Università deli Studi dell'Insubria
Dipartimento di Scienze Teoriche e Applicate
Ph.D. Course in Computer Science

Unsupervised Feature Learning using Self-Organizing Maps

Advisor: Dr. Ignazio Gallo

Ph.D. Major thesis of
Marco Vanetti

Academic Year 2011-2012

To Laura,
for giving me the best days of my life.

Contents

Acknowledgements **ix**

List of symbols **xi**

1 Introduction **1**

2 Unsupervised Feature Learning **7**

2.1 Introduction 7

2.2 Unsupervised Training 8

2.3 Feature Learning Unit Composition 15

2.4 Summary 17

3 Unsupervised Texture Descriptor **19**

3.1 Introduction and State of the Art 19

3.2 The Proposed Configurations 23

3.3 Experiments 25

3.4 Summary 35

4 Image Classification **37**

4.1 Introduction and State of the Art 37

4.2 Image Representation 39

4.3 Experiments and Analysis 40

4.3.1 Standard Classification Methods 41

4.3.2 FLU Configuration 44

4.3.3 Reducing the Feature Size 45
4.3.4 Effect of Color, Local Image Normalization and Receptive Field Size 46
4.3.5 Other datasets 48
4.4 Summary 54

5 Conclusions **55**

List of Figures

1.1 This figure shows 16 images containing 8 classes of objects. Images are taken from the Caltech-256 dataset. 3

1.2 Two texture mosaics each one composed of 5 distinct textured regions (a and b) and the solution to the segmentation problem (c). . 4

2.1 Scheme used to represent a Feature Learning Unit. (a) Configuration for the training, using a set of input images. (b) Configuration for the remapping of a single image. 9

2.2 Schemes for the 1-dimensional (a) and the 2-dimensional (b) versions of the SOM neural network. 10

2.3 Schema representing the image remapping strategy, performed using a 2-dimensional SOM. 13

2.4 (a) A sample textured image and a receptive field of 20×20 pixels. (b) The same image with a 10-pixel padding obtained using a mirror strategy. 14

2.5 (a) Two FLU in series. (b) Parallel composition of two FLUs and a third FLU that aggregates the results. 16

3.1 Segmentation between two areas with different textures obtained using the proposed descriptor. Segmentation border is depicted with a white line. 20

3.2 The FLU configurations used as texture descriptor. (a) intensity configuration called *Config. A*. (b) histogram configuration called *Config. B*. 24

3.3 (a) Input image, a mosaic composed of 5 textures. (b)(c)(d) Two channels images remapped by FLU_1 (b), FLU_2 (c) and FLU_3 (d). (e) From the left, the final segmentation, the ground truth segmentation and the segmentation error map. Wrong pixels are shown in black. 26

3.4 Results obtained on the texture mosaic in Figure 3.3(a) by [1] with (a) and without (b) boundary region localization. (c) Our result for comparison. 27

3.5 On the right, a false-color image representing the feature extracted by the FLU_3 and, on the left, a detail of the boundary between three textures. 27

3.6 (a) Texture mosaic composed of 5 textures. (b)(c)(d) Two channels images remapped by FLU_1 (b), FLU_2 (c) and FLU_3 (d). (e) From the left, the final segmentation, the ground truth segmentation and the segmentation error map. Wrong pixels are shown in black. . . . 29

3.7 Comparison of results for the texture mosaic in Figure 3.6. (a) Results obtained using the full method proposed by [2] and (b) using only a single Gabor magnitude distribution. (c) Our result for comparison. 30

3.8 Segmentations obtained using different subsets of the 3-layers architecture proposed in *Config. A*. 31

3.9 (a) Three synthetically created texture-non texture mosaics. (b) Segmentation results obtained using the features extracted by the *Config. A*. 32

3.10 Sample images taken from the Mosaic-5 dataset and the ground truth segmentation image. 33

4.1 Encoding of a pyramidal histogram feature with 3 levels using a 4-neurons FLU. Each red square yields a 4-bins histogram. Each 4-bins histogram represents the activation of the four neurons when we slide the receptive field on the local area surrounded by the red square. 40
4.2 Some example images extracted from the CIFAR-10 dataset. 41
4.3 Overall accuracy obtained using the *icon classifier* with pixel intensities and colors. 42
4.4 Overall accuracy obtained using the PHOG based classifier, with and without colors. 43
4.5 Overall accuracy obtained using a pyramidal histogram on RGB pixel values and varying the number of the bins. 44
4.6 Overall accuracy obtained varying the number of neurons of the FLU and the pyramidal histogram levels. 45
4.7 Overall accuracy obtained using applying the topological grouping to reduce the number of bins in the histogram. In this test we used a 1 Level pyramidal histogram. 47
4.8 Effect of color, local brightness and contrast normalization for a 64-neurons FLU. 48
4.9 Effect of receptive field size in a 64 and 128-neurons FLUs, varying the number of pyramid levels. In this test we used color and local brightness and contrast normalization. 49
4.10 Weights plot obtained from a 64-neurons FLU trained with color $\theta = \{6, 6\}$ receptive fields. Effects of training with (a) and without (b) local contrast and brightness normalization of patches. Notice that the features extracted, plotted from top-left to bottom-right, are topologically ordered. 50
4.11 Confusion matrix obtained with a 128-neurons FLU, color, local contrast/brightness normalization and a $\theta = \{4, 4\}$ receptive field. . 51

List of Tables

3.1 Segmentation results obtained using different features and subsets of the 3-layers architecture proposed in Figure 3.2(a). 31
3.2 Mean segmentation error obtained with the proposed configurations on Mosaic-5 dataset. 35

4.1 Classification results obtained with five different datasets. In the last column the difference in OA% between the classification carried out using the PHOG feature and the proposed method. 52
4.2 Classification results obtained with other state of the art image classification methods. 53

Acknowledgements

I would like to express my deep gratefulness to Professor Ignazio Gallo, my research supervisor, for its guidance and encouragement during my research work.

I would also like to thank 7Pixel, the company that funded my PhD and the 7Pixel's CEO Nicola Lamberti, for his deep and vivid interest in scientific research.

My special thanks goes to my fellow Ph.D. students and friends in the department: Angelo Nodari, Valentina Pedoia, Moreno Carullo, Simone Albertini, Lorenzo Bossi and Alessandro Zamberletti, thank you for providing a cheerful yet challenging environment for working and learning.

My special thanks are extended also to all my co-workers in 7Pixel and, in particular, to: Marco Polita, Alessandro Cavallaro, Antonio Tirendi, Roberto Zanon and Andrea Broglia, I want to thank you for giving me technical advices but also for having lifted up my mood.

Finally, I wish to thank my parents, Alessandro and Antonietta, my brother Luca, my girlfriend Laura and all my family for their encouragement and support throughout my study.

Marco Vanetti
Induno Olona, December 2012

List of symbols

$\mathbf{b}$	Input vector, presented to the SOM.
$\mathbf{w_i}$	SOM neural weights for the neuron with index i.
k	Index of the winning neuron, it can be obtained after an input pattern $\mathbf{b}$ is presented to the SOM.
R, C	Size in neurons of the SOM lattice (Rows and Columns).
m	Number of neurons in the SOM.
$\mathbf{q_i}$	Vector containing the coordinates of the neuron i within the SOM lattice.
W, H	Size in pixels of the input image (Width and Height).
α	Learning rate, used during the SOM training.
σ	Standard deviation of the gaussian function, used during the SOM training.
$I(x, y)$	Intensity of the pixel with coordinates x, y within the image I.
I_0	Input image, coming from the dataset.
FLU_i	Feature Learning Unit with index i.
$I_i, (i > 0)$	Resampled image, obtained as output of FLU_i.
$I_{i,j}$	Image obtained concatenating the two channels of images I_i and I_j.
$\theta = \{\theta_W, \theta_H\}$	Size in pixels of the receptive field (Width and Height).
f_i	Value of the bin i in the histogram f.
L	Number of levels of the pyramidal histogram.

1

Introduction

When we want to solve a complex Artificial Intelligence (AI) problem like the automatic categorization or labeling of images, the recognition of objects classes inside an image or the conversion of speech into written text, a typical approach is to transform the interest data from the low-level representation used to store it in the machine into a high-level representation, called feature space. Although this definition can be considered unnatural to the eyes of a layman, human beings almost always describe objects using a more or less complex feature space.

Think of a customer in a clothing store that wants to buy a new coat and ask help to the sales assistant. To be helpful, probably the sales assistant would ask the customer some information about the garment like the color, the size, the material or the shape. Let suppose now that the customer find a coat that he really likes and he takes a photo of it with his smartphone. The content of the jpeg file saved on the flash memory of the smartphone is far from the description that the

customer gave to the sales assistant in order to find the coat. This happens because the jpeg file is a low level representation of the coat and, although is indispensable to display it on a monitor or to print it on paper, it can be hardly used directly to classify the color or the shape of the garment. A further feature extraction phase is thus required to create a semantic description of the object.

We are completely unaware of the process that, inside our brain, extracts high-level information from the images allowing us to make a clear and detailed description of the objects with which we interact. The same argument can be made for the sounds that we hear, tastes that we feel or scents that we smell, although the view is for the majority of us the most pervasive sense.

Now we are convinced that the feature extraction phase plays a key role in the semantic analysis of data, but the question is: given a specific AI problem, which are the best features? The answer to this question depends very much on our test configuration and, in particular, on some issues such as how many examples our dataset have, how many the distribution of examples in the dataset reflects the distribution of the data in the real world, how good is our evaluation metric, etc. Let's suppose that our dataset and our evaluation metrics have been chosen well. Given an initial set of features, we can write down a simple greedy-like algorithm that selects the best features according with our evaluation metric. However, even in this case the quality of our AI method depends a lot on the set of initial features. This observation is still valid even in the case of a more refined feature selection algorithm like the well-known Boosting [3].

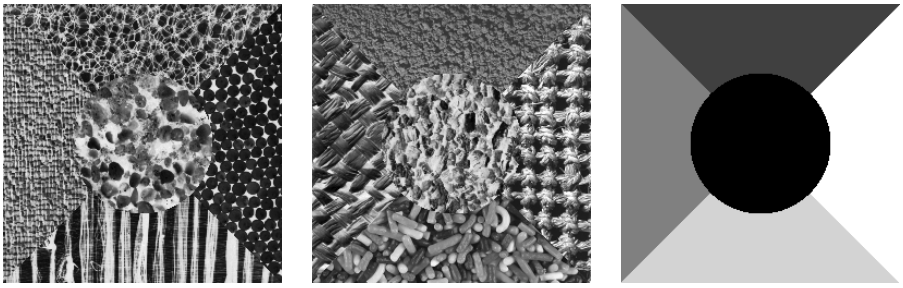
Consider now the problem of classifying images depicting fabric textures in two classes: horizontal stripes and vertical stripes. In this case it is very important that the set of initial features contains some edge-related feature like the Histogram of Oriented Gradients (HOG) or some simpler features describing the distribution of horizontal and vertical edges. Probably, at the end of the features selection, the edge-related features will be selected and other features like the histogram of colors will be discarded. Normally in such simple problems our intuition is enough to select the best features, but if we treat much more complex problems, such as the

Figure 1.1: This figure shows 16 images containing 8 classes of objects. Images are taken from the Caltech-256 dataset.

classification of 30000 images containing 256 categories of objects[1], we could no longer be guided by our intuition. Even defining a large number of features, it is not obvious that a combination of them giving decent classification results exists. Look, for example, at the images in Figure 1.1, you will probably find it difficult to propose few simple features capable of separating the 8 classes of objects. The problem is that most of the features used to analyze images are handcrafted and meant to solve specific problems, not very general problems like the one seen above.

A problem that we will discuss extensively in this thesis is the automatic unsupervised segmentation of images according to their texture properties. A fact that seems obvious is that separate different textured areas require less experience than separate different classes of objects. Look for example at Figure 1.2, after a glance the texture segmentation is automatically done by our visual system without troubles. Now, the goal is to try to do it automatically and, since the problem seems so easy, in an unsupervised way. As we shall see in later chapters, many features and methods have been proposed in the literature to try to address this problem, but we are still far from a robust and general solution.

[1]We refer to the Caltech-256 dataset, available at `http://www.vision.caltech.edu/Image_Datasets/Caltech256/`



(a) (b) (c)

Figure 1.2: Two texture mosaics each one composed of 5 distinct textured regions (a and b) and the solution to the segmentation problem (c).

Our conviction is that, in order to find a robust solution to these difficult problems, it is necessary to leave the old "handcrafted feature" approach in favor of a new methodology in which, using unsupervised techniques, the features arise directly from the data. Proceeding in this way, the feature definition and selection phases are not required since they are performed automatically by a *feature learning* module that does not require particular knowledge-based considerations. Fortunately, this is not just our conviction, since in literature many methods that learn features using unsupervised pre-training were proposed. A detailed survey on the subject is contained in [4]. However the problem is that, with regard to images, most of these methods are related only to the supervised classification of images and recognition of handwritten characters [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 14, 15].

In this thesis we still focus on images but we show that, even for a problem like the texture segmentation, relevant features can be found using an unsupervised approach. In particular we employ a standard Self Organizing Map (SOM) neural network and some very simple local pixel encodings, keeping the method as simple as possible.

The main contributions of this thesis are the definition of a simple feature learning method, based on the well-known SOM neural network, and its application to the unsupervised description and segmentation of textures, a problem that is usually dealt with using ad-hoc computer vision and image processing methods. We further demonstrate that, the use of our method to learn features for a super-

vised image classifier, provides some interesting additional properties compared with other similar methods in literature.

2

Unsupervised Feature Learning

2.1 Introduction

Given an image or a set of images, our target is to analyze a large set of small patches, extracted from the images and composed of few squared pixels, in order to find any regularity that allows us to moving away from the standard, "hardware oriented", RGB representation and getting closer to a more semantic representation. This chapter starts describing the methods used to extract local patterns from the image and the feature learning phase performed by the SOM. Then we explain how to combine multiple feature extraction stages in order to create a much powerful feature extractor that, as experimentally demonstrated in Chapter 3, gives a representation which is very suitable to describe textures.

Since the model proposed in this chapter has been successfully applied to two different branches of computer vision, unsupervised texture segmentation and su-

pervised image classification, we performed two separate studies of the state of the art and reported them in Chapter 3 and Chapter 4. This chapter is purely descriptive and does not contain experiments, in the following chapters we analyze how the model described here can be used, as mentioned before, to solve practical computer vision problems.

2.2 Unsupervised Training

Before delving into the details, we anticipate that the method described in this section can be seen as a *black box* where, firstly, a set of images $\mathbf{I_0}$ is used to train the model and, after the training, any input image I_0 can be remapped into the new image I_1. We call this black box Feature Learning Unit (FLU) and we represent it as reported in Figure 2.1. We have to introduce the FLU concept because it is useful to explain, in the following section, the composition of more than one feature learning units.

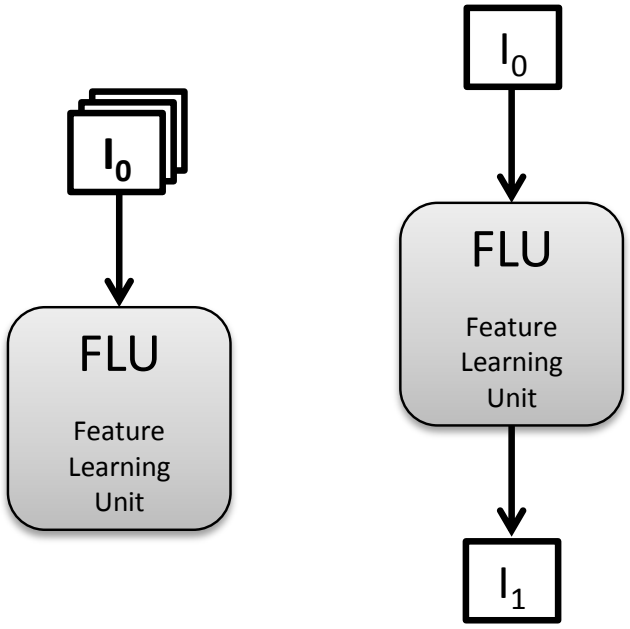
Let's see now how FLU works. As discussed in the Chapter 1, the proposed feature learning model is based on the SOM, an artificial neural network first proposed by Teuvo Kohonen in early 1981 and published in [16]. This neural network is able to produce, without supervision, a spatially organized internal representation of various features of input signals [17].

Figure 2.2(a) depicts a typical SOM, a neural network composed of m neurons, where each neuron with index i is fully connected to the input layer through a series of weighted links

$$\mathbf{w_i} = [w_{i1}, w_{i2}, \ldots, w_{in}]^T \tag{2.1}$$

where $0 \leq w_{ij} \leq 1$ and n is the dimension of the input data. The general SOM model admits a N-dimensional grid of neurons, but in this thesis we will focus only on the 1-dimensional and 2-dimensional versions of the network, reported in Figure 2.2.

The proposed method involves an initial unsupervised training phase, where a large number of vectors are presented to the network and the neural weights are updated according to a particular rule. The training vectors are extracted



(a) (b)

Figure 2.1: Scheme used to represent a Feature Learning Unit. (a) Configuration for the training, using a set of input images. (b) Configuration for the remapping of a single image.

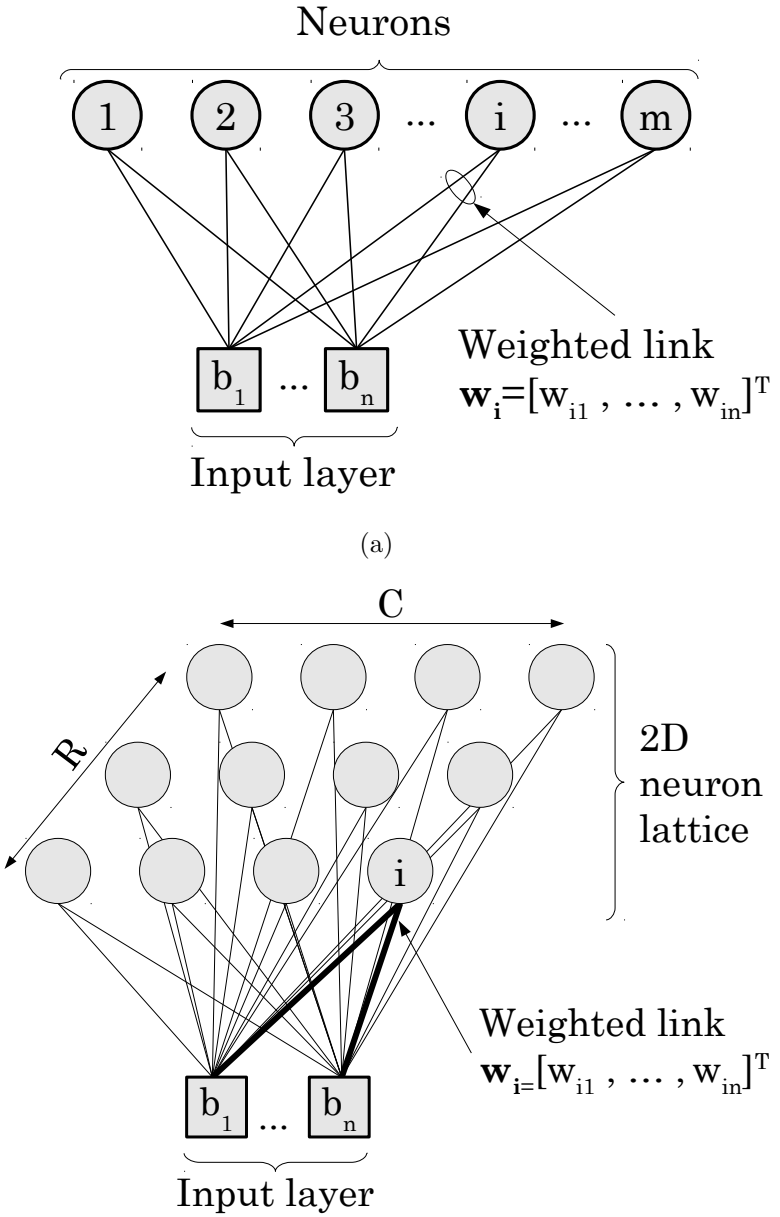


Figure 2.2: Schemes for the 1-dimensional (a) and the 2-dimensional (b) versions of the SOM neural network.

from the input images using an overlapping window that slides over each image, in literature this approach is known as *receptive field* [18] and is widely used for its simplicity and efficiency [15, 14, 13]. In order to make patterns that the SOM can learn, the pixels within the receptive field should be encoded in some way. We used two encodings, the first called *intensity encoding* and the second that we call *histogram encoding*.

Intensity encoding involves the concatenation of pixel intensities within the receptive fields, where intensities are real numbers, normalized in order to vary within 0 and 1. Formally

$$\mathbf{b} = [b_1, b_2, \ldots, b_j, \ldots, b_n]^T \tag{2.2}$$

where $0 \leq b_j \leq 1$ is the intensity value of the pixel j. The size of the pattern obtained using the intensity encoding strategy is determined by the size of the receptive field, in particular with a receptive field $\theta = \{\theta_W, \theta_H\}$ of $\theta_W \times \theta_H$ pixels, the pattern size $n = \theta_W \theta_H$.

With histogram encoding, the receptive field is used to compute an histogram with n bins and each bin $i = 1, 2, \ldots, n$ in the histogram f_i represents the number of pixel in the receptive field that have an intensity value $(i-1)/n < b < i/n$. A typical value for n is 256, that gives one histogram bin to each intensity value of a 8-bit grayscale image. Once the value of each bin f_i is determined, the pattern is normalized so that $\sum_{i=1}^{n} b_i = 1$, in particular

$$\mathbf{b} = \left(\frac{1}{\sum_{i=1}^{n} f_i} \right) \cdot [f_1, f_2, \ldots, f_i, \ldots, f_n]^T \tag{2.3}$$

Let us describe now how the unsupervised learning happens. The learning process requires t_{tot} iterations and, at the first iteration, the values of the neural weights are randomly chosen between 0 and 1. At each iteration t, a new input vector is presented to the SOM and a single neuron k is activated in a particular location of the network. We call this neuron *the winner*. The winner selection occurs by satisfying the identity

$$\|\mathbf{b} - \mathbf{w_k}\| = \min_i \{\|\mathbf{b} - \mathbf{w_i}\|\} \tag{2.4}$$

The step previously described is followed by the update of the weights in the neighborhood of the winner. The update, described by the equation

$$\mathbf{w_i}(t+1) = \mathbf{w_i}(t) + \alpha(t)h_{ik}(t)[\mathbf{b}(t) - \mathbf{w_i}(t)] \tag{2.5}$$

depends on two functions: $\alpha(t)$, called *adaptation gain* or *learning rate*, and $h_{ik}(t)$, called *neighborhood function*, that is a *bell curve* kernel function.

Since the SOM only updates neurons near the winner, the function $h_{ik}(t)$ depends on the dimensionality of the neuron lattice. For the 1-dimensional SOM we have

$$h_{ik}(t) = \exp\left(-\frac{\|i-k\|^2}{2\sigma^2(t)}\right) \tag{2.6}$$

while for the generic N-dimensional SOM

$$h_{ik}(t) = \exp\left(-\frac{\|\mathbf{q_i} - \mathbf{q_k}\|^2}{2\sigma^2(t)}\right) \tag{2.7}$$

where $\|\mathbf{q_i} - \mathbf{q_k}\|^2$ denotes the Euclidean distance between the coordinates of the winning neuron k and the neuron to be updated i. In the case $N = 2$, depicted in Figure 2.2(b), we can consider $\mathbf{q_k} = [q_{kr}, q_{kc}]^T$ and $\mathbf{q_i} = [q_{ir}, q_{ic}]^T$, where $1 \leq r \leq R$ in subscript is in reference to the row number inside the $2D$ neuron lattice and $1 \leq c \leq C$ refers to the column number.

In order to speed up the convergence of the learning process, the two parameters σ^2 and α can be chosen as time-variable functions that decrease monotonically and linearly with the iterations. How to configure σ^2 and α and how many iterations are required depend on the problem to be treated, this details are discussed in the following experimental chapters. The one just described is the standard setup for a SOM, for an extended discussion on the parameters and on the model itself we recommend the book [19]. We also tested other weights initialization methods and other non-standard functions for σ^2 and α, but during the experiments we did not noticed significant changes that may justify the increase in complexity of the method.

Since we have an arbitrarily large set of input (or training) images, the number of iterations will probably not match the total number of patches t_{tot} extracted by

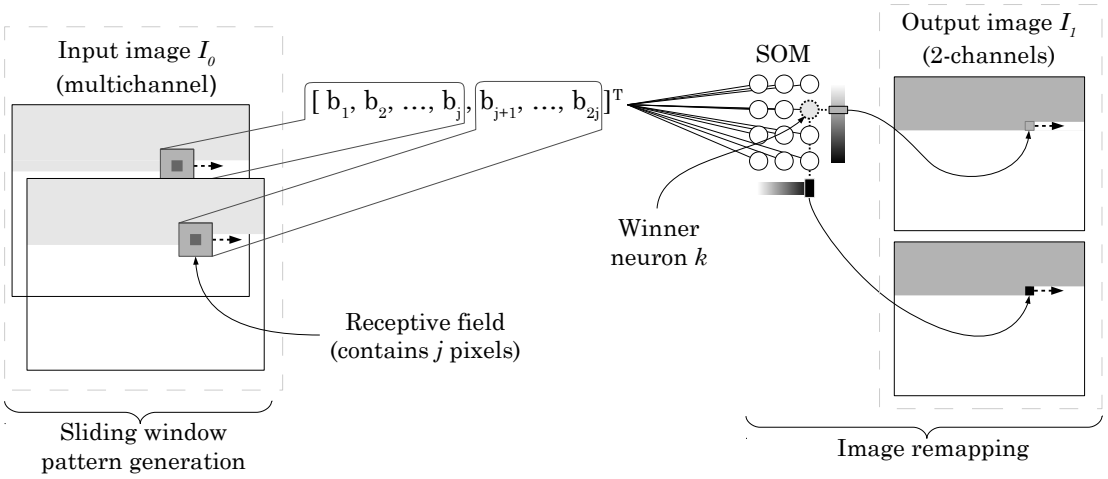
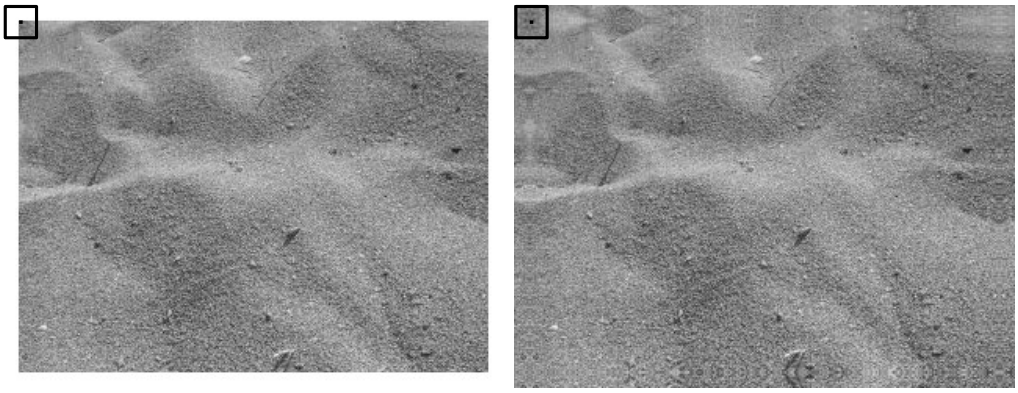


Figure 2.3: Schema representing the image remapping strategy, performed using a 2-dimensional SOM.

sliding the receptive field. The strategy adopted is to extract all the patches and sort them randomly, then the first t_{tot} patches were involved in the training process while other patches are discarded. If the number of patches is less than t_{tot}, the patches are cyclically presented to the SOM, this event may happen, for example, when we have a single input image. To create a random permutation of the patches we use a modern version of the Fisher-Yates shuffle algorithm, proposed in [20], that has a $O(n)$ complexity, where n is the number of patches.

At the end of the training phase, the spatial location, represented by the coordinates of each neuron in the network, corresponds to a particular domain or feature of input signal patterns [17] and the weights of each neuron contain a good prototype of the input patches [21]. Summarizing, by using a small window of local context around each pixel, the proposed method tries to associate to each neuron a particular local feature.

Once the SOM is trained, its neural weights $\mathbf{w}$ can be treated as constant values, and employing the same sliding window approach used during the previous training phase, we can map each pixel of the input image in the N-dimensional Euclidean space of the activated neurons within the SOM lattice. Using again Equation 2.4, we thus generate a new image with N-channels that we call *remapped image*.



(a) (b)

Figure 2.4: (a) A sample textured image and a receptive field of 20×20 pixels. (b) The same image with a 10-pixel padding obtained using a mirror strategy.

The remapped image $I_1(x, y)$ can be formally computed from the input image $I_0(x, y)$ using

$$I_1(x, y) = \frac{k}{m} \tag{2.8}$$

and, for the 2-dimensional SOM

$$I_1(x, y) = \left[\frac{q_{kr}}{R}, \frac{q_{kc}}{C}\right]^T \tag{2.9}$$

where, for each pattern $\mathbf{b}$ centered on the pixel (x, y) of the input image I_0, the winner neuron k is found using Equation 2.4. R and C refer to the size, in rows and columns, of the 2-D neural lattice. Note that the image defined in Equation 2.9 is a two-channels image, therefore each pixel contains two intensity values. The resampling procedure for a 2-dimensional SOM is visually explained in Figure 2.3.

For some applications, like texture segmentation, it may be necessary that the remapped image I_1 has the same size of the input image I_0. For this purpose the input image can be border-padded so that, in total, $H \cdot W$ training vectors will be extracted, where H is the height and W is the width in pixels of the input image I_0. For texture segmentation, we tested the "mirror" border padding strategy, as explained in [22] and depicted in Figure 2.4.

2.3 Feature Learning Unit Composition

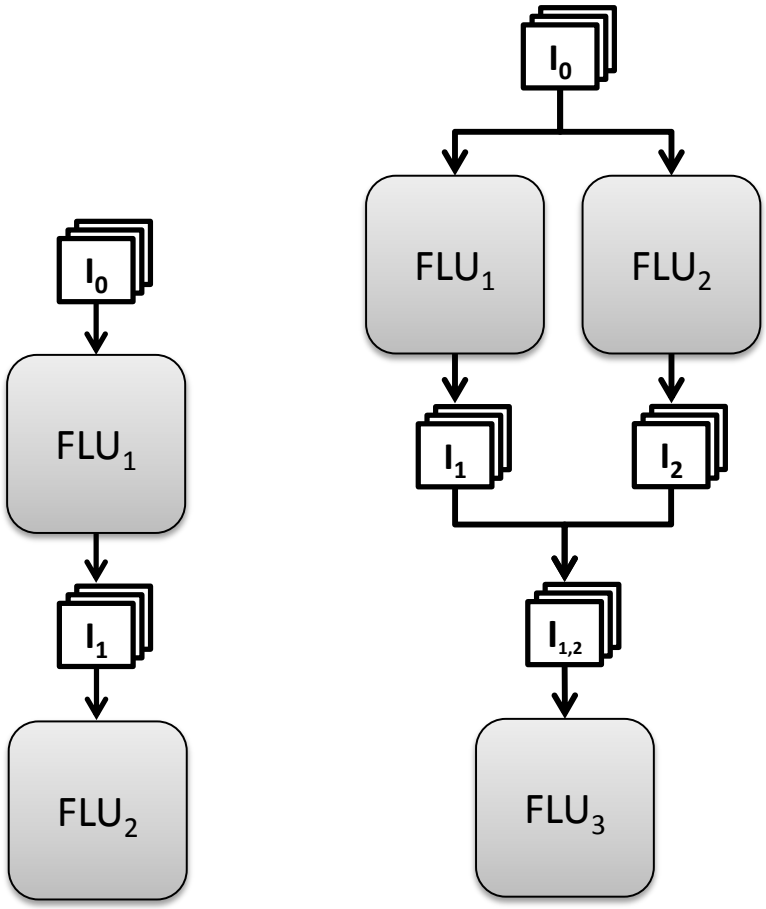
As we will see in Chapter 3 and Chapter 4, the FLU is a powerful tool that is able to process images and find, in an unsupervised way, salient local features. For problems like the classification of scene images the features can be extracted by the FLU, sliding the receptive field over the whole image, and then globally aggregated using some encoding like a pyramidal histogram. However, when we deal with the segmentation of images, a highly accurate local feature description is needed because we can no longer rely on a global encoding [23]. For this reason we empirically found that the composition of more than one FLUs with different receptive field sizes and local encodings can provide a highly reliable local features.

This property was largely demonstrated in the convolutional neural networks literature, a model biologically inspired by the cells in the mammalian visual cortex and successfully applied to handwritten character recognition [5, 6] and class object recognition [10]. In [24], convolutional networks were also applied to face recognition and a SOM is used to reduce the dimensionality of input images.

In this section we describe how the learning process happens when we compose more FLUs. The proposed learning algorithms are easy and intuitive, but a formal description is required.

Let's start with the serial composition of two FLU. The learning scheme for the serial composition is depicted in Figure 2.5(a). The learning process starts training the FLU_1 with a set of input images $\mathbf{I_0}$. The method used to train the unit is the one described in the previous section. When the training of the first unit is done, FLU_1 is used to remap all input images, creating a new set of images $\mathbf{I_1}$, that is in turn used to train FLU_2, with the same method used until now.

For the parallel composition, depicted in Figure 2.5(b), the learning process starts training FLU_1 and FLU_2 with the same set of input images $\mathbf{I_0}$. It is important to notice that the two trainings are completely independent and can be carried out in parallel. Once FLU_1 and FLU_2 are trained, the sets of remapped images $\mathbf{I_1}$ and $\mathbf{I_2}$ are created and, for each image in this two sets, the channels are concatenated in order to create a new set of images $\mathbf{I_{1,2}}$. Formally, given the image $I_1 \in \mathbf{I_1}$ with two channel $I_1 = \{I_1^a, I_1^b\}$ and the single-channel image



(a) (b)

Figure 2.5: (a) Two FLU in series. (b) Parallel composition of two FLUs and a third FLU that aggregates the results.

$I_2 \in \mathbf{I_2}, I_2 = \{I_2^a\}$, the resulting image $I_{1,2} \in \mathbf{I_{1,2}}$ is defined as $I_{1,2} = \{I_1^a, I_1^b, I_2^a\}$. The set $\mathbf{I_{1,2}}$ is then used to train FLU_3.

The two cases of composition discussed in this section should be seen as building blocks that give the essential tools to create more complex models.

2.4 Summary

In this chapter we introduced the concept of FLU and explained how the unsupervised learning happens. Summarizing, the FLU is model that uses local patterns from input images to train a SOM neural network and then, using a process called remapping, acts as a local feature extractor. Then we described the basic building blocks to combine multiple FLUs and increase the local accuracy of the feature. An application of the FLUs composition will be used as a texture descriptor and discussed in Chapter 3. However, even the features yielded by a single well-configured FLU can be successfully used, with a proper encoding, to solve a complex problem like the classification of scene images, discussed in Chapter 4.

3

Unsupervised Texture Descriptor

3.1 Introduction and State of the Art

In order to automatically produce a description of a natural image, a fundamental role is played by texture descriptors. Images representing real objects often do not exhibit regions with uniform intensities but, due to the physical properties of real surfaces, they contain frequent variations of brightness which form certain repeated patterns called visual texture or, more simply, texture.

Over the years, many problems involving texture analysis have been proposed, the main ones are listed below. *Texture classification* aims to produce a classification map of an image where each uniform textured region is identified by a particular texture class which belong to. *Texture segmentation* is focused on finding texture boundaries even if it is not possible to classify each region. Figure 3.1 shows an example of unsupervised texture segmentation obtained applying

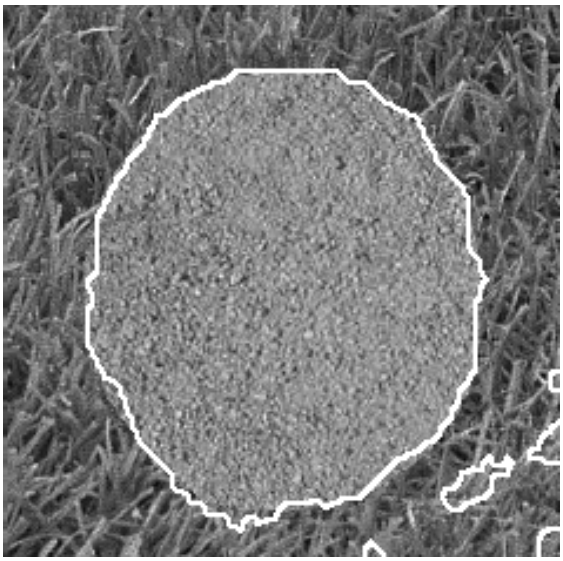


Figure 3.1: Segmentation between two areas with different textures obtained using the proposed descriptor. Segmentation border is depicted with a white line.

a K-means clustering to the local descriptor proposed in this chapter. *Texture synthesis* is used for image compression and in computer graphics, with the aim of rendering object surfaces which need to be as realistic as possible. Finally, with *shape from texture*, we aim to extract the three-dimensional shape of objects in a scene using texture information, distorted by imaging process and the perceptive projection [25]. Despite the final purpose is quite different, each of the problems listed above requires a texture descriptor, which becomes an essential tool in many applications.

A common denominator for most successful texture descriptors is that the textured image is submitted to a linear transform, filter or filter bank. Methods using this common scheme are called *filtering approaches*, and received an extensive survey in [26], a comparative study where various filtering approaches have been evaluated within a texture classification framework.

An important issue that characterizes most of the filtering approaches is the selection of an appropriate filter bank. The most commonly are the Gabor filters, inspired by experiments with animal visual systems [27], and signal-processing based filters, designed with desirable band-pass properties in the Fourier domain

[28]. However, the optimal choice of a filter bank is often influenced by the particular application and may require a lot of experimentation.

A simple and promising strategy to combine multiple filters, resulting in a compact description of the texture, is the spectral histogram, first suggested in psychophysical studies on texture modeling [29] and later used for texture analysis and synthesis [30] [31]. Spectral histogram is based on the assumption that all of the spatial information characterizing a texture image can be captured in the first order statistics of an appropriately chosen set of linear filter outputs. Spectral histogram can also be used as a local descriptor, using and appropriately sized receptive field, in this case the descriptor is often called Local Spectral Histogram (LSH).

LSH is a powerful local texture descriptor, able to seize general aspects of texture as well as non-texture regions. In [1] a LSH based on a receptive field $\theta = \{19, 19\}$ and on a filter bank based composed of eight filter (pixel intensity, two gradient filters, two-scales Laplacian of Gaussian and three Gabor filters) has been used for texture segmentation, attaining the state of the art in the field of unsupervised texture segmentation methods based on filter bank.

The choice of a suitable filter bank can be carried out automatically with applications involving a supervised learning, where a filter-selection algorithm may choose, from among a set of pre-configured filters, a subset that maximizes the quality of the result [32]. However, the parameters of a good generic descriptor should not depend on the particular application.

The main drawback of LSH is that it requires large integration windows to extract meaningful texture features from the image, this results in a poor reliability of the description along texture boundaries. A solution to the aforementioned problem has been proposed in [1] by using asymmetric windows and a refined probability model based on seed points automatically extracted from the segmented regions.

Some work tried to generalize the methods based on multichannel filtering by training, in a supervised fashion, a neural network in order to find a minimal set of specific filters. These methods may delegate to the neural network the dual task

of extracting features and classifying textures [33] [34], or perform separately the second phase using a more powerful classifiers such as Support Vector Machines (SVMs) [35].

In this chapter we propose an application of the model presented in Chapter 2 as texture descriptor. The potential of our method is its total independence from a feature bank, since the unsupervised training is able to automatically extract salient information using only simple pixel encodings from small image patches. We exploit two topological configurations. The first is based on a pyramidal composition of three FLUs that use intensity as input encoding. The second configuration has a more complex topology and use a histogram-based input encoding. For both configurations, the unsupervised image analysis is distributed across multiple FLU modules, using a local receptive field which become progressively larger. At each node of the composition, only the most relevant feature for the particular context will be extracted by the FLU and the image will be "redrawn" deprived of redundant information.

Considering the complexity of the non-linear dimensionality reduction introduced by each FLU, the validity of the proposed approach is difficult, if not impossible, to prove analytically. However, to evaluate the method, we used a very simple unsupervised texture segmentation strategy, based on a K-means clustering algorithm applied on the remapped image yielded by the terminal FLU node. In this way we highlight the goodness of the features, extracted in an unsupervised fashion by the FLU composition, and we exclude any contributions attributable to a supervised machine learning method or a post-processing/refinement phase.

In the experimental phase, we will use only mosaics of textures with a fixed number of regions forming the same shape. This choice is justified by the fact that we are proposing and evaluating a texture descriptor able to describe complex textures but not segment scene images. To segment a complex scene image, a simple K-means clustering is not adequate and other grouping strategies such as Region Growing or Watershed can be used in its place [36]. However, the definition of a texture segmentation method is beyond the scope of this thesis and can be a future work.

The main results collected in this chapter have been published in [37].

3.2 The Proposed Configurations

As argued in Chapter 2.3, image segmentation and consequently texture segmentation, requires a highly accurate local feature description. We experimentally verified that, to increase the accuracy of the local feature, we can combine more FLUs using some simple learning algorithms. The final descriptor is therefore based on a proper composition of FLUs. In particular we used the two configurations depicted in Figure 3.2.

The first configuration, reported in Figure 3.2(a), is based on a pyramidal composition of three FLU elements. We called it *Config. A*. The encoding used to create patterns from the input images (and from images resampled by each FLU) is the intensity encoding. At each level of the pyramid the size of the receptive is increased by a factor 2, starting from a receptive field $\theta = \{2, 2\}$ applied on the input images and ending with a receptive field $\theta = \{8, 8\}$ used to yield the output remapped image.

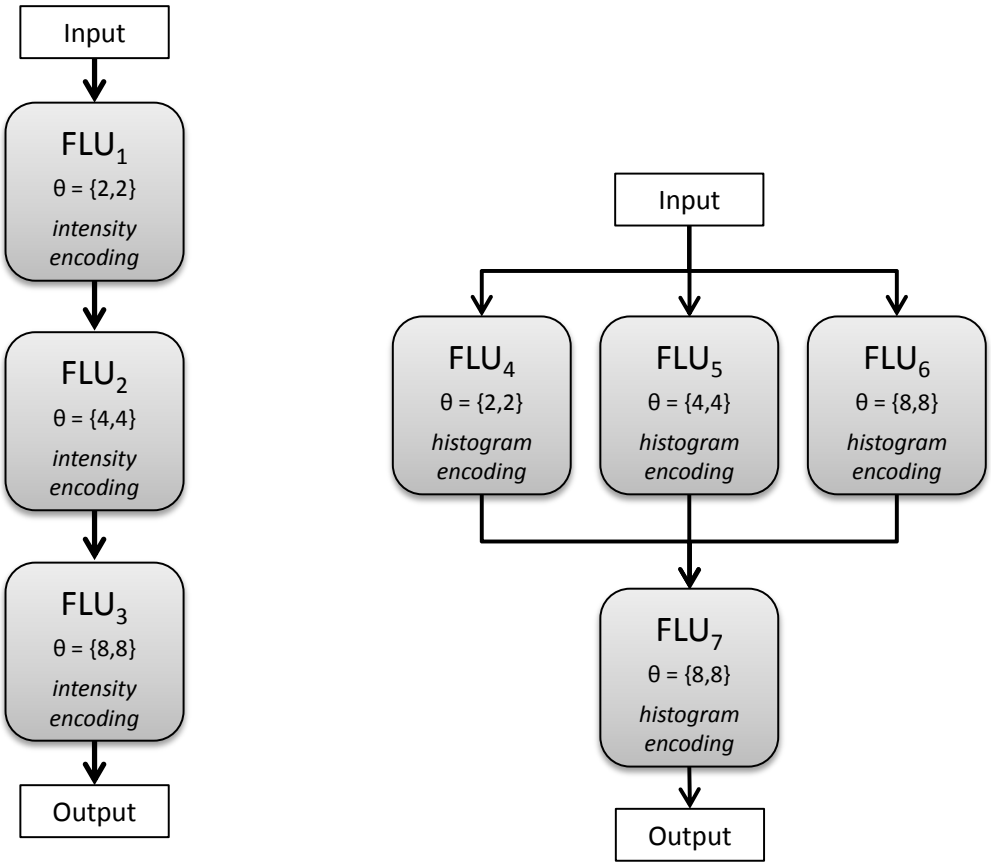
The second configuration, called *Config. B* and represented in Figure 3.2(b), processes the input images with a parallel composition of three FLU with different receptive field sizes, $\theta = \{2, 2\}$, $\theta = \{4, 4\}$, $\theta = \{8, 8\}$. Then the three remapped images are aggregated with a further $\theta = \{8, 8\}$ node. All the FLU involved in this second configuration use a receptive field with histogram encoding.

For the configurations used in this chapter we use 2-dimensional SOM with 20×20 neuron in each FLU. The SOM parameters varies within two learning phases, known in literature as the *ordering* and *tuning* phase.

During the ordering phase, that involves the first 1000 iterations, parameters vary as follows:

- α linearly decreases from 0.1 to 0.01
- σ linearly decreases from $\max(R, C)/2$ to 1

The tuning phase involves a number of iterations that depends on the size of



(a) (b)

Figure 3.2: The FLU configurations used as texture descriptor. (a) intensity configuration called *Config. A*. (b) histogram configuration called *Config. B*.

the network, in particular it is $500 \cdot m$. During this second phase the parameters vary as follows:

- α linearly decreases from 0.01 to 0.001
- σ linearly decreases from 1 to 0

This parameter configuration is widely used and documented in many works using the SOM model [17, 38].

We found these two FLU compositions trying a lot of configurations and using a trial and error strategy on the evaluation images. In the following section we analyze the results obtained using the two configurations and compare them with other state-of-the-art texture segmentation methods.

3.3 Experiments

In this section we evaluate the two configurations, introduced in the previous section, within a simple segmentation framework based on the K-means algorithm. We also evaluate the contribution of each individual FLU to the overall segmentation. This last aspect is very important and justifies the use of a FLU composition.

For each image, the set of patterns to be clustered is created by concatenating pixel intensities, taken from the remapped image, to their normalized coordinates. This simple strategy is done in order to create a raw topological constraint that leads to a more reliable segmentation. Formally

$$P = \bigcup_{x=1}^{W} \bigcup_{y=1}^{H} \left(I_{FLU}(x,y) || \left[\frac{x}{W}, \frac{y}{H} \right]^T \right) \tag{3.1}$$

where I_{FLU} is usually called the remapped image obtained as the output of a FLU composition, composed of $W \times H$ pixels.

Our first experiments involves the *Config. A*, since it has a plain topology and uses the simplest encoding, based on the pixel intensity. However, in the second part of this section we show that the *Config. B* can be used to further improve the results. We used as evaluation metric the percentage of pixels incorrectly segmented, this metric is a common choice in the texture segmentation literature.

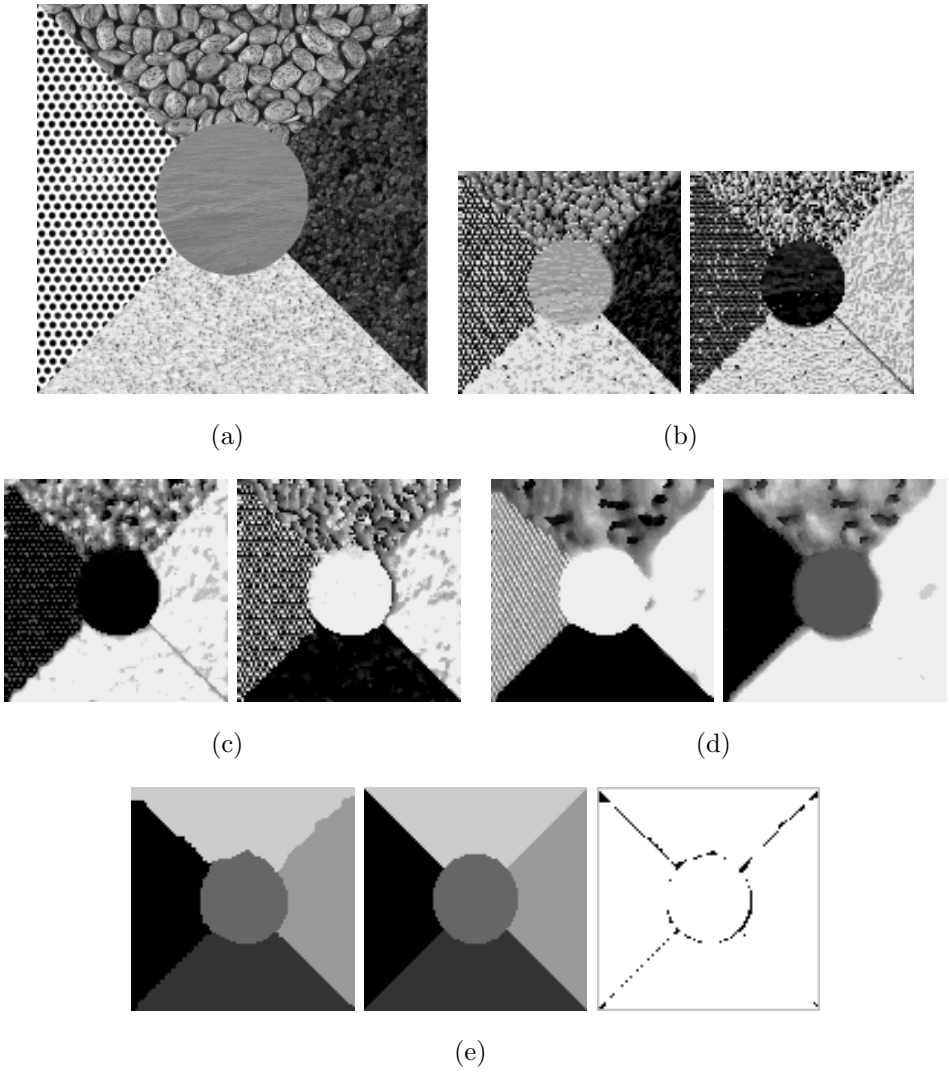


Figure 3.3: (a) Input image, a mosaic composed of 5 textures. (b)(c)(d) Two channels images remapped by FLU_1 (b), FLU_2 (c) and FLU_3 (d). (e) From the left, the final segmentation, the ground truth segmentation and the segmentation error map. Wrong pixels are shown in black.

(a) (b) (c)

Figure 3.4: Results obtained on the texture mosaic in Figure 3.3(a) by [1] with (a) and without (b) boundary region localization. (c) Our result for comparison.

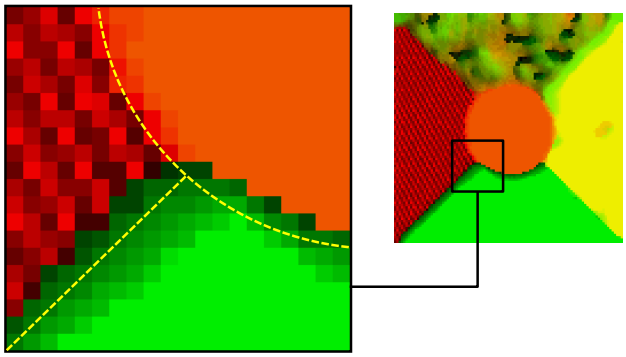


Figure 3.5: On the right, a false-color image representing the feature extracted by the FLU_3 and, on the left, a detail of the boundary between three textures.

Figure 3.3(a) depict a 5-texture mosaic used in [1] to test an unsupervised segmentation method. The authors have obtained a 3.90% error using a LSH texture descriptor with a receptive field $\theta = \{19, 19\}$ and a filter bank composed of one intensity filter, two gradient filters, two-scales Laplacian of Gaussian and three Gabor filters. By applying a refined probability model to localize the region boundaries, they have reduced the error to 0.95%. The proposed method using *Config. A* performs with an error of 1.75%, Figure 3.3(e) shows the resulting segmentation, the ground truth segmentation and a map that highlights wrong segmented pixels. Figure 3.4 reports a visual comparison between results obtained by [1] and the proposed segmentation method.

A typical problem shared by several texture descriptors, and visible in Figure 3.4(b), is the bad handling of the area near texture boundaries, this happens when the receptive field overlaps two areas with different textures. Luckily, the topological ordering property owned by the SOM network and inherited by the proposed texture descriptor makes the transition between different textures smoothed. Since the local texture description has a gradient-like trend near texture boundaries, the K-means clustering that uses the common Euclidean distance as a metric of distance, is able to recognize and separate with a good precision the two textures along the real boundary. Figure 3.5 shows the feature extracted by the FLU_3 in false-colors[1] and highlights the boundary area where the feature descriptor is smoothed. Considering that we do not use any handcrafted feaure/filter and mostly that our method does not rely on a specific border localization technique, the result obtained is very challenging.

Figure 3.6(a) is another 5-texture mosaic used in [2] to test a supervised approach based on empirical marginal distributions of local texture features, in particular a co-occurrence distribution and 2 Gabor magnitude distributions extracted using a $\theta = \{11, 11\}$ receptive field. They have obtained an error of 22.87% training a model with only one Gabor magnitude distribution and 3.1% using all the three distributions. Our segmentation error is 4.51%. The two results are comparable,

[1]The two channels of the remapped image are merged into the R and G channels of an RGB image.

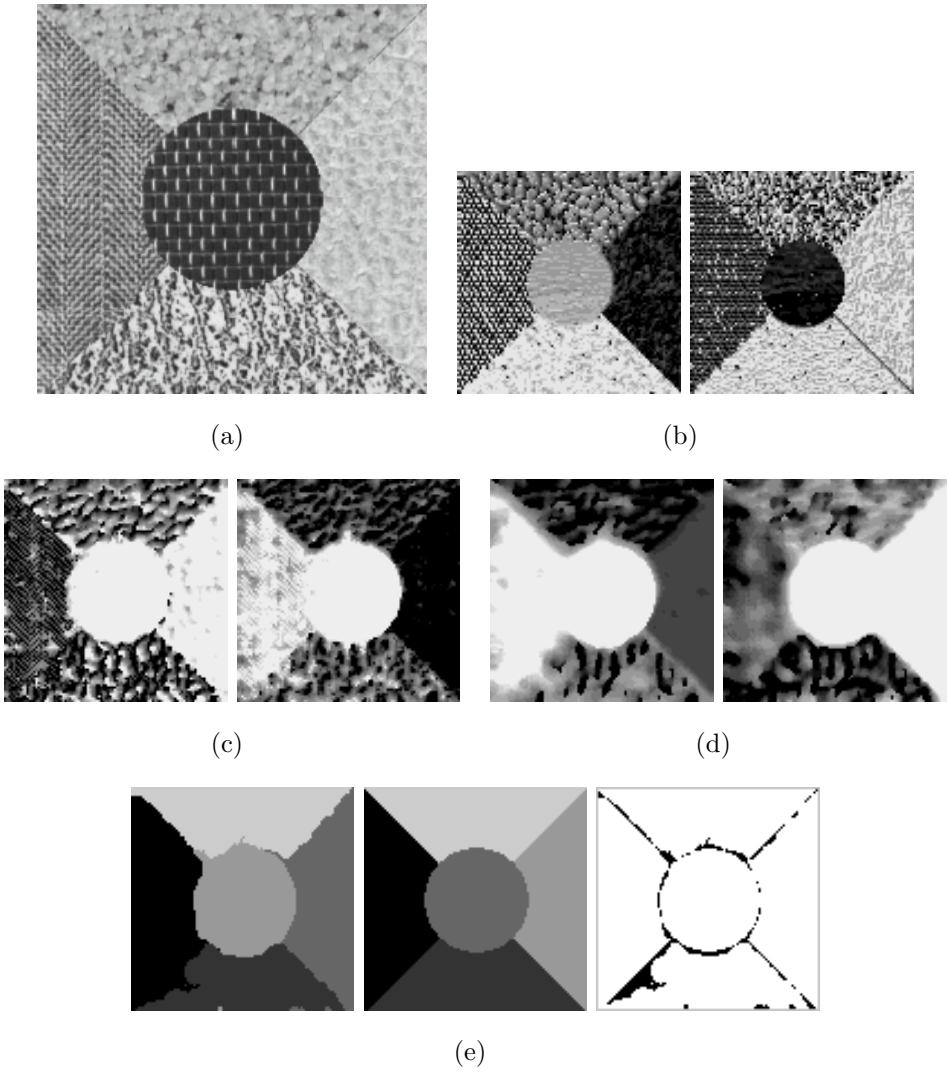
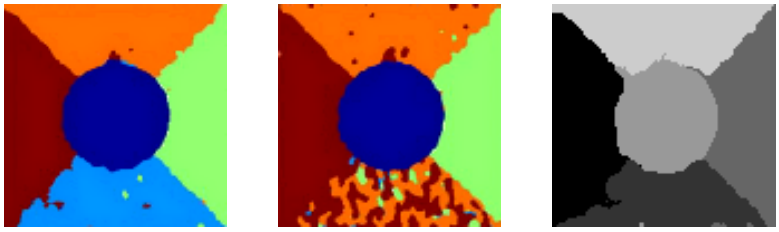


Figure 3.6: (a) Texture mosaic composed of 5 textures. (b)(c)(d) Two channels images remapped by FLU_1 (b), FLU_2 (c) and FLU_3 (d). (e) From the left, the final segmentation, the ground truth segmentation and the segmentation error map. Wrong pixels are shown in black.



(a) (b) (c)

Figure 3.7: Comparison of results for the texture mosaic in Figure 3.6. (a) Results obtained using the full method proposed by [2] and (b) using only a single Gabor magnitude distribution. (c) Our result for comparison.

but the problem studied here is essentially more difficult, given the unsupervised nature of the feature extraction process and, then, of the image segmentation.

[39] proposed the 2-class mosaic in Figure 3.1 as a challenging image since it show two textures that are both irregular and have similar means and gradient-magnitudes. No numerical result is available in their paper, but the results that we achieved is qualitatively comparable with that shown in [39], obtained using an unsupervised approach that minimizes the entropy-based metric on the probability density functions of image neighborhoods.

To investigate the contribution of each FLU in the overall process, we evaluated the method by excluding different subsets of FLUs. The worst result is obtained without any FLU, just applying the K-means clustering directly on the intensity levels of the input image, while the best configuration involves all the three FLUs composed as in the *Config. A*. Results in Table 3.1 show that the strength of the descriptor lies primarily in the pyramidal approach that at each level refines the quality of the descriptor. This process is also visible in Figure 3.8 that shows the segmentation images obtained segmenting the remapped image at each level of the pyramid. It's clear that a shallow architecture, based on a single FLU, is not able to provide a feature that can be used with the K-means segmenter to create a satisfactory segmentation.

We also tested a set of images artificially generated in order to prove if the proposed method is only able to segment textures or also non-textured areas. The first

Table 3.1: Segmentation results obtained using different features and subsets of the 3-layers architecture proposed in Figure 3.2(a).

	Figure 3.3(a) error (%)	Figure 3.6(a) error (%)
Raw pixels	29.45	52.45
Only FLU_1	17.12	18.89
Only FLU_2	27.56	23.70
Only FLU_3	28.88	34.90
Composition of $FLU_1 \rightarrow FLU_2$	11.03	9.75
Composition of $FLU_1 \rightarrow FLU_3$	8.52	4.83
Full *Config. A*	**1.75**	**4.51**

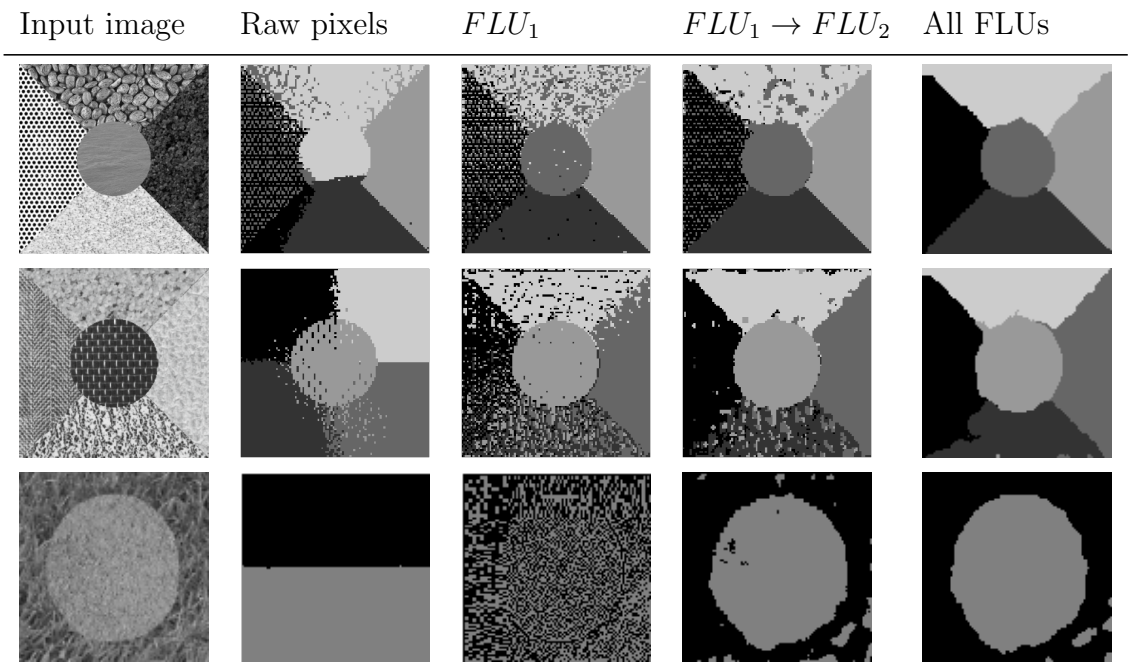


Figure 3.8: Segmentations obtained using different subsets of the 3-layers architecture proposed in *Config. A*.

(a)

(b)

Figure 3.9: (a) Three synthetically created texture-non texture mosaics. (b) Segmentation results obtained using the features extracted by the *Config. A*.

image is composed by two wave-gradient regions with two different orientations. The mean intensity is constant within the two regions and the only discriminant information is the orientation of the wave pattern. The second image shows two regions, one with a wave-gradient texture and one with a solid color. Also in this case both regions have the same mean intensity. The third image contains two non-textured areas with different intensities. These test images are depicted in Figure 3.9(a) and, as can be seen in Figure 3.9(b), in all three cases, the proposed method has been able to distinguish the two regions almost perfectly. This result suggests that the proposed descriptor can handle, at the same time, texture regions as well as non-textured regions.

Results obtained with the 3-stacked FLUs are very promising, but we want to further stress the method using a much larger and complex dataset. The problem is that, to the best of our knowledge, in the literature there are no standard datasets suitable to test an unsupervised texture segmentation method. For this reason

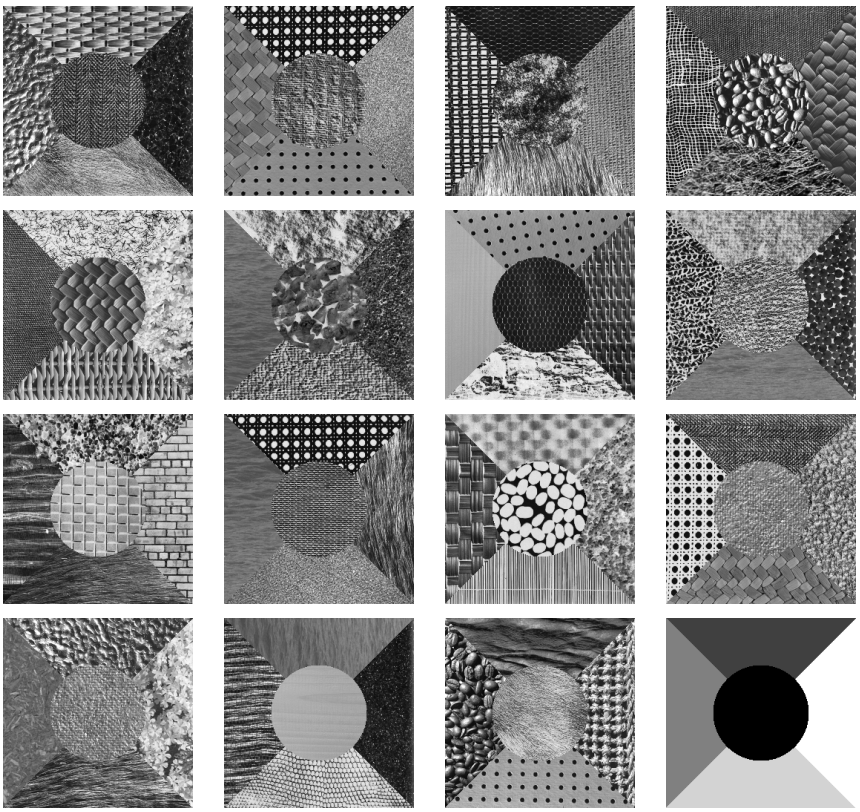


Figure 3.10: Sample images taken from the Mosaic-5 dataset and the ground truth segmentation image.

we created a new dataset, called *Mosaic-5*, composed of 100 images containing 5-texture mosaics. Textures was randomly taken from the well-known Brodatz album [40] and the *Vision Texture Dataset*[2]. There is only one ground truth segmentation for the entire Mosaic-5 dataset and this is shared by all the 100 images. As evaluation metric we still use the percentage of pixels incorrectly segmented, but in this case the error is averaged between all the images, resulting in more reliable evaluation metric. Figure 3.10 contains some sample images taken from the Mosaic-5 dataset and the ground truth segmentation. The Mosaic-5 dataset is publicly available on the web site `http://www.dicom.uninsubria.it/~marco.vanetti/`.

Testing the *Config. A* we obtained an high mean error of 36.8%, this is probably due to the high complexity of the dataset, that combines textures with very different spatial homogeneities. This fact convinced us to find a new composition of FLUs able to provide acceptable errors on the Mosaic-5 dataset. After an extensive trial and error phase, we found the configuration referred as *Config. B* and depicted in Figure 3.2(b). *Config. B* is based on a parallel composition of three FLU with different receptive field sizes and a histogram encoding. Remapped images produced by the first three FLUs are aggregated by a fourth FLU in order to yield the final texture description. The use of a local receptive field with a histogram encoding of pixel intensities was inspired by the texture segmentation method proposed in [1], based on a bank of different filters.

In Table 3.2 we collect the results obtained on the Mosaic-5 dataset using the two configurations in Figure 3.2. As done before for the single evaluation images, we isolate the contribution of each FLU element to the final texture description. Results shows that the *Config. B*, scoring a mean error of 10.7%, is a very powerful texture descriptor that, although is not based on any handcrafted feature or filter, can well characterize even very hard texture mosaics. It's important to notice that the training and the following remapping and segmentation are performed serially and independently for each image of the dataset. For this reason the proposed

[2]The Vision Texture Dataset is provided by the MIT Vision and Modeling Group, available on the web site `http://vismod.media.mit.edu/`

Table 3.2: Mean segmentation error obtained with the proposed configurations on Mosaic-5 dataset.

	Mean segmentation error (%)
K-means on pixels	64.9
Only FLU_1	57.2
Composition of $FLU_1 \rightarrow FLU_2$	43.1
Full *Config. A*	36.8
Only FLU_4	38.8
Only FLU_5	23.7
Only FLU_6	15.5
Full *Config. B*	**10.7**

method, in order to be used as a generic texture segmenter, does not require to be pre-trained because the training is limited only to the image to be segmented.

As a final point, we want to consider the time complexity of each FLU that is, fixing the SOM size, $O(\theta_W \theta_H)$. Before being processed, each input image is scaled to 100×100 pixels. Under these conditions, the time required to process an image with a receptive field of $\theta = \{2, 2\}$ and any type of encoding is about 3 seconds. With a receptive field of $\theta\{8, 8\}$, the time required rises to about 8 seconds. Tests showed that a complete training and resampling of the 3-stacked FLUs model requires about 15 seconds using an unoptimized, single C# thread, on an Intel(R) Core(TM) i5 mobile CPU at 2.30Ghz.

3.4 Summary

In this chapter we have applied two FLU compositions as a new texture descriptor able to characterize textured as well as non-textured regions with high accuracy. The potential of the method lies in its independence from a feature bank and its ability to automatically extract, without supervision, salient information us-

ing only simple pixel encodings from small image patches. Our method exploits the important topological ordering property of the SOM and allows a smoothed and reliable image description even in areas with strong transitions, such as the boundary between two different textures or two different solid colors.

Comparison with other state-of-the-art methods shows that our solution gives comparable results even without a directly managing of difficult areas, such as texture boundaries. The provided configurations offers good results on different types of evaluation images and on the very challenging Mosaic-5 texture segmentation dataset, that we proposed in order to allow a reliable comparison with future methods.

4

Image Classification

4.1 Introduction and State of the Art

For the automatic categorization of scene images, a very complex computer vision problem, a common trend in recent years consists in the use of feature learning and deep learning algorithms to learn a set of features from unlabeled data in an unsupervised way. Features learned are typically used to train a supervised discriminative model, e.g. a SVM classifier. As discussed in previous chapters, feature learning algorithms are opposed to methods that use specific *handcrafted* features, chosen by a domain expert.

In deep learning literature many methods such as K-means and Gaussian Mixtures [14], Autoencoder [8, 11], Restricted Boltzmann Machine [7, 12] and Sparse Coding [9] have been successfully applied to the problem of single-layer feature learning and multi-layers deep learning. Even in the computer vision have been

proposed methods that exploit the K-means algorithm to create a dictionary or *bag of visual words* used as a feature in many visual class recognition problems [41, 42].

In this chapter we train a single FLU node to learn single-layer features from the extremely challenging CIFAR-10 dataset, containing 60.000 tiny natural images belonging to 10 classes, with 6.000 images per class [12]. As remarked in the previous chapters, the features learned by the FLU arise straight from the raw pixel values within a local receptive field and not from a knowledge-based feature selection. In the experimental section we show that a supervised linear SVM classifier trained with opportunely encoded learned features provides significantly better results than using raw pixels values or the Pyramid Histogram of Oriented Gradients (PHOG), a popular handcrafted feature used in computer vision to represent the shape of objects and to perform visual class recognition in natural images [43, 44]. Contrary to most feature learning algorithms, the proposed method is fast and requires just few minutes to train the FLU, despite the large number of images involved in the process. It may seem strange that a method successfully employed as a texture descriptor can be used to classify complex scene images. However it has been shown in literature that early scene identification can be explained with a simple texture recognition model [45].

The major contribution of this chapter is the empirical study of the SOM neural network used to learn features from a very big and challenging datasets, the CIFAR-10. The unsupervised learning process is fast and can be controlled by adjusting the size of the SOM. Moreover our results show that using the proposed method it is possible to arbitrarily reduce the number of features without repeating the feature learning process by combining topologically close neurons. This interesting property follows directly from the topological ordering property of the SOM neural network.

The main results collected in this chapter have been published in [46].

4.2 Image Representation

This section describes the feature learning phase, performed by the FLU on a large set of images, and the encoding that we used to represent the input image and then perform the supervised classification.

Given a large set of input images (in our case the training set of the CIFAR-10 dataset) we train a single FLU with the method described in Chapter 2.2. The FLU used in this chapter is based on a one-dimensional SOM with m neurons, as depicted in Figure 2.2(a). Once the FLU is trained, the neural weights $\mathbf{w}$ can be treated as constant values and, given a new input, according to Equation 2.4, a single neuron is selected as the winner and is therefore activated. To represent an image we slide the receptive field, pixel by pixel, over the whole image obtaining a distribution of neurons activations. These activations are then encoded using a histogram representation, where each bin $i = 1, 2, \ldots, m$ in the histogram f_i represents the activation count for a single neuron.

Following the spatial pyramid scheme proposed in [41], we compute more local histograms on the same image, starting from a single histogram at the first level and quadrupling the number of histograms for each new level of the pyramid. Considering only the histograms on a single level: they are computed in order to cover non-overlapping regions of the image, have always a rectangular shapes and have all the same area. To form the final feature that describes the image, the histograms from all levels and all regions are concatenated as can be seen in Figure 4.1, showing an example of a pyramidal histogram with 3 levels.

The final feature is a vector with dimensionality $\sum_{l=1}^{L} \left(m \cdot 4^{l-1}\right)$, where L is the number of levels and m is the number of neurons in the SOM. Each histogram in the pyramid is individually normalized in order to satisfy the identity $\sum_{i=1}^{m} f_i = 1$.

This encoding is similar to the PHOG feature, where each bin in the histogram represents the number of edges having orientations within a certain angular range [43].

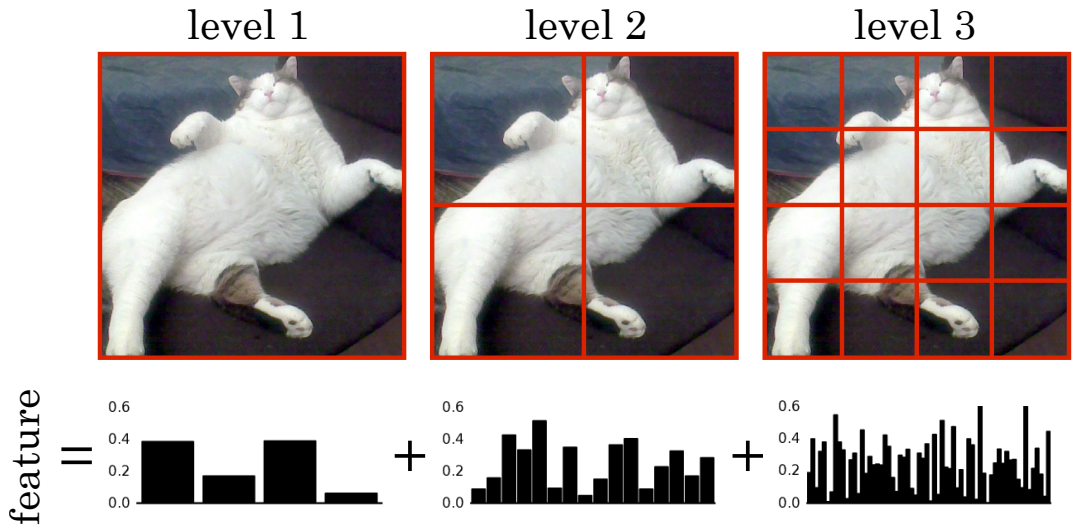


Figure 4.1: Encoding of a pyramidal histogram feature with 3 levels using a 4-neurons FLU. Each red square yields a 4-bins histogram. Each 4-bins histogram represents the activation of the four neurons when we slide the receptive field on the local area surrounded by the red square.

4.3 Experiments and Analysis

In this section we conduct several experiments using features extracted from images with the FLU-based method just described and a linear SVM as supervised training classifier [47].

As specified in Section 4.1, the dataset used to analyze the method is the CIFAR-10, a very challenging image classification dataset that contains 60.000 tiny annotated natural images divided into 10 classes, with 6.000 images for each class [12]. The images, each with a resolution of 32×32 pixels, contain different classes of objects, in particular animals and vehicles. Figure 4.2 shows some example images taken from the CIFAR-10 dataset. In all experiments we used the training set, composed by 50.000 images, to learn features and to train the SVM and the test set, composed by 10.000 images, to test the overall classification accuracy. As evaluation metric we used the percentage *overall accuracy*, which represents the number of images correctly classified on the total number of images

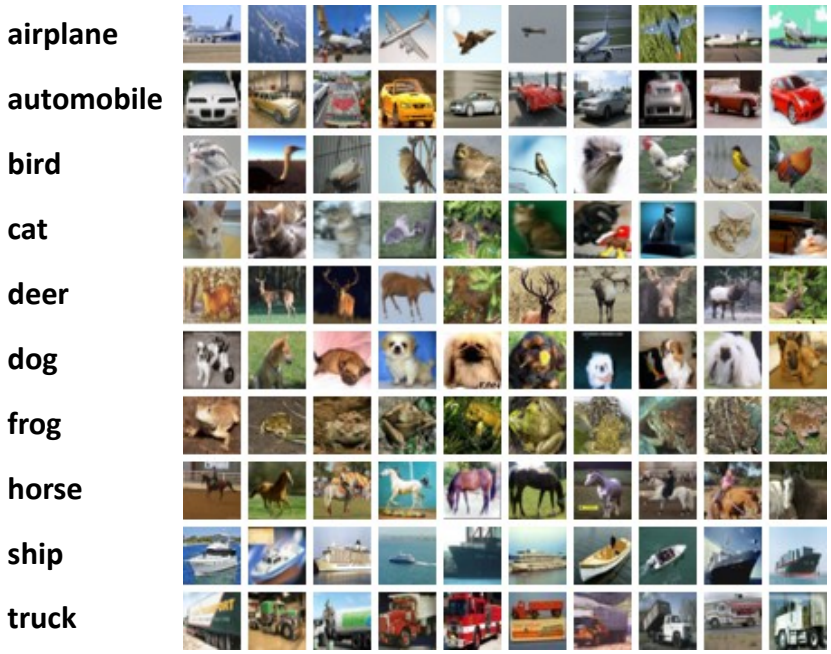


Figure 4.2: Some example images extracted from the CIFAR-10 dataset.

of the test set.

In order to improve the statistical reliability of accuracy values, for each experiment we trained the SVM 5 times, using 5 disjoint sets of training images, and we have averaged the test results, obtained each one on the whole test set. We found experimentally that a third level in the pyramidal histogram increases too much the size of the training vectors reducing the OA in all experiments, for this reason we reported results only for the first two levels.

All the tests reported in the following sections, except those in Section 4.3.4 and Section 4.3.5, were performed using "grayscale" pixel intensities within receptive field of $\theta = \{6, 6\}$ pixels with no local brightness and contrast normalization.

4.3.1 Standard Classification Methods

We now describe the results obtained on the CIFAR-10 dataset using three standard image classification methods. The first method, which we call *icon classifier*, represents each image as the concatenation of the intensity values of the pixels.

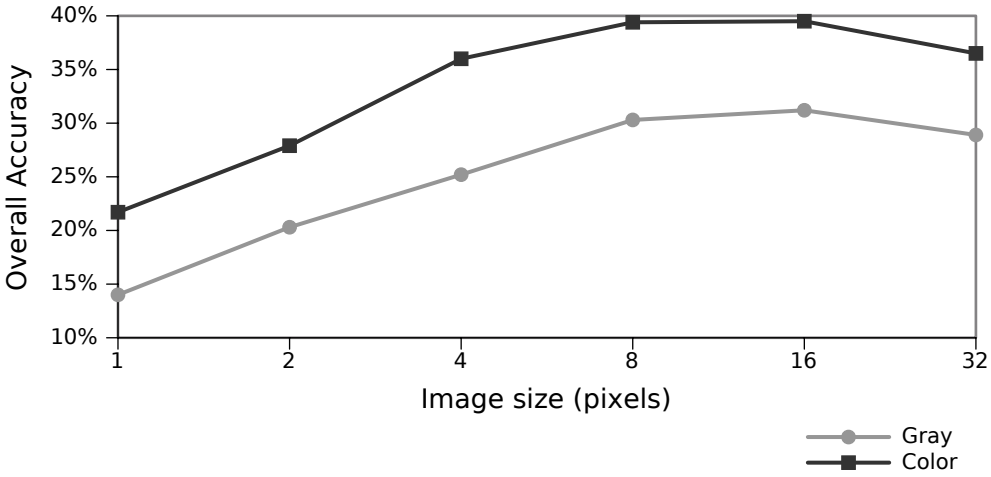


Figure 4.3: Overall accuracy obtained using the *icon classifier* with pixel intensities and colors.

For the *color* version of the icon classifier, the feature is formed by concatenating for each pixel the value of the three RGB channels. To control the size of the feature vector we scaled the image to different sizes using linear interpolation. Results obtained with the icon classifier are presented Figure 4.3. Notice how, by reducing the entire image to a single pixel corresponding to the average value of all pixels, the SVM is able to correctly classify almost 15% of the images. We also verified that the color is very important for the classification process, giving an improvement to the accuracy from 8 to 11%.

We then tested the classifier proposed in [43], based on a SVM and using the PHOG feature. Results are shown in Figure 4.4. We trained the SVM using a pyramidal histogram of the gradients computed on both the intensities of the pixels and the RGB channels. With the PHOG feature the performance is always acceptable and grows increasing the levels of the pyramid. Due to the small size of the images we could not test the PHOG with 4 levels.

To exclude that the classification results obtained with features learned by the FLU can be due only to the pyramidal encoding, we performed a test using a pyramidal histogram of RGB pixel values. To form the histogram feature, the RGB space is linear quantized over the bins of the histograms. Figure 4.5 shows

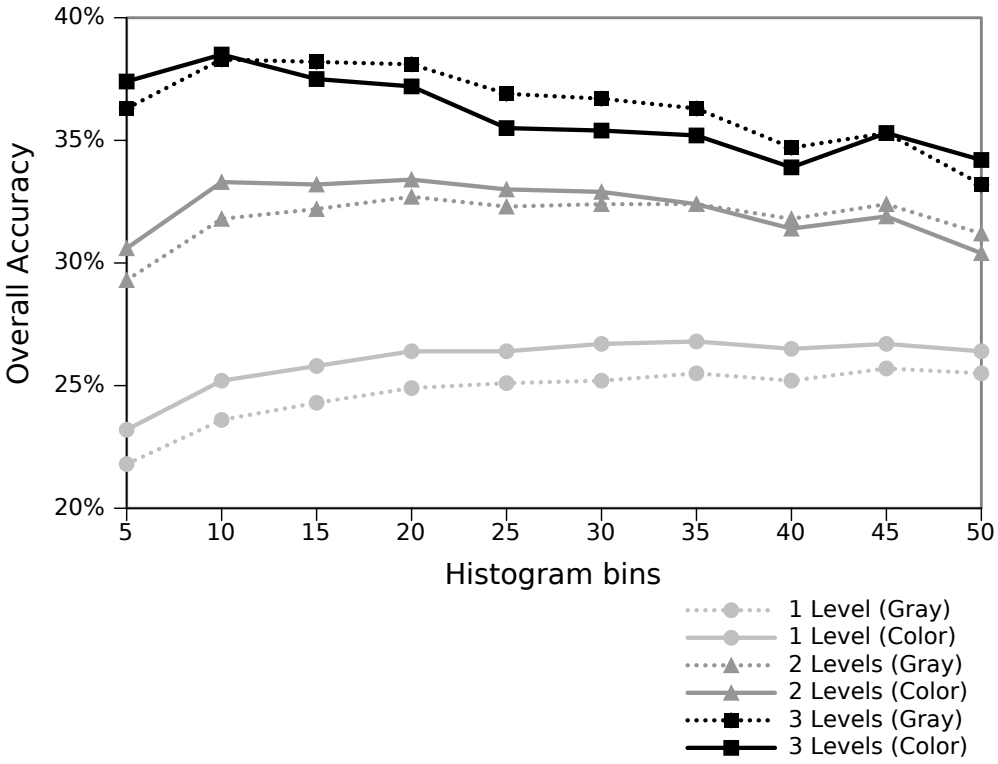


Figure 4.4: Overall accuracy obtained using the PHOG based classifier, with and without colors.

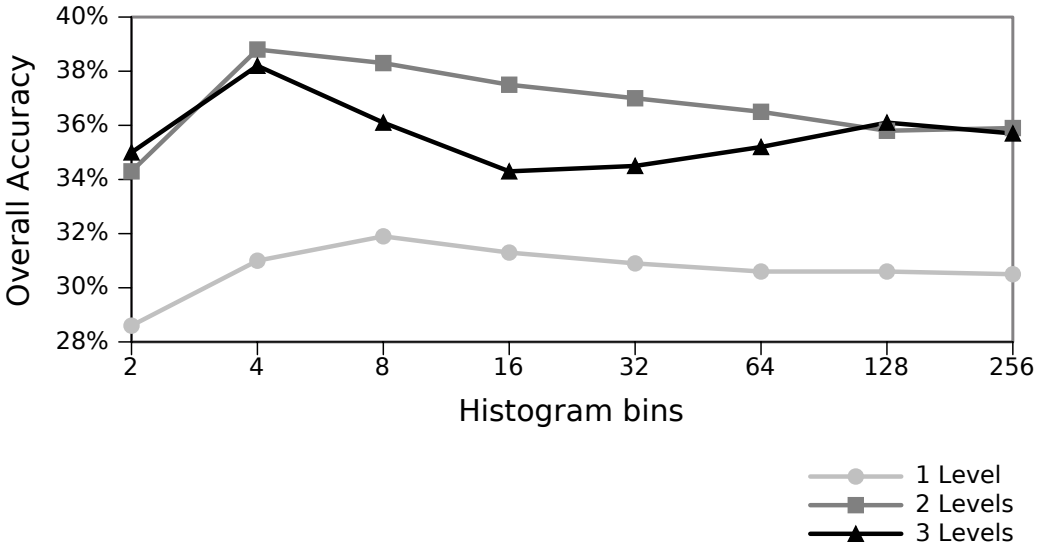


Figure 4.5: Overall accuracy obtained using a pyramidal histogram on RGB pixel values and varying the number of the bins.

the results obtained with the RGB pyramidal histogram classifier, using histograms with different number of bins. It appears that the contribution of the pyramidal coding is not sufficient to outperform the results obtained with the previously analyzed icon classifier. The last experiment confirms that the CIFAR-10 dataset is very hard and we need to learn ad-hoc features from the dataset itself in order to achieve results that exceed the 40% accuracy.

4.3.2 FLU Configuration

In all experiments presented in this chapter we used a FLU configured according to the following specifications. The learning rate α decreases linearly with the first 1000 ordering iterations from 0.1 to 0.01 and for the next $500 \cdot m$ tuning iterations from 0.01 to 0.001. The parameter σ decreases linearly from $m/2$ to 1 during the ordering phase and from 1 to 0 during the tuning phase. As explained in the previous chapter, this parameter configuration is widely used across the literature on SOM. We tried to double, triple and quadruple ordering and tuning iterations, but this did not lead to any change of more than 0.5% in the classification accuracy.

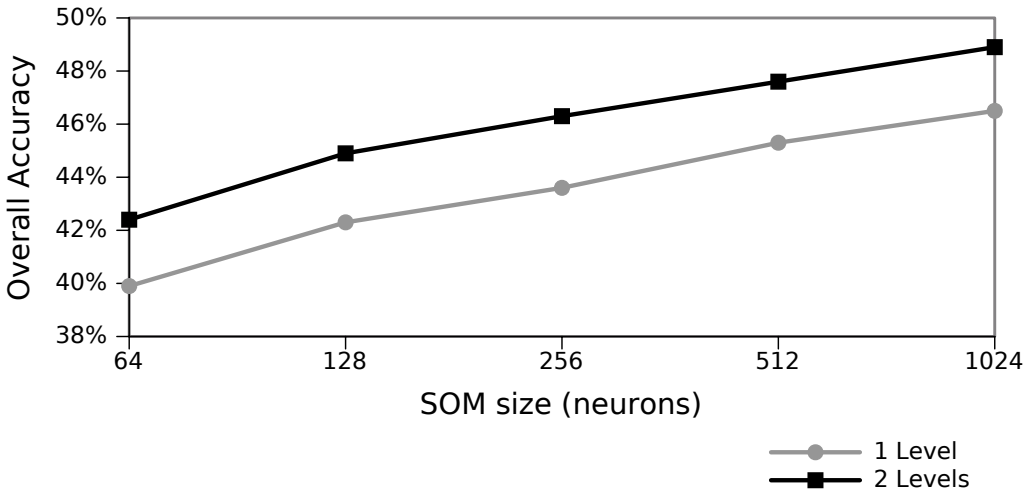


Figure 4.6: Overall accuracy obtained varying the number of neurons of the FLU and the pyramidal histogram levels.

We conducted first experiments using FLUs with 64 to 1024 neurons, doubling at each experiment the number of neurons. The receptive field was set to $\theta = \{6, 6\}$. Figure 4.6 shows overall accuracies in function of the size of the FLU and the number of levels in the pyramid. In accordance with the literature on feature learning, increasing the number of features leads to improved results, in particular in our case there is a linear relationship between the square of the number of neurons involved in the unsupervised learning and the overall classification accuracy. Using the second level of the pyramid, the accuracy increases from 2.3% to 2.7%.

The computational time required to train a FLU with 512 neurons, using $\theta = \{4, 4\}$ receptive fields, was about 20 minutes, or 10 minutes using a 256-neurons FLU. Our implementation is a single threaded C# code on an Intel(R) Xeon(TM) @ 2.66GHz CPU.

4.3.3 Reducing the Feature Size

An important property of the SOM model is that the weights of spatially close neurons correspond to similar features [17]. This property is called *topological*

ordering and is a consequence of the Equation 2.5 that forces the weight vector of the winning neuron and its neighborhood to move toward the input vector. Exploiting this property we can arbitrarily reduce the number of features used to describe an image by grouping neighboring neurons in the same histogram bin. For example, by grouping all pairs of neighboring neurons it is possible to halve the size of the final feature. Grouping more close neurons, we can further reduce the size of the feature and significantly speedup the supervised learning performed by the SVM [1].

We performed some experiments grouping neurons from FLUs with different sizes in order to obtain several description of images involving histograms with different number of bins. For example, the representation obtained by a 256-neurons FLU was reduced in size obtaining histograms with 128, 64 and 32 bins. We also performed a test with a 1024-neurons FLU where, at the end of the unsupervised learning process, the neurons were randomly ordered in order to nullify the effect of the topological ordering.

Results reported in Figure 4.7 clearly shows that the topological ordering of the SOM allows to efficiently reduce the size of the features without having to retrain the unsupervised model and without sacrificing the classification quality for more than $1-2\%$ accuracy. The procedure described above can not be carried out in such a simple way using other not supervised methods that do not have the topological ordering property, such as the K-means clustering.

4.3.4 Effect of Color, Local Image Normalization and Receptive Field Size

In this section we report the results obtained using different receptive field sizes, adding the RGB color information and applying a local brightness and contrast normalization to the patches extracted from the image. Let's assume that the intensity value of the pixels varies between 0 and 1, we employed on every patch extracted from the image a simple normalization, subtracting the mean intensity

[1] In our tests we noticed a $40-50\%$ speedup every time we halved the size of the features used to train the SVM.

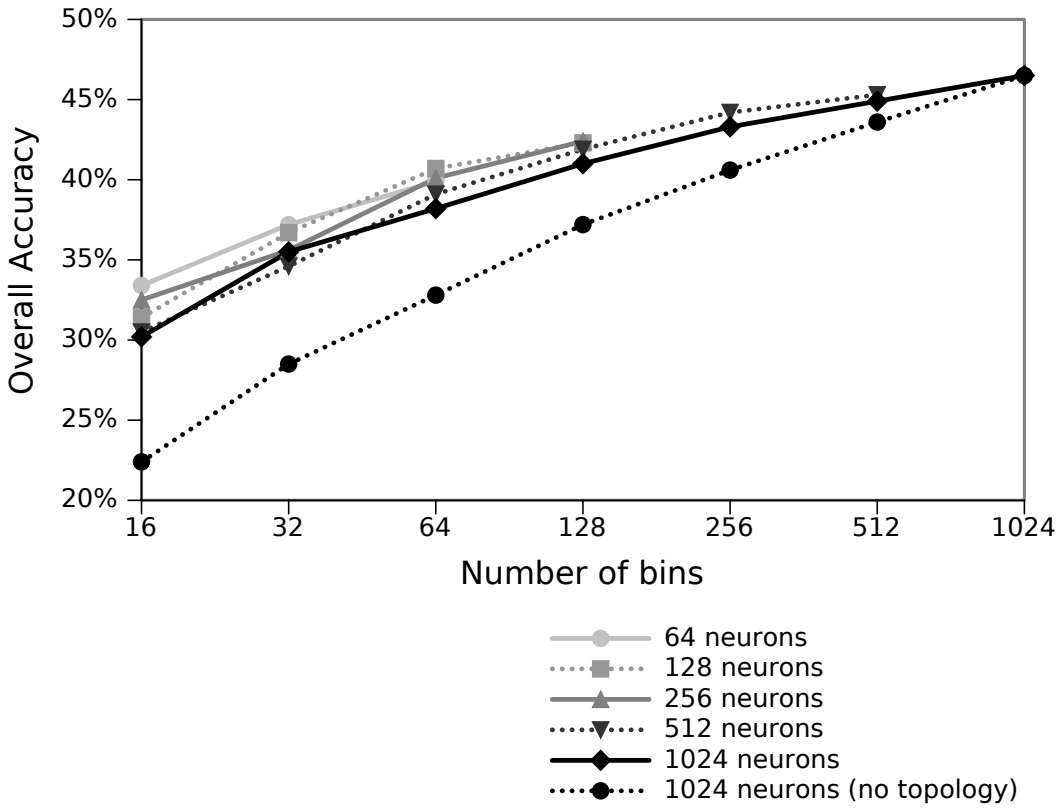


Figure 4.7: Overall accuracy obtained using applying the topological grouping to reduce the number of bins in the histogram. In this test we used a 1 Level pyramidal histogram.

value, dividing by the standard deviation of its elements and summing 0.5. Pixel intensities that fall outside the 0 to 1 range after the process are clipped to lie within this range. Local brightness and contrast normalization is one of many methods used in feature learning algorithms to improve the quality of the classification results [14].

Figure 4.8 shows the effects of the introduction of color and local brightness and contrast normalization, while in Figure 4.9 we have shown how the classification accuracy varies in function of the receptive field size. It is interesting to notice that the use of local normalization makes the contribution of the color less important,

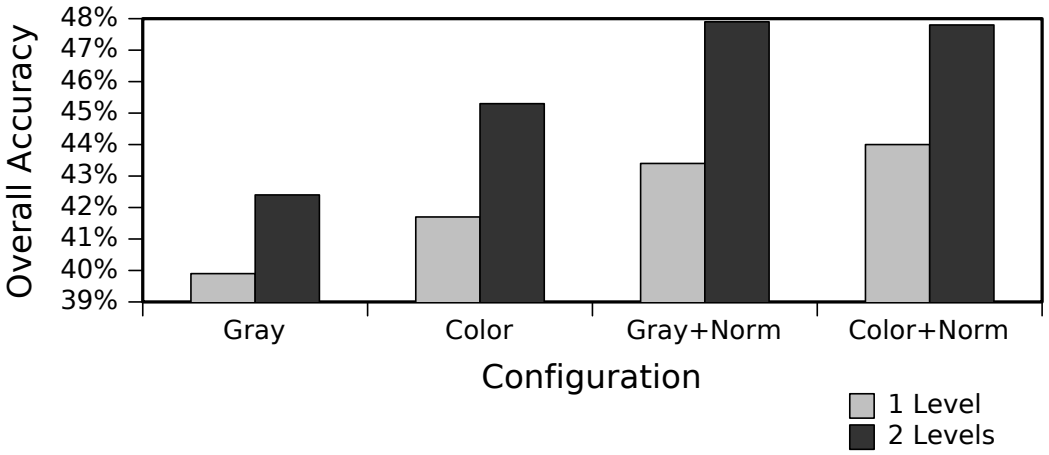


Figure 4.8: Effect of color, local brightness and contrast normalization for a 64-neurons FLU.

this fact can be seen also in Figure 4.10, where the weights of a 64-neurons FLU, trained with and without the local normalization are shown.

An overall accuracy of 54% was obtained using a 128-neurons FLU, $\theta = \{4, 4\}$ receptive field, color and local brightness and contrast normalization, and is comparable with results obtained by [14] using a K-means with a hard pooling feature encoding and a number K of centroids similar to the number of neurons in our FLU. Figure 4.11 shows the confusion matrix for this last experiment.

4.3.5 Other datasets

Cifar-10 is a very challenging dataset and is well accepted in literature to test machine learning and feature learning methods. However, for completeness, we want to test the applicability of our method to other classes of images and, in particular, to some datasets widely used in computer vision. In particular we chose to test other two standard image classification dataset, the *Caltech-101* [48] and the *Caltech-256* [49]. The Caltech-101 dataset contains 9145 images belonging to 101 classes plus a *clutter* class, while the Caltech-256 contains 30.608 images

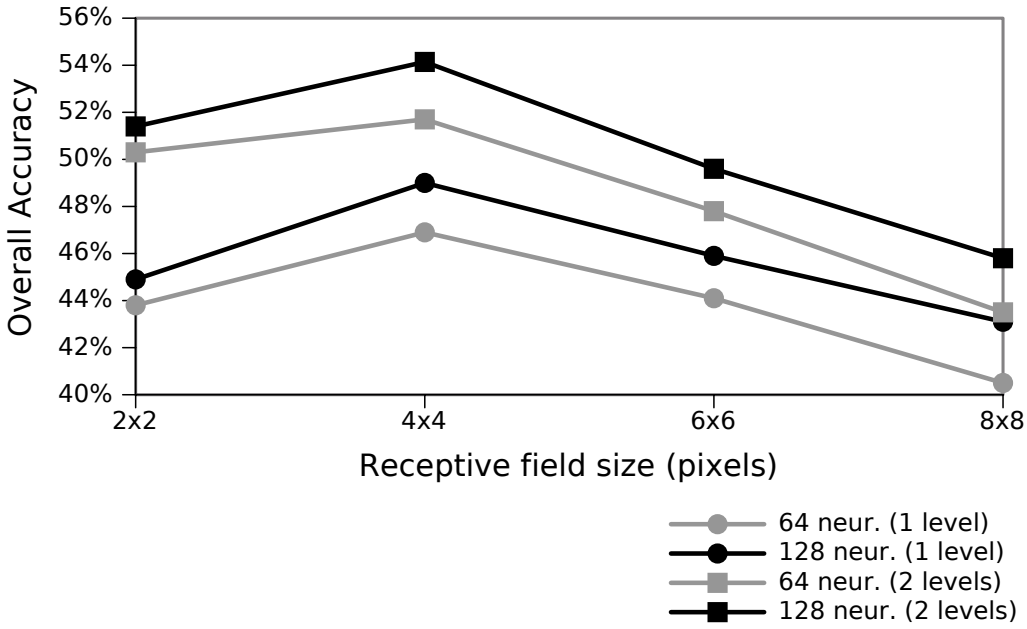
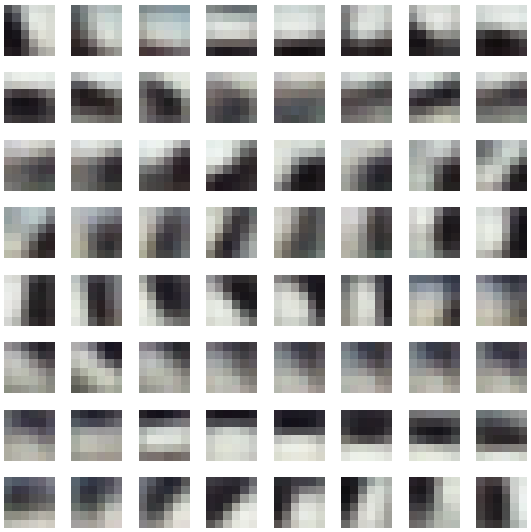
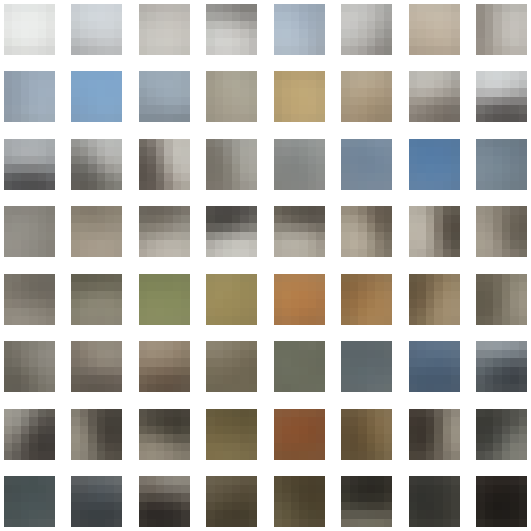


Figure 4.9: Effect of receptive field size in a 64 and 128-neurons FLUs, varying the number of pyramid levels. In this test we used color and local brightness and contrast normalization.



(a)



(b)

Figure 4.10: Weights plot obtained from a 64-neurons FLU trained with color $\theta = \{6, 6\}$ receptive fields. Effects of training with (a) and without (b) local contrast and brightness normalization of patches. Notice that the features extracted, plotted from top-left to bottom-right, are topologically ordered.

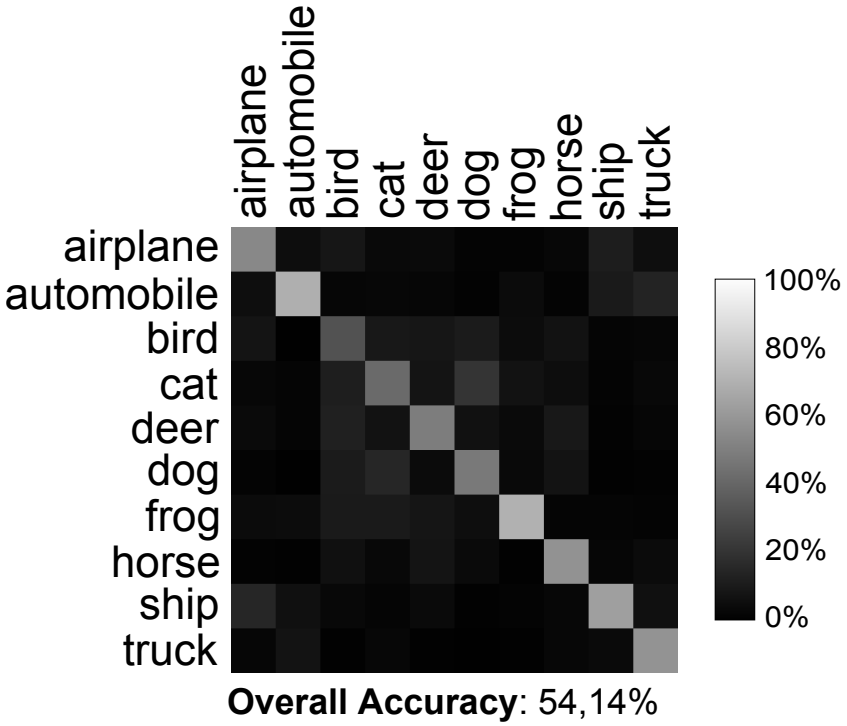


Figure 4.11: Confusion matrix obtained with a 128-neurons FLU, color, local contrast/brightness normalization and a $\theta = \{4, 4\}$ receptive field.

Table 4.1: Classification results obtained with five different datasets. In the last column the difference in OA% between the classification carried out using the PHOG feature and the proposed method.

Dataset	Images	Classes	PHOG	FLU	Δ
	Number		*OA%*		*OA%*
CIFAR-10	50.000	10	38	**56**	+18
CALTECH-256	30.607	256	49	**75**	+26
CALTECH-101	9.144	101	79	**87**	+8
Drezzy-46	4.841	46	87	**90**	+3
Artelab Mobile Fashion	502	5	**96**	95	-1

and $256 + 1$ classes [2].

We also tested the proposed method on two much smaller dataset, to see the influence of the number of training images on the quality of learned features. In particular we employed the *Drezzy-46* and *Artelab Mobile Fashion* datasets, proposed in our previous work [50].

We chose the standard 2:1 train-test split for all these datasets, so each dataset has been split into two sets, 2/3 of the images for training purposes and 1/3 for testing. In order to be processed efficiently, each image has been scaled to fit inside a 64×64 pixels square.

The model used for this test is a FLU with a 512 neurons SOM, a $\theta = \{4, 4\}$ receptive field, 2 pyramid levels, colors and local brightness and contrast normalization. For comparison we tested also the classifier based on a SVM and the PHOG feature, configured with 15-bins and 3 levels.

Table 4.1 reports information about the four datasets and the results obtained. Results on the two Caltech dataset are very good and the standard PHOG method is significantly outperformed. As expected, the quality of the learned features decreases if the number of images used in the unsupervised learning is too low

[2]The two Caltech datasets are available for download on the web page `http://www.vision.caltech.edu/archive.html`

Table 4.2: Classification results obtained with other state of the art image classification methods.

Method	Caltech-101	Drezzy-46
	$OA\%$	
FLU	**87**	**90**
LP-β	82	80
MKL	74	89
VLFeat	66	71
R.Forests*	80	-

*the source code is not available

[4, 51], consequently we have a very little gain in performances using the FLU with the Drezzy-46 dataset and a very little regression with the 502-images Artelab Mobile Fashion dataset.

We also performed tests with other state-of-the-art methods for image classification proposed in the computer vision literature. In particular, we considered

- LP-β, a multi-class classification method based on a Boosting of SVM weak learners [52]
- MKL, a classifier based on multiple kernel [53]
- VLFeat, a method based on bag of visual words and Random Forests contained within the VLFeat framework [54]
- RForests, an image classification approach that combines the PHOG feature with a bag of visual words strategy [44]

It is interesting to notice that the LP-β and MKL methods require, using the Caltech-101 dataset on our platform, days for the training phase, minutes to classify each image during the testing phase and gigabytes of disk space to store all the features data extracted from the training dataset. Results reported in Table 4.2

show an improvement of 5% over the LP-β method using the Caltech-101 dataset and a slightly improvement of 1% over the MKL method on the Drezzy-46.

4.4 Summary

In this chapter we presented a model that exploits a single FLU node to learn features from images without requiring any supervision. Our experiments performed on the very challenging CIFAR-10 and on other computer vision datasets show that the features learned by the FLU and encoded using a pyramidal histogram approach significatively outperform the classification methods based on raw pixels values and other state-of-the-art methods designed specifically for image classification.

Despite the large number of images processed in the datasets, the proposed feature learning process is fast and requires few minutes also using FLUs with hundreds of neurons. Moreover, employing the presented model it is possible to control the size of the features used to train the supervised classifier by grouping close neurons in the histogram encoding scheme. This property allows to speed up the learning process without having to repeat the unsupervised feature learning.

Experiments show that the accuracy of the classification can be improved by applying appropriate normalizations and fine tuning to the receptive field. Other normalization methods, such as whitening [55], and feature encoding schemes, such as hard or soft pooling [41, 56], can be applied to improve the results and can be considered in future work.

5

Conclusions

In recent years a great amount of research has focused on algorithms that learn features from unlabeled data. These approaches are known as feature learning or deep learning methods and have been successfully applied to classify scene images and recognize with high precision handwritten characters.

In this thesis we showed that a feature learning approach can be used to segment complex textures, a problem for a long time addressed proposing a large amount of handcrafted descriptors and local optimization strategies. We employed the SOM neural network for its ability to natively provide a set of topologically ordered features. These features allowed us to obtain a highly accurate local description, even in areas characterized by a transition from one texture to another. We also showed that a single feature learning unit can be combined with others in order to significantly improve the quality of the texture description and, consequently, reduce the segmentation errors. The results obtained proved that the proposed

segmentation method is valid and provides a real alternative to other state-of-the-art methods.

Since the proposed framework is simple, we easily combined it with a pyramidal histogram encoding and a SVM supervised network in order to classify scene images. We showed that the important topological ordering property, inherited from the SOM network, allow us to resize the feature set, obtained during the initial unsupervised learning, avoiding an unpredictable performance loss. Moreover, the results obtained on the standard Caltech-101 dataset proved a significant improvement on some state-of-the-art computer vision methods, designed specifically for image classification.

Future research could be made in order to further improve the texture segmentation accuracy by defining a method that automatically find a FLU configuration and an optimal parameters setting for the receptive field size and the type of encoding. As regards the classification of scene image, to improve the results, other normalization methods and feature encoding schemes can be tested and integrated within the classification framework.

Bibliography

[1] X. Liu and D. Wang, "Image and texture segmentation using local spectral histograms," *IEEE Transactions on Image Processing*, vol. 15, pp. 3066–3077, 2006.

[2] I. Karoui, R. Fablet, J.-M. Boucher, W. Pieczynski, and J.-M. Augustin, "Fusion of textural statistics using a similarity measure: application to texture recognition and segmentation," *Pattern Analysis and Applications*, vol. 11, pp. 425–434, 2008.

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[4] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[5] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[6] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of The IEEE*, vol. 86, pp. 2278–2324, 1998.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, pp. 1527–1554, 2006.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.

[9] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Neural Information Processing Systems*, 2006, pp. 801–808.

[10] T. Serre, L. Wolf, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 411–426, 2007.

[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Neural Information Processing Systems*, 2007, pp. 153–160.

[12] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[13] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *International Conference on Machine Learning*, 2009, pp. 110–880.

[14] A. Coates, H. Lee, and A. Y. Ng, "An analysis of single-layer networks in unsupervised feature learning," in *AISTATS*, 2011.

[15] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *International Conference on Machine Learning*, 2012.

[16] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[17] ——, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, 1990.

[18] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[19] T. Kohonen, M. R. Schroeder, and T. S. Huang, Eds., *Self-Organizing Maps*, 3rd ed. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2001.

[20] R. Durstenfeld, "Algorithm 235: Random permutation," *Communications of The ACM*, vol. 7, 1964.

[21] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Norwell, MA, USA: Kluwer Academic Publishers, 1991.

[22] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2010.

[23] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277–1294, 1993.

[24] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural network approach," *IEEE Transactions on Neural Networks*, vol. 8, pp. 98–113, 1997.

[25] M. Tuceryan and A. K. Jain, "Texture analysis," *The Handbook of Pattern Recognition and Computer Vision*, 1998.

[26] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 291–310, 1999.

[27] J. Daugman, "Two-dimensional spectral analysis of cortical receptive field profiles," *Vision Research*, vol. 20, pp. 847–856, 1980.

[28] A. C. Bovik, "Analysis of multichannel narrow-band filters for image texture segmentation," *IEEE Transactions on Signal Processing*, vol. 39, pp. 2025–2043, 1991.

[29] J. R. Bergen and E. H. Adelson, "Early vision and texture perception," *Nature*, vol. 333, pp. 363–364, 1988.

[30] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Annual Conference on Computer Graphics*, 1995, pp. 229–238.

[31] S. C. Zhu, Y. N. Wu, and D. Mumford, "Minimax entropy principle and its application to texture modeling," *Neural Computation*, vol. 9, pp. 1627–1660, 1997.

[32] X. Liu and D. L. Wang, "Texture classification using spectral histograms," *IEEE Transactions on Image Processing*, vol. 12, pp. 661–670, 2003.

[33] A. K. Jain and K. Karu, "Learning texture discrimination masks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 195–205, 1996.

[34] K. I. Kim, K. Jung, S. H. Park, and H. J. Kim, "Support vector machines for texture classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1542–1550, 2002.

[35] J. Melendez, X. Gironés, and D. Puig, "Supervised texture segmentation through a multi-level pixel-based classifier based on specifically designed filters," in *ICIP*, 2011, pp. 2869–2872.

[36] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (2nd Edition)*. Prentice Hall, 2002.

[37] M. Vanetti, I. Gallo, and A. Nodari, "Unsupervised self-organizing texture descriptor," in *Images: Fundamentals, Methods and Applications (CompIMAGE2012)*, 2012.

[38] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[39] S. P. Awate, T. Tasdizen, and R. T. Whitaker, "Unsupervised texture segmentation with nonparametric neighborhood statistics," in *European Conference on Computer Vision*, 2006, pp. 494–507.

[40] P. Brodatz, "Textures: a photographic album for artists and designers," 1966.

[41] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2169–2178.

[42] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *European Conference on Computer Vision*, 2004.

[43] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Conference on Image and Video Retrieval*, 2007, pp. 401–408.

[44] ——, "Image classification using random forests and ferns," in *International Conference on Computer Vision*, 2007, pp. 1–8.

[45] L. W. Renninger and J. Malik, "When is scene identification just texture recognition?" *Vision research*, vol. 44, no. 19, pp. 2301–2311, September 2004.

[46] M. Vanetti, I. Gallo, and A. Nodari, "Unsupervised feature learning using self-organizing maps," in *International Conference on Computer Vision Theory and Applications (VISAPP2013)*, 2013.

[47] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[48] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *CVPR 2004, Workshop on Generative-Model Based Vision*, 2004.

[49] G. Griffin, A. Holub, and P. Perona, "The caltech-256," Tech. Rep., 2007.

[50] A. Nodari, M. Ghiringhelli, A. Zamberletti, M. Vanetti, S. Albertini, and I. Gallo, "A mobile visual search application for content based image retrieval in the fashion domain," in *CBMI*, 2012, pp. 1–6.

[51] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" vol. 11, pp. 625–660, February 2010.

[52] P. V. Gehler and S. Nowozin, "On feature combination for multiclass object classification," in *ICCV*. IEEE, 2009, pp. 221–228.

[53] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman, "Multiple kernels for object detection," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2009.

[54] A. Vedaldi and B. Fulkerson, "Vlfeat: an open and portable library of computer vision algorithms," in *ACM Multimedia*, A. D. Bimbo, S.-F. Chang, and A. W. M. Smeulders, Eds. ACM, 2010, pp. 1469–1472.

[55] A. Hyvarinen and E. Oja, "Independent component analysis: algorithms and applications," *Neural Networks*, vol. 13, pp. 411–430, 2000.

[56] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *International Conference on Computer Vision*, 2009, pp. 2146–2153.