Neurocomputing 268 (2017) 76–86

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

An analysis of Convolutional Long Short-Term Memory Recurrent Neural Networks for gesture recognition

Eleni Tsironi, Pablo Barros*, Cornelius Weber, Stefan Wermter[1]

Department of Computer Science, Knowledge Technology (WTM), University of Hamburg, Vogt-Koelln-Strasse 30, Hamburg D-22527, Germany

ARTICLE INFO

Article history:
Received 8 July 2016
Revised 29 November 2016
Accepted 12 December 2016
Available online 2 May 2017

Keywords:
Gesture recognition
CNN
LSTM
CNN visualization

ABSTRACT

In this research, we analyze a Convolutional Long Short-Term Memory Recurrent Neural Network (CNNLSTM) in the context of gesture recognition. CNNLSTMs are able to successfully learn gestures of varying duration and complexity. For this reason, we analyze the architecture by presenting a qualitative evaluation of the model, based on the visualization of the internal representations of the convolutional layers and on the examination of the temporal classification outputs at a frame level, in order to check if they match the cognitive perception of a gesture. We show that CNNLSTM learns the temporal evolution of the gestures classifying correctly their meaningful part, known as Kendon's stroke phase. With the visualization, for which we use the deconvolution process that maps specific feature map activations to original image pixels, we show that the network learns to detect the most intense body motion. Finally, we show that CNNLSTM outperforms both plain CNN and LSTM in gesture recognition.

1. Introduction

Gesture recognition is a field of particular interest to researchers varying from virtual reality to Human–Robot interaction. In this paper, we contribute to a more detailed description and analysis of the Convolutional Long Short-Term Memory Recurrent Neural Network (CNNLSTM) that we introduced in our former work [1] to deal with gesture recognition. CNNLSTM was inspired by the efficiency of Convolutional Neural Networks in implicit feature extraction and the ability of Recurrent Neural Networks in modeling time series. We proposed CNNLSTM based on Multichannel Convolutional Neural Networks (MCCNN) [2] for gesture recognition and extended the concept by adding Long Short-Term Memory units for addressing the problem of dynamic gesture recognition as a sequence-labeling task and not as a simple classification task as in the case of MCCNN.

In this paper, we provide a further analysis on our previously reported results and we further examine what the trained convolution kernels of the convolutional layers of the proposed system learn to detect from the input image given the influence of the joint training with the recurrent layer. Such combinations of jointly trained convolutional and recurrent models are very promising for many temporal classification tasks (e.g. activity recognition [3], speech recognition [4]) and have been gathering a lot of attention. Therefore, it is interesting to find out what such architectures are able to learn.

A gesture recognition system needs to take care of three subtasks: the segmentation of the body parts involved in the gesture, the feature extraction necessary for the representation of the segmented region and the recognition algorithm which labels a gesture. Those processes depend highly on the input sensor that captures the body posture data and may vary from wearable joint coordinate tracking sensors over depth sensors to simple RGB cameras. For example, in the case of wearable joint tracking sensors, the segmentation is self-explanatory, while RGB cameras, as the ones used for the development of our system, provide more noisy input that requires a more difficult segmentation process, e.g. skin color segmentation or motion analysis.

The feature extraction process can be either implicit [5] or explicit [6]. Explicit feature extraction requires a predefined representation by the system developer e.g. a hand orientation process. The problem with explicit feature extraction methods is that they are domain specific [7], and usually do not provide generalization [8]. For each type of environment, a new set of feature extractors is proposed.

For implicit feature extraction, Convolutional Neural Networks (CNNs) are well known for several computer vision tasks [6,9–11]. CNNs are trained either on the raw input or a smoothly prepro-

* Corresponding author.
E-mail address: barros@informatik.uni-hamburg.de (P. Barros).
[1] http://www.informatik.uni-hamburg.de/WTM/

http://dx.doi.org/10.1016/j.neucom.2016.12.088

cessed version of it, and can learn how to represent images across domains [12].

Besides feature extraction, another important part of a gesture recognition system is classification. Gestures are sequences and to deal with these there have been proposed numerous classifiers varying from simple rule-based classifiers to complex neural network architectures (e.g. Recurrent Neural Networks [13], Echo State Networks [14]). However, the models that gather particular interest due to their efficiency in dealing with sequence labeling are Hidden Markov Models [15] and Dynamic Time Warping models [16]. Most of these models deal with sequence classification as a stochastic task; however, these models usually cannot deal with problems such as a highly varying number of images in one sequence.

The gesture recognition system that we analyze requires the input stream of an RGB camera and uses motion representation in order to extract the body parts involved in the gesture. The motion is represented by differential images calculated on an RGB input stream that is further given to the CNNLSTM temporal classifier, which jointly performs the feature extraction, representation and recognition tasks. CNNLSTM is a Recurrent Neural Network, trained with Back Propagation Through Time (BPTT). It was inspired by the efficiency of Convolutional Neural Networks (CNNs) in implicit feature extraction and their successful application in the form of Multichannel Convolutional Neural Networks (MCCNNs) in gesture recognition [2] and by the ability of Long Short-Term Memory Recurrent Neural Networks (LSTMs) with forget gates and peephole connections [17] in modeling long-range dependencies of sequential data. It deals with the limitations caused by the use of feed-forward networks in sequential tasks such as the requirement of using fixed-size windows and their lack of flexibility in learning sequences of different sizes. At the same time, it avoids the process of explicit feature extraction by jointly training the CNN and the LSTM using the convolutional layers as a trainable feature detector. Similar forms of combinations of CNNs and LSTMs have been successfully used in other sequential tasks such as activity recognition [3] and speech recognition [4].

The aim of this work is to provide a visualization of the convolutional kernels of the architecture so that we can find out what features the architecture learns to extract from the visual input when trained with BPTT. We also perform an extensive evaluation of the previously published results carrying out an error analysis of both sequence and frame-level classification. The main contribution of this work is that for the first time to the best of our knowledge a deconvolution process is used for the visualization of those visual stimulus components that excite the kernels of the convolutional layers in a convolutional recurrent network like CNNLSTM. The frame-level classification analysis also reveals that CNNLSTM learns to recognize correctly the meaningful part of a gesture also known as the stroke phase [18].

In Section 2 we present our gesture recognition system using the CNNLSTM architecture. Based on the experiments described in Section 3, we analyze the capacities of the model as a jointly trained feature extractor and temporal classifier in Sections 4 and 5. Finally, in Section 6, we carry out an error analysis and in Section 7 we briefly discuss our findings and share some ideas about future work.

2. Gesture recognition with CNNLSTM

The first step for the development of a gesture recognition system, as already mentioned, is the process of segmenting the moving body parts. In order to segment those regions, our model requires the calculation of differential images on which the representation of the motion change is depicted and is given as input to the CNNLSTM temporal classifier which is trained to be both an implicit feature extractor and a gesture sequence classifier.

More specifically, at each time step, the system creates a motion representation image, called differential image, by means of a three-consecutive frame subtraction (Eq. (1)).

$$\Delta = (I_t - I_{t-1}) \wedge (I_{t+1} - I_t), \tag{1}$$

where Δ is the differential image, I_t, I_{t-1}, I_{t+1} are the frames at the current time step t, the previous time step $t-1$ and the next time step $t+1$, respectively, and $\wedge$ is the *bitwise AND* operation.

Since we use a single immobile RGB camera as the input sensor and since there are no other moving objects nor any other people except for the subject performing the gesture in front of the sensor, we can use motion analysis to detect whole body motion. Differential images [19] are more robust to noise in comparison to the simple image subtraction approach.

Recurrent Neural Networks (RNNs) require temporally pre-segmented inputs in order to be trained. Since CNNLSTM is also a type of RNN, it also requires manually annotated gestures in which the beginning and end frames of each gesture are marked in advance. Moreover, our temporal classifier, like most RNN classifiers, offers a frame-level classification which, however, due to the internal memory, learns to classify based on the previously processed frames (see Section 2.1). Therefore, we have a frame-level classification, but what is useful in the so-called sequence labeling tasks, like gesture recognition, is a sequence-level classification. In other words, we want to receive a single label per gesture sequence, a task that requires the post-processing of the frame-level classification output of the classifier. For this reason, in the test phase, CNNLSTM uses a heuristic, which is further discussed in Section 2.2.

2.1. CNNLSTM – frame-level classification

As mentioned above, for the training of CNNLSTM a temporally pre-segmented gesture sequence is converted to a sequence of differential images and at each time step the CNNLSTM architecture processes each differential image separately. The CNNLSTM consists of two consecutive convolutional layers, each layer implementing subsequently convolution and pooling operations. The flattened output of the last convolutional layer is given to the input nodes of a hidden LSTM Recurrent Neural Network, which is finally connected to a softmax output layer. Fig. 1 shows a simplified version of the CNNLSTM. Both the convolutional as well as the LSTM layers are trained jointly using Backpropagation Through Time [20]. During model training, each of the differential images is given as input to the CNNLSTM, followed by its label. The activation of each unit of the output layer corresponds to a specific gesture class.

The convolution layer and the pooling layer are integrated with one squashing function through which the output of a max-pooling layer is passed in combination with an additive bias:

$$x_j^l = \tanh\left(\text{pooling}_{\max}\left(\sum_i x_j^{l-1} * k_{ij}\right) + b_j^l\right), \tag{2}$$

where x_j^l are the feature maps produced by the convolutional layer l, x_j^{l-1} are the feature maps of the previous convolutional layer $l-1$, k_{ij} are the i trained convolution kernels and b_j^l the additive bias. Finally, $\text{pooling}_{\max}(\cdot)$ is the max-pooling operation and $\tanh(\cdot)$ is the hyperbolic activation function.

The hidden LSTM units consist of memory blocks which are controlled by memory cells, an input and output gate, a forget gate and peephole connections. The equations below describe the activations of a memory block of the recurrent hidden LSTM layer.

$$i_t = \sigma(x^t W_{xi} + h_{t-1} W_{hi} + c_{t-1} W_{ci} + b_i),$$
$$f_t = \sigma(x^t W_{xf} + h_{t-1} W_{hf} + c_{t-1} W_{cf} + b_f),$$
$$o_t = \sigma(x^t W_{xo} + h_{t-1} W_{ho} + c_t W_{co} + b_o),$$

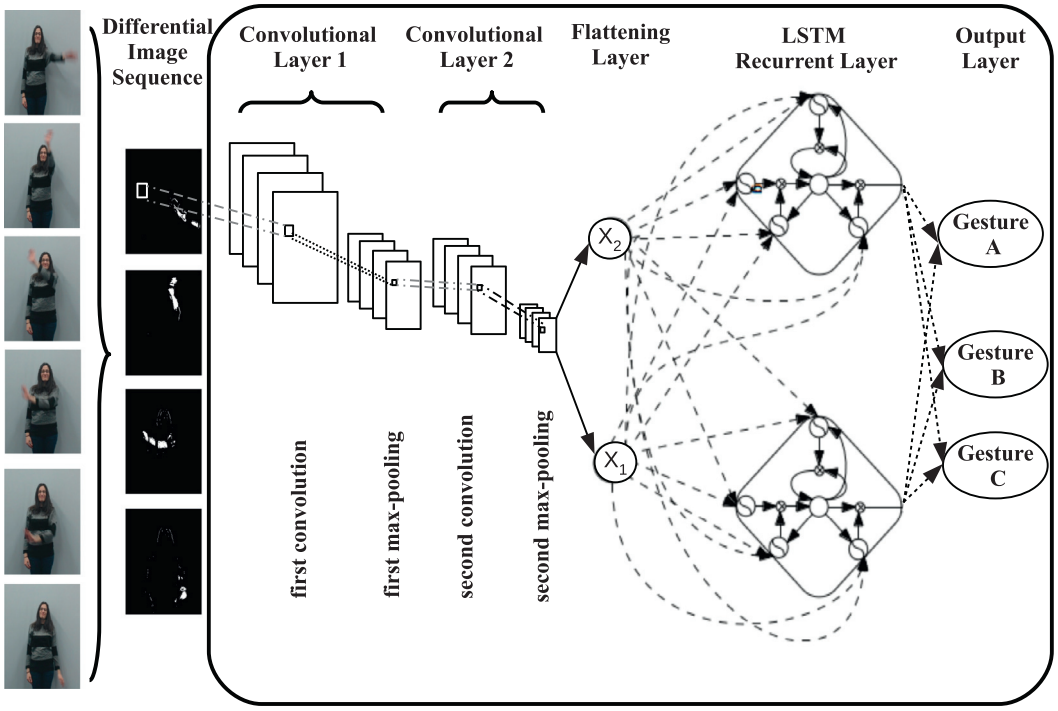

Fig. 1. The CNNLSTM architecture, simplified for visualization purposes.

$$c_t = f_t \circ c_{t-1} + i_t \circ tanh(x^t W_{xc} + h_{t-1} W_{hc} + b_c),$$
$$h_t = o_t \circ tanh(c_t), \tag{3}$$

where x^t is the input to the LSTM block, i_t, f_t, o_t, c_t, h_t are the input gate, the forget gate, the output gate, the cell state and the output of the LSTM block, respectively, at the current time step t. W_{xi}, W_{xf}, W_{xo} are the weights between the input layer and the input gate, the forget gate and the output gate, respectively. W_{hf}, W_{hi}, W_{ho} are the weights between the hidden recurrent layer and the forget gate, the input gate and the output gate of the memory block, respectively. W_{ci}, W_{cf}, W_{co} are the weights between the cell state and the input gate, the forget gate and the output gate, respectively, and finally, b_i, b_f, b_o are the additive biases of the input gate, the forget gate and the output gate, respectively. The set of activation functions consists of the sigmoid function $\sigma(\cdot)$, the element-wise multiplication $\circ$ and the hyperbolic activation function $\tanh(\cdot)$.

CNNLSTM is a recurrent network and therefore it is trained with Backpropagation Through Time. The process starts with random initialisation of all the model parameters, except for the biases which are initialized with zeros. During the training phase, the weight update takes place after a whole sequence has been propagated forward through the network. The error signals are calculated with respect to the Mean of Cross Entropy Losses cost function. The loss function was chosen as the natural cost function to be used for a softmax output layer aiming to maximize the likelihood of classifying the input data correctly.

Finally, we need to calculate the time complexity of CNNLSTM. In order to do that, we have to calculate first the time complexity of the convolutional layers and that of the LSTM layer. The complexity of all convolutional layers [21] is estimated as $O(\Sigma_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2)$, where d is the number of convolutional layers, n_l is the number of filters in the lth layer, n_{l-1} is the number of input channels of the lth layer, s_l is the spatial size of the filter, and m_l is the spatial size of the output feature map. The complexity of the CNN layers occurs due to the convolution operation (the number of operations per input grows quadratically with the length of the kernel), the number kernels and the cache unfriendly memory access [22].

On the other hand, LSTM is local in space and time [23], which means that the input length does not affect the storage requirements of the network and for each time step, the time complexity per weight is $O(1)$. Therefore, the overall complexity of an LSTM per time step is equal to $O(w)$, where w is the number of weights.

The CNNLSTM complexity per time step can be calculated as the sum of the complexity of the convolutional layers and the LSTM layer: $O(\Sigma_{l=1}^{d}(n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2) + w)$ and for all the training process $O((\Sigma_{l=1}^{d}(n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2) + w) \cdot i \cdot e)$, where i is the input length and e the number of epochs. Therefore, we can conclude that our model has O complexity in the typical asymptotic notation.

2.2. Sequence-level classification

The CNNLSTM provides an output at every frame based on previous and current input. In other words, CNNLSTM classifies gestures at a frame level, which means that each differential image in a gesture sequence is separately classified based on the information extracted from the previously processed differential images. Although the model is able to classify long streams of consecutive body postures, the proposed system is developed to classify temporally pre-segmented gestures, which means that the frames of the approximate start and end of a gesture are annotated in advance. For a sequence-level/gesture-level classification we used simple heuristics according to which the whole sequence is assigned the gesture label with the highest frequency in the frame-level classification of the temporally pre-segmented sequence. This approach does not take into account the topological order of the frames and just depends on a frequency measure, which, as we will later show, is consistent with the evaluation method we will discuss below.

We evaluated the efficiency of our heuristics based on the cognitive perception of gestures as presented by Kendon [18]: the process of gesture recognition can be divided into five gesture phases

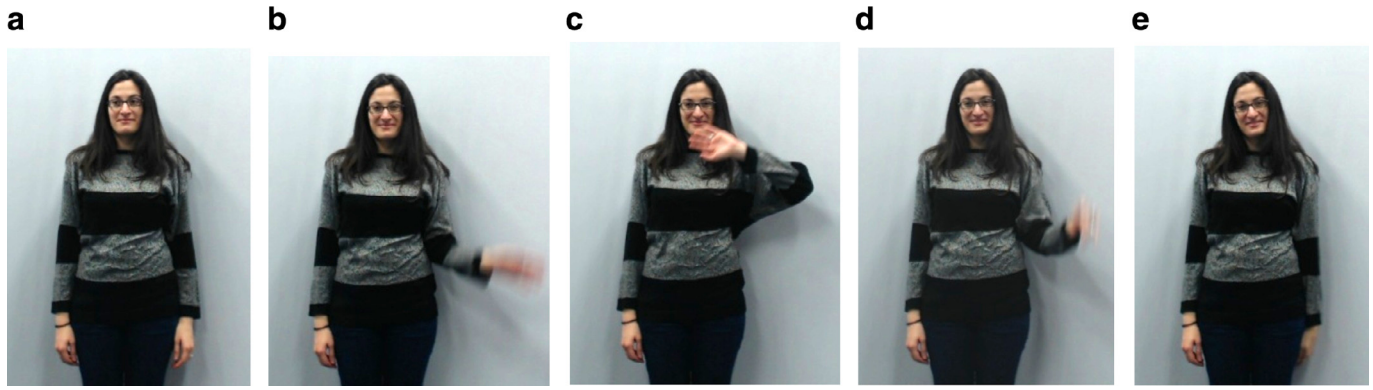

Fig. 2. Gesture phases based on the cognitive perception of gestures: (a) rest position, (b) preparation, (c) stroke, (d) retraction, (e) rest position.

[18] (Fig. 2). During the first phase the arms stay in a rest position (Fig. 2(a)), for example the arm is resting on the table or next to the torso. The next phase is a preparation phase (Fig. 2(b)) with the body parts involved in the gesture getting ready to perform the meaningful part of the gesture. The climax of the gesture takes place in the third phase (Fig. 2(c)), the stroke phase, during which the actual gesture is performed. After the stroke phase, a retraction phase (Fig. 2(d)) brings the body parts involved in the gesture back to their resting position (Fig. 2(e)). Based on this phase classification, we can infer that the frames that correspond to the stroke phase of a gesture, are the ones that correspond to the meaningful part of it and therefore they have to be classified correctly by a gesture recognition system.

3. Experiments

Our model was evaluated on the Gesture commands for Robot InTeraction – GRIT corpus,[2] known previously as TsironiGR corpus [1], consisting of nine gesture classes performed by six subjects each. The purpose of using our own dataset is that none of the publicly available datasets meet our requirements for our Human–Robot Interaction scenarios (e.g. ChaLearn).[3] Furthermore, other set-ups do not match a Human–Robot Interaction scenario e.g. inconvenient camera positions restricted to capture specific body parts (right above the user's palm). In addition, the majority of datasets were created collecting poses or postures to address the problem of static and not dynamic gesture recognition. Some other dynamic datasets, created for gesture recognition, include interaction with objects, which according to our point of view are meant for activity recognition and were particularly created for other specific scenarios such as driving. Moreover, the majority of datasets consist of single palm or hand gestures. Finally, some other datasets that were closer to our Human–Robot Interaction scenario were lacking in subject variability (gathered on only one or two subjects) or offer joint coordinates, e.g. extracted from RGB-D sensors, but without any visual stream.

The quantitative evaluation of the proposed CNNLSTM architecture in terms of precision and recall has been previously presented in [1]. However, the evaluation was very brief, comparing the system performance results just to a CNN architecture (CNN_{MHI}) trained on motion history images (a variation of MCCNN as suggested by Barros et al. [2]). Therefore, in the current work, we include two new experimental set-ups, which we use for the comparison of our model, a plain CNN_{frame} architecture and a plain LSTM architecture. The parameters of all the architectures were selected empirically, after a set of experiments. The depth of two convolutional layers was based on the MCCNN [2] and after a set of experiments was shown to perform better than using one or three covolutional layers. The small number of convolutional layers can be explained by the small size of the input images and by the use of differential images, which remove the background and irrelevant features. The size of the images was also determined based on the empirical experiments and was sufficient for training the network in the minimum amount of time. For all experiments we used the same configuration; the same number of convolutional layers, feature maps, kernel size, and pooling size, as well as the same number in the nodes of the fully-connected layers, which in the cases of CNN_{MHI} and CNN_{frame} is a feed-forward layer.

The formerly used CNN_{MHI} was developed to classify the temporally pre-segmented sequences receiving as input their motion signature. This motion signature is a motion history image, which is the sum of all differential images of the temporally pre-segmented frames that correspond to a gesture sequence. The architecture consists of two convolutional layers, a feed-forward fully connected layer and a softmax output layer. In other words, the architecture is the same as CNNLSTM with the only difference that the recurrent LSTM layer is replaced by the feed-forward fully connected layer.

In this work, we also want to examine whether CNNLSTM performs better or worse than a plain CNN architecture named CNN_{frame} which ignores completely the temporal evolution of a gesture, except for the three consequent frames that are required for the calculation of a differential image. Therefore, CNN_{frame} gets as input a single differential image and classifies the differential image to one of the gesture categories. The architecture is exactly the same as of CNN_{MHI}, with the only difference being the type of input and that in order to assign a single label to the gesture sequence the same heuristics, described in Section 2.2, are used as for CNNLSTM.

We also compare CNNLSTM to a plain LSTM architecture, that receives as input the extracted features of a temporally pre-segmented sequence. The features were extracted by using the pre-trained convolutional layers of the CNN_{frame} architecture as a feature extractor in order to have the same feature extraction technique as in CNNLSTM. The architecture consists of one LSTM recurrent layer and a softmax output layer. For the sequence-level classification the same approach as in Section 2.2 is used. Important to note that using this architecture, no joint training was done.

All the architectures were evaluated by performing five experiments for each one, with the same set-up, with a random selection of the gesture sequences in the database and a distribution of 60% for training, 20% for validation and 20% for testing. By doing empirical evaluations and no special parameter optimization, we chose for all the architectures the same input size for the differential and motion history images to be 64 × 48 pixels. For all the models

[2] Available in https://www2.informatik.uni-hamburg.de/wtm/datasets/
[3] http://gesture.chalearn.org/2013-multi-modal-challenge.

Table 1
Accuracy, precision, recall, *F*1-score, training time per epoch and the number of epochs.

Model	Accuracy	Precision	Recall	*F*1-score	Time (s)	Epochs
CNN_{frame}	71.85% ± 3.74%	77.93% ± 5.14%	71.66% ± 3.39%	74.63% ± 3.84%	66	35
CNN_{MHI}	77.78% ± 4.19%	79.29% ± 3.80%	78.56% ± 4.09%	78.91% ± 3.80%	2	160
LSTM	74.63% ± 4.37%	75.59% ± 4.44%	74.53% ± 4.39%	75.06% ± 4.39%	45	150
CNNLSTM	91.67% ± 1.13%	92.35% ± 0.98%	91.90% ± 1.05%	92.13% ± 1.00%	130	19

Table 2
Error analysis results: mean recall, precision and *F*1 measure.

$\text{Accuracy}_{\text{frame}}$	$\text{Accuracy}_{\text{seq}}$	$\text{Recall}_{\text{seq}}$	$\text{Precision}_{\text{seq}}$	$F1_{\text{seq}}$
81.97%	92.59%	92.85%	93.46%	93.15%

with convolutional layers the parameters were the same; the size of the first convolutional layer kernels was 11 × 11, while the size of the kernels of the second convolutional layer was 6 × 6. The size of the max-pooling window 2 × 2 was applied in both layers. The feature maps of the first layer were 5 while those of the second were 10. The number of the hidden neurons of the LSTM layer was 500. At the same time the number of neurons of the feed-forward fully connected layers in the CNN_{frame} and CNN_{MHI} was also 500. The number of the hidden neurons was chosen empirically, after a set of experiments within the range of 200 and 750 neurons.

As Table 1 reveals, CNNLSTM outperforms in terms of all measures the other baseline architectures. We showed that in this way, the joint training takes advantage of both the feature extraction power of CNNs and the ability of learning from former inputs of LSTMs. As expected the training time per epoch[4] of CNNLSTM is higher than for the other models, however, in terms of training cycles, it needs only about 19 epochs, while CNN_{frame}, CNN_{MHI} and plain LSTM reaches optimal results, respectively, on average after 35, 160 and 150 epochs. We stopped the training of the network when the validation and training error increased. We state that the network reached a convergence, when the network validation and training error starts to increase. It is obvious that the effect of the training in CNNLSTM is positive not only in terms of precision and recall, but it also converges faster than the isolated CNN and the plain LSTM.

We would also like to note, that since CNNLSTM outperformed the other models and based on the ability of convolutional layers in learning to extract meaningful information from raw image data, we performed experiments training the network on raw image sequences. However, using the same parameters as described above, the network did not learn any meaningful information. This could be an indication that unlike the training with differential images, training a network with raw RGB data could require more training data, or input images with bigger size or even a deeper network topology with more convolutional layers.

Finally, in order to carry out the error analysis we used the results of one of the best runs, shown in Table 2. All the measures are computed for sequence-level classification, except for accuracy which is calculated also at frame level.

4. Analysis of convolutional layers

There is a serious criticism on CNNs that the learned features are not interpretable. As an answer to that, there have been visualization approaches in order to provide a clear understanding of how such architectures work, so that we can avoid the improvement of them based just on blind trial-and-error attempts.

In order to understand the operation of convolutional layers of an architecture or, in other words, to figure out which input stimulus excites individual feature maps at each convolutional layer, we need a deconvolutional neural network [24] i.e. a reverse convolutional network that, unlike a typical convolutional network, maps features to pixels and not pixels to features. In order to do that, we first need to pass an input image through the trained convolutional layers of the architecture. Once, we have computed the feature maps of each desired layer to be deconvolved, we pass them to the deconvolution network where we repeatedly apply the following consecutive operations: unpooling, rectification and filtering to reconstruct the activity in the previous layer that caused the activation until we reach the original image pixel level (for more information see [25]).

The visualization process is very important in order to understand whether good system performance can be reasonably justified. More specifically, we can see what kind of features the trained kernels learn to extract based on a specific kind of input, whether those features are the desired ones (e.g. arm motion detection) and whether they are semantically related to the nature of the task of gesture recognition. Moreover, in the particular case of CNNLSTM, the visualization process allows us to study whether the temporal aspect affects the implicit feature extraction positively or negatively. Furthermore, we can justify the misclassifications produced by the system and in this manner, we can improve our model by applying hyperparameter optimization.

From our CNNLSTM architecture we isolated the trained convolutional layers from the rest of the network and we repeated the previously described deconvolution process. We applied the deconvolution process in each of the kernels, or filters, of each of our convolutional layers. Each kernel is a weight matrix which is convolved to the image, filtering different patterns of the image.

Since the visualization process based on the deconvolutional neural network is computationally expensive and highly time-consuming, we visualized just one whole gesture sequence per gesture class. For each of the differential images of each of the selected gesture sequences, we generated 5 visualizations for the first convolutional layer and 10 for the second convolutional layer. Moreover, we chose some simple random and previously unseen images, which did not have any relation to the gesture domain, so that we can verify some low-level features that the trained kernels learn to extract. Those were particularly interesting for the interpretation of the first layer kernels, which function as feature extractors, in other words, they are supposed to extract low-level features. We do not provide the visualization of them though for the second layer kernels, based on the assumption that the deeper the layers are, the higher-level the extracted features and since the semantic domain is not the same (not related to gesture recognition), it would be redundant. In the next two subsections, we present the visualizations of the kernels in the first and second layer.

4.1. Visualizing the kernels of the first convolutional layer

Kernel 0. Examining carefully the visualization of the non-domain examples (Fig. 3(a)) in combination with the visualization of the

[4] We used a device with Intel Core i7-2670QM CPU 2.20 GHz 8 and a 7.7 GiB memory.

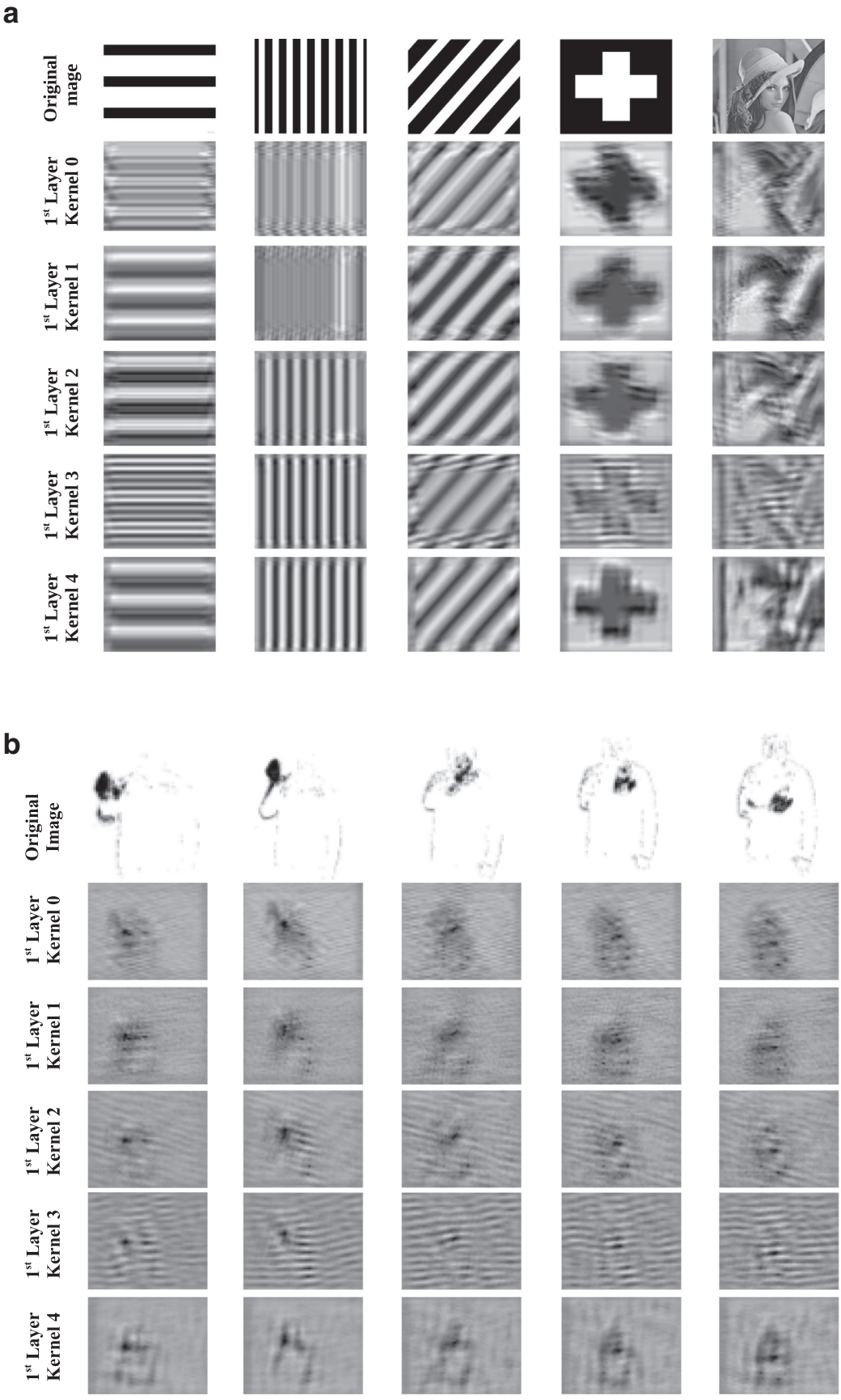

Fig. 3. (a) Visualization of the results of the deconvolution process of the 1st layer kernels for previously unseen images. From left to right; horizontal lines, vertical lines, diagonal lines, white cross, the standard image processing "Lena" image. (b) Visualization of the results of the deconvolution process of the 1st layer kernels for an example gesture "Circle". The colors of the original images are inverted for better visualization.

sequence of differential images (Fig. 3(b)) we can observe the following findings: Kernel 0 creates a strong blurring effect, while at the same time creating a high contrast between bright and dark values with a focus on vertical edge detection on the right side of the image. We could also say that it works as a very coarse contrast detector. Based on the domain gesture sequences, it is clear that the filter detects very high-intensity pixel values which correspond to the most intense movement and the detection in the deconvolved image corresponds to black edges or big black dot blobs. The shadows created by the gesture execution are only detected when they are very intense, so in general the visualization is smooth and blurred and it is only merged with the main motion when the shadow pixel and main motion pixels are in the same neighborhood. Respectively, the body pose or the body parts such as head are detected when they are substantially moved. In general, there is a very smooth or a quite unclear body pose detection. The background is normally smooth, but depending on the shadows and reflections some noise is also detected.

Kernel 1. Following the same analysis methodology, Kernel 1 also blurs the image but it seems to offer more compact structure results than Kernel 0 and less blurry edge detection. Moreover, it ignores finer information that corresponds to low (dark) pixel values and it seems that it also specializes on the detection of vertical edges on the right side of the image (Fig. 3(a), Kernel 1, horizontal lines). In the differential images, except for the above characteristics, Kernel 1 offers finer motion and shadow detection than Kernel 0, therefore, as in Kernel 0, most moving parts are detected such as palm, arm or head, but the motion is mostly in a form of a moving small dot blob; however, when the arms are stretched, horizontal edges can be detected clearly. When the body is substantially moved, some body parts are detected too. The background seems to contain less noisy information, however, that depends on the amount of information in the differential image.

Kernel 2. In general, Kernel 2 detects the highest intensity values, captures less information than the previous two kernels and provides finer edge detection. In the gesture sequences analysis, this kernel offers clearer motion detection than Kernels 0 and 1. At the same time, because it has a stronger blurring effect, motion detection has the form of more uniform blobs. In terms of the detection of shadows formed due to body and arm motion, there are more shadow responses which are thicker than for kernel 0. The background is noisy with diagonal wavy edge responses. The human body borders are sometimes detected, when the body parts are intensely moved.

Kernel 3. The brightest values both on the general images, as well as on the differential images, are represented as very blurred horizontal edges that correspond mostly to arm or head motion. There is no concrete body figure detection, except for some cases where the most moved parts are detected. There is no clear difference between the body figure and the background. In this sense, shadows are also detected but not as intensely as in the previous filters, corresponding also to horizontal edges. The background is wavy, but not intense when the shadow and the background noise are not very intense.

Kernel 4. In contrast to Kernel 3, this kernel seems to detect mostly vertical responses and offers sharper vertical edge detection than Kernels 0–2. In general, we notice more abrupt changes and Kernel 4 has a stronger blurring effect than Kernel 2. As far as the differential images are concerned, there is a more compact blob detection corresponding to the body parts that were mostly moved, usually reflecting the arm or head. The most intense motion is also darker on the deconvolved image. Moreover, it has a better visualization of the motion and offers a cleared visual perception of the human body and its separation from the background. The body is mostly not detected as a compact, uniform blob but mostly its contours are pointed out. However, it is worth mentioning that when the body motion is very light, Kernel 4 also fails to detect it. The background seems quite distinct but it gets affected by the reflections and shadows and their respective intensity. Shadows are detected but not affecting the detection of the main motion. After Kernel 0, Kernel 4 is the one that offers the most information on motion detection but without redundant noisy information that Kernel 0 also detects.

4.2. Visualizing the kernels of the second convolutional layer

The second convolutional layer consists of 10 trained kernels. Like the kernels of the first convolutional layer, those of the second layer detect the most intense values of the image, while at the same time filtering out less intense values that were detected by the first layer kernels. Therefore, the output of the visualization of the second layer kernels appears to be more blurred than the output of the first layer kernels (Figs. 4 and 5). In general, Kernels $1_{2ndlayer}$, $4_{2ndlayer}$, $5_{2ndlayer}$, $6_{2ndlayer}$ and $9_{2ndlayer}$ cause horizontal edge responses. Kernel $5_{2ndlayer}$ seems to produce the most blurring output in comparison to the rest. The kernels with the horizontal responses discard most of the input passed through the previous layer and are more sensitive to the most intense pixel values and less sensitive to less intense pixel values. Due to this fact, the information that gets captured mostly corresponds to the arm motion since arm motion is usually the most intense one. Information about the shadow created due to the motion of the arm and the rest of the body is not captured.

On the other hand, Kernels $0_{2ndlayer}$ and $7_{2ndlayer}$ seem to have both horizontal and vertical edge responses. The kernels $0_{2ndlayer}$, $2_{2ndlayer}$, $3_{2ndlayer}$, $7_{2ndlayer}$ and $8_{2ndlayer}$ seem to detect both horizontal and vertical edges and capture more information. Thereafter, not only the main arm motion is detected, but also high intensity values created due to shadows or due to motion caused by other human body parts (e.g. the head). The difference among them seems also to be the intensity of each of them. For example, Kernel $7_{2ndlayer}$ provides more information in comparison to $8_{2ndlayer}$, while kernels $0_{2ndlayer}$ and $3_{2ndlayer}$, which seem to offer similar output, are less sensitive to the background noise.

As shown by the extensive analysis above, the visualization of the trained kernels does not reveal a lot of high-level semantic information related to the gesture recognition task, which can be reasonably justified by the fact that our architecture does not use a lot of convolutional layers. The deeper a convolutional model is, the higher level is the information it learns to extract. More precisely, our visualization shows that the visual stimuli to which the kernels respond are low-level and learn to detect high-intensity pixels, which correspond mostly to the arm motion (to avoid any confusion, for visualization purposes, the original images in Figs. 3(b), 4 and 5 are shown in inverted colors). It also seems that the training with Back Propagation Through Time allowed the convolutional layers to learn interesting visual cues since the system performance as we showed in Section 3 for CNNLSTM is quite high.

5. Analysis of the temporal aspect of CNNLSTM

Before carrying out our CNNLSTM experiments there were two fundamental questions: firstly, whether the convolutional layers could be trained using the error signals of BPTT; secondly, if yes, in what way the error signals of the LSTM affected the training of the convolutional layers, in other words, what kind of features they learn to extract, taking into account the temporal effect during the

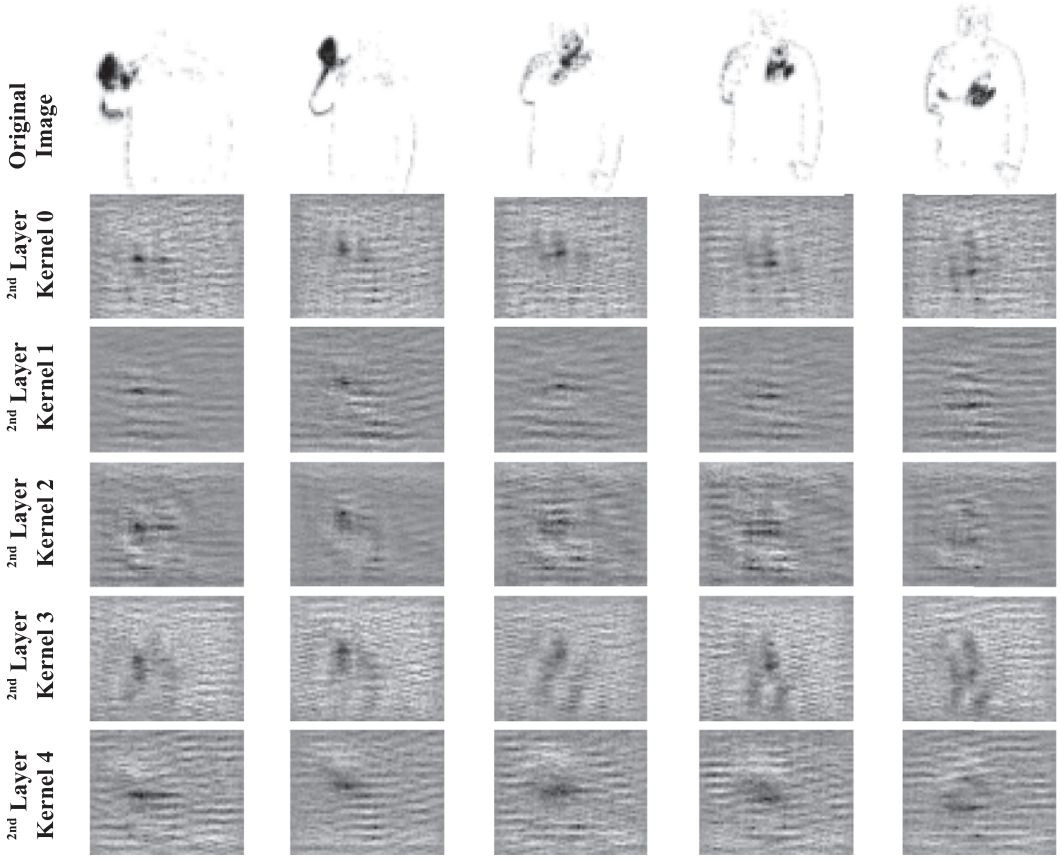

Fig. 4. Visualization of the results of the deconvolution process of the 2nd layer Kernels 0–4 for an example gesture "Circle". The colors of the original images are inverted for better visualization.

Table 3
Patterns of correct frame classification.

Pattern	Percentage
All frames classified correctly	19.0
First front misclassified frames (less than 1/3 of the sequence)	40.0
Less than 4 misclassified frames (at the beginning)	32.0
Pattern XLabelX (stroke phase correctly classified)	4.0
Pattern 1–2 wrong frames (middle or end)	2.0
Pattern XXLabel (first 2/3 of the sequence wrong)	1.0
Pattern XLabel (first half of the sequence wrong)	1.0
Pattern LabelXLabelX	1.0

training. After exploring what kind of features convolutional layers learn, it is clear that the combined training with LSTM does not affect negatively the training of the convolutional layers that learned to detect the highest intensity pixels, which due to the nature of the differential images correspond to the body motion, more specifically arm motion for the majority of images, excluding of course, images of the rest position phase of the gesture.

We will now analyze the frame-level classification of the gesture sequences. The goal of this analysis is first, to verify whether the heuristics used for assigning the most frequent label among the frames to a sequence can be justified by the results and second, to discover frame patterns that might correspond to the cognitive perception of gestures as proposed by Kendon [18]. The names of the patterns, except for those where the name explicitly denotes it, are composed of the symbols *X* and *Label. X* corresponds to any subsequence of incorrectly labeled frames and *Label* corresponds to a sequence of frames classified with the correct label.

In Table 3, we see that there are three main patterns extracted that cover 91% of the correctly classified gestures. The most frequent pattern shows that 40% of the correctly classified sequences begin with misclassifications. The number of those misclassified frames is less than 1/3 of the gesture sequence length. This is something totally justified, since those frames correspond to a resting position of the subject and a preparation phase. Most gestures start from the same resting position and it is not clear during the preparation what the exact gesture is going to be.

The second most frequent category corresponds to less than four misclassified frames at the beginning of the gesture sequence. While this pattern can be considered a subcategory of the previous one, where 1/3 of the gesture was misclassified, we think it is necessary to differentiate them, since the network is able to recognize the gesture correctly almost from the start.

The third most frequent category includes the gestures in which all the frames were correctly classified. The fourth most frequent pattern, *XLabelX*, corresponds also to the correct classification of the stroke phase, according to Kendon, and misclassifies the frames of the retraction phase.

The other patterns are outliers but also explainable. For example, the patterns *XLabel, XXLabel* and *LabelXLabelX* suggest that the front frames were misclassified but the remaining half or at least a third of the frames were classified correctly, showing that the network needs more time to distinguish between two different gestures or between the correct gesture class and a combination of others. This makes sense considering the preparation and the retraction phases of the gesture that can be very fuzzy. Moreover, in the case of *XXLabel*, we can also explain that the speed the subject performed the gesture with could matter and therefore, more time is needed until the correct class label is found. Of course, the problem that occurs here is that, if the label of each of the classes of *XX* was the same, or in other words the total number

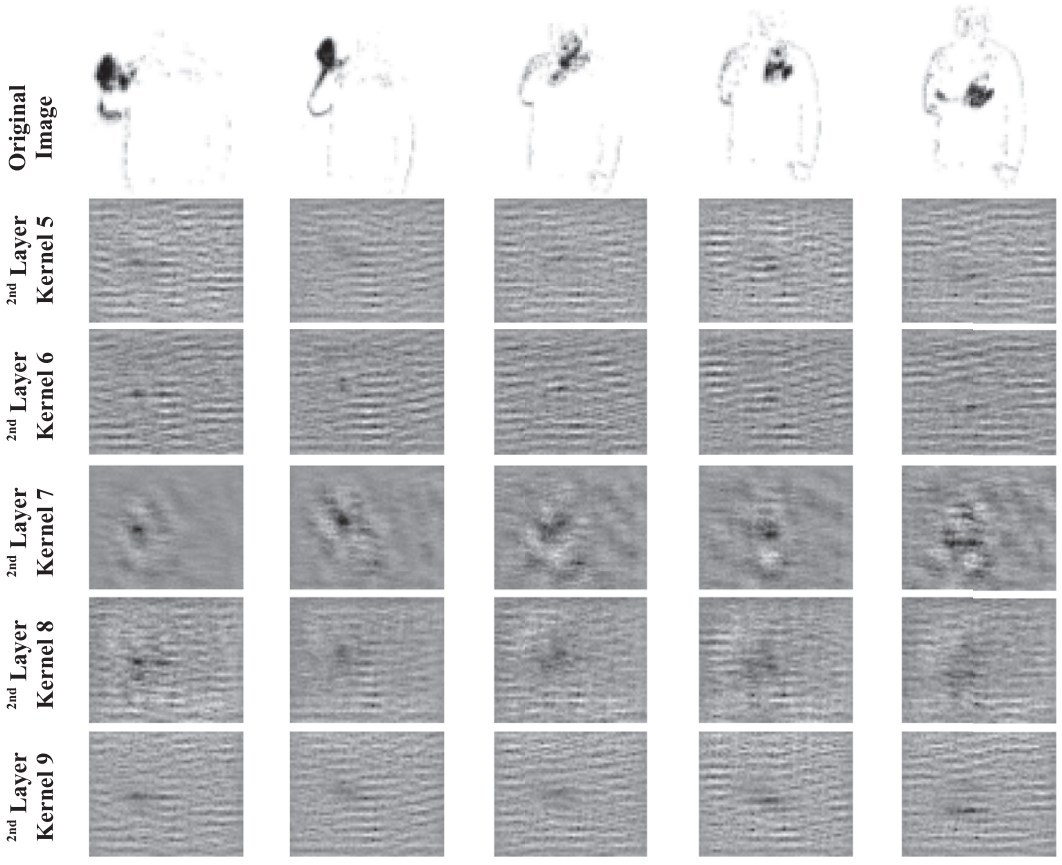

Fig. 5. Visualization of the results of the deconvolution process of the 2nd layer Kernels 5–9 for an example gesture "Circle". The colors of the original images are inverted for better visualization.

to the misclassified frames was assigned the same incorrect label and not a different one for each of the *X* labels, the gesture at the sequence level would have been assigned the label *X* and not the correct one, which is also a problem that can occur to the Kendon-like, correctly classified stroke pattern (*XLabelX*). However, this pattern corresponds only to 1% of the cases and therefore represents a very small part of the correctly classified sequences. Moreover, in general, the most important feature of the sequence classification heuristics, is that there are no inconsistent classifications, such that correctly classified frames are randomly mixed with other misclassified frames in the sequence, there is no other correctly classified sequence the frame distribution of which does not correspond to the Kendon gesture perception phases.

At this point, in order to highlight the ability of CNNLSTM and LSTM in learning the temporal evolution of a gesture based on their internal memory, we note that such a pattern analysis was not possible on the outputs of CNN_{frame}. For the sequence-level classification we used the same heuristics and after carrying out the frame analysis, we could not identify any patterns such as the ones of Table 3. The correct sequence label might have been assigned correctly to the majority of the gestures but the correct label distribution did not have any cohesion at a frame level, as it did for the CNNLSTM and the correct frame labels were interfering with a lot of wrong frame labels. So even if the frame-level accuracy was low (Table 4), the sequence-level accuracy was much higher, which, however, could not be justified by the Kendon gesture phases. The meaningful part of a gesture corresponds to the Kendon stroke phase and therefore it is crucial for a good gesture recognition system to classify those frames consistently correctly. However, CNN_{frame} fails to do that and it fails to capture the temporal evolution of a gesture. The fact that the correct sequence label is assigned to the gesture is justified by the nature of the heuristics used for sequence-level classification, which assigns the label with the maximum number of votes.

Table 4
Frame level accuracy.

Model	Frame level accuracy
CNNLSTM	80.10% ± 1.33%
LSTM	63.53% ± 1.91%
CNN_{frame}	37.53% ± 1.45%

On the other hand, LSTM frame-level classification analysis offered results similar to CNNLSTM. The percentage of sequences in which the stroke phase of the gesture was correctly classified (all correctly classified frames, less than 1/3 of the first frames misclassified and *XLabelX*) was 74.44% and just in about 11% of the cases the frame distribution had no cohesion or patterns that could not be justified by Kendon's perceptual phases. The rest belonged to the pattern of the first half of the frames being misclassified, which shows that in these cases, it takes quite long for the correct label to be recognized by the network.

In any case, the analysis of CNNLSTM reveals that the network learns to classify in a more coherent way than the isolated CNN or the isolated LSTM. We need to admit though, that if a better image processing technique for feature extraction is applied, then maybe LSTM will have a more competitive performance to the one of CNNLSTM. We show however, that the jointly trained CNNLSTM architecture outperforms the separately trained CNN and LSTM.

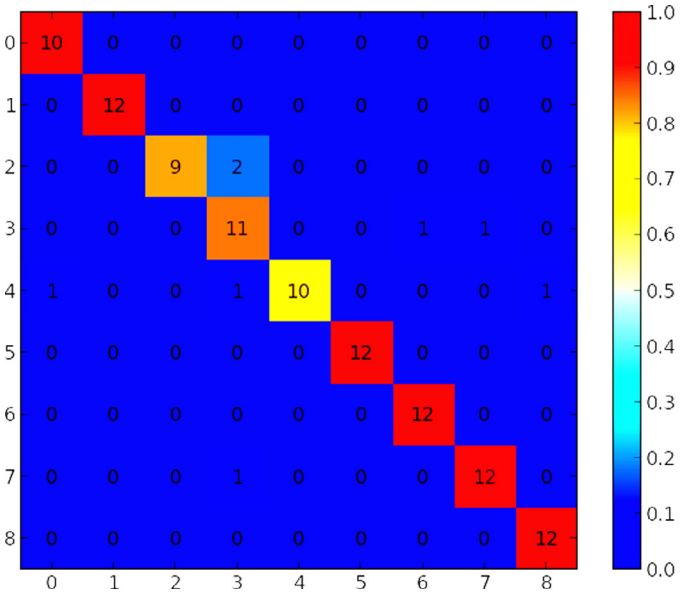

Fig. 6. Confusion matrix of the gesture-level classification. The labels correspond to: abort (0), circle (1), hello (2), no (3), turn left (4), turn right (5), stop (6), turn (7) and warn (8).

Table 5
Recall, precision and *F*1 measure per class.

Gesture	Abort (0)	Circle (1)	Hello (2)
Recall (%)	100.00	100.00	81.82
Precision (%)	90.91	100.00	100.00
***F*1 score (%)**	95.24	100.00	90.00
Gesture	**No (3)**	**Turn left (4)**	**Turn right (5)**
Recall (%)	84.62	76.92	100.00
Precision (%)	73.33	100.00	100.00
***F*1 score (%)**	78.57	86.96	100.00
Gesture	**Stop (6)**	**Turn (7)**	**Warn (8)**
Recall (%)	100.00	92.31	100.00
Precision (%)	92.31	92.31	92.31
***F*1 score (%)**	96.00	92.31	96.00

6. Error analysis

Since there were only 8 sequence-level misclassifications, the analysis below was combined with the examination of the classification at the frame level. Fig. 6 presents the confusion matrix for all the gesture classes and Table 5 presents evaluation metrics.

The most occurring confusion was found between the gesture classes “hello” and “no”. After examining the frame-level classification of the two misclassified instances of the class, it was revealed that they were clearly classified as “no”. The confusion can be explained by the fact that both gestures consist of the same arm movement, with the only difference that in the “hello” class, the palm is fully open, while in the “no” class, just the index finger is lifted, a detail that is not always captured as the visualization of the convolutional layers reveals. On the other hand, the misclassified “no” gesture as “turn” could be due to the fact that the cyclic movement of “turn” is performed by some subjects in the dataset just with the index finger. In the case of the instances of “no” misclassified as “stop”, checking the classification at frame level as produced by the CNNLSTM, we can notice that there is a confusion between the classes “stop” and “no”, which shows that the network is not confident about the previously seen frames and cannot clarify this confusion with the evolution of time.

As far as the class “turn” is concerned, one of its instances was misclassified as “no”. Studying the classification per frame, the label “turn” interchanges with “no”, even if in terms of frequency “no” occurs significantly more. As mentioned above, this could be due to the fact that “turn” is performed many times in the training set as an index finger gesture, like the “no”, even if their trajectories are not the same.

Moreover, even if every instance classified as “turn left” is correct, the system failed to catch three instances, which are wrongly classified as either “abort”, “no” and “warn”. It is worth mentioning that at the frame level seven out of the ten correctly classified sequences have all frames classified correctly. One instance, however, in terms of frame classification was very inconsistent, with a lot of different labels interchanged. However, because the classification strategy on the sequence level depends on the label with the maximum frequency, it was correctly classified as “turn left”.

7. Conclusion and future work

In this paper we studied in depth the formerly introduced Convolutional Long Short-Term Memory Recurrent Neural Network (CNNLSTM) architecture for the task of gesture recognition. We used a Deconvolutional Neural Network to visualize the features of the original input image that each of the kernels of the convolutional layers of our CNNLSTM model learned to extract. As discussed in Section 4, the majority of kernels learn to detect the most intense pixel values which correspond to the main arm motion depicted in the differential images that are given at each time step as input to CNNLSTM.

Moreover, we also showed that the CNNLSTM frame-level classification is consistent with the cognitive perception of gestures as defined by Kendon, ensuring that the model mostly classifies correctly all the frames belonging to the stroke phase during which the meaningful part of the gesture is performed. By doing that, we can explain the performance of the model, and illustrate what the network is learning. In addition, based on the above error analysis, we can state that the remaining confusion of the model can be explained considering the nature of some gestures with similar body postures or gesture signatures and the number of inconsistent classifications is very low.

Visualizing what the network learned gives us a powerful tool to investigate how the gestures were recognized, and understand the behavior of our network. It was possible to identify that the motion representation actually makes the network learn only movement, which is reflected in what the filters capture. This means that the CNNLSTM network is specialized to detect movement. Also, by detailing each kernel, it is possible to see that the movements on the right side of the image are mostly captured, indicating that the data used is highly biased by gestures happening in the right side of the scene. This gives us an insight that the dataset could be improved by adding a uniform distribution of movements, and retraining the network to generalize movements in all possible positions.

Further work will include the extension of the current CNNLSTM architecture to train the model for direct sequence-level classification. This can be achieved by training the model with a Connectionist Temporal Classification (CTC) loss function [26]. CTC eliminates both the need for training with temporally presegmented gestures as well as the post-processing of the frame-level classification outputs in order to achieve sequence-level classification. In this way, we will be able to compare our system to other systems that perform gesture recognition and temporal gesture segmentation at the same time.

Moreover, part of future work will consist of experimentation with different kinds of network inputs, e.g., using raw visual and depth information instead of training the system on differential images extracted from RGB images. Finally, part of the future work, based on the frame-level classification analysis, will try to explore why fewer errors are made in the retraction phase than in the

preparation phase of the gesture. Based on the internal memory of LSTM which learns by what has already been observed in the stroke phase, it makes sense that the LSTM outputs the correct gesture label even after the gesture has already been made. We will try to verify this assumption by checking the evolution of output of the LSTM gates, in other words if they are closed or open and when they allow the cell state to be updated.

Acknowledgment

The authors gratefully acknowledge partial support from the CAPES Brazilian Federal Agency for the Support and Evaluation of Graduate Education (p.n.5951-13-5), the German Research Foundation DFG under project CML (TRR 169) and the Hamburg Landesforschungsförderungsprojekt.

References

[1] E. Tsironi, P. Barros, S. Wermter, Gesture recognition with a convolutional long short-term memory recurrent neural network, in: Proceedings of the Twenty-Fourth European Symposium on Artifical Neural Networks, Computational Intelligence and Machine Learning, 2016, pp. 213–218.
[2] P. Barros, G.I. Parisi, D. Jirak, S. Wermter, Real-time gesture recognition using a humanoid robot with a deep neural architecture, in: Proceeding of the Fourteenth IEEE-RAS International Conference on Humanoid Robots (Humanoids), 2014, pp. 646–651, doi:10.1109/HUMANOIDS.2014.7041431.
[3] J. Donahue, L.A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, T. Darrell, K. Saenko, Long-term recurrent convolutional networks for visual recognition and description, in: Proceeding of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2625–2634, doi:10.1109/CVPR.2015.7298878.
[4] T.N. Sainath, O. Vinyals, A. Senior, H. Sak, Convolutional, long short-term memory, fully connected deep neural networks, in: Proceeding of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2015, pp. 4580–4584, doi:10.1109/ICASSP.2015.7178838.
[5] S. Rautaray, A. Agrawal, Vision based hand gesture recognition for human computer interaction: a survey, Artif. Intel. Rev. 43 (2012) 1–54, doi:10.1007/s10462-012-9356-9.
[6] J. Nagi, F. Ducatelle, G. Di Caro, D. Ciresan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, L. Gambardella, Max-pooling convolutional neural networks for vision-based hand gesture recognition, in: Proceeding of the 2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), 2011, pp. 342–347, doi:10.1109/ICSIPA.2011.6144164.
[7] S. Bilal, R. Akmeliawati, M. El Salami, A. Shafie, Vision-based hand posture detection and recognition for sign language, in: Proceeding of the Fourth International Conference on Mechatronics (ICOM), 2011, pp. 1–6, doi:10.1109/ICOM.2011.5937178.
[8] A. Jmaa, W. Mahdi, Y. Jemaa, A. Hamadou, Hand localization and fingers features extraction: application to digit recognition in sign language, in: Proceeding of the 2009 Intelligent Data Engineering and Automated Learning – IDEAL, 5788, 2009, pp. 151–159, doi:10.1007/978-3-642-04394-9_19.
[9] G.E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Comput. 18 (7) (2006) 1527–1554.
[10] D. Cireşan, U. Meier, J. Masci, J. Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural Netw. 32 (2012) 333–338. http://dx.doi.org/10.1016/j.neunet.2012.02.023.
[11] M. Khalil-Hani, L.S. Sung, A convolutional neural network approach for face verification, in: Proceeding of the 2014 International Conference on High Performance Computing Simulation (HPCS), 2014, pp. 707–714, doi:10.1109/HPCSim.2014.6903759.
[12] T.P. Karnowski, I. Arel, D. Rose, Deep spatiotemporal feature learning with application to image classification, in: Proceeding of the 2010 Ninth International Conference on Machine Learning and Applications (ICMLA), 2010, pp. 883–888, doi:10.1109/ICMLA.2010.138.
[13] N. Neverova, C. Wolf, G. Paci, G. Sommavilla, G.W. Taylor, F. Nebout, A multi-scale approach to gesture detection and recognition, in: Proceeding of the 2013 IEEE International Conference on Computer Vision Workshops (ICCVW), 2013, pp. 484–491, doi:10.1109/ICCVW.2013.69.
[14] D. Jirak, P. Barros, S. Wermter, Dynamic gesture recognition using echo state networks, in: Proceedings of the Twenty-third European Symposium on Artifical Neural Networks, Computational Intelligence and Machine Learning, 2015, pp. 475–480.
[15] G. Rigoll, A. Kosmala, S. Eickeler, High performance real-time gesture recognition using hidden Markov models, in: Proceeding of the 1998 Gesture and Sign Language in Human–Computer Interaction, 1371, 1998, pp. 69–80, doi:10.1007/BFb0052990.
[16] T. Darrell, A. Pentland, Space-time gestures, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'93), 1993, pp. 335–340, doi:10.1109/CVPR.1993.341109.
[17] F. Gers, Long Short-Term Memory in Recurrent Neural Networks, Universität Hannover, 2001 Ph.D. thesis.
[18] A. Kendon, Gesticulation and speech: two aspects of the process of utterance, The Relationship of Verbal and Nonverbal Communication, Walter de Gruyter, 1980, pp. 207–227, doi:10.1515/9783110813098.207.
[19] R.T. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, et al., A System for Video Surveillance and Monitoring, 2, Technical Report CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, 2000.
[20] P.J. Werbos, in: Backpropagation through time: what it does and how to do it, 78, 1990, pp. 1550–1560, doi:10.1109/5.58337.
[21] K. He, J. Sun, Convolutional neural networks at constrained time cost, in: Proceeding of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 5353–5360, doi:10.1109/CVPR.2015.7299173.
[22] K. Chellapilla, S. Puri, P. Simard, High performance convolutional neural networks for document processing, in: Proceeding of the Tenth International Workshop on Frontiers in Handwriting Recognition, SuviSoft, 2006.
[23] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, doi:10.1162/neco.1997.9.8.1735.
[24] M.D. Zeiler, G.W. Taylor, R. Fergus, Adaptive deconvolutional networks for mid and high level feature learning, in: Proceeding of the 2011 International Conference on Computer Vision, 2011, pp. 2018–2025, doi:10.1109/ICCV.2011.6126474.
[25] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceeding of the European Conference on Computer Vision, 8689, 2014, pp. 818–833, doi:10.1007/978-3-319-10590-1_53.
[26] A. Graves, Supervised sequence labelling, in: Supervised Sequence Labelling with Recurrent Neural Networks, 385, Springer, 2012, pp. 5–13, doi:10.1007/978-3-642-24797-2_2.

Eleni Tsironi received her Bachelor degree in Business Administration with a major in Business Information Systems from the Athens University of Economics and Business (A.U.E.B.), Greece, and Master degrees in Science Informatics from the University of Hamburg, Germany and in Natural Language Processing and Human Language Technology from the University of Wolverhampton, Universit de Franche-Comt and Universidade do Algarve. Her main research interests include neurocognitive systems for human–robot interaction, natural language processing, image summarization, and bio-inspired gesture recognition.

Pablo Barros received his Bachelor degree in Information Systems from the Federal Rural University of Pernambuco and his Master degree in Computer Engineering from the University of Pernambuco, both in Brazil. Since 2013, he is a research associate and Ph.D. candidate in the Knowledge Technology Group at the University of Hamburg, Germany, where he is part of the research project CML (Cross-Modal Learning). His main research interests include deep learning and neurocognitive systems for multimodal emotional perception and learning, human–robot interaction and cross-modal neural architectures.

Cornelius Weber is Lab Manager at the Knowledge Technology group at the University of Hamburg. He graduated in physics, Bielefeld, Germany, and received his PhD in computer science at the Technische Universität Berlin in 2000. Then he was a postdoctoral fellow in Brain and Cognitive Sciences, University of Rochester, USA. From 2002 to 2005 he was a research scientist in Hybrid Intelligent Systems, University of Sunderland, UK. Then, Junior Fellow at the Frankfurt Institute for Advanced Studies, Frankfurt am Main, Germany, until 2010. His core interest is computational neuroscience with foci on vision, unsupervised learning and reinforcement learning.

Stefan Wermter received the Diplom from the University of Dortmund, the M.Sc. from the University of Massachusetts, and the Doctorate and Habilitation from the University of Hamburg, all in Computer Science. He has been a visiting research scientist at the International Computer Science Institute in Berkeley before leading the Chair in Intelligent Systems at the University of Sunderland, UK. Currently Stefan Wermter is Full Professor in Computer Science at the University of Hamburg and Director of the Knowledge Technology institute. His main research interests are in the fields of neural networks, hybrid systems, cognitive neuroscience, bio-inspired computing, cognitive robotics and natural language processing. In 2014 he was general chair for the International Conference on Artificial Neural Networks (ICANN). He is also on the current board of the European Neural Network Society, associate editor of the journals Transactions on Neural Networks and Learning Systems, Connection Science, International Journal for Hybrid Intelligent Systems and Knowledge and Information Systems and he is on the editorial board of the journals Cognitive Systems Research and Journal of Computational Intelligence.