Action Recognition based on Hierarchical Self-Organizing Maps

Miriam Buonamente[1], Haris Dindo[1], and Magnus Johnsson[2]

[1] RoboticsLab, DICGIM, University of Palermo,
Viale delle Scienze, Ed. 6, 90128 Palermo, Italy
{miriam.buonamente,haris.dindo}@unipa.it
http://www.unipa.it
[2] Lund University Cognitive Science,
Helgonavägen 3, 221 00 Lund, Sweden
magnus@magnusjohnsson.se
http://www.magnusjohnsson.se

Abstract. We propose a hierarchical neural architecture able to recognise observed human actions. Each layer in the architecture represents increasingly complex human activity features. The first layer consists of a SOM which performs dimensionality reduction and clustering of the feature space. It represents the dynamics of the stream of posture frames in action sequences as activity trajectories over time. The second layer in the hierarchy consists of another SOM which clusters the activity trajectories of the first-layer SOM and thus it learns to represent action prototypes independent of how long the activity trajectories last. The third layer of the hierarchy consists of a neural network that learns to label action prototypes of the second-layer SOM and is independent - to certain extent - of the camera's angle and relative distance to the actor. The experiments were carried out with encouraging results with action movies taken from the INRIA 4D repository. The architecture correctly recognised 100% of the actions it was trained on, while it exhibited 53% recognition rate when presented with similar actions interpreted and performed by a different actor.

Keywords: Self-Organizing Map, Neural Network, Action Recognition, Hierarchical models, Intention Understanding

1 Introduction

Recognition of human intentions is becoming increasingly demanded due to its potential application in a variety of domains such as assisted living and ambient intelligence, video and visual surveillance, human-computer interfaces, gaming and gesture-based control. Typically, an intention recognition system is focused on a sequence of observed actions performed by the agent whose intention is being recognised. To provide the system with this component, it is necessary to use activity recognition together with the intention recognition. The purpose

of action recognition is an analysis of ongoing events from data captured by a camera in order to track movements of humans and to identify actions.

Many challenges make the action recognition task extremely difficult to imitate artificially, each person differs in terms of height, weight, shape of the human body and gender. Another important aspect to be considered is the impact of the camera viewing angle variations on the action recognition performance. Multi-camera setups have been employed to implement view independent methods [1], [2], [3]. These methods are based on the observation of the human body from different angles, obtaining in this way a view-invariant representation.

Dealing with action recognition, it is important to give a brief definition of what we mean by action. We adopt the following action hierarchy: actions and activities. The term action is used for simple motion patterns typically executed by a single human. An example of an action is crossing arms. A sequence of actions represents an activity, such as the activity dancing. Activities usually involve coordination among persons, objects and environments. In this paper, we focus only on the recognition of actions, where actions can be viewed as sequences of body postures.

An important question is how to implement the action recognition ability in an artificial agent. We tried to find a suitable neural network architecture having this ability. In our previous work, we have focused on the representational part of the problem. We endowed an artificial agent with the ability to internally represent action patterns [4]. Our system was based on the Associative Self-Organizing Map [5], a variant of the Self-Organizing Map (SOM) [6], which learns to associate its activity with additional inputs. The solution was able to parsimoniously represent human actions.

In this paper, we present a novel architecture able to represent and classify others' behaviour. In order to get a more complete classification system we adopt a hierarchical neural approach. The first level in the system is a SOM that learns to represent postures - or posture changes - depending on the input to the system. The second level is another SOM that to represent the superimposed activity trace in the first level SOM during the action, i.e. it learns to represent actions. Thus, the second layer SOM provides a kind of time independent representation of the action prototypes. The third level is a supervised artificial neural network that learns to label the action.
In our previous paper [7] we showed that we could get discriminable activity traces using an A-SOM, which corresponds to the first level SOM in the current system.The system was able to simulate the likely continuation of the recognised action. Due to this ability, the A-SOM could receive an *incomplete* input pattern (e.g. an initial part of the input sequence only) and continue to elicit the most likely evolution of the action, i.e. to carry out sequence completion of perceptual activity over time. In the present system, instead, we focus on the probem of robust action representation and recogniton, given the whole (noisy) input sequence. We are currently working towards an integration of the two approaches.

We have tested the ability of our architecture to recognise observed actions on movies taken from the "INRIA 4D repository [3]", a publicly available dataset of movies representing 13 common actions: check watch, cross arms, scratch head, sit down, get up, turn around, walk, wave, punch, kick, point, pick up, and throw (see Fig. 1).

The implementation of all code for the experiments presented in this paper was done in C++ using the neural modelling framework "Ikaros" [8].

This paper is organized as follows: A short presentation of the proposed architecture is given in section II; section III presents the experiment for evaluating the model; and finally conclusions are outlined in section IV.

2 Proposed Architecture

The architecture presented in this paper is composed of three layers of neural networks, see Fig. 3. The first and the second layers consist of SOM networks whereas the third layer consists of a custom made supervised neural network. The first layer SOM receives sequences of vectors representing preprocessed sequences of posture images. The activity trajectories, Fig. 2, elicited during the time actions last are superimposed and vectorized into a new representations before entering the layer two SOM as input. This superimposition process can be imagined as the projection of the matrices representing the activity in the grid of neurons in the SOM for all the iterations an action lasts onto a new matrix of the same dimensionality, followed by a vectorization process. The second layer SOM thus clusters the activity trajectories and learns to represent action prototypes independent of how long the activity trajectories in the first layer SOM last. Thus the second layer SOM provides a kind of time independent representation of the action prototypes. The activity of the second layer SOM is conveyed to a third level neural network that learns to label the action prototypes of the second layer SOM independent of the camera's capturing angle and distance to the actor.

2.1 First and Second Layers

The first and the second layers of the architecture consist of SOMs. The SOM is one of the most popular neural networks and has been successfully applied in pattern recognition and image analysis. The SOM is trained using unsupervised learning to produce a smaller discretized representation of its input space. In a sense it resembles the functioning of the brain in pattern recognition tasks. When presented with input, it excites neurons in a specific area. The goal of learning in the SOM is to cause nearby parts of the network to respond to similar input patterns while clustering a high-dimensional input space to a lower-dimensional

[3] The repository is available at http://4drepository.inrialpes.fr. It offers several movies representing sequences of actions. Each video is captured from 5 different cameras. For the experiments in this paper we chose the movie "Andreas2" for training and "Hedlena2" for testing, both with frontal camera view "cam0".

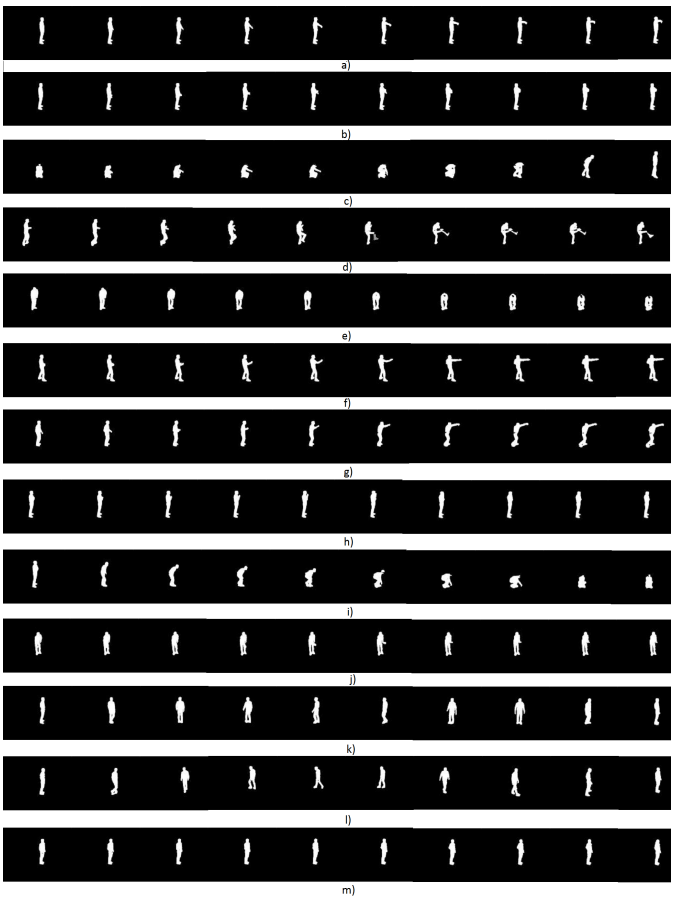

Fig. 1. Prototypical postures of 13 different actions in our dataset: check watch, cross arms, get up, kick, pick up, point, punch, scratch head, sit down, throw, turn around, walk, wave hand.

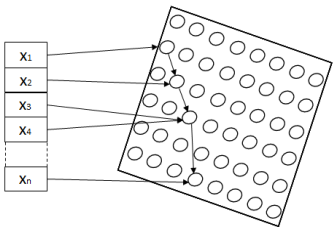

Fig. 2. The trajectory resulting from the neurons activated by the input sequence.

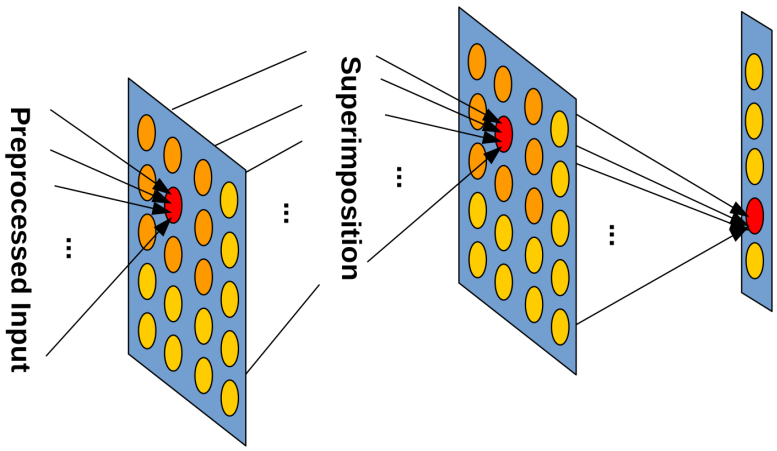

Fig. 3. The proposed architecture is composed of three layers of neural networks. The first and the second layers consist of SOM networks whereas the third layer consists of a custom made supervised neural network.

output space. SOMs are different from many other artificial neural networks because they use a neighbourhood function to preserve the topological properties of the input space. The SOM algorithms adapt a grid of neurons, so that neurons located close to each other respond to similar features.

The SOM structure is made of one input layer and one output layer, the latter also known as the Kohonen layer. The input layer is fully connected to the neurons in the Kohonen layer. The weight vectors of the neurons in the Kohonen layer are modified iteratively in the training phase. When a new input arrives, every neuron competes to represent it. The Best Matching Unit (BMU) is the neuron that wins the competition. The BMU together with its neighbours in the grid are allowed to adapt to the input. The neighbouring neurons less so than the BMU. Neighbouring neurons will gradually specialise to represent similar inputs, and the representations will become ordered in the map. Another important characteristic of the SOM is its ability to generalise, i.e. the network can recognise or characterise input it has never encountered before.

The SOM consists of a grid of neurons with a fixed number of neurons and a fixed topology. Each neuron n_i is associated with a weight vector w_i. All the elements of all the weight vectors are initialized by real numbers randomly selected from a uniform distribution between 0 and 1, after which all the weight vectors are normalized, i.e. turned into unit vectors.

At time t each neuron n_i receives an input vector $x(t)$.

The BMU n_b at time t is the neuron with the weight vector w_b that is most similar to the input $x(t)$ and is obtained by:

$$b = \text{argmax}_i \frac{x(t) \cdot w_i(t)}{||x(t)||||w_i(t)||}, \tag{1}$$

The neurons of the Kohonen layer adapt to increase their representation of the current input by modifying their weight vectors to become more similar to it with an amount that depends on a Gaussian function of the neuron's distance to the BMU:

$$\Delta w_i = \gamma(t) G_{ib}(t)(x(t) - w_i(t)) \tag{2}$$

where the learning rate $\gamma(t)$ is a monotonically decreasing function of time. $G_{ib}(t)$ is a Gaussian function, with a radius $\sigma(t)$ monotonically decreasing with time, of the distance in the map between the neuron n_i and the BMU:

$$G_{ib}(t) = \exp \frac{-d(i,b)^2}{\sigma(t)^2} \tag{3}$$

2.2 Third Layer

The third layer, which is the output layer of the architecture, consists of an array of a fixed number of neurons. Each neuron n_i is associated with a weight vector $w_i \in R^n$, where n is equal to the number of neurons in the second layer SOM. All the elements of the weight vector are initialized by real numbers randomly

selected from a uniform distribution between 0 and 1, after which the weight vector is normalized.

At time t each neuron n_i receives an input vector $x(t) \in R^n$, which is the vectorized activity of the second layer SOM.

The activity y_i in the neuron n_i is calculated using the standard cosine metric:

$$y_i = \frac{x(t) \cdot w_i(t)}{||x(t)||||w_i||} \tag{4}$$

During the learning phase the weights w_{ij} are adapted by

$$w_{ij}(t+1) = w_{ij}(t) + \beta x_j(t)[y_i - d_i] \tag{5}$$

where β is the adaptation strength and d_i is the desired activity for the neuron n_i.

3 Experiment

We have tested our architecture (Fig. 3) in an experiment to verify that it is capable of recognising and properly classifying observed actions, overcoming problems related with the action recognition task.

To this aim we created training and test sets for the architecture by choosing two movies from the INRIA 4D repository. In the movies, two different actors (Andreas and Hedlena) perform the same set of 13 actions. Each actor interprets and performs actions as individuals and thus they tend to differ slightly in how they perform the same actions. We chose to use one of the movies (performed by Andreas) to create a training set for the architecture and the other movie (performed by Hedlena) to create a test set. In this way, we wanted to demonstrate that our architecture is able not only to properly recognize action instances it has observed during training, but that it is also able to recognise the actions when they are performed by someone else, i.e. to recognise action instances it never encountered before. To create the training and test sets from the original movies, we split each of the original movie into 13 new movies, one for each action (see Fig. 1).

Before entering the architecture, the input goes through a preprocessing phase. This is done to reduce the computational load and improve architecture performances. In the preprocessing phase the number of images for each movie is reduced to 10 without affecting the quality of the action reproduction and guaranteeing seamless and fluid actions, see Fig. 4 a).

Consecutive images are then subtracted to catch only the dynamics of the action, focusing in this way the attention on the movement exclusively. This operation further reduced the number of frames for each movie to 9. As an example, we can see in Fig. 4 that in the *check watch* action only the arm is involved in the movement.

In the next step of the preprocessing phase, a fixed boundary box is used to cut the images and produce binary images of a fixed and small size while

Fig. 4. a) The *check watch* action with a reduced number of images; b) The sequence of images obtained by subtracting consecutive images of the *check watch* action.

eliminating anything not significantly involved in the movement. In this way an attentive process, similar to how the human eye observes and follows only the salient parts of an action, is simulated. The binary images are then shrunk to 50×50 matrices and vectorized before entering the first layer SOM.

The architecture was trained in two phases. First the first layer SOM was trained for 20000 iterations by randomly selecting actions performed by the actor Andreas. Then the fully trained SOM of the first layer received each action performed by Andreas again and the corresponding sequence of activity matrices elicited by each action was superimposed and vectorized. Each such new superimposed activity vector represents the activity trajectory in the first layer SOM elicited by a particular action. More in detail, to superimpose the activity matrices, before the vectorization, can be seen as the creation of matrices, one for each action, with dimensions equal to the neuron grid of the first layer SOM. The value of the elements of these matrices are either zero or one. All elements corresponding to a neuron in the first layer SOM, which was most activated for at least one of the inputs during the action is set to one and all the other elements are set to zero.

In the second training phase the second layer SOM and the third layer neural network were trained. In this process the second layer SOM received randomly selected input from the set of superimposed activity vectors for 20000 iterations, and the third layer neural network received the corresponding target output (action labels). The target output consists of 13-dimensional vectors, with one element set to one and the other elements set to zero.

To show how the activity trajectories in the first layer SOM in the fully trained architecture differ we have depicted these for the actions carried out by the actor Andreas in Fig. 5. This was, for each action, done by recording the neuron in the first layer SOM most activated by each input in the sequence composing the action. The most activated neurons for each of the actions were then depicted and connected with arrows to show how the trajectories evolves over time. Each picture in Fig. 5 shows the grid of neurons forming the first layer SOM and illustrates the sequence of most activated neurons, represented by black dots, during the corresponding action. The black dots were connected with arrows to show how the trajectories evolve over time. The locations of the neurons activated most by the first and the last inputs of an action are represented by empty dots.

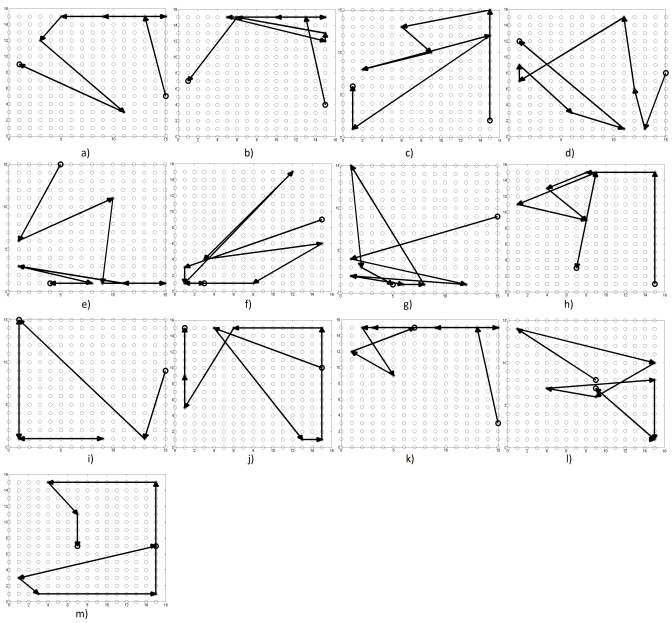

Fig. 5. Activity trajectories in the first layer SOM for the 13 actions carried out by Andreas: a) Check Watch; b) Cross Arms; c) Scratch Head; d) Sit Down; e) Get Up; f) Turn Around; g) Walk; h) Wave Hand; i) Punch; j) Kick; k) Point; l) Pick Up; m) Throw. The dots in each diagram represent the most activated neurons (centres of activity) during the action and the arrows indicate the action's evolution over time. Actions composed of similar postures present fewer centres of activity, whereas actions composed of postures with more different characteristics present more centres of activity. The diagrams indicate the ability of the SOM to create topology preserving maps in which similar postures are represented close to each other.

We tested the fully trained architecture with all 13 actions performed both by the actor Andreas (the action instances the architecture was trained on) and by the actor Hedlena (the action instances the architecture was not trained on). During testing, the input went through the preprocessing described above before entering the first layer SOM, which activity in turn were superimposed and vectorized as described above before entering the second layer SOM.

During the testing we recorded the most activated neuron in the third layer neural network to see if the actions were labelled correctly. This was done for both the actions carried out by Andreas as reported in Table 1 and by Hedlena as reported in Table 2. The architecture was able to recognise 100% of the actions performed by Andreas and 53% of the actions performed by the actor Hedlena, which the architecture was not trained on.

Andreas			
Actions	Most activated neuron	Expected neuron	Correctenss
Check Watch	0	0	correct
Cross Arm	1	1	correct
Scracth Head	2	2	correct
Sit Down	3	3	correct
Get Up	4	4	correct
Turn Around	5	5	correct
Walk	6	6	correct
Wave Hand	7	7	correct
Punch	8	8	correct
Kick	9	9	correct
Point	10	10	correct
Pick Up	11	11	correct
Throw	12	12	correct
%Correctness			100

Table 1. Recognition rate for the actions carried out by Andreas. Our architecture recognises 100% of the actions performed by Andreas, which the architecture was trained on.

4 Conclusion

We have proposed a novel hierarchical SOM based architecture that recognises actions. Our architecture is composed of three layers of neural networks. The first layer consists of a SOM that learns to represent the dynamics of sequences of postures composing actions. The second layer consists of another SOM, which learns to represent the activity trajectories in the first layer SOM, which also means that it learns to represent action prototypes. The third layer consists of a custom made supervised neural network that learns to label the action prototypes represented in the second layer SOM.

In an experiment we verified the architecture's ability to recognise observed actions as well as to recognise the same actions interpreted and performed by someone else.

As reported in Table 1 the actions used to train the architecture and performed by the actor Andreas were recognised to 100%. In Table 2 we can see

Hedlena				
Actions	Most activated neuron	Expected neuron	Place in order of activation of the expected neuron	Correctenss
Check Watch	12	0	4	
Cross Arm	1	1	1	correct
Scracth Head	2	2	1	correct
Sit Down	3	3	1	correct
Get Up	5	4	2	
Turn Around	5	5	1	correct
Walk	6	6	1	correct
Wave Hand	2	7	5	
Punch	8	8	1	correct
Kick	3	9	5	
Point	10	10	1	correct
Pick Up	2	11	11	
Throw	4	12	2	
%Correctness				53

Table 2. Recognition rate for the actions carried out by Hedlena. Our architecture recognises 53% of the actions performed by Hedlena, which the architecture was not trained on. In the cases of failed recognition the place in the order of activation of the expected neuron could be seen as the order of choice, i.e. if the place in the order of activation of the expected neuron is k, then the correct action would be the k:th most likely action according to the architecture.

that the actions interpreted and performed by another actor Hedlena, that the architecture was not trained on, were recognised to 53%. The values reported in the fourth column of Table 2 show that in some of the cases where recognition failed, the expected neuron, i.e. the neuron which if most activated would indicate the correct action, is still one of the most activated. For example, in the case of the action *Get Up*, which was incorrectly recognised as the action *Turn Around*, the architecture's second choice would have been the correct action *Get Up*.

An important observation is that some failed recognitions are plausible. Actions like *check watch*, *throw*, *wave hand* and *scratch head* can easily be confused even by a human observer. Consider, for example, the two actions *wave hand* and *scratch head*. The only part involved in the movement is the arm and the movement for both actions is the same, i.e. to raise the arm to the head. This could easily confuse the architecture to label both actions equally. The same reasoning can be applied to other actions that involve the movement of the same part of the body. Other considerations can be done for actions that involve movement of different parts of the body such as *kick* and *sit down*. In this case, the preprocessing operation such as subtraction of consecutive frames, gives rise to new sequences that sometimes can contain very similar frames, or frames that can be confused with each other, leading to a failed recognition of the observed action.

The promising experimental results show the potential of this hierarchical SOM based action recognition architecture. Potential future extensions include a more elaborate preprocessing procedure to enable a more potent view and size independence as well a explicit action segmentation.

Acknowledgements The authors gratefully acknowledge the support from the European Community's Seventh Framework Programme under grant agreement no: 612139.

References

1. Ahmad, M., Lee, S.W.: Human action recognition using shape and clg-motion flow from multi-view image sequences. Pattern Recognition **41**(7) (2008) 2237 – 2252
2. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. Computer Vision and Image Understanding **104**(23) (2006) 249 – 257 Special Issue on Modeling People: Vision-based understanding of a persons shape, appearance, movement and behaviour.
3. Ahmad, M., Lee, S.W.: Hmm-based human action recognition using multiview image sequences. In: Pattern Recognition, 2006. ICPR 2006. 18th International Conference on. Volume 1. (2006) 263–266
4. Buonamente, M., Dindo, H., Johnsson, M.: Recognizing actions with the associative self-organizing map. In: 2013 XXIV International Symposium on Information, Communication and Automation Technologies (ICAT), IEEE (2013) 1–5
5. Johnsson, M., Balkenius, C., Hesslow, G.: Associative self-organizing map. In: Proceedings of IJCCI. (2009) 363–370
6. Kohonen, T.: Self-Organization and Associative Memory. Springer Verlag (1988)
7. Buonamente, M., Dindo, H., Johnsson, M.: Simulating actions with the associative self-organizing map. In Lieto, A., Cruciani, M., eds.: AIC@AI*IA. Volume 1100 of CEUR Workshop Proceedings., CEUR-WS.org (2013) 13–24
8. Balkenius, C., Morén, J., Johansson, B., Johnsson, M.: Ikaros: Building cognitive models for robots. Advanced Engineering Informatics **24**(1) (2010) 40–48