Learning Attribute Representation for Human Activity Recognition

Fernando Moya Rueda, Gernot A. Fink
TU Dortmund University
Department of Computer Science
Dortmund, Germany
{fernando.moya, gernot.fink}@tu-dortmund.de

Abstract—**Attribute representations became relevant in image recognition and word spotting, providing support under the presence of unbalance and disjoint datasets. However, for human activity recognition using sequential data from on-body sensors, human-labeled attributes are lacking. This paper introduces a search for attributes that represent favorably signal segments for recognizing human activities. It presents three deep architectures, including temporal-convolutions and an IMU centered design, for predicting attributes. An empiric evaluation of random and learned attribute representations, and as well as the networks is carried out on two datasets, outperforming the state-of-the art.**

I. Introduction

Human activity recognition (HAR) is a classification task for recognizing human movements. It is the core of smart assistive technologies, e.g., in smart-homes, in rehabilitation and health support, and in the industry [1]. HAR uses as inputs signals from videos or a set of on-body sensors. This paper covers HAR tasks using multichannel time series signals acquired from a set of on-body sensors. The recognition of human activities is a complicated task due to the large intra- and inter-class variability of human actions. Humans carry out the same tasks in different ways; even, a single person realizes a task differently. Furthermore, HAR is difficult due to the unbalance problem, where there are more samples of certain actions than others. Based on the assumption that body movements present certain patterns, HAR's idea is to classify them with different techniques.

Multichannel time-series based HAR uses a combination of signals recorded from different types of sensors, e.g., accelerometers, gyroscopes, magnetometers, and heart rate monitors [2], [3], [4]. Usually, segmentation by means of a sliding window, extraction of engineered-features followed by a classification constitute the standard pipeline for recognizing human actions [5]. Nevertheless, relevant features are hard to compute, not accurate, and not scalable. Convolutional neural networks (CNNs) have been successfully applied to HAR tasks, unifying conveniently the feature extraction and classification [6]. These architectures extract hierarchically the basic and complex features of the human body movements and learn their temporal dependencies.

Motivated by the success of attribute representations for image classification [7], human actions can be likewise represented by a collection of attributes. These attributes describe semantically and coarsely human actions, i.e., attributes like

moving left foot and right foot, forward, and sequential can be taken as the "walking" action [8]. Common attributes represent a set of similar human actions. For example, "walking" and "running" could have the movement of the feet as common attributes. The usage of this representation is suitable for recognition tasks where the data is unbalanced or training and testing sets are disjoint, e.g., zero shot learning. As such collection of attributes in the context of multichannel time series-based HAR is not available, we propose learning an attribute representation for HAR using deep architectures.

The paper is structured as follows: section II will discuss the related work in the field of multichannel time series based HAR. In section III, attribute representations in images and sequences will be introduced. In section IV and section V, deep architectures and learning of attribute representations are described. Experiments on two datasets, the Opportunity-gestures and locomotion datasets, and the Pamap2 datasets, will be presented in section VI and section VII. In the last section, conclusions will be drawn

II. Related Work

Traditionally, HAR using multichannel time series from on-body sensors has been solved by using engineered features obtained by statistical processes in a sliding window manner. The mean, median, min, max and magnitude area are examples of statistical features. Classifiers, e.g., SVM, Random Forest, Dynamic Time Warping or Hidden Markov Models use these features for predicting action classes of windowed sequences [5]. These approaches work relatively well, when data is scarce and highly unbalanced. However, identifying relevant features is time consuming, leading to difficulty in scaling up activity recognition of complex high level behaviours [3].

As deep neural networks have become the state-of-the-art in different tasks, e.g., image classification [9], image segmentation [10], [11], speech recognition and word spotting [12], [13], they haven been, recently, deployed in HAR in a sliding window framework [1], [2], [3], [4], [6]. Deep neural networks allow to learn the features and the classifier in an end-to-end manner directly from the raw data of multichannel time series. CNNs exploit the hierarchical composition of human movements, where complex activities are a combination of basic movements. The authors in [2] proposed an architecture with temporal-convolutions applied over all

sensors simultaneously. Their architecture is composed of two or three temporal-convolution layers with ReLU activation functions followed by a downsampling (max-pooling), and a softmax classifier. The authors in [6] introduced a deeper network with four temporal-convolution layers over individual sensors followed by a fully-connected layer and a softmax classifier. The fully-connected layer finds correlations between different sensors. An architecture that combines temporal-convolutions and Long Short-Term Mermory (LSTM) units is presented by the authors in [3]. LSTMs are recurrent units with memory cells and a gating system that find long temporal dependencies in time-series problems without the limitations of exploiting or vanishing gradients during learning [3], [14]. Their architecture is composed of four temporal-convolution layers with ReLU activation functions followed by two LSTM layers and a softmax classifier. While temporal-convolutions are applied to individual sensors de-noising and capturing local dependencies, the LSTM layers find temporal dependencies over all the pre-convolved sensor sequences.

The authors in [1] introduced a CNN based on IMUs for HAR in the context of the Order Picking process. The network has parallel convolutional blocks, one per IMUs' data. These blocks find intermediate representations of the IMUs' sequences, which are then concatenated by means of fully-connected layers. This network becomes robust against IMU's faults and asynchronous data. The authors in [4] compared different architectures including CNNs, similar to [6], and LSTMs. They used a three-layered LSTM and one-layered bi-directional LSTM (B-LSTM) configurations directly on the raw data from the time-series. The B-LSTM processes the input sequences following two directions (forward, and backward). This network shows the state-of-the-art performance.

III. Attribute representation

Attributes provide high level semantic descriptions of objects, categories, and scenes in images [7], [12], [8]. For example, in object classification, attributes can be colour, shape, texture, or size of objects. In HAR, collection of verbs and objects have been used for representing human actions in images and videos [8], [15]. Attribute representations have been used for zero-shot and transfer learning on different tasks like object recognition [7] and word spotting [12], [13]. The advantage of using an attribute representation is that recognition tasks can be carried out without much annotations, or in cases where data are highly unbalanced, the quantity of samples is large, and the testing set contains unseen object classes at training [7].

Consider for HAR a dataset with tuples $(x_1, y_1), ..., (x_n, y_n)$ with $X = x_1, x_2, ..., x_n$ being N sample sequences from an arbitrary space and $Y = y_1, y_2, .., y_n$ their respective classes from a set of K number of classes, the idea is to learn the function $f : X \rightarrow Y$. Using an attribute representation A, an additional mapping is introduced $f : X \rightarrow A \rightarrow Y$, see Figure 1. This additional mapping serves as an intermediate layer that allows sharing high-level concepts among classes, making full usage of the whole data. Classes with higher

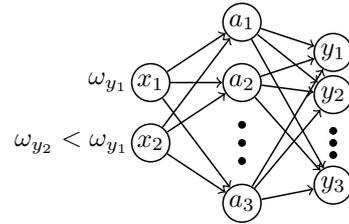amount of samples can borrow attributes to lesser frequent classes.

Fig. 1: Graphical representation of the HAR using attribute representations as an intermediate layer. HAR using attribute representation could be enhanced as classes with more samples, e.g., y_1 with ω_{y_1} could help learning lesser frequent classes, e.g., y_2.

In the context of multichannel time-series based HAR, datasets are highly unbalanced, specially towards the "NULL" class. This class covers all the human actions that are not relevant for the task, sometimes involving more than 75% of the recorded data [3]. An attribute representation is beneficial for solving this case, where sequences of the "NULL" class provide good material for learning shared attributes that are contained in less frequent classes, see Figure 1.

IV. Attribute-based CNN architecture

In this paper, we proposed three attribute-based architectures based on the CNN, deepConvLSTM presented in [3], [6], and the CNN-IMU introduced in [1]. These architectures have in common temporal-convolution layers and fully-connected layers. The convolutional layers extract temporal local features providing and creating an abstract representation of the input sequence. The fully-connected units connect all the local features giving a global view of the input data. Specifically, the CNN architecture contains four temporal-convolution and two fully-connected layers with ReLU activation functions. The deepConvLSTM has also four temporal-convolution layers with ReLU activation functions, but it uses two LSTM layers instead of the fully-connected ones. The LSTMs capture the global temporal dynamics of the input data. The CNN-IMU network contains parallel temporal-convolution blocks, each with an additional fully-connected layer, see Figure 2. These parallel branches process and merge input sequences from IMUs individually. They create an intermediate representation per IMU. The network combines these intermediate representations into a global one, forwarding them to a softmax layer. Depending on the dataset, the architectures contain also max-pooling layers. For all of the three architectures, each of the convolutional layers has $C = 64$ filters of size $[5, 1]$ performing convolutions only in the time-axis. Both the fully-connected layer and the LSTMs layer contain 128 units. Max-pooling of $[2, 1]$ with stride of 1 is used.

The network's input consists of sequence segments of length T measurements composed of multiple sensor channels D extracted from a sliding window approach with step size s. So, the input's size is $[T, D]$. Having a feature map x^i of size $[T, D, C]$ in layer i, and a set of C_i filters w^{i,c_i} of size
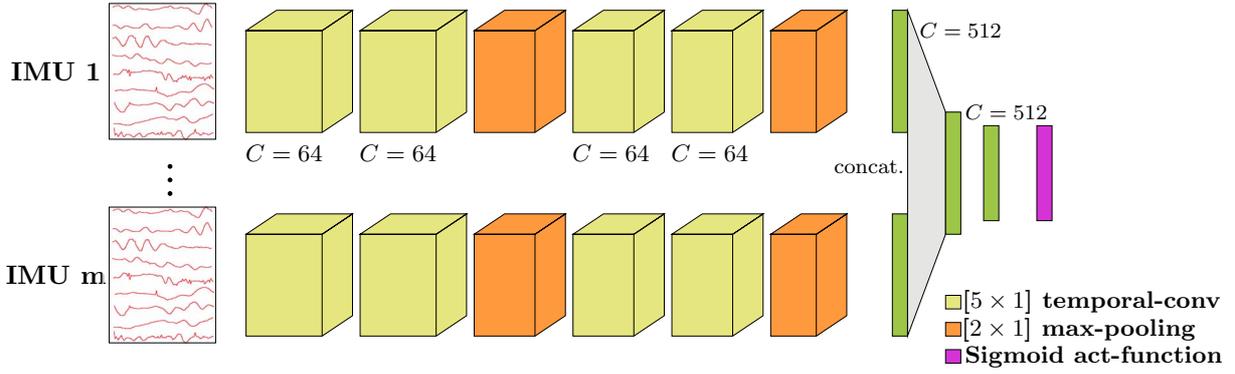
Fig. 2: The attrCNN-IMU architecture contains m parallel temporal convolutional blocks, one per IMU. The ouputs of the blocks are concatenated and forwarded to a fully connected layer. The output layer is the sigmoid function. The attrCNN and the attrCNNLSTM have only one block for all of the IMUs, and they have fully connected and LSTM layers respectivelly.

$[F, 1]$ and biases b^{i,c_i} connecting layers i and j, a temporal-convolution for each d sensor is:

$$x_{t,d}^{j,c_j} = \sigma \left(\sum_{c=0}^{C_i} \sum_{f=0}^{F-1} w_f^{i,c} \cdot x_{t+f,d}^{i,c} + b^{i,c} \right) \qquad \forall d = 1, ..., D \tag{1}$$

where σ is the activation function. The filters w^i are shared among the sensors D.

A max-pooling operation between layer i and j for a single channel c reduces the size of a feature map finding the maximum among a set of P values, and is given by:

$$x_{t,d}^{(j),c_j} = \max_{0 < p \leq P} \left(x_{t+p,d}^{i,c_i} \right) \quad \forall d = 1, ..., D \tag{2}$$

A standard CNN for classification uses a softmax activation, see Equation 3, which produces pseudo-probabilities per class $k_i \in K$. For training, the cross-entropy loss between the estimated probabilities x_k^j, and the target label $y_k \in Y$ is used.

$$x_k^j = \frac{e^{x^i}}{\sum_{k=1}^K e^{x_k^i}} \quad (3) \qquad sigmoid(x) = \frac{1}{1 + e^{-x}} \quad (4)$$

However, the proposed network will compute an attribute representation of an input sequence, rather than classify it using a softmax function, being different from [3] and [1]. As the representation will contain multiple $0s$ and $1s$ for an attribute being present or not, the softmax function is replaced by a sigmoid activation function, see Equation 4, following the architecture described in [16] in the context of word spotting. The sigmoid activation function is applied to each element of the output layer. Its output corresponds to pseudo-probabilities for each attribute a_i being present in the representation.

Such a network must be trained using the binary-cross entropy loss given by.

$$E(a, \tilde{a}) = -\frac{1}{n} \sum_{i=1}^n \left(a_i log \, \tilde{a}_i + (1 - a_i) log \, (1 - \tilde{a}_i) \right) \tag{5}$$

where a is the target attribute representation and $\tilde{a}$ is the CNN's output.

Figure 2 shows the CNN-IMU architecture for attribute representation, called attrIMU-CNN. The other two networks (attrCNN and attrCNNLSTM) are simplifications of the first one. Both are a branch of the CNN-IMU, where their inputs are sequences including all the sensors from the IMUs. The attrdeepCNNLSTM contains LSTMs layers instead of fully-connected ones.

V. LEARNING ATTRIBUTE REPRESENTATIONS

In the context of HAR using multichannel time-series signals and having introduced the attribute representation and the attribute-based deep networks, datasets do not possess annotated attribute representations. Their annotations are related with specific coarse actions, e.g., walk, jump, run. However, there is no annotation of small and granulate actions that could describe a coarse one. For example, for walking, a person should move first one foot and then the other in a coordinate way and speed, and for running these movements are faster. For standing, the person does not move any of its foot. One might think of discriminating walking, running and standing by the attributes moving or not moving, the left or right foot, and the speed. Nevertheless, these annotations do not exist. Besides, humans can not provide easily annotations by just observing the data, as for example they can on images or videos.

Considering the literature in random sub-space projection and random indexing for representing words, phrases and documents [17], [18], [19], a set of different sequence segments could be projected to, or described by a random representation. This random representation should be adequate such that one can differentiate the actions. As this random space is spanned by unknown attributes, it might not be suitable. One needs to search for the best possible representation that will describe properly a set of sequence segments. An evolutionary algorithm can be used for performing this search, where attribute representations are seen as genotypes of the classes. By evaluating, selecting and mutating those genotypes, a proper representation of the classes can be learned.

Algorithm 1 shows the evolutionary algorithm (EA) used for finding the best attribute representation of a certain dataset. The algorithm evaluates the fitness of an attribute representation by training and validating a deep network. The validation weighted $F1$ serves as the performance metric, in which case, first, the cosine distance between the computed $\tilde{a}$ and $a \in A$ target attributes is estimated, finding class predictions. Second, the precision and recall of the predictions are used, see Equation 6. The algorithm mutates, evaluates and selects the best attribute representation for a certain number of iterations starting from a random representation. A global mutation in $\mathbb{B}^n$ is selected to change the attribute representation; that is, the attribute a_i for $i = 1, ..., n$ of a single target attribute representation a flips with probability $p_i \in (0, 1)$.

Algorithm 1 Evolutionary algorithm

1: **procedure** EVOLUTION($niter, K, tr, vl$)   ▷ Input niter:number of iterations, K: number of classes, tr: training set, vl: validation set
2:     $A_{gen} \leftarrow random(K)$
3:     $F_1^{best} \leftarrow 0.0$
4:     **for** $i = 1 : niter$ **do**   ▷ Evaluate each generation
5:         $network \leftarrow trainCNN(tr, A_{gen})$
6:         $F_1^i \leftarrow testCNN(network, vl, A_{gen})$
7:         **if** $F_1^i > F_1^{best}$ **then**
8:             $A_{best} \leftarrow A_{gen}$
9:             $F_1^{best} \leftarrow F_1^i$
10:         $A_{gen} \leftarrow mutate(A_{gen})$
11:     **return** A_{best} ▷ Best attribute generation

As shown in Section VI, the datasets are highly unbalanced and the classification accuracy is not a convenient measurement for evaluating the performance of the networks. The weighted $F1$ considers the correct classification of each class equally, using the precision, recall and the proportion of a class in the dataset [3].

$$F_1 = \sum_i 2 \times \frac{n_i}{N} \times \frac{precision_i \times recall_i}{precision_i + recall_i} \qquad (6)$$

where, n_i is the number of samples for each class $k \in K$, and N is the total number of samples in the dataset.

The evolution is performed for a fix number of iterations $niter$ in which a single attribute generation is used for training a network. The network is trained from scratch within each iteration, i.e., weights from previous generations are not used. The performance on the validation set is taken as the generation's fitness. The best generation or learned attribute set is used for a final training and deploying on a testing set.

VI. DATASETS

An evaluation of the proposed approach is carried out for HAR on the datasets: Opportunity [20], and Pamap2 [21]. In general, the performance of three CNNs using a learned attribute representations is evaluated. First, A proper attribute representation for each dataset and the networks is found by means of the evolutionary algorithm, see Section V. Second, Using this learned representation, a final training of the networks is done.

A. Opportunity

The Opportunity dataset [22], [20] contains recordings of four participants carrying out activities of daily living (ADL). Seven IMUs, placed on both wrists, upper arms, shoes, and back, provide recordings of 113 sensors. Each participant executed five sessions of ADLs. There were no restrictions on performing the activities. In addition, participants carried out a drill session with a predefined sorted set of 17 activities. The dataset contains two different label annotations: **Gestures** with $K = 18$ classes and **Locomotion** with $K = 5$ classes. Therefore, one obtains two different classification tasks. The gestures task has, for example, activities like open and closing doors or drawers, and drinking coffee, and the locomotion has activities related with motion like walking, standing, and sitting. Following [3], [20], sessions ADL3 of participants 2 and 3 are used as validation set, and sessions ADL4 and ADL5 of participants 2 and 3 as testing set. The recording's sample rate is $30Hz$. A sliding window of 720ms or $T = 24$ and a step size of 360ms or $s = 12$ (50% overlapping as [3]) are used for segmenting the sequences.

B. Pamap2

The Pamap2 dataset [21] contains recordings from 9 participants carrying out 12 activities, e.g., walking, standing, cycling. Three IMUs (on the hand, chest and ankle) and a HR-monitor provide recordings of 40 sensors, e.g., accelerometers, gyroscope, magnetometer, temperature and heart rate. Recordings from participants 5 and 6 are used as validation and testing sets respectively. Following [4], IMUs recordings are downsampled to $30Hz$. A sliding window of 3s or $T = 100$ and step size of $660ms$ or $s = 22$ (78% overlapping as in [4], [21]) are used for segmenting the sequences. The window size is smaller than the one in [4] for generating a larger number of segments.

C. Preprocessing

As suggested in [3], [1], input sequences were normalized per sensor to the range $[0, 1]$. Moreover, a Gaussian noise of $\mu = 0$ and $\sigma = 0.01$ is added, simulating sensors' inaccuracies.

VII. EXPERIMENTS

The algorithm 1 finds an attribute representation A_{best} suitable for each dataset, by iteratively training and validating the three architectures using attribute configurations, which mutate starting from random. Different numbers n of attributes $\in \mathbb{B}^n$ for representing a class k are proposed depending on the number of total classes K per dataset. Based on training and validating the architectures on random attributes for different n, we set $n = [10, 32, 24]$ number of attributes for the Opportunity-Locomotion, -Gestures datasets, and Pamap2 dataset. Depending on the dataset and network, the following configurations have been employed. The parameters are updated by minimizing the binary-cross entropy, see Equation 5, using batch gradient descent with RMSProp update rule [3]. We used a learning rate of 0.0001, RMS decay of 0.9, and
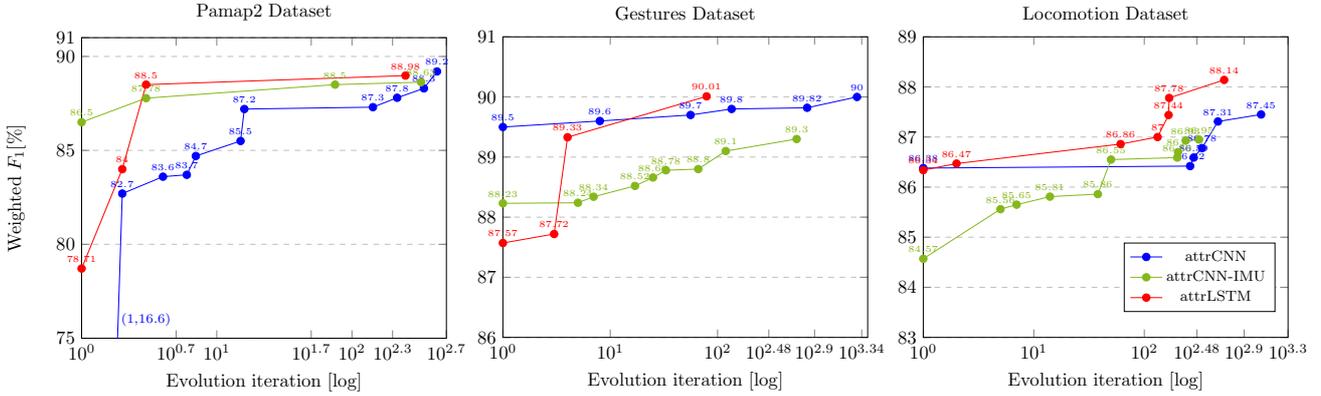
Fig. 3: Weighted $F1[\%]$ of the attributes evolution for the three attrCNNs on the three datasets.

batch size of 100 for Gestures and Locomotion and of 50 for Pamap2. In addition, we used Dropout on the feature-maps inputs of the first, and second fully-connected layers. Networks are trained from scratch for a fixed number of epochs, see Table I, in each EA-iteration. A nearest neighbour approach was used for predicting the class k by measuring the cosine distance from the attribute vectors $\tilde{a}$ to the set $a \in A_{best}$. We used the validation weighted $F1$ as the fitness metric.

TABLE I: Number of epochs for training the architectures on the evolutionary algorithm.

Dataset	Network		
	attrCNN	attrCNN-IMU	attrdeepConvLSTM
Gestures	12	5	10
Locomotion	10	5	10
Pamap2	25	5	25

Figure 3 shows the weighted F1 evolution for the three networks on the three validation datasets. The weighted $F1$ on the Pamap2 presents the biggest improvements of 72.65%, 2.51%, 11.42% using the attrCNN, attrCNN-IMU, and attrdeepCNNLSTM between the first and the last attribute generation in $niter = [426, 75, 249]$ iterations respectively. The weighted $F1$ on the Gestures and Locomotion shows average improvements of 1.39% and 2.39% respectively. It is noticeable that mostly the performance of the networks present a relative good performance despite the randomness of the initial attribute generations. This could be explained as the Gestures and Locomotion datasets are strongly unbalanced towards the NULL class, and they predict correctly sequence segments with that label. However, as the attributes evolve, the correct predictions increase, showing the potential of the attribute search. Moreover, common attributes among classes are found, which helps sharing strength between the more and the less frequent classes. In addition, there is a substantial dimensionality reduction.

Table II shows the learned attribute representation for attrCNN on the Opportunity dataset with Locomotion labels after 1140 iterations. Analyzing the attribute sharing among action classes, there is a sort of semantic relation. For example, the "Stand" class shares 4 and 5 attributes with classes "Sit" and "Lie". These three action classes have lesser sharing with

"Walk" class with 3 and 2 shared attributes. One advantage of this attribute sharing is that the "Null" class attributes become a source for learning attributes shared with infrequent classes. There is not a specific definition of each attribute, but the sort of basic movements or states in each action class keeps certain relation. One uses these relations for learning a better representation of the actions. Observing the attributes found using the attrCNN-IMU on the same dataset, the classes "Stand" and "Walk" shared more attributes than with the class "Sit". These examples show that the learned attribute representation depicts a certain logic, but it changes with respect to the networks.

TABLE II: Attribute representation $A \in \mathbb{B}^{10}$ of the Locomotion dataset found using the evolutionary algorithm and the attrCNN

Classes	Attributes									
	1	2	3	4	5	6	7	8	9	10
Null	1	0	0	0	0	1	0	0	1	1
Stand	0	1	1	0	1	1	1	1	1	1
Walk	0	1	0	0	0	0	0	0	1	1
Sit	0	0	0	0	1	1	0	0	1	1
Lie	1	1	1	0	1	0	0	1	1	0

The learned attribute representation for Gesture dataset using the three networks show multiple relations. As the dataset contains mostly opening and closing of doors, fridges, dishwashers, and drawers, the attributes are mixed. These labels also explains in part the good performance of the initial random representation utilizing the three networks on this dataset. However, after the attribute evolution, classes involving only closing or only opening movements display a strong attribute sharing with $20 - 22$ common attributes.

For the Pamap2 dataset, the learned attribute representation for the networks presents different relations. This dataset has different activities, including moving and static ones, but without the "Null" class. The classes are more diverse, having more marked action classes. So, the shared attribute are also more diverse, e.g., it keeps relations among classes "rope jumping", "lying", "cycling", "ironing".

A. Final training

Finally, we evaluate the performance of the networks using the learned attribute representations on the testing sets. We

have trained the three attrCNNs on the training and validation sets using the learned attribute representations. Similar settings on learning rates, momentum, and updating rule from the evolution are used. However, the number of epochs is increased. In addition, random attributes were used for training and testing the attrCNNs for $niter = 100$. In that case, we report the average of their weighted $F1[\%]$ performances. Table III presents the testing weighted $F1$.

TABLE III: Weighted $F1[\%]$ of the three attrCNNs on the testing sets using the evolved and random attribute representations. In addition, it shows a comparison with the state-of-the-art networks. Results marked with '*' were obtained following the networks and configurations of the original publications.

Architecture	Pamap2	Locomotion	Gestures
CNN [3]	87.37*	87.8	85.1
deepCNNLSTM [3]	87.63*	89.5	91.5
CNN [4]	87.2*	-	90.8
CNN-IMU [1]	89.01*	88.23*	92.15*
attrCNN random	84.72	85.9	88.96
attrdeepCNNLSTM random	85.15	86.64	88.58
attrCNN-IMU random	86.26	86.85	89.92
attrCNN evol	90.55	**90.0**	91.94
attrdeepCNNLSTM evol	88.0	89.01	90.97
attrCNN-IMU evol	**90.88**	89.75	**92.9**

It is noticeable the comparable performance of these representations. Deep networks are capable of learning features and classify input segments when the targets are represented by a set of attributes either random or learned. The attrCNN's and the attrCNN-IMU's performances using the learned attributes are higher in comparison with the random ones. These results demonstrate that the attributes found by means of the evolutionary algorithm on the validation set keep a relation with action classes, and represent better the input sequences. Moreover, these performances also show that the networks benefit from the shared attributes, using information from the most frequent actions reducing the effects of the unbalanced problem. The attrCNN and the attrCNN-IMU using the learned attributes show better performance with respect to similar architectures with action labels as targets from [3], [1]. The attrdeepCNNLSTMs do not benefit greatly in comparison with the deepCNNLSTM; however, their performances are comparable.

VIII. CONCLUSION

We have presented a new application of attribute representations for solving human activity recognition from multichannel time-series signals acquired from on-body sensors. Considering that human-labeled attributes annotations for HAR are not existing, we proposed a method for learning attributes that better represent the time-series signals, starting from a random representation. Three deep architectures for mapping input sequences into a set of attributes for recognition are introduced. We show empirically that these networks using either random or learned attributes, in general, present a comparable or even better performance contrasting with similar networks that predict classes directly.

REFERENCES

[1] R. Grzeszick, J. M. Lenk, F. M. Rueda, G. A. Fink, S. Feldhorst, and M. ten Hompel, "Deep neural network based human activity recognition for the order picking process," in *In Proc. of the 4th Int. Workshop on Sensor-based Activity Recognition and Interaction*. ACM, 2017.

[2] C. A. Ronao and S.-B. Cho, "Deep convolutional neural networks for human activity recognition with smartphone sensors," in *Int. Conf. on Neural Information Processing*. Springer, 2015, pp. 46–53.

[3] F. J. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[4] N. Y. Hammerla, S. Halloran, and T. Ploetz, "Deep, convolutional, and recurrent models for human activity recognition using wearables," *arXiv preprint arXiv:1604.08880*, 2016.

[5] S. Feldhorst, M. Masoudenijad, M. ten Hompel, and G. A. Fink, "Motion classification for analyzing the order picking process using mobile sensors," in *In Proc. of the 5th Int. Conf. on Pattern Recognition Applications and Methods*. SCITEPRESS-Science and Technology Publications, Lda, 2016, pp. 706–713.

[6] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *IJCAI*, 2015, pp. 3995–4001.

[7] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 951–958.

[8] J. Zheng, Z. Jiang, and R. Chellappa, "Submodular attribute selection for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 11, pp. 2242–2255, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[11] V. Badrinarayanan, A. Handa, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling," *arXiv preprint arXiv:1505.07293*, 2015.

[12] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 12, pp. 2552–2566, 2014.

[13] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Proc. Int. Conf. on Frontiers in Handwriting Recognition*, 2016.

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

[15] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, "Human action recognition by learning bases of action attributes and parts," in *Computer Vision (ICCV), IEEE Int. Conf.* IEEE, 2011, pp. 1331–1338.

[16] S. Sudholt and G. A. Fink, "PHOCNet: A Deep Convolutional Neural Network for Word Spotting in Handwritten Documents," in *Proc. Int. Conf. on Frontiers in Handwriting Recognition*, Shenzhen, China, 2016.

[17] M. Sahlgren, "An introduction to random indexing," in *Methods and Applications of Semantic Indexing Workshop at the 7th Int. Conf. on Terminology and Knowledge Engineering, TKE 2005*, 2005.

[18] P. Kanerva, "The spatter code for encoding concepts at many levels," in *ICANN94*. Springer, 1994, pp. 226–229.

[19] R. D. Paradis, J. K. Guo, J. Moulton, D. Cameron, and P. Kanerva, "Finding semantic equivalence of text using random index vectors," *Procedia Computer Science*, vol. 20, pp. 454–459, 2013.

[20] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, vol. 34, no. 15, pp. 2033–2042, 2013.

[21] A. Reiss and D. Stricker, "Introducing a new benchmarked dataset for activity monitoring," in *Wearable Computers (ISWC), 2012 16th Int. Symposium on*. IEEE, 2012, pp. 108–109.

[22] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Networked Sensing Systems (INSS), 2010 Seventh Int. Conf. on*. IEEE, 2010, pp. 233–240.