chen, C., Ma, Y., Ren, G.: Hyperspectral classification using deep belief networks based on conjugate gradient update and pixel-centric spectral block features. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 13, 4060–4069 (2020)
5. Khan, M.J., Khurshid, K., Shafait, F.: A spatio-spectral hybrid convolutional architecture for hyperspectral document authentication. In, IEEE (sep 2019)
6. Ahmad H.M., Khan M.J., Yousaf A., Ghuffar S., Khurshid K. Deep Learning: A breakthrough in medical imaging. Current Medical Imaging 16, (8), 946–956 (2020).
7. Yousaf A., Khan M.J., Khan M.J., Siddiqui A.M., Khurshid K. A robust and efficient convolutional deep learning framework for age-invariant face recognition. Expert Systems 37, (3), e12503 (2020).
8. Khan, M.J: Deep learning for automated forgery detection in hyperspectral document images. J. Electron. Imaging 27(05), 1 (2018)
9. Shafer, M.A., et al.: Hyperspectral image classification using deep learning technique. In: Joint European-US Workshop on Applications of Invariance in Computer Vision, Springer, pp. 334–342 (2020)
10. Islam, A.U, et al.: Hyperspectral image analysis for writer identification using deep learning. In: 2019 Digital Image Computing: Techniques and Applications (DICTA), IEEE (2019)
11. Khan, M.J., et al.: Modern trends in hyperspectral image analysis: A review. IEEE Access 6, 14 118–14 129 (2018)
12. Ozdemir, A., Polat, K.: Deep learning applications for hyperspectral imaging: A systematic review. J. Inst. Electron. Comput. 2(1), 39–56 (2020)
13. Liu, Q., et al.: Bidirectional-convolutional lstm based spectral-spatial feature learning for hyperspectral image classification. Remote Sens. 9(12), 1330 (2017)
14. Hamouda, M., Ettabaa, K.S., Bouhlel, M.S.: Framework for automatic selection of kernels based on convolutional neural networks and k-means clustering algorithm. Int. J. Image Graph. 19(04), 1950019 (2019)
15. Fotiadou, K., Tsagkatakis, G., Tsakalides, P.: Deep convolutional neural networks for the classification of snapshot mosaic hyperspectral imagery. Electron Imaging 2017(17), 185–190 (2017)
16. Yang, J., Zhao, Y.-Q., Chan, J.C.-W.: Learning and transferring deep joint spectral–spatial features for hyperspectral classification. IEEE Trans. Geosci. Remote Sens. 55(8), 4729–4742 (2017)
17. Santara, A., et al.: BASS net: Band-adaptive spectral-spatial feature learning neural network for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. 55(9), 5293–5301 (2017)
18. Li, X., et al.: Gell classification using convolutional neural networks in medical hyperspectral imagery. In: 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE (2017)
19. Luo, F., et al.: Dimensionality reduction with enhanced hybrid-graph discriminant learning for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. 58(8), 5336–5353 (2020)
20. Luo, F., et al.: Sparse-adaptive hypergraph discriminant analysis for hyperspectral image classification. IEEE Geosci. Remote Sens. Lett. 17(6), 1082–1086 (2020)
21. Lorenzo, P.R., et al.: Hyperspectral band selection using attention-based convolutional neural networks. IEEE Access 8, 42 384–42 403 (2020)
22. Hamouda, M., Ettabaa, K.S., Bouhlel, M.S.: Modified convolutional neural network based on adaptive patch extraction for hyperspectral image classification. In: 2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) IEEE, pp. 1–7 (2018)
23. Liu, W., Lee, J.: An efficient residual learning neural network for hyperspectral image superresolution. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12(4), 1240–1253 (2019)
24. Mei, S., et al.: Unsupervised spatial-spectral feature learning by 3d convolutional autoencoder for hyperspectral classification. IEEE Trans. Geosci. Remote Sens. 57, 6808–6820 (2019)
25. Hamouda, M., Ettabaa, K.S., Bouhlel, M.S.: Smart feature extraction and classification of hyperspectral images based on convolutional neural networks. IET Image Processing 14(10), 1999–2005 (2020)
26. Hamouda, M., Bouhlel, M.S.: Dual convolutional neural networks for hyperspectral satellite images classification (DCNN-HSI). In: 27th International Conference on Neural Information Processing (ICONIP 2020) (2020)
27. LeCun, Y., et al.: Gradient-based learning applied to document recognition. Proc. IEEE 86(11), 2278–2324 (1998)
28. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Adv. Neural Inform. Process. Syst. 1, 1097–1105 (2012)
29. Hamouda, M., Ettabaa, K.S., Bouhlel, M.S.: Automatic selection of kernels to classify hyperspectral imagery based on convolution neural network and clustering algorithm. In: Proceedings of the 31st International Business Information Management Association Conference. (Apr. 2018)
30. Phung, S.L., Bouzerdoum, A.: Visual and audio signal processing lab. University of Wollongong. (2009). https://documents.uow.edu.au/docs/cnn-matlab/cnn-matlab.pdf
31. Feng, J., et al.: CNN-based multilayer spatial–spectral feature fusion and sample augmentation with local and nonlocal constraints for hyperspectral image classification. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12(4), 1299–1313 (apr 2019)
32. Hamouda, M., Ettabaa, K.S., Bouhlel, M.S.: Hyperspectral imaging classification based on convolutional neural networks by adaptive sizes of windows and filters. IET Image Process. 13(2), 392–398 (2018)
33. Guo, Y., et al.: High efficient deep feature extraction and classification of spectral-spatial hyperspectral image using cross domain convolutional neural networks. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12, 345–356 (2019)
34. Yu, C., et al.: Hyperspectral image classification method based on CNN architecture embedding with hashing semantic feature. IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 12, 1866–1881 (2019)
35. Gong, Z., et al.: A CNN with multiscale convolution and diversified metric for hyperspectral image classification. IEEE Trans. Geosci. Remote Sens. 57(6), 3599–3618 (2019)
36. GHC: Hyperspectral remote sensing scenes. Grupo de Inteligencia Computacional (2014)

How to cite this article: Hamouda M, Bouhlel MS. Modified Convolutional Neural Networks Architecture for Hyperspectral Image Classification (Extra-Convolutional Neural Networks). IET Image Process. 2021;1–8. https://doi.org/10.1049/ipr2.12169