Information Theoretic Learning for Pixel-Based Visual Agents

Marco Gori, Stefano Melacci, Marco Lippi, and Marco Maggini

Department of Information Engineering, University of Siena
Via Roma 56, 53100 - Siena, Italy
{marco,mela,lippi,maggini}@dii.unisi.it

Abstract. In this paper we promote the idea of using pixel-based models not only for low level vision, but also to extract high level symbolic representations. We use a deep architecture which has the distinctive property of relying on computational units that incorporate classic computer vision invariances and, especially, the scale invariance. The learning algorithm that is proposed, which is based on information theory principles, develops the parameters of the computational units and, at the same time, makes it possible to detect the optimal scale for each pixel. We give experimental evidence of the mechanism of feature extraction at the first level of the hierarchy, which is very much related to SIFT-like features. The comparison shows clearly that, whenever we can rely on the massive availability of training data, the proposed model leads to better performances with respect to SIFT.

1 Introduction

The research in computer vision is still looking for unifying mechanisms to handle low level computations, as well as the extraction of symbolic representations. Convolutional learning of hierarchical features is a research topic that has received an increasing interest from the machine learning and computer vision communities in the last few years [1–5] and that offers intriguing insights on bridging low and high level vision computations. Even though its roots can be traced back to more than a decade ago (Convolutional Neural Networks, [6]), the growing availability of computer resources have encouraged the research on deep convolutional architectures, showing several improvements and interesting results [1–3]. These architectures are able to extract high level semantics, and they are not restricted to the description of low level features like, for instance, the popular Scale Invariant Feature Transform (SIFT) [7].

Following the research guidelines of convolutional neural networks, this paper gives insights on a truly new challenging approach in which we promote the idea of using pixel-based models for any computer vision problem, including those in which we need to extract high level symbolic representations. This paper borrows the idea of multi-layer convolutional architectures and receptive fields [8], but it provides an in-depth re-thinking of the way features are extracted and propagated through the layers. Motivated by the recent efforts in defining an

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part VI, LNCS 7577, pp. 864–875, 2012.

unsupervised algorithm to learn invariant features [2, 4], we use computational units based on very natural invariance mechanisms. The learning principles are based on Information Theoretic Learning [9] and, particularly, on a proper unsupervised learning algorithm [10], which is aimed at minimizing the conditional entropy, under the soft-constraint of optimizing the development of different features. We sketch the general idea of the *Developmental Visual Agents* (DVAs) and then focus attention on the mechanisms for feature extraction, under the fundamental assumption of relying on computational units that are connected to a pixel-centered receptive field that is located at different levels of the hierarchy. In so doing, the units extract features depending on image portions that are larger and larger as we move towards higher levels of the architecture. Unlike other deep architectures, the one we propose has the distinctive property of relying on computational units that incorporate classic computer vision invariances and, especially, the scale invariance. The proposed learning algorithm develops the parameters of the computational units and, at the same time, makes it possible to detect the optimal scale for each pixel.

We give experimental evidence of the mechanism of feature extraction at the first level of the hierarchy, which is very much related to SIFT-like features. The comparison shows clearly that, whenever we can rely on the massive availability of training data, the proposed model leads to better performances with respect to SIFT. This is extremely important, especially when considering that we have only exploited the first level of the deep learning architecture, whose most remarkable features are likely to emerge when extracting higher level features and when closing the loop with supervised learning schemes.

2 The DVA Architecture

The DVA model consists in a stack of L processing layers that progressively extract a set of features associated to each pixel of an input image. Each layer l computes a pixel-based probability distribution over d_l values, such that $p_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1})$ is the probability of the feature h for the pixel at coordinates $\boldsymbol{x}$, given the input $\boldsymbol{v}_{l-1}$ provided by the previous layer. Hence, the output of layer l is a probability vector field $\boldsymbol{v}_l \in [0,1]^{d_l \times M \times N}$, being $M \times N$ the input image size. The only exception to the previous forward processing scheme is represented by the first layer $l = 1$ for which $\boldsymbol{v}_0$ is the input image, represented by the RGB color levels or luminance for gray level images.

The feature detectors for layer l are implemented by a set of d_l adaptive functions $f_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1} | \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,l})$ that estimate the presence of each feature h at pixel $\boldsymbol{x}$ given the output of the previous layer. The functions depend on a set of parameters that are to be adapted through the developmental process. In particular, the *local scale* $\sigma_l(\boldsymbol{x}) \in \mathbb{R}$ and the *local rotation* versor $\boldsymbol{\nu}_l(\boldsymbol{x}) \in [0,1]^2$ are computed for each input to provide scale and rotation invariance in the feature extraction process, whereas the function parameters $\boldsymbol{\alpha}_{h,l}$ are determined through a learning procedure that aims at developing an optimal set of features for layer l. In particular, as it will be detailed in the following section, the feature detectors implement their computation considering a *receptive field* around

the current pixel $\boldsymbol{x}$ whose extension depends on the local scale $\sigma_l(\boldsymbol{x})$ and that is aligned considering the local rotation $\boldsymbol{\nu}_l(\boldsymbol{x})$. Given the feature functions $f_{h,l}$, $h = 1, \ldots, d_l$, the corresponding probability distribution is simply computed by applying the softmax function as

$$p_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1}) = \frac{e^{f_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1} | \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,l})}}{\sum_{k=1}^{d_l} e^{f_{k,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1} | \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,l})}} \ , \tag{1}$$

where the dependence of $p_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1})$ on the feature detector parameters has been neglected to simplify the notation. The feature probability distribution for the pixel $\boldsymbol{x}$ is the vector $\boldsymbol{p}_l(\boldsymbol{x}, \boldsymbol{v}_{l-1}) = [p_{1,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1}), \ldots, p_{d_l,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1})]'$. The DVA structure is depicted in Fig. 1.

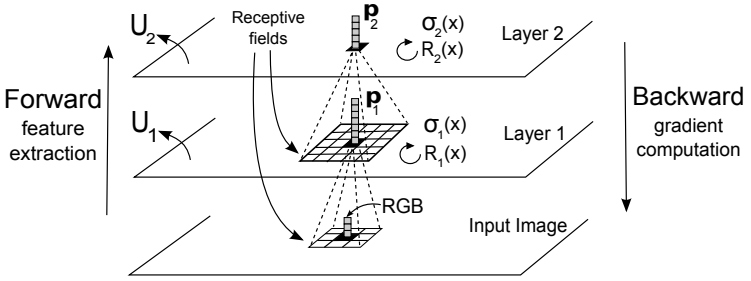

Fig. 1. Sketch of the layered computation in the DVA model. The forward step allows the extraction of the features for each pixel in the input image. The backward step is exploited in the learning of the feature detectors.

The development of the feature detectors is performed in order to optimize a given criterion through a learning process. The objective function can incorporate both supervised and unsupervised criteria at a layer level. For instance, once a semantic role is assigned to the features of a given layer or set of layers, supervisions can be provided as a target value for a subset of pixels. However, in this paper we do not consider any supervised contribution, neither contributions that implement constraints (e.g. geometrical, semantic) among the feature values in the same or different layers. These supervised criteria can be straightforwardly incorporated into the learning process once an appropriate cost function is defined and they will be at the basis of the future implementations of the DVA model.

The unsupervised criteria are generally defined at a layer level. In particular, the learning objective is to favor the development of feature detectors that are selectively triggered by specific characteristics of the local neighborhood of each pixel (i.e. only one or just few conflicting features are detected for each pixel), while, at the same time, trying to exploit all the available codebook (in principle made up of d_l symbols). The selectivity of the feature detector can be measured by the entropy of the distribution computed for each pixel, whereas the overall

use of the codebook is measured by the entropy of the overall distribution of codes. In the current implementation we considered the Rényi entropy for both the measures. In particular, the *average local entropy* for each layer l is computed as

$$\overline{H}_p(\{\boldsymbol{p}_l(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) = -\frac{1}{TNM} \sum_t \sum_{ij} \log \left[\sum_{k=1}^{d_l} p_{k,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))^2 \right] \quad (2)$$

for a batch of T input images $\boldsymbol{v}_0(t)$, $t = 1, \ldots, T$, yielding the feature vectors $\boldsymbol{p}_l(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))$. The *global entropy* of the codebook of layer l is computed as

$$H_P(\{\boldsymbol{p}_l(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) = -\log \left[\sum_{k=1}^{d_l} \left(\frac{1}{TNM} \sum_t \sum_{ij} p_{k,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t)) \right)^2 \right] . \quad (3)$$

The optimization process aims at minimizing the local entropy in order to develop selective feature detectors while maximizing the global entropy to exploit the maximum number of codebook symbols. This second contribution is needed to avoid trivial solutions in which the same feature is detected on all the input pixels [10].

Finally, the learning objective incorporates also a regularization term to favor parsimonious solutions. In particular, we can introduce a cost term to favor the development of smooth feature detection functions. In the current implementation, we considered the gradient as differential operator and the objective is to minimize its average norm for all the feature detectors by adding the cost term

$$N_l = \frac{1}{2TNMd_l} \sum_t \sum_h \sum_{ij} \|\nabla_{\boldsymbol{x}} f_{h,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\|^2 \quad (4)$$

where the dependence of the feature functions on the set of parameters was omitted to simplify the notation. The gradient can be explicitly computed given the definition of the functions $f_{h,l}$.

The three contributions can be combined to obtain the unsupervised objective function that is to be minimized in the long-term learning process to determine the optimal feature detector parameters $\boldsymbol{\alpha}_{h,l}$, $h = 1, \ldots, d_l$, $l = 1, \ldots, L$. Hence, the objective function is

$$U_l = \eta \overline{H}_p(\{\boldsymbol{p}_l(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) + (\eta - 1) H_P(\{\boldsymbol{p}_l(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) + \lambda N_l \quad (5)$$

where $\eta \in [0, 1]$ and $\lambda > 0$ are used to weight the contributions of each term. The sketch of the learning process, that requires also to determine the optimal local scale values $\sigma_l(\boldsymbol{x})$ for each input image, is reported in section 2.3.

2.1 The Feature Detectors

The adaptive feature detectors, exploited in the current implementation of the DVA model, are basically convolutional filters whose output for a given pixel $\boldsymbol{x}$

is computed as

$$f_{h,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1}) = \sum_{q=1}^{d_{l-1}} \sum_{ij} \left[\sum_{k=1}^{|\mathcal{N}_l|} \alpha_{hqk,l} e^{-\frac{\left\| \sigma_l(\boldsymbol{x}) R_l(\boldsymbol{x}) \breve{\boldsymbol{x}}_{k,l} + \boldsymbol{x} - [i,j]' \right\|^2}{2\mu_l \sigma_l(\boldsymbol{x})^2}} \right] v_{qij,l-1} \ , \qquad (6)$$

where in the left side we omitted the list of the function parameters to simplify the notation. Basically for each component q of the input field $\boldsymbol{v}_{l-1}$, the detector computes a local convolution between the input $v_{qij,l-1}$ and a kernel (the term in square bracket in the equation) that is computed as a mixture of gaussians centered in a predefined set of sample points around the local origin, $\mathcal{N}_l = \{\breve{\boldsymbol{x}}_{k,l}\}$. The coefficients $\alpha_{hqk,l}$ of the mixture are the parameters used to adapt the feature detector behavior during the learning process, whereas the set of points in $\mathcal{N}_l$ is an a-priori design choice. The local scale $\sigma_l(\boldsymbol{x})$ is used to *stretch* or *compress* the convolution kernel, in order to obtain scale invariance in the feature detector. The rotation matrix $R_l(\boldsymbol{x})$, that is defined by the local reference versor $\boldsymbol{\nu}_l(\boldsymbol{x})$, rotates the convolution kernel in order to obtain rotation invariance. The *receptive field* of the feature detector is defined by the area in which the samples of the convolution kernel are not negligible and, hence, its size varies depending on the local scale. The parameter μ_l is predefined and allows the tuning of the gaussian widths with respect to the position of their centers in the sample set $\mathcal{N}_l$, for a given reference scale, i.e. $\sigma_l(\boldsymbol{x}) = 1$.

2.2 The Local Rotation

The local rotation is determined by defining a reference direction that depends on the input configuration around the current pixel, such that any rotation in the input field yields the same direction. In the current implementation, the reference direction is computed by considering the average gradient of an appropriate norm of the input field, as

$$\boldsymbol{s}_l(\boldsymbol{x}, \boldsymbol{v}_{l-1}) = \sum_{ij} \nabla_{\boldsymbol{x}} \|\boldsymbol{v}_{l-1}\|_r \mid_{[i,j]'} e^{-\frac{||\boldsymbol{x} - [i,j]'||^2}{2\gamma\sigma_l(\boldsymbol{x})^2}} \qquad (7)$$

where the contributions around the considered pixel $\boldsymbol{x}$ are weighted by a gaussian whose width depends on the local scale $\sigma_l(\boldsymbol{x})$. The parameter γ is used to tune the width for computing the average with respect to the scale range. For the first layer, we use $r = 1$ such that the computed direction is the average of the gradients for each of the channels in the input image. For the following layers, we consider $r = 2$ since it has a nice probabilistic interpretation. In fact, in this case

$$\boldsymbol{s}_l(\boldsymbol{x}, \boldsymbol{v}_{l-1}) = \sum_{ij} \sum_{q} v_{qij,l} \nabla_{[x,y]'} v_{qxy,l-1} \mid_{[i,j]'} e^{-\frac{||\boldsymbol{x} - [i,j]'||^2}{2\gamma\sigma_l(\boldsymbol{x})^2}} \ , \qquad (8)$$

that corresponds to an average of the gradients of each component of the probability field, weighted by the corresponding probability[1].

[1] In the implementation the gradients are computed with the 3×3 Sharr filter.

Finally the direction versor, to be used to compute the rotation matrix $R_l(\boldsymbol{x})$, is obtained by the normalization $\boldsymbol{\nu}_l(\boldsymbol{x}) = \frac{\boldsymbol{s}_l(\boldsymbol{x},\boldsymbol{v}_{l-1})}{||\boldsymbol{s}_l(\boldsymbol{x},\boldsymbol{v}_{l-1})||}$. If $||\boldsymbol{s}_l(\boldsymbol{x},\boldsymbol{v}_{l-1})||$ is below a given threshold, as it happens in regions with constant or isotropic input values, a predefined default direction is chosen.

2.3 The DVA Learning Algorithm

The DVA model features two different sets of parameters that have to be determined by an optimization procedure: the feature detector convolutional kernel coefficients $\boldsymbol{\alpha}_{h,l}$ and the local scale fields $\sigma_l(\boldsymbol{x})$. Even if these two sets of parameters are related to each other, their nature is quite different. In fact, the feature detector coefficients are to be developed through a long-term learning procedure that requires to extract sufficient statistics from a significant set of images. On the contrary, the scale parameters are basically a local property of each image, even if their optimal values depend also on the current configuration of the convolutional kernel coefficients. Basically, the parameters $\boldsymbol{\alpha}_{h,l}$ determine the DVA behavior, whereas the local scales $\sigma_l(\boldsymbol{x})$ are a by-product of the feature extraction process on each single image.

We briefly sketch the optimization policy adopted in the early experimentation with the DVA model, but many aspects related to the learning procedure are still an open research activity.

The development of appropriate feature detectors requires a long-term learning process. Hence, the update policy follows an EM-like procedure in which the scale is estimated to compute the extracted features with the current $\boldsymbol{\alpha}_{h,l}$ values (forward step). In particular, since the scale is a local property, its values are optimized with respect to the local entropy computed at each pixel $\boldsymbol{x}$ as

$$H_p(\boldsymbol{p}_l(\boldsymbol{x},\boldsymbol{v}_{l-1}(t))) = -\log\left[\sum_{k=1}^{d_l} p_{k,l}(\boldsymbol{x},\boldsymbol{v}_{l-1}(t))^2\right] , \tag{9}$$

when processing the input image $\boldsymbol{v}_0(t)$. Starting from the bottom layer $l = 1$, layer by layer we select the scale map that yields the most discriminative decision for the feature detector at each pixel (i.e. the one that minimizes the local entropy of eq. (9)). Hence, the scale optimization is part of the forward step that allows the computation of the feature maps for all layers.

After the forward step is performed on all the T images in the learning batch, it is possible to evaluate the objective function U_l for each layer, given the current values of the parameters $\boldsymbol{\alpha}_{h,l}$. In order to optimize U_l, we exploit a gradient descent technique[2]. More precisely, once U_l has been computed, the evaluation of the partial gradient for a given input image follows a back-propagation like scheme among the functional blocks that compose the architecture (i.e. the feature detection functions, the local rotation module, etc.).

The procedure described so far requires large memory resources to store the forward variables computed for all the T images, since the global entropy term

[2] We employed a version of the Resilient Backpropagation (RPROP) algorithm [11].

H_P of eq. (3) combines their contributions using a non linear function. To overcome this issue, it is possible to approximate the global entropy computation by applying an incremental moving average technique, so that the partial gradient can be evaluated on each image independently. Basically, the argument of the $\log(\cdot)$ function in H_P is computed by mixing the statistics obtained for the images already processed with the current $\boldsymbol{\alpha}_{h,l}$ parameters together with a contribution derived from the statistics computed at the previous iteration of the learning procedure (i.e. with the previous parameter configuration). Formally,

$$\nabla_{\boldsymbol{\alpha}_{h,l}} U_l \approx \frac{1}{T} \sum_{t=1}^{T} \nabla_{\boldsymbol{\alpha}_{h,l}} \hat{U}_l(t) \ , \tag{10}$$

where $\hat{U}_l(t)$ is the partial objective function which considers the local entropy and the regularizer only for the t-th image, whereas the global entropy is the approximated estimate for the whole batch up to image t.

The back-propagation technique could also be used to compute the gradient of the local entropy with respect to the scale. However, in the implementation we exploited a grid search since the scale values are quantized for efficiency reasons.

The development of an appropriate set of feature detectors requires several learning iterations. We can design the learning process considering a set of *developmental stages* that focus on easier tasks first. These stages are implemented by a schedule of the learning algorithm parameters. The possible schedules are currently under study. For instance, the scale range can be progressively widened, starting with large constant scales (i.e. blurred images) in order to first learn features at a coarse level, and the focus the learning towards the development of feature related to small scale details. Future directions concern the definition of developmental stages related to the layers in the DVA (the learning process should focus first on the lower layers), and to the progressive introduction of external knowledge in the learning process (like supervisions and constraints derived from rules).

3 Experimental Results

We evaluated the *low-level* feature extraction ($L = 1$) mechanism of DVAs on several experimental settings aimed at showing different properties of the model. In all the experiments, we set $\mu_1 = \gamma = \frac{1}{9}$, so that for integer values of the scale $\sigma_1(\boldsymbol{x})$ we get receptive fields defined on $(2\sigma_1(\boldsymbol{x}) + 1) \times (2\sigma_1(\boldsymbol{x}) + 1)$ patches (± 3 standard deviations from the mean). The $\boldsymbol{x}'_k$ points are placed on the vertices of the 3×3 grid centered around the origin.

The first test aims at showing the ability of learning distinct features. A DVA was trained to learn 4 different filters with $\sigma_1(\boldsymbol{x})$ set to the fixed value 1 (3×3 patches), when processing a pattern consisting of a grid whose lines are 1 pixel wide and separated by 1 pixel (see Fig. 2). The functions, reported in grayscale in Fig. 2, detect 4 distinct patterns (+, O, H, and rotated H) that can be easily distinguished on the input grid. As long as the objective function decreases (see

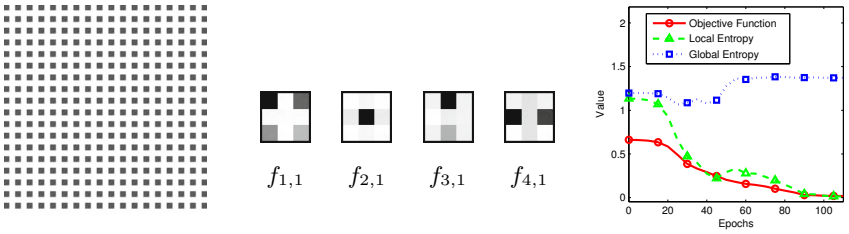

Fig. 2. A white grid on a dark background (lines are separated by 1 pixel). Each receptive field is a 3×3 patch ($\sigma_1(\boldsymbol{x}) = 1$). Left to right: the input pattern, the 4 learned filters, the objective function and entropies with respect to the training iterations.

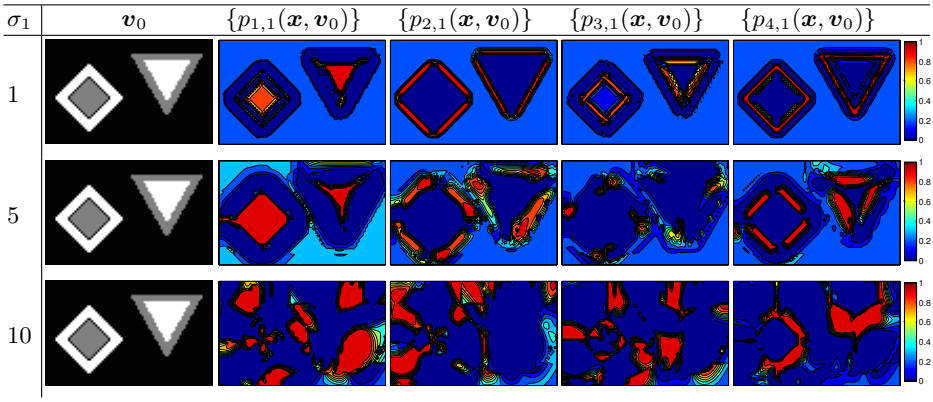

Fig. 3. Four features learned at different scales σ_1. The input example $\boldsymbol{v}_0$ and the feature maps $\{p_{h,1}(\boldsymbol{x}, \boldsymbol{v}_0)\}$, $h = 1, \ldots, 4$ are shown.

the plot in Fig. 2), the local and global entropy are minimized and maximized, respectively (the maximum value of the latter is $\ln(4) \approx 1.38$, that is also added to U_1 to set its minimum value to zero).

In order to evaluate the type of features extracted at different scales, we trained a 4-feature DVA on a grayscale image composed of two traffic signs, downscaled to 80 × 50. Fig. 3 reports the resulting 4 feature maps, i.e. the probabilities $p_{h,1}(\boldsymbol{x}, \boldsymbol{v}_0)$, $h = 1, \ldots, 4$ for all the image pixels. The DVA learns finer features as long as the scale decreases. We can clearly distinguish edge-like features, corners, uniform regions. When $\sigma_1(\boldsymbol{x}) = 10$, the features become hardly interpretable, but it is still appreciable how they capture different properties.

The DVA model includes rotation invariance, and the scale of each pixel is automatically adjusted. The next experiment is composed of two pictures at the same resolution, representing the same object at a different scale/angle. Fig. 4 shows the case of two guitars, out of which a DVA has extracted 4 features using a variable scale ($\sigma_1(\boldsymbol{x}) \in [2, 10]$). Comparing the pairs of corresponding feature maps (each column of Fig. 4), it can be noticed that the DVA has learned the same features from both images, exploiting the invariance properties with respect

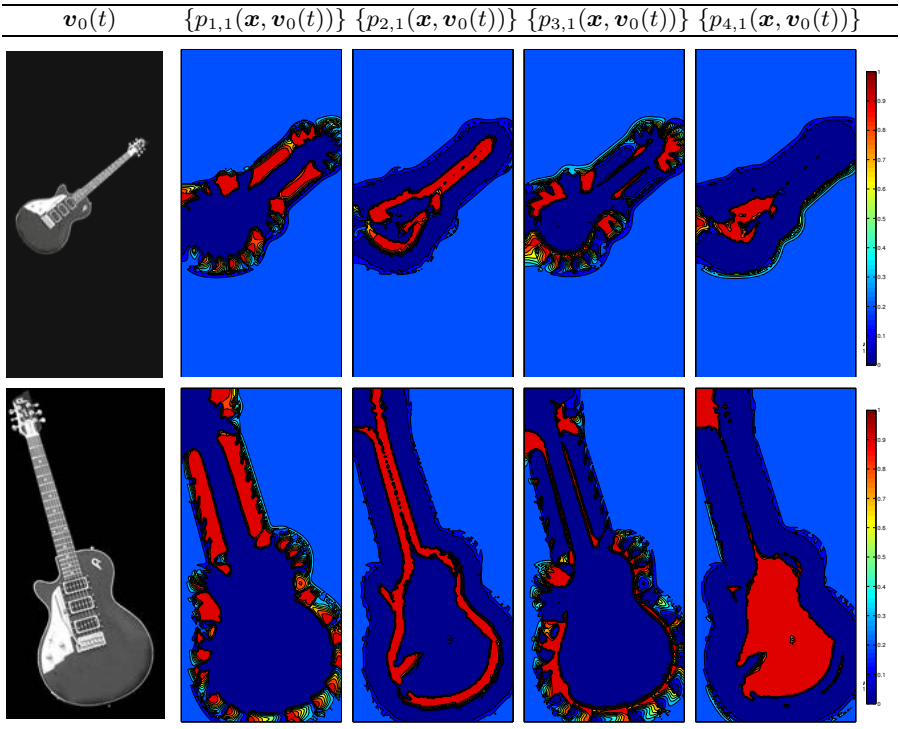

Fig. 4. Two images of the same guitar represented at a different orientation/scale. DVA extracts similar features for the two pictures (variable scale, $\sigma_1(\boldsymbol{x}) \in [2, 10]$).

to rotation and scale changes. The two leftmost graphs of Fig. 5 depict the scale of each pixel: smaller scales are found along the borders of the guitar as well as in regions in which there are more details. The rightmost graph represents the percentage of pixels (excluding the background) at each scale level. The scale of the pixels on the external borders of the guitars and in internal uniform regions are comparable between the two images, so that the plots show a similar trend. However we can appreciate that the number of pixels at smaller scales is larger in the case of the small guitar.

Fig. 6 shows the results of an experiment in the case of a multichannel input. An RGB image (Fig. 6, top-left) is fed to DVA that is requested to learn 6 distinctive features (right portion of Fig. 6) as well as the scale of each pixel (Fig. 6, bottom-left). The 3 features on the top-row correspond to the detection of red regions (the tomatoes), of the transitions between red and green, and of information on the gray levels (notice the gray leaf on the bottom right corner), respectively. The features on the bottom-row are related to details on the green background region. This illustrates that DVA has nicely exploited the colors to learn distinguishable and relevant low level features. As in the previous experiment, the scale ($\sigma_1(\boldsymbol{x}) \in [1, 5]$) has been adjusted by DVA to better capture

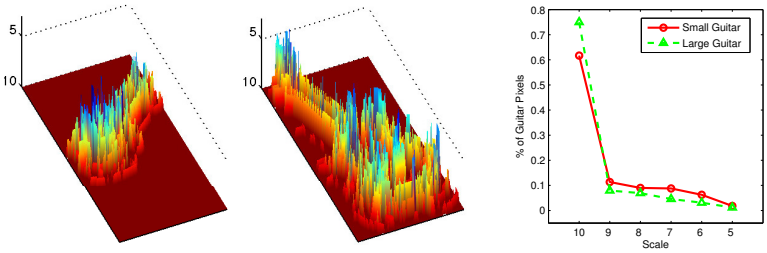

Fig. 5. The scale maps for the pixel in the two images of Fig. 4 as determined by DVA. The rightmost graph is the percentage of pixels (discarding the background) that are associated to a certain scale (from larger to smaller scales).

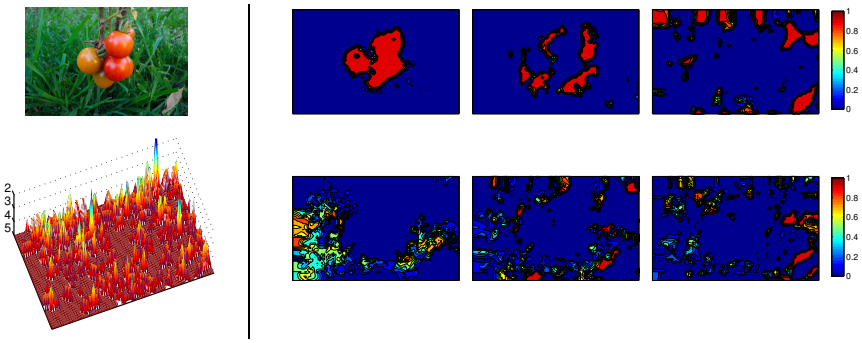

Fig. 6. Left column: an RGB input image and the scale developed by DVA ($\sigma_1(\boldsymbol{x}) \in [1, 5]$). Right column: the 6 features maps learned by DVA on the RGB input.

the details at the appropriate grain. For instance, the region of the tomatoes is detected at a large scale since it is close to be uniform.

In order to better investigate the role of color, a DVA has been trained to learn 3 features on the 80×50 images of Fig. 7 (strawberries, vegetables, ocean). A large (fixed) scale was used ($\sigma_1(\boldsymbol{x}) = 20$), so that the receptive field of each pixel involves a neighboring region of 41×41 pixels. The extracted features are shown on the right of each image. At such a large scale the image under each receptive field is strongly blurred, and the DVA has automatically learned to distinguish the RGB values, by developing a specific detector for each color.

3.1 Image Classification

DVA are general purpose agents performing pixel-based labeling. The information at the lower level must be hierarchically processed to capture high level image properties such as the class to which it belongs. However, in order to investigate the quality of the *low level* features, we simulated a scenario that allowed a qualitative comparison of a single-level DVA with image classification

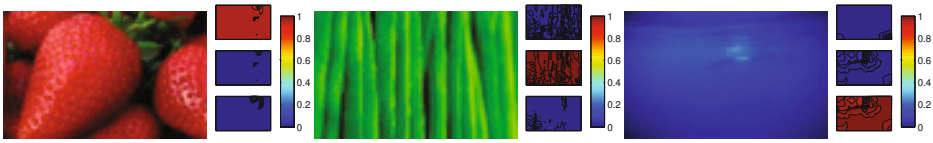

Fig. 7. Three RGB images. Due to the large scale, the learned features (on the right side of each image) correspond to the RGB values ($\sigma_1(\boldsymbol{x}) = 20$) .

Table 1. Average accuracy (and std) on the test dataset from the Caltech 101

Training Images	**10**	**50**	**150**
SIFT	38.3 (10.1)	45.5 (0.9)	46.7 (4.0)
DVA	37.8 (9.6)	47.5 (5.8)	48.7 (3.1)

procedures based on Scale Invariant Feature Transform descriptors (SIFT) [7]. In detail, first a gaussian smoothing was applied to each feature map, to better capture the relationships of neighboring pixels (simulating an artificial higher level of computation). Then, the average of the feature vectors of the pixel in the image was used as input for a linear Support Vector Machine (SVM) classifier.

We selected the data from the Caltech 101 collection [12], composed of several real-world images grouped in 101 categories. We considered the first 10 classes with ≥ 50 examples each (in alphabetical order) and we trained a DVA on a variable number D of images ($D = 1, 5, 15$ per class) converted to grayscale and scaled to 151×143 (preserving proportions by zero padding the shorter edge, if needed). The number of DVA features was set to 15. For an accurate comparison, SIFTs descriptors were computed for each pixel and a codebook of 15 words was built by K-Means (as popularly done [13]), so that each image was represented by a visual word of length 15. Classifier parameters were tuned by a 5-fold cross-validation (or using an additional example per class when $D = 1$), and the accuracy was measured on 200 test instances (20 per class). The results in Table 1 indicate a good quality of the features learned by DVA, that benefits from a larger number of examples, encouraging the research along this direction.

4 Conclusions

The paper introduces the idea of using deep architectures for computer vision under the strong hypothesis of developing pixel-based features at all the layers of the hierarchy. The most remarkable feature with respect to others deep learning schemes is that we incorporate crucial invariances (e.g. scale invariance), that emerge thanks to the minimization of a proper information-based criterion. The very promising experimental results given at the first layer of the architecture shed light on most important future challenges, where those invariance properties are expected to arise at higher levels of the hierarchy.

Acknowledgments. This research was partially supported by the research grant PRIN2009 "Learning Techniques in Relational Domains and Their Applications" (2009LNP494) from the Italian MURST.

References

1. Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks. In: Int. Joint Conference on Neural Networks, pp. 2809–2813. IEEE (2011)
2. Kavukcuoglu, K., Sermanet, P., Boureau, Y., Gregor, K., Mathieu, M., LeCun, Y.: Learning convolutional feature hierarchies for visual recognition. In: Advances in Neural Information Processing Systems (2010)
3. Lee, H., Grosse, R., Ranganath, R., Ng, A.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: International Conference on Machine Learning, pp. 609–616. ACM (2009)
4. Kavukcuoglu, K., Ranzato, M., Fergus, R., LeCun, Y.: Learning invariant features through topographic filter maps. In: CVPR, pp. 1605–1612. IEEE (2009)
5. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: CVPR, pp. 2146–2153. IEEE (2009)
6. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)
7. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
8. Fukushima, K.: Neocognitron: A hierarchical neural network capable of visual pattern recognition. Neural Networks 1(2), 119–130 (1988)
9. Principe, J.: Information theoretic learning: Renyi's entropy and kernel perspectives. Springer (2010)
10. Melacci, S., Gori, M.: Kernel methods for minimum entropy encoding. In: International Conference on Machine Learning and Applications, pp. 352–357. IEEE (2011)
11. Riedmiller, M., Braun, H.: A direct algorithm method for faster backpropagation learning: the RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 1, pp. 586–591 (1993)
12. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. Computer Vision and Image Understanding 106(1), 59–70 (2007)
13. van Gemert, J., Veenman, C., Smeulders, A., Geusebroek, J.: Visual word ambiguity. IEEE TPAMI 32(7), 1271–1283 (2010)