Sparse Shift-Invariant Representation of Local 2D Patterns and Sequence Learning for Human Action Recognition

Moez Baccouche, Franck Mamalet
Orange Labs R&D
4 rue du Clos Courtel, F-35510, France
firstname.surname@orange.com

Christian Wolf, Christophe Garcia, Atilla Baskurt
Université de Lyon, CNRS
INSA-Lyon, LIRIS, UMR 5205, F-69621, France
firstname.surname@liris.cnrs.fr

Abstract

Most existing methods for action recognition mainly rely on manually engineered features which, despite their good performances, are highly problem dependent. We propose in this paper a fully automated model, which learns to classify human actions without using any prior knowledge. A convolutional sparse auto-encoder learns to extract sparse shift-invariant representations of the 2D local patterns present in each video frame. The evolution of these mid-level features is learned by a Recurrent Neural Network trained to classify each sequence. Experimental results on the KTH dataset show that the proposed approach outperforms existing models which rely on learned-features, and gives comparable results with the best related works.

1. Introduction and related work

In the last years, human action recognition has become an increasingly active research area due to its large number of applications. Among the most popular state-of-the-art methods, we can mention those proposed by Laptev et al. [7], Niebles et al. [9] and others [6, 8, 2], whose common point is the use of so-called *engineered* features, which are manually designed to be optimal for a specific task, and thereby are highly problem dependent. By opposition to this dominant methodology, there has been a growing interest in approaches that can automatically learn salient patterns for human action recognition, directly from the raw image pixels [4, 5, 12]. In our previous work [1], we have presented a neural model which can achieve excellent performance for human action recognition, when trained to extract features in a supervised manner. In this paper, we propose to investigate the other paradigm of pattern recognition, in which an unsupervised feature extraction step precedes the supervised classification.

Most unsupervised learning techniques for feature extraction rely on auto-encoders to produce a compact representation of the input content, which generally decreases the classification performances [10]. However, several recent works advocated the use of *sparse-overcomplete* representations, i.e whose dimension is larger than the input one, but where only a small number of components are non-zero. The overcompleteness ensures a high separability between classes, while the sparsity provides a simple part-based interpretation of the input content. Several sparsifying procedures have been presented in the literature, including the one introduced by Ranzato et al. in [10], in which a nonlinear *sparsifying logistic* is placed between the encoder and the decoder. Ranzato et al. proposed a learning algorithm to train the model, and a specific procedure to handle shift-invariance. The approach obtained outstanding results in object recognition [10]. In this article, we propose a solution based on a similar principle. A convolutional sparse auto-encoder is trained to automatically build a sparse representation of local 2D patterns of each video frame. We propose a novel approach for handling shift-invariance, by including an additional hidden variable to the objective function. The entire video sequence is then labelized considering the temporal evolution of these learned features, using a recurrent neural network.

The rest of the paper is organized as follows. Section 2 presents the proposed model for feature learning, and the corresponding training algorithm. We present in section 3 the architectures of the encoder and decoder used in our experiments. The recurrent neural scheme for the entire sequence labelling is then described in section 4. Experimental results on the KTH dataset are given in section 5. Finally, we conclude and give some perspectives of this work.

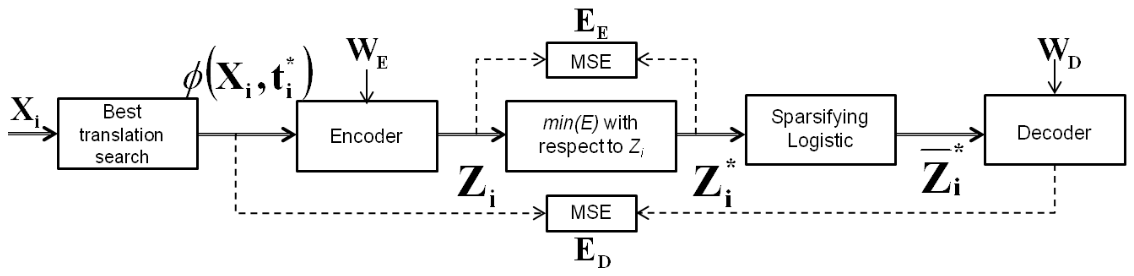

Figure 1. Overview of the proposed model for learning sparse shift-invariant features.

2. Learning sparse shift-invariant representation of local 2D patterns

The proposed model can be schematized by two main modules: an encoder (with trainable parameters W_E), which builds a code vector representing the salient information contained in the input, and a decoder (with trainable parameters W_D), which learns to reconstruct the input from a sparse version of the obtained code. Instead of operating on the complete image, the auto-encoder takes as input small patches $\{X_i\}_{i\in[1..P]}$ of size $M \times M$ each (where P is the number of training samples).

As in [10], the system learns a compact code which can reconstruct the input. A *sparsifying logistic* between the encoder and the decoder restrains the size of the code. It is a non linear function which can be seen as a SoftMax applied on consecutive training samples. Given the i^{th} training sample, and its corresponding non-sparse code $Z_i = \left\{z_i^{(k)}\right\}_{k\in[1..N]}$, where N is the code size, the sparse code $\overline{Z}_i = \left\{\overline{z}_i^{(k)}\right\}_{k\in[1..N]}$ will be expressed by: $\overline{z}_i^{(k)} = \frac{\eta e^{\beta z_i^{(k)}}}{\xi_i^{(k)}}$ with $\xi_i^{(k)} = \eta e^{\beta z_i^{(k)}} + (1-\eta)\,\xi_{i-1}^{(k)}$ where η and β are positive parameters controling the code sparsity and softness. Due to the resulting strong non-linearities, the encoder and the decoder are learned separately: when learning one of the two parts, the weights of the other part are kept constant. A third (and also separate) step produces optimal code (i.e Z_i is considered as an additional parameter). All three steps minimize a single global objective function with respect to different parameters at each step - details are given below.

In contrast to [10], in order to handle the shift-invariance of the learned representations, we propose to introduce an additional hidden variable t_i (a translation vector), on which the optimization is done. The idea is to represent the neighbourhood of a given patch X_i by a single translated patch $\phi(X_i, t_i)$, i.e the model assigns the same code to the shifted versions of a given input.

Figure 1 summarizes the different modules of our model, and the steps involved during training.

The objective function E is a sum of two terms, representing respectively the encoder prediction and the decoder reconstruction mean square errors (MSE), as expressed by the following equation:

$$\begin{aligned} &E(X_i, t_i, Z_i, W_E, W_D) \\ &= E_E(X_i, t_i, Z_i, W_E) + E_D(X_i, t_i, Z_i, W_D) \\ &= \|Z_i - Enc(W_E, \phi(X_i, t_i))\|^2 \\ &+ \left\|Dec\left(W_D, \overline{Z}_i\right) - \phi(X_i, t_i)\right\|^2 \end{aligned} \quad (1)$$

Optimal parameters $(W_E^*, W_D^*, Z_i^*, t_i^*)$ are those minimizing E. Each one is optimized separately as follows:

$$t_i^* = \underset{t_i}{\arg\min}\, E(t_i | X_i, Z_i, W_E, W_D) \quad (2)$$

$$Z_i^* = \underset{Z_i}{\arg\min}\, E(Z_i | X_i, t_i^*, W_E, W_D) \quad (3)$$

$$(W_E^*, W_D^*) = \underset{W_E, W_D}{\arg\min}\, E(W_E, W_D | X_i, t_i^*, Z_i^*) \quad (4)$$

Learning is performed for each input patch X_i in three steps: *best translation search* is performed to find t_i^* that minimizes (2) ; a steepest descent is done on the Z_i parameter to find an optimal code Z_i^* that minimizes (3) ; then W_E and W_D are updated using standard back-propagation given t_i^* and Z_i^* as expressed in (4).

The next section details the architectures of the different modules.

3. Encoder/Decoder architectures

Figure 2-(a) represents the architecture of the proposed convolutional sparse auto-encoder.

The encoder is a 2D convolutional neural network with 64 trainable 2D convolution kernels of size 8×8 each. It takes as input a small 8×8 image patch and

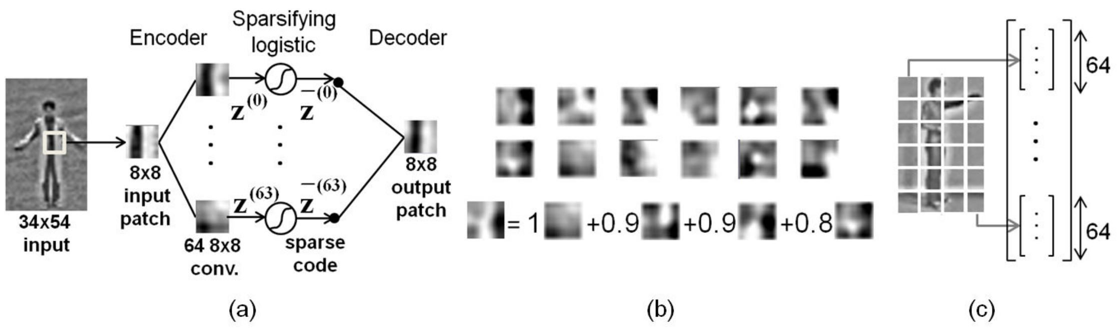

Figure 2. (a)- Architecture of the convolutional sparse auto-encoder. (b)- A subset of 12 learned basis elements. (c)- Feature vector generation from the patches responses.

learns to compute a non-sparse code of size 64 corresponding to the response of each convolution. Note that the representation is overcomplete since the code size is the same as the input dimension. The total number of trainable parameters of the encoder is 4160. t_i optimization is performed among values in $[-2, 2]$ for each component, i.e the selected patch is located in a 12×12 neighbourhood around the initial position. After the training, the output of the encoder is considered as optimal and no steepest descent is performed.

The decoder consists of a set of 64 output neurons fully connected to the sparse code $\overline{Z}$. The total number of parameters of the decoder is 4097. The output is a weighted sum of elementary patches (which will be called "basis" in the following). Since the code is sparse, only few elements of the basis are used to reconstruct the output (typically, less than 8 over the 64). We depict in Figure 2-(b) a subset of the learned basis on the KTH dataset. Note that no element of the basis is a shifted version of another one. The auto-encoder can reconstruct each input patch by combining a few elements of this basis (see the example in the last row of Figure 2-(b)). For each reconstructed input, the coefficients of the basis elements correspond to the input coordinates in the projection space. These coordinates are used hereafter as features to represent the patch content. The feature vector corresponding to the entire image is obtained by concatenating the patches responses, as illustrated in Figure 2-(c).

In the next section, we describe how these learned features are used to feed a recurrent neural network classifier, which is trained to recognize the actions based on the temporal evolution of these features.

4. Sequence classification

Among state-of-the-art learning machines, Recurrent Neural Networks (RNNs) are widely used for temporal analysis of data, because of their ability to take into account the context using recurrent connections. The *Long Short-Term Memory* model (LSTM-RNN) is a particluar recurrent architecture which was introduced by Schmidhuber et al. [3] to address some shortcomings of RNNs (especially their inability to deal with long sequences, e.g. video sequences). The LSTM-RNN learns to assign a label at each timestep of the input sequence, and the final label is obtained by averaging all individual decisions (one per timestep). The LSTM-RNN has been tested in many applications, like phoneme classification [3] and action recognition [1], and generally outperformed existing methods.

We propose to use a bidirectional LSTM-RNN model with one hidden layer to classify sequences. At each timestep, the LSTM-RNN takes as input a feature vector generated by concatenating the sparse codes of the patches, placed at different locations in the original image (see Figure 2-(c)). The LSTM-RNN is fed with sequences of feature vectors (one per timestep). Note that each feature vector has approximately the same size as the image, but only a few input connections are activated at each timestep due to the code sparsity. The LSTM cells are fully connected to these inputs and have also auto-recurrent connections. For the output layer, we used K output neurons and the SoftMax activation function, which is standard for 1 out of K classification tasks. The hidden layer contains 10 LSTM neurons for each direction. This architecture corresponds to about 6.10^4 parameters learned with standard online *backpropagation through time*.

Dataset	Features	Method	Acc.
KTH1	Learned	**Ours**	**93.70**
		Jhuang et al. [4]	91.70
	Engineered	Gao et al. [2]	96.33
		Liu et al. [8]	94.20
		Niebles et al. [9]	81.50
KTH2	Learned	**Ours**	**90.76**
		Ji et al. [5]	90.20
		Taylor et al. [12]	90.00
	Engineered	Kim et al. [6]	95.33
		Gao et al. [2]	92.45
		Laptev et al. [7]	91.80

Table 1. Obtained results on KTH dataset.

5. Experimental results

The KTH dataset [11] is the most commonly used human actions dataset. It contains 6 types of actions (walking, jogging, running, boxing, hand-waving and hand-clapping) performed by 25 subjects in 4 different scenarios. As in [2], we rename the KTH dataset in two ways: the first one (the original one) where each person performs the same action 3 or 4 times in the same video, is named KTH1 and contains 599 long sequences with several "empty" frames between action iterations. The second, named KTH2, is obtained by splitting videos in smaller ones where no empty frames are present, and contains 2391 sequences. Original videos underwent the following steps: spatial down-sampling by a factor of 2 horizontally and vertically, extracting the person-centred bounding box as in [4], and applying spatio-temporal local contrast normalization as in [12]. The model was trained as described above. For the *sparsifying logistic*, η and β were fixed respectively to 0.02 and 1.5. The LSTM-RNN input size is 1536 per time step, which corresponds to the 4×6 possible locations of the 8×8 patches in the original image of size 34×54. We performed leave-one-out cross validation as recommended by Gao et al. in [2] and reported the average accuracies in Table 1.

For each dataset, Table 1 is divided into two groups: the first group consists of the methods using automatically learned features, and the second those using manually engineered ones (i.e. specialized for the KTH dataset). Among the methods of the first group, to our knowledge, our method obtained the best results, both on KTH1 (93.70%) and KTH2 (90.76%). Our method also obtained comparable results with the approaches relying on engineered features.

To end with, after training phase, feature extraction process from a single frame takes in average $146ms$ (on a 3 GHz processor). Besides, LSTM-RNN classification of a video sequence containing 25 frames requires about $250ms$.

6. Conclusion

In this paper, we have presented a neural model for human action recognition using a fully automated feature construction process. A convolutional sparse auto-encoder is trained to build a sparse shift-invariant representation of 2D patterns present in each image of the video, and the temporal evolution of these features is used to classify the actions. Experimental results on the KTH dataset show that the proposed approach gives the best results among methods using learned features, and competitive results compared to those using engineered ones. Our ongoing work aims at extending this method to the 3D case, by the use of 3D convolutions in the auto-encoder to capture spatio-temporal saliency.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential Deep Learning for Human Action Recognition. In *HBU*, 2011.

[2] Z. Gao, M.-y. Chen, A. Hauptmann, and A. Cai. Comparing evaluation protocols on the kth dataset. In *HBU*. 2010.

[3] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 2005.

[4] H. Jhuang, T. Serre, L. Wolf, and T. Poggio. A biologically inspired system for action recognition. In *ICCV*, 2007.

[5] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *ICML*, 2010.

[6] T.-K. Kim, S.-F. Wong, and R. Cipolla. Tensor canonical correlation analysis for action classification. In *CVPR*, 2007.

[7] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[8] J. Liu and M. Shah. Learning human actions via information maximization. In *CVPR*, 2008.

[9] J. Niebles, H. Wang, and L. Fei-Fei. Unsupervised learning of human action categories using spatial temporal words. *IJCV*, 2008.

[10] M. Ranzato, F. Huang, Y. Boureau, and Y. Lecun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR*, 2007.

[11] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *ICPR*, 2004.

[12] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *ECCV*, 2010.