Information-based learning of deep architectures for feature extraction

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

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche
Università degli Studi di Siena, Via Roma 56, I-53100 Siena (Italy)
{mela,lippi,marco,maggini}@diism.unisi.it

Abstract. Feature extraction is a crucial phase in complex computer vision systems. Mainly two different approaches have been proposed so far. A quite common solution is the design of appropriate filters and features based on image processing techniques, such as the SIFT descriptors. On the other hand, machine learning techniques can be applied, relying on their capabilities to automatically develop optimal processing schemes from a significant set of training examples. Recently, deep neural networks and convolutional neural networks have been shown to yield promising results in many computer vision tasks, such as object detection and recognition. This paper introduces a new computer vision deep architecture model for the hierarchical extraction of pixel–based features, that naturally embed scale and rotation invariances. Hence, the proposed feature extraction process combines the two mentioned approaches, by merging design criteria derived from image processing tools with a learning algorithm able to extract structured feature representations from data. In particular, the learning algorithm is based on information-theoretic principles and it is able to develop invariant features from unsupervised examples. Preliminary experimental results on image classification support this new challenging research direction, when compared with other deep architectures models.

1 Introduction

Nowadays, deep architectures are receiving a growing attention in the areas of machine learning and computer vision [1–3]. In this paper, we propose a new challenging approach to computer vision, which is inspired by the literature on deep architectures [4]. Like for low–level vision, it is a truly pixel–based model, which carries out unsupervised feature extraction by information-theoretic learning principles. This hierarchical model is named a *Developmental Visual Agent* (DVA). While the proposed learning algorithm is formulated in an unsupervised setting, future directions of research will also include the possibility of enforcing supervisions and other types of semantic constraints coming from prior knowledge of the problem.

The focus of this paper is on hierarchical feature extraction, a fundamental task of computer vision where DVA can represent a sound and valuable alternative to classic deep architecture models, such as Deep Belief Networks [1] and

Convolutional Neural Networks [5]. Although deep architectures have long been introduced in the nineties, recent years have seen a second birth of this research direction, due both to the massive increase of the available computational resources, and to the connections with biological models of the visual cortex [6]. Unlike these well–known models which are nowadays widely employed in many tasks, our approach naturally incorporates classic properties of computer vision, such as scale and rotation invariance, by relying on computational units which are connected to pixel–centered receptive fields at each layer in the hierarchy. In so doing, moving to upper layers, each computational unit will process a growing portion of the input image, therefore being able to construct more sophisticated features. The proposed learning scheme could also be naturally adapted to managing input videos or streams by extending the receptive fields to consider the time dimension. At each level of the hierarchy, a set of discrete–value features is developed while processing the input image stream by an unsupervised learning algorithm based on information–theoretic criteria. The algorithm both estimates the local invariance parameters for each pixel (i.e. the local scale and eventually the rotation) and adapts the behavior of the computational units on a long–term horizon.

The paper is organized as follows. The next section presents the DVA deep architecture describing the computational units and how scale and rotation invariance is embedded into the model. Then, Section 3 introduces the learning algorithm, that is based on unsupervised criteria defined by information–theoretic principles. In Section 4 some preliminary results are presented, showing how invariant features are developed and comparing the proposed architecture with Deep–Belief networks on a multi–class image classification benchmark. Finally, Section 5 draws the conclusions and describes the future research directions.

2 The DVA deep architecture

In the DVA architecture, the feature extraction is performed by a layered deep network, that hierarchically computes a set of features associated to each pixel $\boldsymbol{x}$ of a given $N \times M$ input image (see Fig. 1). Each layer l in the deep architecture ($l = 1, \ldots, L$) computes a set of probability distributions for C_l features, $p_{c,l}(s_{c,h}|\boldsymbol{x}, \boldsymbol{v}_{l-1})$, $c = 1, \ldots, C_l$, over sets of $d_{c,l}$ discrete symbols for each pixel. In particular, $p_{c,l}(s_{c,h}|\boldsymbol{x}, \boldsymbol{v}_{l-1})$ denotes the probability of the h–th symbol $s_{c,h}$ for the c–th feature of layer l, computed for the pixel $\boldsymbol{x}$, given the input $\boldsymbol{v}_{l-1}$ provided by the previous layer. The input image is fed to the first layer as $\boldsymbol{v}_0$, that is a $d_0 \times N \times M$ tensor, where d_0 is the number of image channels (i.e. $d_0 = 1$ for grayscale images, $d_0 = 3$ for color images). The probability distribution for each feature is a histogram with $d_{c,l}$ entries that can be collected into the vector $\boldsymbol{p}_{c,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1})$. The output of the l-th layer is a $d_l \times N \times M$ tensor $\boldsymbol{v}_l$, where $d_l = \sum_{c=1}^{C_l} d_{c,l}$, and it is obtained by the concatenation of the $\boldsymbol{p}_{c,l}$ histograms over all the input pixels, i.e. $\boldsymbol{v}_l = \{\boldsymbol{p}_{c,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}) \mid c = 1, \ldots, C_l;\ i = 1, \ldots, N;\ j = 1, \ldots, M\}$. The tensor $\boldsymbol{v}_l$ is fed as input to the $(l+1)$-th layer.

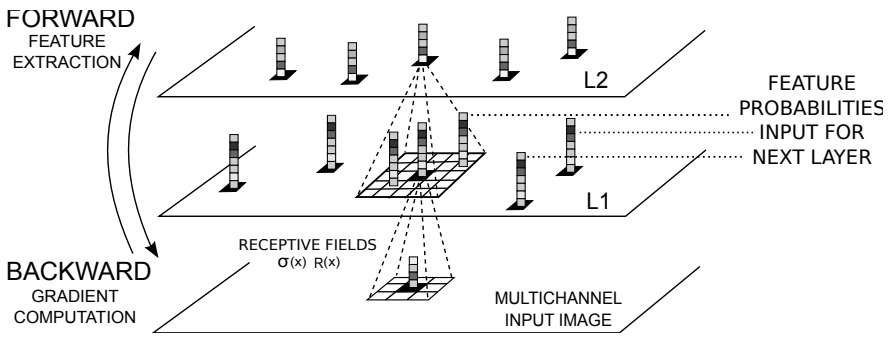
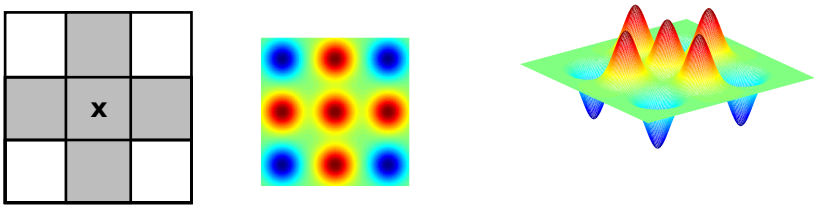


Fig. 1. An example of DVA deep architecture. The forward step extracts the pixel-based features for the input image, while the backward step exploits the gradient computation.

Each probability distribution is associated to a feature extractor that is basically an adaptive filter $f_{h,cl}(\boldsymbol{x}, \boldsymbol{v}_{l-1}|\sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,cl})$ depending on a set of parameters: (1) the *local scale* $\sigma_l(\boldsymbol{x}) \in I\!R^+$ and (2) the *local rotation* versor $\boldsymbol{\nu}_l(\boldsymbol{x}) \in [0,1]^2$ are computed for each pixel in order to provide scale and rotation invariance for the features, while (3) the *filter parameters* $\boldsymbol{\alpha}_{h,cl}$ are learned in order to develop an optimal set of features for layer l. As highlighted in Fig. 1, each feature detector considers a *receptive field* around each pixel $\boldsymbol{x}$. The extension of such area depends of the value of the local scale $\sigma_l(\boldsymbol{x})$, while the local rotation $\boldsymbol{\nu}_l(\boldsymbol{x})$ determines its orientation. The output of the filters $f_{h,cl}$ is computed as

$$f_{h,cl}(\boldsymbol{x}, \boldsymbol{v}_{l-1}|\sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,cl}) = \sum_{q=1}^{m_{cl}} \sum_{ij} \pi_{q,cl}(\boldsymbol{v}_{l-1}[\boldsymbol{x}_{ij}]) \cdot \left[\sum_{k=1}^{|\mathcal{N}_l|} \alpha_{hqk,cl} e^{-\frac{\left\| \sigma_l(\boldsymbol{x}) R_l(\boldsymbol{x}) \hat{\boldsymbol{x}}_{k,l} + \boldsymbol{x} - [i,j]' \right\|^2}{2\mu_l \sigma_l(\boldsymbol{x})^2}} \right], \tag{1}$$

where $\pi_{q,cl}(\boldsymbol{v}_{l-1}[\boldsymbol{x}_{ij}])$ is a projection of the input $\boldsymbol{v}_{l-1}$ of layer l onto a subspace with m_{cl} dimensions, that favors the development of distinct features. In fact, a different projection $\pi_{\cdot,cl}$ is exploited for each feature c of layer l. For instance, a random projection can be employed trying to reduce the dependences among the inputs processed by each feature function $f_{\cdot,cl}$. Given the input tensor, each component $\pi_{q,cl}(\boldsymbol{v}_{l-1}[\boldsymbol{x}_{ij}])$ is spatially convoluted with a kernel (the term in square bracket in Eq. 1) defined as a mixture of gaussians centered in a predefined set of sample points around the local origin, $\mathcal{N}_l = \{\hat{\boldsymbol{x}}_{k,l}\}$ (see Fig. 2 for an example). The convolutional kernel is scaled and rotated by optimally setting the values $\sigma_l(\boldsymbol{x})$ and $R_l(\boldsymbol{x})$, that is the rotation matrix induced by the versor $\boldsymbol{\nu}_l(\boldsymbol{x})$. 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. $\sigma_l(\boldsymbol{x}) = 1$. The filter equation can be rewritten by



(a) (b) (c)

Fig. 2. Example of a filter layout. The filter detects a cross (a) with a 3×3 grid of gaussians. The resulting convolutional kernel is shown in (b) and (c) for a reference scale and rotation. The scale and rotation parameters modify the filter pattern to yield invariance in the detection of the cross pattern.

introducing the variables $\xi_{qk,cl}(\boldsymbol{x}, \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}))$ defined as

$$\xi_{qk,cl}(\boldsymbol{x}, \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x})) = \sum_{ij} \pi_{q,cl}(\boldsymbol{v}_{l-1}[\boldsymbol{x}_{ij}]) e^{-\frac{\left\| \sigma_l(\boldsymbol{x}) R_l(\boldsymbol{x}) \hat{\boldsymbol{x}}_{k,l} + \boldsymbol{x} - [i,j]' \right\|^2}{2\mu_l \sigma_l(\boldsymbol{x})^2}} , \qquad (2)$$

such that

$$f_{h,cl}(\boldsymbol{x}, \boldsymbol{v}_{l-1} | \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}), \boldsymbol{\alpha}_{h,cl}) = \sum_{q=1}^{m_{cl}} \sum_{k=1}^{|\mathcal{N}_l|} \alpha_{hqk,cl} \ \xi_{qk,cl}(\boldsymbol{x}, \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x})) \ . \qquad (3)$$

The equation shows that the filter implements a linear function parametrized by the coefficients $\alpha_{hqk,cl}$ given the input variables $\xi_{qk,cl}(\boldsymbol{x}, \sigma_l(\boldsymbol{x}), \boldsymbol{\nu}_l(\boldsymbol{x}))$, that depend on the local scale and rotation parameters. This formulation allows a more efficient computation and delineates some alternative approaches both for the optimization of the invariance parameters, that actually affect the distribution of the $\xi_{qk,cl}$ variables, and for the implementation of the feature detectors, that can be realized also by non–linear adaptive functions depending on the $\xi_{qk,cl}$ inputs (e.g. SVMs can be employed). These directions are currently under investigation.

The probability distribution over the set of values for the c-th feature of layer l is simply computed via the softmax function

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

The feature development process is driven by the optimization of an objective function, which in principle can incorporate both supervised and unsupervised

criteria at each layer. Although the use of supervisions and geometrical/semantic constraints will be at the basis of the future development of DVA models, in this paper we focus on unsupervised feature extraction.

At each layer l and for each feature c, the objective function is designed to favor the development of a discriminative value for each pixel, while trying to exploit the whole available codebook (i.e., all the $d_{c,l}$ symbols). The first condition can be met by minimizing the entropy of the probability distribution for the values of the feature c at each pixel, while the second can be obtained by maximizing the entropy of the overall distribution of codes for the feature c on the processed pixels. The two contributions will be referred to as *average local entropy* and *global entropy*, respectively [7]. We employed the Rényi entropy to carry out both measures, but the objective functions can be reformulated using other similar information–theoretic criteria. The *average local entropy* of the values for each feature c of layer l is computed as

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

for a batch of T input images $\boldsymbol{v}_0(t)$, $t = 1, \ldots, T$. On the other hand, the *global entropy* of the codebook for the feature c in layer l is computed as

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

A regularization term $N_{c,l}$ is included in the objective function as well, to favor smooth solutions on the image plane,

$$N_{c,l} = \frac{1}{2TNMd_{c,l}} \sum_t \sum_h \sum_{ij} \|\nabla_{\boldsymbol{x}} f_{h,cl}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\|^2 ,$$

where we considered the gradient as differential operator, which can be analytically computed given the definition of the functions $f_{h,cl}$.

These three contributions are combined to obtain the unsupervised objective function driving the learning algorithm for the feature c of layer l, as

$$U_{c,l} = \eta \overline{H}_p(\{\boldsymbol{p}_{c,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) + (\eta - 1) H_P(\{\boldsymbol{p}_{c,l}(\boldsymbol{x}_{ij}, \boldsymbol{v}_{l-1}(t))\}) + \lambda N_{c,l} \quad (4)$$

where $\eta \in [0, 1]$ and $\lambda > 0$ are used to weigh the contributions of each term. Note that the sign of the combination coefficients, $\eta > 0$ and $\eta - 1 < 0$, requires to minimize $\overline{H}_p$ and to maximize H_P, as described above. The minimization of these functions for each feature c and layer l yields the optimal feature detector parameters $\boldsymbol{\alpha}_{h,cl}$, $h = 1, \ldots, d_{c,l}$, $c = 1, \ldots, C_l$, $l = 1, \ldots, L$.

3 Entropy-based learning

The DVA model described in the previous section can be trained following different strategies, according to the scenarios where the agent operates. A layer–by–layer procedure can be simply employed, similarly to most deep architecture

systems, where each layer l is trained independently from the others, starting from the bottom. A different learning strategy, which is currently under investigation and is beyond the goals of this paper, is to jointly learn the parameters of all the layers, possibly in a supervised setting, similarly to what happens in the fine-tuning phase of deep belief networks [1].

Within a single layer l of the DVA architecture, the learning algorithm has to optimize two distinct sets of parameters: (1) the feature detector convolutional kernel coefficients $\boldsymbol{\alpha}_{h,cl}$ for each feature c and (2) the local scale, $\sigma_l(\boldsymbol{x})$, and rotation, $\boldsymbol{\nu}_l(\boldsymbol{x})$, fields. There is a key difference between these two sets of parameters. While the $\boldsymbol{\alpha}_{h,cl}$ coefficients have to model a pool of adaptive feature–extractor filters and are, therefore, developed through a long–term learning procedure, the invariance parameters have to be *locally* adapted for each pixel of each image, even if their optimal values will depend on the current configuration of the convolutional kernel coefficients.

The $\boldsymbol{\alpha}_{h,cl}$ parameters are optimized with respect to the objective functions $U_{c,l}$ defined in Eq. 4, following a gradient descent procedure over a batch of T training images. More precisely, for each image in the training set a forward and a backward step are performed, following a backpropagation–like schema (see Fig. 1). During the forward phase on the input image $\boldsymbol{v}_0(t)$, the scale parameters $\sigma_l(\boldsymbol{x}, t)$ are first adjusted, using the current values of the coefficients $\boldsymbol{\alpha}_{h,cl}$. Since the scale is a local property, its values are optimized with respect to the local entropy for all the features c in the layer[1] computed at each pixel $\boldsymbol{x}$ as

$$H_l = \frac{1}{C_l} \sum_{c=1}^{C_l} \overline{H}_p(\{\boldsymbol{p}_{c,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1}(t))\}) \ . \tag{5}$$

In this way, the scale is estimated in order to yield the most discriminative configuration for the values of the features, given the current $\boldsymbol{\alpha}_{h,cl}$ values. This solution reduces the complexity of the architecture since the scale is independent on the specific feature. This choice favors the development of features that depend on input patterns that cover a similar area around the considered pixel at a given layer. Basically, all the C_l features at layer l will detect configurations at the same scale. In general, it is possible to consider a different local scale for each feature by a straightforward extension of the optimization algorithm, but we preferred not to adopt this solution to reduce the number of parameters. After adjusting the scale, the forward step is completed by computing the value of the objective functions $U_{c,l}$, using the computed $\sigma_l(\boldsymbol{x})$. During the backward phase, the evaluation of the partial gradient of the functions $U_{c,l}$ for a given input image follows a back–propagation like scheme. When the gradient computation is completed, the parameters $\boldsymbol{\alpha}_{h,cl}$ are updated using a gradient descent technique. In the implementation, we employed a version of the Resilient Backpropagation (RPROP) algorithm [8].

[1] For efficiency reasons, since the scale values are quantized, we exploited a grid search to find the optimal parameters of $\sigma_l(\boldsymbol{x})$, rather than computing the gradient of the local entropy.

Finally, given the scale $\sigma_l(\boldsymbol{x})$, the local rotation versor $\boldsymbol{\nu}_l(\boldsymbol{x})$ is computed by considering the (weighted) average gradient over the receptive field. At the first layer ($l = 1$), $\boldsymbol{\nu}_l(\boldsymbol{x})$ is chosen as the local average of the gradients for each channel in $\boldsymbol{v}_0$. At the other layers ($l > 1$), an appropriate norm of $\boldsymbol{p}_{c,l}(\boldsymbol{x}, \boldsymbol{v}_{l-1})$ is selected, due to the probabilistic nature of such vector, such that

$$\boldsymbol{\nu}_l(\boldsymbol{x}) = \frac{1}{Z} \sum_{ij} \nabla_{\boldsymbol{x}} \|\boldsymbol{v}_{l-1}\|_r|_{[i,j]'} e^{\frac{\|\boldsymbol{x}-[i,j]'\|^2}{2\gamma\sigma(\boldsymbol{x})^2}}$$

where Z is the normalization factor, $\gamma > 0$ is a parameter that affects the extension of the average area with respect the local scale, and r is the index of the exploited norm (in the experiments we used $r = 2$).

Developing appropriate and discriminant feature detectors is a learning process requiring several iterations. In order to facilitate such process, a procedure based on a set of *developmental stages* can be designed, so that the agent can focus on easier tasks first, and therefore developing low-level features. We are currently investigating possible strategies to implement this stage-based learning procedure. One possibility is to progressively widen the scale range, starting with large constant scales, corresponding to feeding the agent with blurred input images, in order to first learn features at a coarse level, while focusing, in a second stage, on the development of features related to small scale details. Within this context, other possible research directions include the progressive introduction of external knowledge within the learning process, for example in the form of logic rules, constraints or simply supervisions.

4 Experiments

We trained DVAs by setting $\mu_l = \frac{1}{9}$ for all layers, so that for integer values of the scale $\sigma_l(\boldsymbol{x})$ we get receptive fields defined on $(2\sigma_l(\boldsymbol{x}) + 1) \times (2\sigma_l(\boldsymbol{x}) + 1)$ patches (± 3 standard deviations from the mean). The $\hat{\boldsymbol{x}}_k$ points are placed on the vertices of the 3×3 grid centered in [0,0], while the $\boldsymbol{\nu}_l(x)$ versor was discretized along 16 different angle directions.

We first present some insights on the *low-level* ($L = 1$) feature extraction. In order to show how DVA models can exploit scale and rotation invariance, we considered three pictures at the same resolution, representing the same object at different scales/angles. Fig. 3 shows the case of three teapots, out of which a DVA has extracted a feature ($C_1 = 1$) with 3 different values using a variable scale ($\sigma_1(\boldsymbol{x}) \in [1, 5]$). Comparing the corresponding feature maps (each column of Fig. 3), it can be noticed that the DVA has learned the same features from all the images, exploiting the invariance properties with respect to rotation and scale changes. The rightmost graph of Fig. 3 depicts the scale of each pixel: smaller scales are found along the borders of the teapot as well as in more detailed internal regions. Moving from larger to smaller teapots, some details are lost due to the size reduction/subsampling.

Image Classification. With the aim of investigating the impact of a multi–layer feature extraction mechanism, compared to a single–layer architecture, we

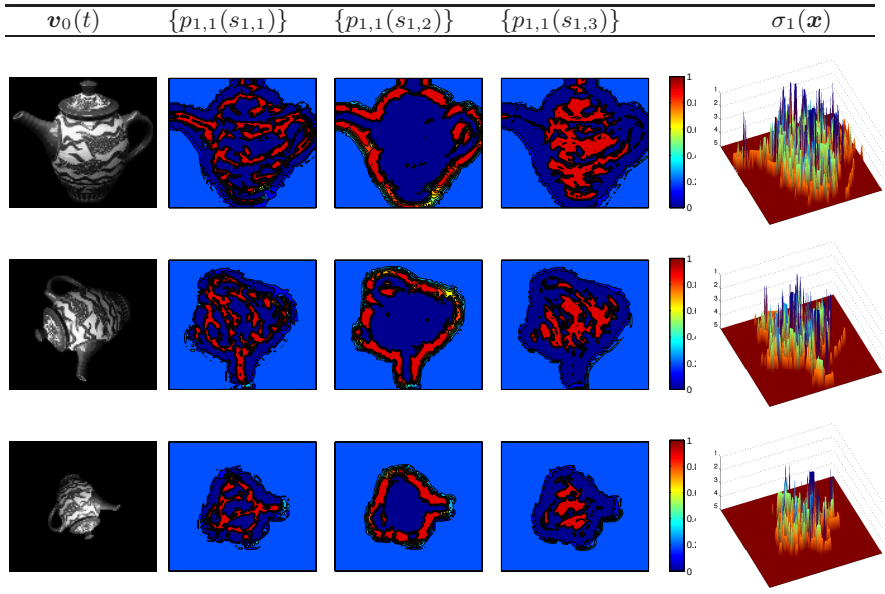


Fig. 3. Three images of the same teapot represented at different orientations/scales. DVA extracts similar features for the three pictures. The scale for each pixel is variable ($\sigma_1(\boldsymbol{x}) \in [1, 5]$) and it is reported in the rightmost graph (vertical axis). Smaller scales are used to capture the details of the internal pattern of the teapot. Some of them are lost in the bottom image due to the size reduction/subsampling.

considered a classic image classification task. Since DVAs are general purpose agents performing pixel–based feature extraction, in order to perform image classification the information extracted at the higher level is to be aggregated by a pooling stage. In particular, the average of the feature histograms at the top layer was computed and used as input for a linear Support Vector Machine (SVM) classifier. We selected the data from the Caltech 101 collection [9], composed of several real–world images grouped into 101 categories. We considered 10 random classes[2] and a variable number D of training images ($D \in \{1, 5\}$ per class) converted to grayscale and scaled to 151×143 (preserving proportions). We trained a shallow ($L = 1$) and deep ($L = 3$) DVA with one feature per layer ($C_l = 1, l = 1, 2, 3$) with 15 and $5, 10, 15$ feature values at each layer, respectively. The accuracy was compared with the performance of classic 3–layer Deep Belief Networks (DBNs) [1]. We tuned the parameters of all the models by performing 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 examples per class). Results are averaged on multiple random splits of the training data. For DBNs, during model selection we selected the number n_h of hidden units for each layer,

[2] They are: airplanes, car_side, crocodile, dollar_bill, elephant, panda, pizza, scissors, soccer_ball, umbrella.

with $n_h \in \{200, 500, 1000, 5000\}$, while early stopping on the validation set was used in order to select the best model to be used at prediction time. For DVAs, we used $\eta = 0.7$ and $\lambda = 0.001$ to weigh the contributions of the global entropy and the regularizer.

Table 1. Average accuracy (and std) on the test dataset from Caltech 101 data. A Shallow DVA is composed of 1 layer, whereas DBN and Deep DVA are composed of 3 layers.

Training Images	**10**	**50**
DBN	21.67 (2.08)	25.83 (1.61)
DVA (Shallow)	26.33 (3.88)	32.83 (5.01)
DVA (Deep)	27.50 (6.00)	33.50 (5.41)

The results in Table 1 indicate a good quality of the features learned by both DVA models in comparison with DBNs. Increasing the number of layers, the extracted features show better discrimination capabilities. Although research along this direction is still at an early stage, these results are strongly encouraging. Future work will include the use of supervised learning algorithms at the upper layers in the hierarchy, while pixel–wise feature extraction might naturally be applied to object detection tasks.

5 Conclusions

The paper presents a deep architecture for computer vision that develops pixel–based features, while incorporating crucial rotation and scale invariances. The learning algorithm is based on unsupervised criteria derived from principles of information theory. The proposed algorithm has been evaluated in a multi–layer setting and it compared favorably with classic deep architectures on a multi–class benchmark. Future work includes the expansion of several aspects of the feature extraction process, in order to allow the algorithm to naturally embed pixel–wise supervisions as well as different kind of knowledge expressed by means of constraints on the feature extractors.

Acknowledgements

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. Hinton, G.E., Salakhutdinov, R.R.: Reducing the Dimensionality of Data with Neural Networks. Science **313**(5786) (July 2006) 504–507

2. Bengio, Y.: Learning deep architectures for ai. Foundations and Trends in Machine Learning **2**(1) (2009) 1–127
3. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Scene parsing with multiscale feature learning, purity trees, and optimal covers. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, icml.cc / Omnipress (2012)
4. Gori, M., Melacci, S., Lippi, M., Maggini, M.: Information theoretic learning for pixel-based visual agents. In Fitzgibbon, A.W., Lazebnik, S., Perona, P., Sato, Y., Schmid, C., eds.: Proceedings of the European Conference on Computer Vision ECCV 2012. Volume 7577 of Lecture Notes in Computer Science., Springer (2012) 864–875
5. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11) (1998) 2278–2324
6. Serre, T., Wolf, L., Poggio, T.: Object recognition with features inspired by visual cortex. In: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Volume 2. (2005) 994–1000 vol. 2
7. Melacci, S., Gori, M.: Unsupervised learning by minimal entropy encoding. IEEE Transactions on Neural Networks and Learning Systems **23**(12) (2012) 1849–1861
8. Riedmiller, M., Braun, H.: A direct algorithm method for faster backpropagation learning: RPROP. In: Int. Conf. on Neural Networks. (1993) 586–591
9. 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) (2007) 59–70