Adaptive Batch Extraction for Hyperspectral Image Classification Based on Convolutional Neural Network

Maissa Hamouda[1(✉)], Karim Saheb Ettabaa[2], and Med Salim Bouhlel[1]

[1] SETIT, Sfax, Tunisia
maissa_h@yahoo.fr, medsalim.bouhlel@enis.rnu.tn
[2] IMT Atlantique, Brest, France
karim.sahebettabaa@riadi.rnu.tn

Abstract. Deep Learning for Hyperspectral Imaging Classification is a wonderful solution, despite a few fuzzification. Conventional neural networks are very effective for classification tasks which have allowed them to be used by a very large companies. In this paper, we present an approach to initialize the convolutional data: Firstly, an adaptive selection of kernels by a clustering algorithm; Secondly, by the definition of adaptive batches size. In order to validate our proposed approach, we tested the algorithms on three different hyperspectral images, and the results showed the effectiveness of our proposal.

Keywords: Hyperspectral imaging · Feature extraction · Convolutional codes Neural networks · Machine learning

1 Introduction

Hyperspectral Imaging is an advanced technology that solves many classification and identification problems. Thus, the large number of spectral and spatial information complicates its classification. Hyperspectral imaging classification domain is very broad (spatial and/or spectral feature extraction or other, segmentation, band reduction, anomaly detection, large image storage, etc.) [1–4]; There is an infinity of treatment algorithms, among the best we have found CNN. Convolutional neural networks are characterized by their excellent performance. There are several methods, as mentioned in [5]: logical regression, Perceptron Multilayer and Autoencoders Stacked Denoising, etc.; The neural network parameters, the number of neurons and the number of hidden layers, etc., can vary and can greatly influence the classification results. In this paper, we present an initialization approach for data to be processed by the convolutional neural networks. First, we adaptatively select kernels by a clustering algorithm. Next, we calculate the size of the adaptive batches. Extracted batches are processed normally in the CNN. in the last section, we present the test results, applied to three different hyperspectral images.

A. Mansouri et al. (Eds.): ICISP 2018, LNCS 10884, pp. 310–318, 2018.
https://doi.org/10.1007/978-3-319-94211-7_34

2 Proposed Approach

Convolutional neural networks have shown their effectiveness in recent years. In fact, it is characterized by the repetitive processing of the convolution and pooling layers until the fully connected layer is obtained. On the other hand, the hyperspectral image is a cubic image, composed of more than a dozen spectral bands, each of them providing information about the image. We extract each band separately. Then we extract each of the batches of information, which is going through the different stages of CNN. The problem is how to choose the right numbers of kernels? their sizes? their positions? The batches size? The proposed method is shown schematically in Fig. 1.

Fig. 1. Overall architecture of the proposed approach

2.1 Kernel Selection

To select the right position of kernels, first, a feature map is created containing the positions of the relevant pixels (kernels). To adaptively select the kernels, we propose to apply the algorithm CKmeans [6, 7]. CKmeans purposes to stake the $X = \{x_1, x_2, \ldots, x_n\}$ ambiguous pixels in p clusters with μ_{ij} are the x_i membership degree, that belongs to the j^{th} cluster. The clustering result is expressed by the membership degree on the matrix μ.

Before all, initialize the parameters: n: number of training data; p: number of clusters; and m: fuzzification parameter, representing the width of the p dimensional cluster perimeter.

For the p initialization, lets applying the Euclidean division in relative integers: for two integers n and p, with p different of zero, Euclidean division associates n quotient m and n remainder r, both integers, satisfying: $n = pm + r$ where $0 \leq r < |p|$. In this case, r should be minimized, to cover as ample space as possible. The three parameters n, p, and m are all positive, and $m \in [1.25, 2]$.

Algorithm steps are as follows; Firstly, take each band separately, one by one, then generate Xi vectors of data. The i from 1 to n data:

ALGORITHM 1: CKmeans Algorithm

(1) Generate p; it is done after fixing the fuzzification parameter m, a random number chosen in the interval already fixed at the top.

(2) Create a new matrix φ; it is done after setting the μ (membership degree).

$$\varphi_{ij} = max\left(\left[\frac{\mu_{ij}}{\max_{l=1}^{p} \mu_{jl}}\right], \left[\frac{\mu_{ij}}{\max_{l=1}^{n} \mu_{lk}}\right]\right) \tag{1}$$

To calculate the φ matrix, follow the next steps: (i) First, initialize μ, by random values among 0 and 1, their sum in each row or column is equal to 1. (ii) Then, in a new matrix, assign the value 1 to the position of the largest number in such a line of the matrix μ, and the others get the value 0, so that at each line get the sum equal to 1 (only 1 per line). (iii) Finally, create a vector, containing the sum of the values in each column; and if a column contains no 1, give it a value of 1; therefore, this vector contains values greater than or equal to one. And move on to calculating the centroid.

(3) Calculate c_{ij} the centroid of cluster j.

$$c_j = \frac{\sum_{i=1}^{n} x_i \varphi_{ij}}{\sum_{i=1}^{n} \varphi_{ij}} \tag{2}$$

(4) Calculate an initial value for J_A.

$$J_A = \sum_{i=1}^{n}\sum_{j=1}^{p} \mu_{ij}^{m} d_{ij}(x_i, c_j)^2 \tag{3}$$

(5) Calculate the table of the fuzzy membership function μ_{ij}.

$$\mu_{ij} = \frac{\left(\frac{1}{d_{ij}(x_i, c_j)}\right)^{\frac{2}{m-1}}}{\sum_{k=1}^{p}\left(\frac{1}{d_{ik}(x_i, c_k)}\right)^{\frac{2}{m-1}}} \tag{4}$$

(6) Calculate the distance d_{ij} between x_i and c_j; hear used the Euclidean distance.

(7) Verification of the criteria for stopping; after fixing $\varepsilon > 0$ (In our approach, have chosen $\varepsilon = 0.001$);

$$d_{ij}(J_{A-1}, J_A) \leq \varepsilon \tag{5}$$

So, if J_A is the objective function calculated in the previous iteration and J_{A-1} is the objective function of the last iteration; or an already fixed number of iterations is reached, move to (7) otherwise return to (2). Finally, the points j with the distance value J_A are adaptatively chosen as the kernels and thus the number of kernels is also adaptatively determined through the thresholdJ_A.

According to the previous instructions, we get a new clustering matrix, from which we generate a binary matrix indicating the kernels positions. We set a threshold parameter θ, in [2, 8], which presents the number of neighboring similar pixels to the middle pixel (active pixel). For each pixel X, for $i \in [1, 8]$, if $P_i == X$, Neighbors value ℵ++. Then, if $\aleph \geq \theta$, so X is a cluster center, and it must be marked by 1, and 0 otherwise.

2.2 Batch Size Calculation

Let μ the threshold of neighbors to consider, and 4 the numbers of the kernels nearest neighbors (this number depends on the size of the image). For each pixel, select $V = \{v1, v2, v3, v4\}$ nearest kernels neighbors, by calculating the distance between each pixel with all the other kernels and choose the smaller values.

Calculate the median among the active pixel and each of its neighbors. let $W = \{w1, w2, w3, w4\}$, so

$$w_i(x, y) = \left(\frac{x_{v_i} + x_k}{2}; \frac{y_{v_i} + y_k}{2}\right) \tag{6}$$

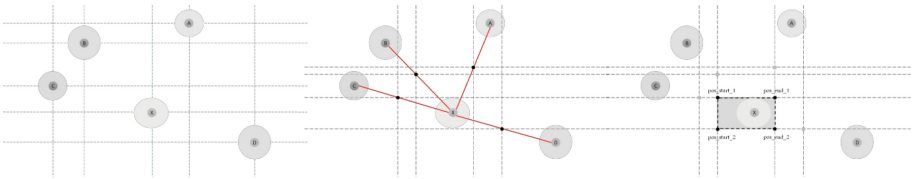
Where x and y present the ordinate and the abscissa of the point successively, and k represents the active pixel.

After selecting new points, select the lines parallel to the x and y-axes passing through each of these points. For that, use constant affine functions, $f(x) = ax + b$ with $a = 0$ and b is the coordinates of the neighboring kernels. So we get four horizontal lines L1, L2, L3 and L4; and four vertical lines (columns) C1, C2, C3, and C4.

For the four new selected points, identify the nearest rows and columns of our active pixel. The intersection of these represents the coordinates of the batch to extract (see Fig. 2). Space is divided into four parts up, down, left and right of the active pixel.

ALGORITHM 2: Iterative Algorithm

for each Ci,
if (Ci $< Ck$)
Cl ← min distance (Ck; Ci)
select left column (Cl)
if (i $> Ck$)
Cr ← min distance (Ck; Ci)
select a right column (Cr)
for each Li,
if (Li $< Lk$)
Lu ← min distance (Lk, Li)
select up the line (Lu)
if (Li $> Lk$)
Ld ← min distance (Lk; Li)
select downline (Ld)



(a) (b) (c)

Fig. 2. (a) Selection of nearest neighbors; (b) Calculation of medians; (c) Coordinates batch selection

The new positions are;
$P_1 = Lu \cap Cl$; $P_2 = Lu \cap Cr$; $P_3 = Ld \cap Cl$ and $P_4 = Ld \cap Cr$;

$$\text{So, } P_i(x, y) = \left(x_{L_{\{u;d\}}} + x_{C_{\{r;l\}}}; y_{L_{\{u;d\}}} + y_{C_{\{r;l\}}}\right) \tag{7}$$

2.3 CNN Treatment

Convolution layer adds to each pixel of the image its local neighbors, weighted by the chosen mask. The central element of the mask is placed on the active pixel. The active pixel is replaced by a weighted sum of the neighbor's pixels and itself.

Let $H_0 \times W_0$ is the input image size (that is the size of the batch B_i), and L the number of layers, where α is a positive integer. Characteristics map n of convolution layer l is calculated as:

$$y_n^l = f_l\left(\sum_{m \in V_n^l} y_m^{l-1} \otimes w_{m,n}^l + \beta_n^l\right) \tag{8}$$

The symbol $\otimes$ presents convolution operator. f_l is activation function of the layer, β_n^l the Bias for characteristics map B in max-pooling layer S^l, and $S^1, S^3 \ldots, S^{2a}$ max-pooling layers. V_n^l presents the list of all the bands in the layer $l-1$ which are connected to the characteristics map n. Finally, $w_{m,n}^l$ is convolution mask from the characteristics map m in layer S^{l-1} to characteristics map n in layer C^l.

The size of the output characteristics map y_n^l is

$$\left(H^{l-1} - r^l + 1\right) \times \left(W^{l-1} - c^l + 1\right) \text{Pixels} \tag{9}$$

Where $r^l \times c^l$ pixels are the size of convolution masks $w_{m,n}^l$, and $H^{l-1} \times W^{l-1}$ pixels are the size of the input characteristics maps y_m^{l-1}. The following image represents an explanatory example of the application of convolution mask on an image.

The max pooling layer divides the batch by making out the maximum of each pixel group, according to a given mask size. And this reduces at least half of its size.

In our work, we applied masks of size 2 x 2. Then the computation equation is as follows;

$$y_n^l(i,j) = \text{Max}\left(y_n^{l-1}(2i-1, 2j-1); y_n^{l-1}(2i-1, 2j); y_n^{l-1}(2i, 2j-1); y_n^{l-1}(2i, 2j)\right) \tag{10}$$

Let $H^{l-1} \times W^{l-1}$ is the input batch size, so the size of the output feature map y_n^l is $H^l = H^{l-1}/2$ and $W^l = W^{l-1}/2$.

In the last convolution layer, each band is connected to exactly one preceding characteristics map. It uses convolution masks that have the same size as its input characteristics maps.

The output layer, called L, is constructed from sigmoidal neurons. Let N^L is the number of sigmoidal output neurons. The corresponding equation to compute y_n^L, the output of sigmoidal neuron n, is:

$$y_n^L = f^L\left(\sum_{m=1}^{N^{L-1}} y_m^{L-1} w_{m,n}^L + \beta_n^L\right) \tag{11}$$

Where β_n^L is the bias associated with the neuron n of the L layer, and $w_{m,n}^L$ is the weight of the characteristic map m of the last convolutional layer, at the neuron n of the output layer. Finally, to calculate the outputs of all the sigmoidal neurons creating the outputs of the network:

$$y = \left[y_1^L, y_2^L, \ldots y_{N_L}^L\right] \tag{12}$$

3 Experiences and Discussion

To prove the effectiveness of our proposed method, we tested our algorithm on three different datasets, SalinasA, Pavia University and Indian Pines, and compared them with other algorithms and situations.

3.1 Datasets

Datasets (1) SalinasA is a small excerpt from the Salinas hyperspectral image and is characterized by its high spatial resolution, 3.7 meters of pixels. Its 204 spectral bands taken by the AVIRIS sensor, in the Salinas Valley area of California. SalinasA is 86×83 pixels in size and is constructed of six classes.

Datasets (2) Indian Pines is composed of 220 spectral bands, reduced to 200 by the removal of water absorption region. It is captured by AVIRIS, in the northwestern Indiana area. Its size is 145×145 pixels and is constructed of sixteen classes.

Datasets (3) University of Pavia is composed of 103 spectral bands. It is captured by ROSIS, in the area of Pavia in Italy, of spatial resolution 1.3 m. Its size is 610×610 pixels, a few of which are not sampled and can be removed from the analysis and is composed of nine classes.

3.2 Experimental Results

For experiments, from each hyperspectral image, we extract the spectral bands. For each of the bands, we compute the grouping matrix, then from each matrix, we generate a new feature map in the form of a binary matrix that specifies the positions and the number of kernels. We extracted a separate feature from the feature map.

We did two different tests: the first by applying fixed size batches (see Table 1). Then, in the second tests, we extracted an adaptive batches sizes; whose size is

Table 1. Batch size, convolution masks size, and max-pooling mask size chosen for each dataset.

Datasets	SalinasA	Indian pines	Pavia University
Batch size	14 × 14	21 × 21	36 × 36
Conv. mask	7 × 7	6 × 6	6 × 6
Max-pooling mask	6 × 6	5 × 5	5 × 5

calculated according to the dimensionality of the input image and the number of generated kernels.

To test the effectiveness of the proposed methods, we first compare the results with a manually selected number of kernels. The tests are applied to the three datasets: SalinasA, Indian Pines, and Pavia University.

Before we start, we have to mention that we have to initialize the number of kernels, so we select it randomly as long as we do not exceed the dimensions of the image; We also let the system choose a random kernel threshold in [2, 8]. In the second tests, for the other parameters, our algorithms calculate the size of the batch, according to the number of kernels calculated and the size of the input band; The average number of layers performed in our three examples is generally between 1 and 3.

First Tests: Fixed Batches Size

We started by testing our method with batches of fixed sizes. we applied the parameters noted in Table 1. We defined by Mk (Manual kernel) and by AutoK (Automatic selection of kernel). The obtained results are written in Table 2.

Table 2. Variation of the results according to the number of kernels used, applied to the three types of hyperspectral data (by fixed batches)

Methods	Datasets (1)	Datasets (2)	Datasets (3)
Mk35	86,36	87,28	79,33
Mk40	86,77	87,49	79,41
Mk45	86,15	87,69	79,73
Mk50	85,52	87,89	80,04
Mk55	85,31	87,51	80,36
Mk60	85,2	87,46	80,43
Mk65	85,14	87,41	80,51
Mk70	85,11	87,36	80,59
Mk75	85,07	87,31	80,67
Mk80	85,05	87,22	80,41
AutoK	86,77	87,89	80,67

In the first experiments, we tested our results with a number of manually selected kernels, which are 35, 40, 45, 50, 55, 60, 65, 70, 75 and 80. We have applied these tests to the three datasets already mentioned above. The results obtained are noted in Table 2. We noticed each time that the best precisions obtained are, using the kernels

of 40, 50 and 75, successively for the data Salinas-A, Indian Pines, and Pavia University. These results are consistent with the results obtained automatically by the CKmeans application. This implies that our method has succeeded in selecting the best number of kernels to use.

Seconds Tests: Adaptatively Batches Size
In the second part, we did the same tests, but by applying adaptive size batches. the results obtained are saved in Table 3.

Table 3. Variation of the results according to the number of kernels used, applied to the three types of hyperspectral data (by adaptive batches)

Methods	Datasets (1)	Datasets (2)	Datasets (3)
Mk35	88,82	89,77	81,84
Mk40	89,23	89,98	81,92
Mk45	88,61	90,18	82,24
Mk50	87,98	90,38	82,55
Mk55	87,77	90	82,87
Mk60	87,66	89,95	82,94
Mk65	87,6	89,9	83,02
Mk70	87,57	89,85	83,1
Mk75	87,53	89,8	83,18
Mk80	87,51	89,71	82,92
AutoK	89,23	90,38	83,18

Again, we have noticed that the best precisions obtained are, using the kernel of 40, 50 and 75, successively for the data Salinas-A, Indian Pines, and the University of Pavia; These results are consistent with the results obtained automatically by the CKmeans application. These results are also better than the accuracies obtained by applying batches of fixed size. this implies that the use of variable batches is a solution to improve the accuracy of the classification.

3.3 Discussions

In Table 4, we compare the classification results obtained by three different algorithms which are K-Means, PCA, Random, (1) and (2). Where (1) the method we tested in the first place, applying the clustering algorithm for calculating the number of kernels. (2) is the application of the steps in (1) but using batches of variable sizes (this is our proposed approach).

Table 4. Comparison of results with other methods

	K-Means	PCA	Random	(1)	(2)
Salinas-A	83,29	79,96	84,96	86,77	89,23
Indian Pines	85,03	87,31	87,13	87,89	90,38
Pavia U	79,82	80,19	80,52	80,67	83,18

Obtained result by testing on the three types of images implies that our proposed approaches are very effective for hyperspectral imaging classification in deep learning.

4 Conclusion

Conventional neural networks are very effective for classification tasks, and especially for complicated or large images such as hyperspectral images. In this paper, we presented a method for initializing data to convolute, by two steps: (1) the adaptive selection of kernels by a clustering algorithm, and (2) the definition of adaptive size batches around pixels. Extracted information batches pass through the different convolution and pooling layers, in order to obtain at the end a simple vector to classify. In order to validate our approach, we tested the algorithms on three different hyperspectral images, and the results showed the effectiveness of our proposal.

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

References

1. Bennour, A., Tighiouart, B.: Automatic highresolution satellite image registration by the combination of curvature properties and random sample consensus. In: 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (2012)
2. Essid, H., Abbes, A.B., Farah, I.R., Barra, V.: Spatiotemporal modeling based on hidden Markov model for object tracking in satellite imagery. In: 6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (2012)
3. Salem, M.B., Ettabaâ, K.S., Bouhlel, M.S.: Anomaly detection in hyperspectral images based spatial spectral classification. In: 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (2016)
4. Loussaief, S., Abdelkrim, A.: Machine learning framework for image classification. In: 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (2016)
5. Jdira, M.B., Imen, J., Kaïs, O.: Study of speaker recognition system based on Feed Forward deep neural networks exploring text-dependent mode. In: 7th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (2016)
6. Vargas, R.R., Bedregal, B.R.C., Palmeira, E.S.: A comparison between KMeans, FCM and CKmeans Algorithms. In: Theoretical Computer Science (2011)
7. Philipp, A., Beck, C., Esteban, P., Kreienkamp, F., Krennert, T., Lochbihler, K., Lykoudis, S. P., PiankoKluczynska, K., Post, P., Alvarez, D.R., Spekat A., Streicher, F.: COST733CLASS v1.2 User guide. Technical report (2014)