Recognition of Human Actions Through Deep Neural Networks for Multimedia Systems Interaction

Marco La Cascia
Dipartimento di Ingegneria
University of Palermo
Palermo, Italy
Email: marco.lacascia@unipa.it

Ignazio Infantino, Filippo Vella
Istituto di calcolo e reti ad alte prestazioni
CNR - National Research Council of Italy
Palermo, Italy
Email: name.surname@icar.cnr.it

***Abstract*—Nowadays, interactive multimedia systems are part of everyday life. The most common way to interact and control these devices is through remote controls or some sort of touch panel. In recent years, due to the introduction of reliable low-cost Kinect-like sensing technology, more and more attention has been dedicated to touchless interfaces. A Kinect-like devices can be positioned on top of a multimedia system, detect a person in front of the system and process skeletal data, optionally with RGBd data, to determine user gestures. The gestures of the person can then be used to control, for example, a media device. Even though there is a lot of interest in this area, currently, no consumer system is using this type of interaction probably due to the inherent difficulties in processing raw data coming from Kinect cameras to detect the user intentions. In this work, we considered the use of neural networks using as input only the Kinect skeletal data for the task of user intention classification. We compared different deep networks and analyzed their outputs.**

Keywords–Multimedia system interaction; gesture recognition; neural networks.

I. INTRODUCTION

In 2010, when Microsoft introduced Kinect, this new device was intended to change the way people play games and how they experience entertainment. However, the potential of the device was immediately clear and applications in different fields leveraging the sensing technology of Kinect have been explored. This is what Microsoft called the ”Kinect Effect” [1]. From the early days of introduction to current days several scientific papers reported possible applications in disparate fields. For example in 2011, in [2], a study on the potential of Kinect in education was published. The author states that the use of Kinect can create unprecedented opportunity to enhance classroom interaction, to improve teachers ability to present and manipulate multimedia and multimodal materials and much more.

Interactive media control using gesture was also immediately perceived as a viable application. In [3], the authors propose probably the first system using a Kinect to interact with multimedia content. They defined gestures to activate controls on a media device and used the depth image to detect and track the hand and gestures. Gestures were defined in terms of distance variances along the 3D axes.

Multimedia presentation systems using a gesture based interface could also be used as information provision systems. In [4], the authors developed a platform to develop interactive systems based on depth image streams and demonstrated its potential in a museum application. The importance of touchless gestural systems has been deeply stressed also in [5] where the authors report a case study on an information provision system in a University campus using Kinect-like devices.

In addition to media and entertainment, these sensors can also be used in very specific professional fields. For example, another interesting use of gestural interaction has been recently introduced in [6] where the authors reported the use of Kinect and gesture recognition to give interactive presentations in a more effective manner compared to a traditional pointer, mouse or keyboard. In other cases, touchless interaction is mandatory as the user cannot touch any device for different reasons. For example in [7], gesture recognition and Kinect have been used for touchless visualization of hepatic anatomical models in surgery.

However, while it is rather easy to collect a significant amount of sensors observations, the great challenge is to properly recognize meaningful patterns in the raw data that can be ascribed to user actions and intentions [8] [9]. Only in simple cases skeletal data or RGBd is sufficient to detect actions with little processing. In many cases quite complex processing is needed to detect user intentions. Machine learning approaches are then exploited to process gestural data. These approaches rely on definition, extraction and analysis of the features most useful to detect the human intention. For example in [10], a simple gesture recognition system based on Kinect skeletal data is proposed. Based on joints information a low-dimensional feature is defined and used for action classification with a support vector machine. The authors claim they can discriminate between 10 different basic actions using 3 seconds sequences at 30fps.

The variable time duration of a gesture performed by different users can also pose significant difficulties. In [11] to cope with different duration of the actions without explicitly use dynamic time warping techniques the authors proposed to model action as the output of a sequence of atomic Linear Time-Invariant (LTI) systems. The sequence of LTI systems generating the action is modeled as a Markov chain, where a Hidden Markov Model (HMM) is used to model the transition from one atomic LTI system to another. LTI systems are modeled in terms of Hankel matrices.

To cope with this increasing complexity and to discriminate between several actions across different users The use of neural networks has recently been explored. For example in [7] the authors use a deep convolutional neural network to recognize various hand gesture. In particular they used a deep network architecture consisting of two convolutional layer, each followed by pooling, and three fully connected layers. The input of the network is a 32x32 segmented and filtered depth image coming from Kinect. The final output is the gesture detected.

Among the existing gestures recognition modules it is also possible to mention the works in [12]-[13]. Some of them suggest heuristics of processing according to specific situations [14] or for identification of the viewed person [15]. Many other techniques for skeleton based action recognition have been proposed in recent years and their description is out of the scope of this paper. The interested reader can refer to [16] for a complete survey.

In this paper we adopted neural networks to process sensed 3D position of human skeleton joints to recognize some gestures that a person can perform to interact with a multimedia system. The neural network can be trained offline and, once trained, it processes only the 3D position of joints leading to very fast processing. The joints position were extracted with a Microsoft Kinect v2. RGBd image stream was not processed.

The paper is organized as follows: in Section II, we briefly introduce two popular architectures of neural networks and their advantages with respect to conventional machine learning approaches. In Section III, we describe the proposed action recognition approach and the pre-processing operations we performed on the sensed data. Some experimental results, comparing our approach with different network architectures and pre-processing strategies are reported in Section IV. Finally, Section V contains some conclusions and a discussion on future directions of the work.

II. NEURAL NETWORKS FOR TOUCHLESS MULTIMEDIA SYSTEMS INTERACTION

The conventional machine learning techniques have shown a set of drawbacks in the processing of raw data from Kinect-like sensors. The pattern recognition approaches, typically, require careful engineering and domain expertise to design feature extraction transforming data in discriminant and significant feature vectors [17].

On the other hand, deep-learning methods usually employ a set of non-linear modules that automatically extract a set of features from the input data and transfer them to the next module [17]. The weights of the layers involved in data processing are learned directly from data, enabling the discovery of intricate structures in high-dimensional data, regardless of their domain (science, business, etc.). Assuming that an adequate amount of training data is available, very complex functions can be learned combining these modules: the resulting networks are often very sensitive to minute details and insensitive to large irrelevant variations.

A. MLP Multilayer Perceptron

A Multi-Layer Perceptron (MLP) is a feedforward network that maps sets of input data onto a set of appropriate outputs; it consists of at least three layers - an input layer, a hidden layer, and an output layer - of fully connected nodes in a directed graph. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function - usually a sigmoid, or the hyperbolic tangent, chosen to model the bioelectrical behaviour of biological neurons in a natural brain. Learning occurs through backpropagation algorithm that modifies connections weights to minimize the difference between the actual network output and the expected result on training data.

B. LSTM

Long Short Term Memory networks (LSTM) have been designed by Hochreiter and Schmidhuber [18]. The key feature of LSTMs is the "cell state" that is propagated from a cell to another. State modifications are regulated by three structures called gates, composed out of a sigmoid neural net layer and a pointwise multiplication operation.

The first gate, called "forget gate layer", considers both the input x_t and the output from the previous step h_{t-1}, and returns values between 0 and 1, describing how much of each component of the old cell state C_{t-1} should be left unaltered: if the output is 0, no modification is made; if the output is one, the component is completely replaced.

New information to be stored in the state is processed afterward. The second sigmoid layer, called the input gate layer, decides which values will be updated. Next, a $tanh$ layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_c)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

To perform a state update, C_{t-1} is first multiplied by the output of the forget gate f_t, and the result is added to the pointwise multiplication of the input gate output i_t and $\tilde{C}_t$.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Finally, the output h_t can be generated. First, a sigmoid is applied, taking into account both h_{t-1} and x_t; its output is then multiplied by a constrained version of C_t, so that we only send to output the parts we are interested in.

III. ACTION DETECTION THROUGH CLASSIFICATION

To interact with a multimedia system, we assumed that the user performs an action and our system detect and classify the action that is mapped to some functions of the system. The set of actions that we want to detect and classify are the following *Hello with the right hand*, *Hello with the left hand*, *Stop with the right hand*, *Stop with the left hand*, *Come Here with the left hand*, *Come Here with the right hand*, *Pass right hand*, *Pass left hand*. At runtime, Kinect continuously acquire data and the classification system should detect actions triggering the corresponding function on the multimedia system The data acquired from Kinect consists of an RGBd dense image stream and a sequence of joint 3D positions. In our approach, we do not use RGBd stream. Nevertheless, we have to pre-process in some way, both spatially and temporally, the sequence of joint positions to let the neural networks coherently process the data.

Kinect estimates the 3D position of 25 joints however not all of them are necessary to recognize simple actions and unnecessary joints information can introduce noise in the system. The joints that have been selected as significant for the problem at hand are the following:

- Left Hand
- Left Wrist
- Left Elbow
- Left Shoulder
- Right Shoulder
- Right Elbow
- Right Wrist
- Right Hand
- Lower Spine
- Middle Spine
- Neck
- Head

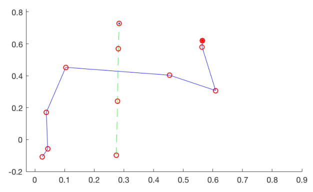
Two examples of the acquisition of depth image and skeleton points are shown in Figure 1. On the left, there is the segmented silhouette while on the right are plot the corresponding points. Kinect cameras can acquire data up to 30 fps but, since we consider that human actions useful to control a multimedia system are quite slow and we don't need a very dense classification of the actions being performed, we down-sampled the data to 2 Hz. Experiments confirmed that this frequency is adequate to capture human actions. We need also to define the duration of the time window to use to capture information and associate the label. In fact, to increase robustness and reduce false positive we consider an action as a sequence of joint positions and not as a single instant snapshot. The simplest, and in many cases adequate, approach consists of defining a fixed time window that contains the development of a typical action. This size is a matter of experience and it depends on the data and on the task. In our case, we considered multiple duration of the time window containing the action and run several experiments on our dataset to understand how to make a reasonable choice. The data in the time window is then used to analyze the occurring action.

To cope with the temporal nature of the problem a sliding window approach has been adopted. As time goes on a new sample is added to the sequence of samples to be classified and the oldest is discarded.

To implement the neural network architectures and perform the tests, we used Keras library [19]. Keras is a high-level Python neural networks library, capable of running on top of two of the most important libraries for numerical computation used for deep learning: TensorFlow [20] and Theano [21]. The use of higher level libraries, like Keras, allows developers and data scientists to rapidly produce and test prototypes, while relaying most implementation details to the chosen lower level library.

Details on the neural networks implementation and testing are given in the next section.

(a) Depth Capture for Hello


(b) Skeleton for Hello

(c) Depth Capture for Stop

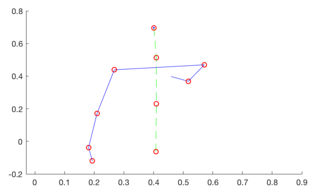
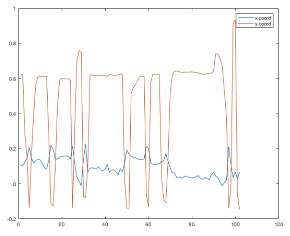
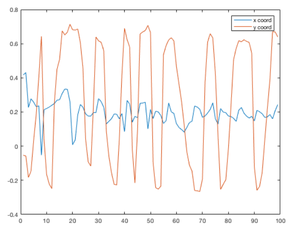

(d) Skeleton for Stop

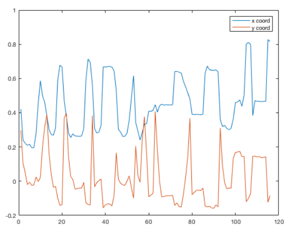
Figure 1. Example of acquisition of poses with depth image and the extracted skeleton position



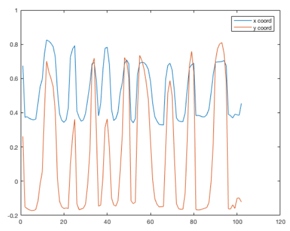
(a) Sample 1 (hello)



(b) Sample 2 (hello)



(c) Sample 1 (wide)



(d) Sample 2 (wide)

Figure 2. Plot of x,y coordinated of Left Hand while repeating the action *Hello*

IV. EXPERIMENTAL RESULTS

The dataset we used to demonstrate our approach is composed by a set of action samples. Each action sample is a matrix representing a single gesture made by a person. These samples were generated from CSV files created with Kinect for Windows SDK 2.0, containing 3D coordinates of the selected skeletal joint listed in previous section (a total of $12 \cdot 3 = 36$ columns).

The number of rows depends on the time interval used for recording. The CSV files were later imported in a Python script to down sample to 2 Hz and segment time window of different duration containing the actions. We also normalized the values in [-1, 1], because we are more interested in the differences between the frames and not the absolute value of the joints position. Some examples of the recordings are shown in Figure 2. The left plot is referred to a person, while on the right the plot is referred to a second person.

The experiments were conducted using different learning architecture. Different network topologies have been used during the training phase to evaluate whether the training process is stable with respect to different training settings. The dataset has been divided in two parts. Four fifths have been used for the training set while the remaining one fifth has been used as test set.

The results reported have been obtained restarting the training multiple times and varying the value of the batch size and averaging the obtained results. Restarting the training is equivalent to use multiple nets and compare their results after the training. Varying the batch size the number of samples that are processed before upgrading the weights of the net is changed. A larger value of the batch size can reduce the noise in the gradient descendent algorithm, that is the base of the backpropagation algorithm and converges quickly towards the minimum. On the other hand, a lower value of the batch size tends to generate a larger noise in the gradient descent algorithm and can help to escape from local minima.

The classification performance is evaluated comparing the label of the sample in the ground truth and the label chosen by the neural network. The value of the True Positive (TP) counts the number of samples that have been correctly classified. False Positive (FP) is the number of times a wrong label has been assigned to a sample. False Negative (FN) is the number of samples that have not been correctly classified. The values of True Negative (TN) is referred to the wrong labels that have not been assigned to a sample. For these experiments it has always been set to zero. The accuracy is then defined as:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision and recall are instead defined as:

$$Prec = \frac{TP}{TP + FP} \qquad Rec = \frac{TP}{TP + FN}$$

The harmonic mean of precision and recall is called F_1-score:

$$F_1 = 2 \times \frac{Prec \times Rec}{Prec + Rec}$$

The F1 metric can be calculated with different modalities. A modality is “micro” calculate the metric globally by counting the total true positives, false negatives and false positives. A “macro” modality calculates the metric for each label, and find their unweighted mean. This does not take label imbalance into account. The last is “weighted”: the metric is calculated for each label, and find their average, weighted by support (the number of true instances for each label). This last modality takes into account the label imbalance; it can result in an F-score that is not between precision and recall.

Once defined the performance metrics, we ran several experiments to understand the behaviour of two neural network architecture, MLP and LSTM, on the problem at hand. The first experiment involved an MLP network. Figure 3 shows the values of precision, recall, accuracy and F1 parameters varying the number of hidden layers in the network. The performance does not have an increasing tendency with the number of layers. An increased number of layers does not imply a better performance. It seems that a reduced number of layers provides a better performance. A reason for this trend could be related to the relatively small number of examples used for the training. The values of accuracy, precision and recall are quite similar for two, three or four layers. The precision drops for four layers. Furthermore, the weighted F1 shows a better value for two layers. Changing the duration of the time window used to represent each action did not affected significantly the performance of the network.

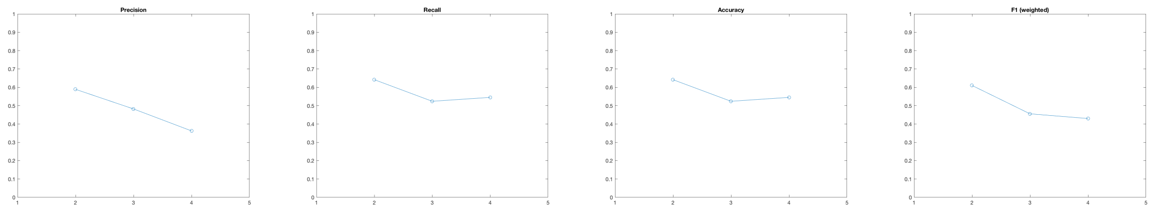
A comparison between the MLP and a network with Long Short Term Memory has been carried on. A first LSTM network with two layers has been created with a layer of one hundred units and a full connected layer with nine out units. A second net with three layers has been tested adding an intermediate LSTM layer with forty units. A further network with four layers, formed adding a layer with seventy units between the first and the second layers has also been tested. The three networks correspond in the horizontal axis to the values 2, 3 and 4.

In the case of LSTM networks, we noticed a significant variation of performance depending on the duration of the time window used to represent the actions. Figure 4 shows the performance for network trained with samples having a time span equal to 2.5 sec. The values are very high both for precision and recall. The averaged value of F1 confirms this trend. Increasing the time span of the samples to be classified performance degraded abruptly. The result for samples with a time span of 5 secs are shown in Figure 5. In this case the results are clearly worse than in the previous case. The best F1 measure is 0.49 when the number of layers is equal to 2. An increased number of layers does not help the performance and in some case, both precision and recall are very low. Considering a larger time span (7.5 sec) results are slightly better (see Figure 6). A higher number of layers helps to increase the performance although the obtained results are still worse than the case when a shorter time span to represent actions is used.

According to our experiments, the best solution consists of considering a time span of 2.5 sec and an LSTM network. The LSTM network showed the best performance also with a not too deep network obtaining a F1 score of 0.904. Although an increased number of layers provides a better result probably the architecture with only two layers is already adequate for home and consumer practical applications of gesture controlled multimedia systems.

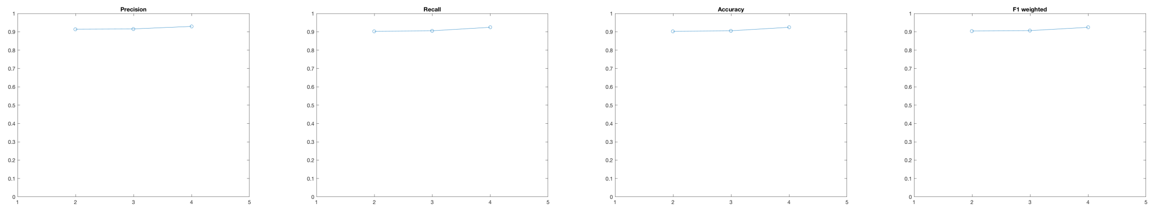
V. CONCLUSIONS

A pre-processing scheme and a few deep neural architectures have been tested for the detection of a set of simple actions to be used in multimedia systems control and interaction. Based on the experiments on an internal dataset both deep neural networks performed well. Most of the samples obtained from the Kinect camera were correctly classified. However, the experiments showed that the LSTM network, even with a very small number of layers, performs better than the considered MLP network. Moreover, the results in terms of accuracy suggest that this simple approach can be usefully



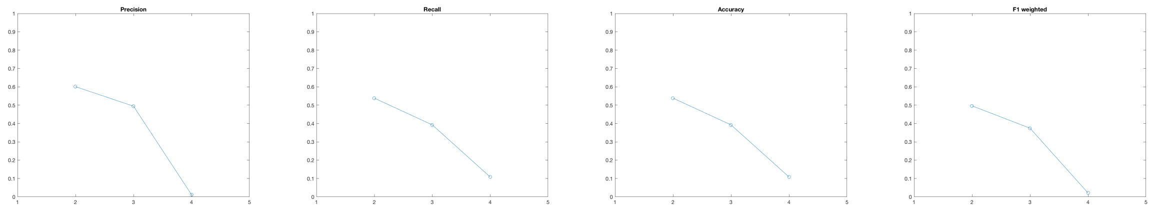
(a) Precision (b) Recall (c) Accuracy (d) F_1 weighted

Figure 3. Comparison of the performances of MLP net vs the number of network layers



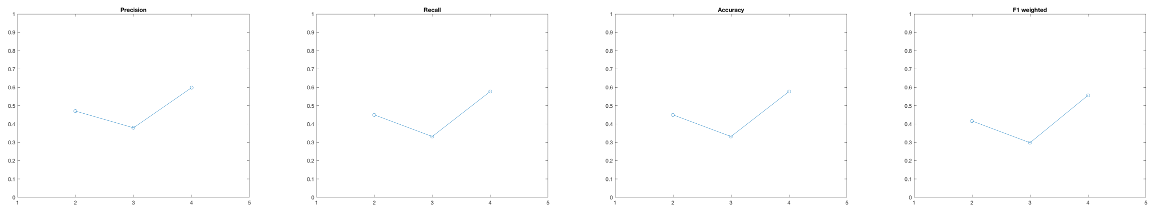
(a) Precision (b) Recall (c) Accuracy (d) F_1 weighted

Figure 4. Comparison of the performances vs number layers of LSTM network with time frame 2.5 sec



(a) Precision (b) Recall (c) Accuracy (d) F_1 weighted

Figure 5. Comparison of the performances vs number layers of LSTM network with time frame 5.0 sec



(a) Precision (b) Recall (c) Accuracy (d) F_1 weighted

Figure 6. Comparison of the performances vs number layers of LSTM network with time frame 7.5 sec

adopted to interpret user intention and control a multimedia system.

In the future, we plan to extend our experiments to more challenging datasets and compare the results obtained processing Kinect sensed skeleton data with results obtained with approaches based on standard RGB cameras. In fact, even tough results with Kinect-like cameras are very promising, it is still not clear if similar performance can be obtained with traditional cameras.

A study on the set of actions a user is more likely to learn and perform (without embarrassment) to control the system and the best mapping of these actions to functions of the system is also on the way.

VI. ACKNOWLEDGMENT

This work has been partially funded by project AMICO (ARS01_00900; CUP B46G18000390005) inside the PON "*Ricerca e Innovazione*" 2014-2020 of the Italian Ministry of

Education, University and Research (MIUR).

References

[1] Z. Zhang, “Microsoft kinect sensor and its effect,” IEEE multimedia, vol. 19, no. 2, 2012, pp. 4–10.

[2] H.-m. J. Hsu, “The potential of kinect in education,” International Journal of Information and Education Technology, vol. 1, no. 5, 2011, pp. 365–370.

[3] M. Maidi and M. Preda, “Interactive media control using natural interaction-based kinect,” in Acoustics, Speech and Signal Processing (ICASSP), International Conference on. IEEE, 2013, pp. 1812–1815.

[4] F.-S. Hsu and W.-Y. Lin, “A multimedia presentation system using a 3d gesture interface in museums,” Multimedia tools and applications, vol. 69, no. 1, 2014, pp. 53–77.

[5] S. Sorce et al., “A touchless gestural system for extended information access within a campus,” in SIGUCCS, International Conference on. ACM, 2017, pp. 37–43.

[6] S. Rhio, Herriyandi, L. Tri Fennia, and S. Edy, “Kinectation (kinect for presentation): Control presentation with interactive board and record presentation with live capture tools,” Journal of Physics: Conf. Series, vol. 801, no. 012053, 2017, pp. 1–6.

[7] J.-Q. Liu, T. Tateyama, Y. Iwamoto, and Y.-W. Chen, “Kinect-based real-time gesture recognition using deep convolutional neural networks for touchless visualization of hepatic anatomical models in surgery,” in International Conference on Intelligent Interactive Multimedia Systems and Services, 2018, pp. 223–229.

[8] J. C. Castillo et al., “A multi-modal approach for activity classification and fall detection,” International Journal of Systems Science, vol. 45, no. 4, 2014, pp. 810–824.

[9] N. C. Krishnan and D. J. Cook, “Activity recognition on streaming sensor data,” Pervasive and Mobile Computing, vol. 10, Part B, 2014, pp. 138 – 154.

[10] S. Saha, B. Ganguly, and A. Konar, “Gesture matching based improved human-computer interaction using microsofts kinect sensor,” in Microelectronics, Computing and Communications (MicroCom), 2016 International Conference on. IEEE, 2016, pp. 1–6.

[11] L. Lo Presti, M. La Cascia, S. Sclaroff, and O. Camps, “Hankelet-based dynamical systems modeling for 3d action recognition,” Image and Vision Computing, vol. 44, no. 1, 2015, pp. 29–43.

[12] P. Barros, G. I. Parisi, D. Jirak, and S. Wermter, “Real-time gesture recognition using a humanoid robot with a deep neural architecture,” in Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on. IEEE, 2014, pp. 646–651.

[13] E. Cipolla, I. Infantino, U. Maniscalco, G. Pilato, and F. Vella, “Indoor actions classification through long short term memory neural networks,” in International Conference on Image Analysis and Processing. Springer, 2017, pp. 435–444.

[14] S. Loth, K. Jettka, M. Giuliani, and J. P. de Ruiter, “Ghost-in-the-machine reveals human social signals for human–robot interaction,” Frontiers in psychology, vol. 6, no. 1641, 2015, pp. 1–20.

[15] F. Vella, I. Infantino, and G. Scardino, “Person identification through entropy oriented mean shift clustering of human gaze patterns,” Multimedia Tools and Applications, vol. 76, no. 2, 2017, pp. 2289–2313.

[16] L. Lo Presti and M. La Cascia, “3d skeleton-based human action classification: a survey,” Pattern Recognition, vol. 53, 2016, pp. 130–147.

[17] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, 2015, pp. 436–444.

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, 1997, pp. 1735–1780.

[19] F. Chollet, “keras,” https://github.com/fchollet/keras, 2015, [retrieved: january, 2019].

[20] M. Abadi et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, [retrieved: january, 2019]. [Online]. Available: http://tensorflow.org/

[21] Theano Development Team, “Theano: A Python framework for fast computation of mathematical expressions,” arXiv e-prints, vol. abs/1605.02688, May 2016, [retrieved: january, 2019]. [Online]. Available: http://arxiv.org/abs/1605.02688