CHAPTER

DEEP LEARNING FOR HUMAN ACTIVITY RECOGNITION

9

Phyo P. San[a], Pravin Kakar[a], Xiao-Li Li, Shonali Krishnaswamy, Jian-Bo Yang, Minh N. Nguyen

*Institute for Infocomm Research, Agency for Science, Technology and Research (A*STAR), Singapore*

ACRONYMS

AC	Accuracy
ADL	Activities of daily living
AF	Average F-measure
CNN	Convolutional neural network
CPU	Central processing unit
DBN	Deep belief network
DT	Decision tree
HA	Hand Gesture
HAR	Human activity recognition
KNN	K-nearest neighbors
LSTM	Long- and short-term memory
MV	Means and variance
NB	Naive Bayes
NF	Normalized F-measure
OAR	Opportunity activity recognition
RAM	Random access memory
ReLU	Rectified linear unit
REALDISP	Real displacement
RNN	Recurrent neural network
SVM	Support vector machine

1 INTRODUCTION

Automatically recognizing a human's physical activities which is commonly referred to as human activity recognition (HAR) has emerged as a key area of interest in several sectors such as the sports and entertainment sectors, office scenarios, and health care. For example, monitoring daily activities in supporting medical diagnosis, assisting patients with chronic impairments, etc., are the key enhancements to traditional medical diagnosis methods. In addition, a vast exploration of

[a]Both authors contributed equally to this work.

Big Data Analytics for Sensor-Network Collected Intelligence. http://dx.doi.org/10.1016/B978-0-12-809393-1.00009-X

related human activities made its debut as key components in several consumer products, such as the Nintendo Wii and the Microsoft Kinect which rely on the recognition of gestures or even full-body movements to fundamentally enhance an immersive gaming experience. All of these useful application examples highlight the significance of HAR in both academia and industry.

Various investigations were performed to study the recognition of human gestures and activities from images and videos in constrained environments or stationary settings [1, 2]. Advances in sensor technologies have given rise to more potential application areas for activity recognition beyond controlled indoor settings and they promise to provide smart assistance and interfaces virtually anywhere and at any time by observing activities from the perspective of users. The first feasibility studies on activity recognition using body-worn sensors [3] dealt with fairly arbitrarily chosen activities, which were not always relevant to real-world applications. To our best knowledge, there are no unified signal processing and classification frameworks for activity recognition that satisfy application and user requirements for scalability and ease of deployment. On the other hand, with the greater availability of data, the motivation to develop activity recognition systems for more challenging and application-oriented scenarios is greater than ever before. Therefore, in this work, we propose an effective and efficient system that is able to recognize a human's physical activities automatically through analysis of the signals acquired (in real time) from multiple body-worn (or body-embedded) inertial sensors.

In the past few years, body-worn sensor-based HAR has made promising progress in applications such as game consoles, personal fitness training, medication intake, and health monitoring. An excellent survey on this topic can be found at Bulling et al. [1]. The HAR signals used in the application acquired by on-body sensors are arguably favorable over the signals acquired by video cameras, due to several reasons: (i) On-body sensors alleviate the limitations of environment constraints and stationary settings that video cameras often suffer from Bulling et al. [1], Ji et al. [4], and Le et al. [5]; (ii) Multiple on-body sensors allow more accurate and effective deployment of signal acquisition sensors on the human body; and (iii) On-body sensors enjoy the advantages of information privacy, as their acquired signals are target-specific while the signals acquired by camera may also contain the information of other nontarget subjects in the scene.

On the other hand, sensor-based HAR systems can suffer several possible issues. One issue is intraclass variability, which is common in experiments done with multiple human subjects. As each subject may perform the same activity with somewhat different movements, it is often difficult for HAR systems to find the commonalities in such cases. Another issue that arises has to do with interclass similarity. Depending on the recognition requirements, some activities may be very similar to be distinguished effectively. As an example, jogging and running share considerable similarity in terms of the motion of human body parts, but may be required to be classified as distinct activities. Besides, class imbalance can also be a problem for training HAR systems, especially in data collected from loosely constrained settings [6, 7]. In such cases, it often happens that certain classes occur much more rarely, yielding fewer training data samples. For instance, in daily life, walking generally occurs far more frequently than running. Data collected from a human subject over the course of such a day will have more samples from the walking activity as compared to the running activity. In addition, annotation of the collected data can also be an issue. It is often difficult to pinpoint precisely when an activity has started and ended. Given the generally high sampling rate of sensors (tens of samples per second),

this can result in considerable amounts of incorrectly annotated data. Sensors can also suffer from inherent variability depending on environmental conditions, and degrade over time which poses a problem for HAR systems.

The key factor attributed to the success of a HAR system is to find an effective representation of the time series collected from the on-body sensors. Though considerable research has been dedicated to investigating this issue, diminishing returns have been obtained. Conventionally, the HAR problem is often taken as one of specific applications of time series analysis. The widely used features in HAR include basis transform coding (e.g., signals with wavelet transform and Fourier transform) [8], statistics of raw signals (e.g, mean and variance of time sequences) [1, 9], and symbolic representation [10]. Although these features are widely used in many time series problems, they are heuristic and not task-dependent. It is worth noting that the HAR task has its own challenges as mentioned above, such as intraclass variability, interclass similarity, the NULL-class dominance (most sensor data arising from activities outside the scope of interest), and the complexity and diversity of physical activities [1]. All these challenges make it highly desirable to develop a systematic feature representation approach to effectively characterize the nature of signals relative to the activity recognition task.

Recently, deep learning has emerged as a family of learning models that aim to model high-level abstractions in data [11, 12]. In deep learning, a "deep" architecture with multiple layers is built up for learning highly discriminative feature representations. Specifically, each layer in a deep architecture performs a non-linear transformation on the outputs of the previous layer, so that the deeply learned model can represent the data by a hierarchy of features that can be interpreted as increasing in complexity from low-level to high-level. The well-known deep learning models include CNNs, deep belief networks (DBNs), and autoencoders. Depending on the usage of label information, the deep learning models can be learned in either a supervised or an unsupervised manner. Though deep learning models have achieved remarkable results in computer vision, natural language processing, and speech recognition, they have not been fully exploited in the field of HAR.

In this chapter, we tackle the HAR problem by adapting one particular deep learning model—the convolutional neural network (CNN) [13]. We build on the work of Yang et al. [13] by providing a more comprehensive experimental evaluation, a deeper analysis of the methods and issues involved with HAR and a more in-depth explanation of the choice of our CNN architecture. A CNN is constructed by stacking different processing units (e.g., convolution, pooling, sigmoid/hyperbolic tangent squashing, rectifier, normalization) in a repetitive manner. This stacking can yield an effective representation of local portions of the signals. Since the deep architecture allows multiple layers of these processing units to be stacked, it can characterize progressively wider portions of the signals by building on the representations learned in the lower layers. Therefore, the features extracted by the CNN are task dependent and nonhandcrafted. Moreover, these features also possess more discriminative power, since the CNN can be trained under the supervision of output labels. All these advantages of the CNN will be further elaborated in the following sections.

As detailed in the following sections, in the application on HAR, the convolution and pooling filters in the CNN are applied along the *temporal* dimension for each sensor, and all these feature maps for different sensors need to be unified as a common input for the neural network classifier. Therefore, a new architecture of the CNN is developed in this paper. In the experiments, we performed an extensive

study on the comparison between the proposed method and the state-of-the-art methods on benchmark datasets. The results show that the proposed method is a very competitive algorithm for the HAR problems. We also investigate the efficiency of the CNN and conclude that the CNN is fast enough for online HAR.

2 MOTIVATIONS AND RELATED WORK

It is highly desirable to develop a systematic and task-dependent feature extraction approach for HAR. Though the signals collected from wearable sensors are time series, they are different from other time series like speech signals and financial signals. Specifically, in HAR, only a few parts of the continuous signal stream are relevant to the concept of interest (i.e., human activities), and the dominant irrelevant part mostly corresponds to the null activity. Moreover, depending on the sensor setup, not all the sensors are relevant to distinguishing each activity. Finally, considering how human activity is performed in reality, we observe that every activity is a combination of several basic continuous movements. Typically, a human activity could last a few seconds in practice, and only a few basic movements could be involved over any given duration in this window. From the perspective of sensor signals, the basic continuous movements are more likely to correspond to smoothly changing signals, and the transitions among different basic continuous movements may cause abrupt, significant changes in the signal values. These properties of signals in HAR require the feature extraction method to be effective enough to capture the nature of basic continuous movements as well as the salience of the combination of basic movements.

As such, we are motivated to build a deep architecture of a series of signal processing units for feature extraction. This deep architecture consists of multiple shallow architectures, and each shallow architecture is composed by a set of linear/nonlinear processing units on locally stationary signals. When these shallow architectures are stacked, the salience of signals in different scales is captured. This deep architecture is not only for decomposing a large and complex problem into a series of small problems, but more importantly for obtaining specific "representations" of signals at different scales. Here, the signal representations reflect the salient patterns of signals. As stated in Bengio [11], what matters for generalization of a learning algorithm is the number of such representations of signals we wish to obtain after learning.

In contrast to this, the traditional features extraction methods such as basis transform coding (e.g., signals with wavelet transform and Fourier transform) [8], statistics of raw signals (e.g., mean and covariance of time sequences) [1], and symbolic representation [10] are deemed to play a comparable role of transforming the data by one or a few of neurons in one layer of a deep learning model. Another type of deep learning model, called DBN [14–16], was also investigated for HAR by Plätz et al. [2]. However, this feature learning method does not employ the effective signal processing units (like convolution, pooling, and rectifier) and also neglects the available label information in feature extraction. The primary use of the CNN mainly lies in 2D image [17, 18], 3D videos [4], and speech recognition [19]. However, in this paper, we attempt to build a *new* architecture of the CNN to handle the unique challenges present in HAR. The most related work is Zeng et al. [20], in which a shallow CNN is used and the HAR problem is restricted to the accelerometer data.

3 CONVOLUTIONAL NEURAL NETWORKS IN HAR

CNNs have great potential to identify the various salient patterns in the sensor signals used for HAR. Specifically, the processing units in the lower layers obtain the local salience of the signals (to characterize the nature of each basic movement in a human activity). The processing units in the higher layers obtain the salient patterns of signals at high-level representations (to characterize the nature of a combination of several basic movements). Note that each layer may have a number of convolution or pooling operators (often with different input and output sizes) as described below, so multiple salient patterns learned from different sections of the signals are jointly considered in the CNN. When these operators with the same parameters are applied on local signals (or their representations) at different time segments, a form of translation invariance is obtained [11, 12, 21]. Consequently, what matters is only the salient patterns of signals instead of their positions or scales. However, in HAR we are confronted with multiple channels of time series signals, in which the traditional CNN cannot be used directly. The challenges in our problem include (i) processing units in CNN need to be applied along temporal dimension and (ii) sharing or unifying the units in CNN among multiple sensors. In what follows, we will define the convolution and pooling operators along the temporal dimension, and then presented the entire architecture of the CNN used in HAR.

We start with the notations used in the CNN. A sliding window strategy is adopted to segment the time series signal into a collection of short segments of signals. Specifically, an instance used by the CNN is a two-dimensional matrix containing r raw samples (each sample with D attributes). Here, r is chosen to be as the sampling rate, and the step size of sliding a window is chosen according to the experimental setup. In general, a CNN requires much more training data than conventional, shallower learning architectures due to the significantly larger number of free parameters involved in the learning process. A smaller step size increases the amount of training instances, but introduces greater redundancy between different instances which may not have an equitable impact in improving test-time performance for the CNN.

For training data, the true label of the matrix instance is determined by the most-frequently occurring label for r raw records. For the jth feature map in the ith layer of the CNN, it is also a matrix, and the value at the xth row for sensor d is denoted as $v_{ij}^{x,d}$ for convenience. This is explained as shown in Fig. 1. For convenience, we show the feature maps for the first convolutional layer. The $r \times D$ input matrix is convolved with J filters of length k along the temporal dimension. This gives rise to J feature maps of size $(r - k + 1) \times D$, each element of which can be referenced by the term $v_{ij}^{x,d}$.

3.1 TEMPORAL CONVOLUTION AND POOLING

In the convolution layers, the previous layer's feature maps are convolved with several convolutional kernels (to be learned in the training process). The output of the convolution operators along with a bias (to be learned) is passed through an activation function to form the feature map for the next layer. Formally, the value $v_{ij}^{x,d}$ is given by

$$v_{ij}^{x,d} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1} w_{ijm}^{p}\, v_{(i-1)m}^{x+p,d}\right), \quad \forall d = 1,\ldots,D \tag{1}$$

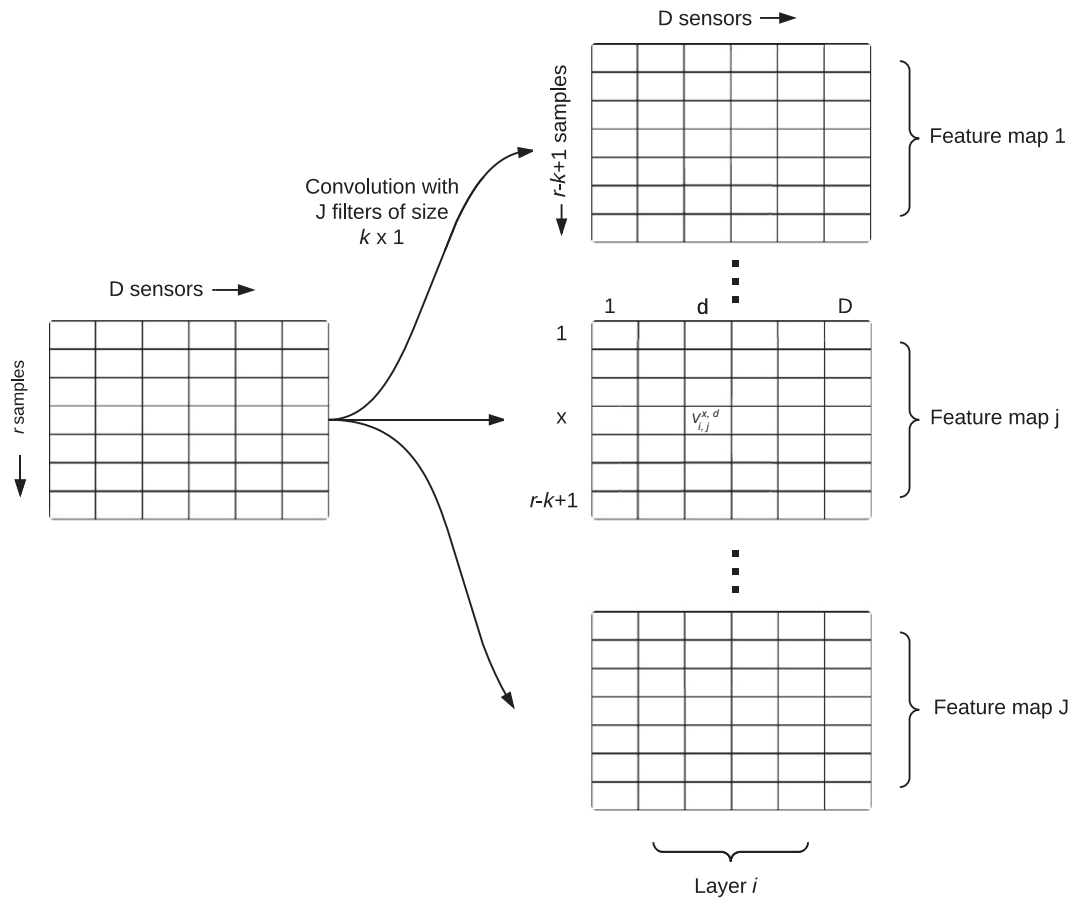

FIG. 1

Illustration of the computation of $v_{i,j}^{x,d}$ at an intermediate layer of the CNN.

where $\tanh(\cdot)$ is the hyperbolic tangent function, b_{ij} is the bias for this feature map, m indexes over the set of feature maps in the $(i-1)$th layer connected to the current feature map, w_{ijm}^{p} is the value at the position p of the convolutional kernel, and P_i is the length of the convolutional kernel.

In the pooling layers, the resolution of feature maps is reduced to increase the invariance of features to distortions on the inputs. Specifically, feature maps in the previous layer are pooled over local *temporal* neighborhood by either max pooling function

$$v_{ij}^{x,d} = \max_{1 \le q \le Q_i} \left(v_{(i-1)j}^{x+q,d} \right), \quad \forall d = 1, \ldots, D \tag{2}$$

or a sum pooling function

$$v_{ij}^{x,d} = \frac{1}{Q_i} \sum_{1 \le q \le Q_i} \left(v_{(i-1)j}^{x+q,d} \right), \quad \forall d = 1, \ldots, D \tag{3}$$

where Q_i is the length of the pooling region.

3.2 ARCHITECTURE

Based on the operators introduced in the above section, we construct a CNN shown in Fig. 2. For convenience, all layers of the CNN can be grouped into five sections. For the first two sections, each section consists of (i) a convolution layer that convolves the input or the previous layer's output with a set of kernels to be learned; (ii) a rectified linear unit (ReLU) layer that maps the output of the previous layer by the function $\text{relu}(v) = \max(v, 0)$; (iii) a max pooling layer that finds the maximum feature map over a local *temporal* neighborhood (a subsampling operator is often involved); (iv) a normalization layer that normalizes the values of different feature maps in the previous layer $v_{ij} = v_{(i-1)j} \left(\kappa + \alpha \sum_{t \in G(j)} v_{(i-1)t}^2 \right)^{-\beta}$, where κ, α, β are hyper-parameters and $G(j)$ is the set of feature maps involved in the normalization.

The use of ReLU units has benefits related to reducing the likelihood of vanishing gradients, which speeds up learning, as well as the introduction of sparsity that helps prevent overfitting. Max pooling helps introduce some temporal translational invariance. Normalization ensures that the distribution of the outputs of each layer does not change too much, which is important for not slowing down learning.

For the third section, it is only constituted of a convolution layer, ReLU layer and a normalization layer, as after the convolution layer the temporal dimension of a feature map becomes one (noting that the size of a feature map output by this layer is $D \times 1$) so the pooling layer is avoided here. For the fourth section, we aim to unify the feature maps output by the third section among all D sensors. Instead of simply concatenating these feature maps, we use a fully connected layer to unify them to achieve the *parametric concatenation* in this layer. An illustrative diagram is shown in Fig. 3. This scheme appears to lead to an explosion in the number of parameters at this layer (in total $d \times 20 \times 400$ parameters). Following the same strategy used in the previous convolution layers, we use different weights for d sensors but share these weights among hidden units in the third layer, as shown in Fig. 3. Mathematically, the value of the jth feature map in this layer is computed by:

$$v_{ij} = \tanh \left(b_{ij} + \sum_m \sum_{d=1}^{D} w_{ijm}^d v_{(i-1)m}^d \right) \tag{4}$$

and this unification is also followed by the ReLU layer and normalization layer.

The fifth section is a fully connected network layer. This layer is same as a standard multilayer perceptron neural network that maps the latent features into the output classes. The output of this layer is governed by the softmax function. This softmax function provides the posterior probability of the classification results. Then, an entropy cost function can be constituted based on the true labels of training instances and probabilistic outputs of softmax function.

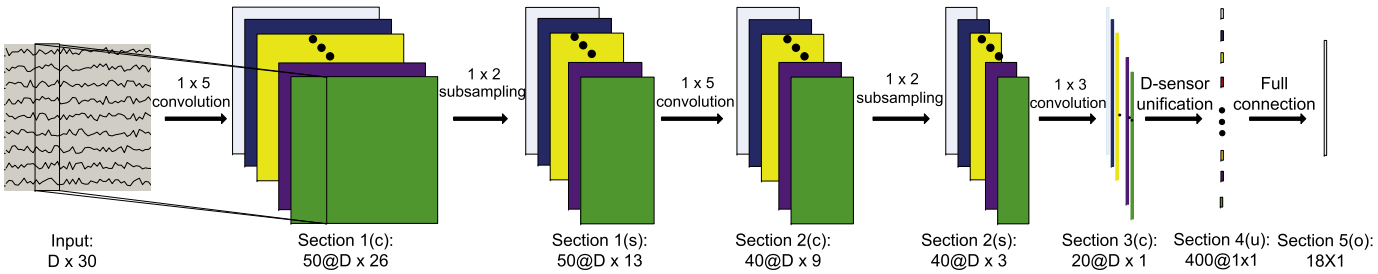

FIG. 2

Illustration of the CNN architecture used for a multisensor based human activity recognition problem. We use the Opportunity Activity Recognition dataset presented in Section 4 as an illustrative example. The *symbols* "c," "s,""u," and "o" in the parentheses of the layer tags refer to convolution, subsampling, unification, and output operations, respectively. The *numbers before and after "@"* refer to the number of feature maps and the dimension of a feature map in this layer. Note that pooling, ReLU and normalization layers are not shown due to the limitation of space.

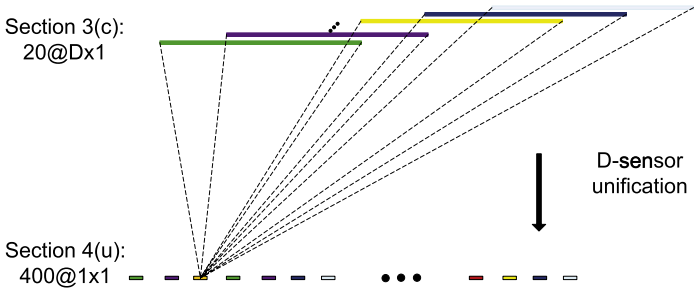

FIG. 3

Illustration of unification layer (i.e., the fourth section in Fig. 2).

$$v_{ij} = \frac{\exp(v_{(i-1)j})}{\sum_{j=1}^{C} \exp(v_{(i-1)j})} \tag{5}$$

where C is the number of output classes.

To convert the matrix-level prediction given by the CNN to originally desired sample-level predictions, the following two steps are used. First, all the samples in a matrix-level instance are labeled by the same predicted label for this matrix-level instance. Second, for a sample lying in the overlapped matrix-level instances, a voting method is used to determine the final predicted label of this sample.

Due to the temporal dependence of sensor signals, the labels of instances often have a smooth trend, as mentioned in Section 2. Recently, Cao et al. [22] have proposed a simple but effective smoothing method to postprocess the predicted labels so as to enhance the prediction performance. The idea is to employ a low-pass filter to remove the impulse noise (potential wrong prediction) and maintain the edges (i.e., the position of activity transition). Specially, for the ith instance, a smoothing filter with a predefined length u_i is applied on the sequence whose center is the ith instance. This filter finds the most frequent label in this sequence and assign it to the ith instance. We shall investigate the prediction results with/without this smoothing method in the experiments.

3.3 ANALYSIS

Note that the ReLU and normalization layers are optional in the first four layers of the CNN in Fig. 2. In our experiments, we found that incorporating these two layers can lead to better results. Furthermore, to avoid overfitting, dropout operations [23] and other regularization methods might be employed in the CNN, though they are not used in our experiments due to the resultant minor performance difference.

Remark 1. The conventional CNN [4, 17, 24] used in the image/video case does not have the unification layer shown in Fig. 3, because the image/video signal is considered to come from a single sensor channel. Thus, the proposed architecture of the CNN is a generalization of the conventional CNN by considering multiple channels of data.

In the CNN, the parameters in all processing units and connection weights are jointly learned through a global objective function (i.e., entropy cost function). In other words, this is an optimization to be performed over a large number of unknown variables. This global objective function can be efficiently optimized by a so-called back-propagation algorithm [25]. The idea of this algorithm is using

the chain rule to derive the gradients of the cost function with respect to the variables in the CNN and then updating these variables by a simple gradient descent method. To speed up the algorithm, a stochastic gradient descent method is often adopted in updating the variables. Investigation on the efficiency of parallel computing of the back-propagation algorithm has attracted considerable research efforts and excellent work along this direction can be found at Jia et al. [26] and Donahue et al. [27].

Remark 2. The global objective function is related to the training error that is computed based on the ground truth labels as well as the outputs of the softmax function in the last layer of the CNN. This function's variables control the various feature maps of the signals. Consequently, through the optimization model, the two tasks of feature learning and classification are mutually enhanced, and the learned features by the CNN have more discriminative power with respect to the ultimate classification task.

4 EXPERIMENTS, RESULTS, AND DISCUSSION

In the experiments, we consider three datasets for HAR with different focuses. The first dataset—the Opportunity Activity Recognition (OAR) dataset—is related to the entire body's movement, the second dataset—the Hand Gesture dataset—particularly focuses on the hand's movement, while the third dataset—the REALDISP dataset—focuses on the impact of wearer-introduced sensor displacement on classification frameworks trained under ideal scenarios. The architecture of the CNN used in the OAR dataset is shown in Fig. 2. The detailed descriptions of the three data sets will be introduced later in this section. The same architecture of the CNN is used for the other two datasets with the only differences in the number of feature maps and the sizes of convolution kernels, since the dimensions of the input and output of the datasets are different. In the normalization operator of the CNN, the parameters are chosen as $\kappa = 1$, $\alpha = 2 \times 10^{-4}$, $\beta = 0.75$ and the size of $G(\cdot)$ is 5 in all experiments. We follow the rules of thumb shown in LeCun et al. [25] to choose other parameters, as how to find the optimal parameters in CNN is still an open question.

4.1 EXPERIMENT ON OAR DATASET

The OAR dataset[1] [22, 28, 29] is about the human activities related to a breakfast scenario. This dataset contains the data collected from the sensors configured on three subjects who perform activities of daily living (ADL). There are 18 classes in this activity recognition task.[2] The Null class refers to the either nonrelevant activities or nonactivities. The used sensors include a wide variety of body-worn, object-based, and ambient sensors—in total, 72 sensors from 10 modalities—with 15 wireless and wired sensor network in home environment. The sampling rate of the sensor signals is 30 Hz. Each record is comprised of 113 real-valued sensory readings excluding the time information. With these sensors, each subject performed one drill session (Drill) which has 20 repetitions of some predefined actions

[1] http://www.opportunity-project.eu/challenge.

[2] The 18 classes are `Null`, `open door 1`, `open door 2`, `close door 1`, `close door 2`, `open fridge`, `close fridge`, `open dishwasher`, `close dishwasher`, `open drawer 1`, `close drawer 1`, `open drawer 2`, `close drawer 2`, `open drawer 3`, `close drawer 3`, `clean table`, `drink cup`, and `toggle switch`.

in one sequence of sensory data, and five ADLs. Following Cao et al. [22], we use Drill and first two sets of ADLs as the training data, and use the third set of ADL as the testing data.

We further remove the time series from six faulty sensors for Subject 1 and also remove the time series from three faulty sensors for Subjects 2 and 3 as suggested in Cao et al. [22]. All these faulty sensors have more than 30% of missing data. For other missing sensor data, we fill the missing values using cubic spline fitting in the temporal dimension. We use a step size of 3 for the training data, and 1 for the testing data in order to have better resolution in the latter case. The network is trained for eight epochs with a learning rate profile of 0.1 for the first two epochs, 0.01 for the next three, and 0.001 for the last three. While a constant learning rate (say, 0.01) can also be used, it is generally found that the number of epochs required in this case is much larger. This would necessitate a much longer training time for a large dataset such as this one. We do demonstrate the use of a constant learning rate for the REALDISP dataset below. The number of epochs and the learning rate profile is determined by using a validation set held out from the training data, and the network is retrained with the entire training data, once the appropriate parameters are determined.

We compare the proposed method with the following four baselines, namely support vector machine (SVM) [22], K-nearest neighbors (KNN) classifier [22, 30], means and variance (MV) [1], and DBN [2]. Among them, the first two methods and the third method show the state-of-the-arts results on the OAR dataset and Hand Gesture datasets, respectively. The fourth method is a recently developed deep learning method for HAR.

- *SVM* with radial basis function (RBF) kernel is used as the classifier. In this baseline, the raw time series samples are directly used as the input of SVM. Additionally, the cross-validation procedure is used to tune the parameters of SVM.
- *KNN* performed a comprehensive empirical evaluation on time series classification problems. Interestingly, the simple technique KNN (specifically, 1NN, i.e., classification based on the top one nearest neighbor) with Euclidean distance was shown to be the best technique. Therefore, we incorporate the KNN with $K = 1$ as the classifier. Same as the SVM baseline, the raw time series samples are directly used as the input of KNN.
- *MV*, which is the same as the proposed CNN method, the sliding window strategy is used to generate a set of $r \times D$ matrix-level instances first. Then the mean and the variance of the signals over the r samples in every $r \times D$ matrix are extracted to constitute the features of the input data for the classifier. The classifier used is the KNN with $K = 1$.
- *DBN*, similar to the CNN and MV methods, a set of $r \times D$ matrix-level instances are generated first. Then, the mean of the signals over the r samples in every $r \times D$ matrix is used as the input of the DBN.[3] The classifier used in this method is chosen between KNN with $K = 1$ and a multilayer perceptron neural network, and the one with better performance is reported.

For MV and DBN methods, matrix-level predictions are converted to the sample-level predictions based on the same strategy used in the CNN method as introduced in Section 3.2. We evaluate all methods' performance under the both settings of with/without the smoothing method mentioned in Section 3.2. As suggested in Cao et al. [22], the parameter u_i in the smoothing method is recommended to be chosen in the range of [60, 100].

[3]The experiment of using the raw inputs as the features in DBN is also performed, but the results are substantially worse than the reported one.

Table 1 AF, NF, and AC Results of the Proposed CNN Method and Four Baselines for the OAR Dataset

	Subject 1			Subject 2			Subject 3		
	AF	**NF**	**AC**	**AF**	**NF**	**AC**	**AF**	**NF**	**AC**
	Without smoothing								
SVM (quoted from Cao et al. [22])	45.6	83.4	83.8	44.4	75.6	79.4	32.1	76.8	78.1
1NN (quoted from Cao et al. [22])	42.7	80.3	79.3	41.1	73.5	73.9	28.5	67.5	63.8
MV	54.2	83.9	83.7	50.8	74.6	74.3	48.6	80.9	80.7
DBN	14.3	75.0	80.0	7.0	66.7	74.1	20.0	73.4	79.3
CNN	**55.5**	**86.4**	**87.0**	**57.1**	**79.5**	**82.5**	**55.8**	**84.0**	**85.8**
	With smoothing								
SVM	48.6	84.7	85.9	43.8	75.9	80.4	27.6	76.7	79.2
1NN	53.9	84.1	84.6	53.2	78.2	79.8	34.0	71.8	69.7
MV	47.9	83.3	84.3	54.3	75.7	75.7	49.6	81.9	82.1
DBN	12.9	74.5	79.5	7.3	66.9	74.9	21.1	73.0	79.7
CNN	**51.6**	**86.5**	**87.7**	**60.0**	**79.7**	**83.0**	**52.6**	**84.7**	**86.7**

Notes: *The best result for each metric is highlighted in bold.*
AC, accuracy; AF, average F-measure; NF, normalized F-measure.

The results of the proposed CNN method and the four baseline methods on OAR dataset are shown in Table 1. Following Cao et al. [22], average F-measure (AF), normalized F-measure (NF), and Accuracy (AC) are used to evaluate the performance of different methods in all experiments. The best performance for each evaluation metric is highlighted in bold.

From the results, we can see that the proposed CNN method consistently performs better than all four baselines in both settings of with/without the smoothing strategy on both datasets in terms of all three evaluation metrics. Remarkably, for Subject 3 in the first dataset and Subject 2 in the second dataset, the proposed method outperforms the best baseline by 5% or so in terms of accuracy both in the absence as well as presence of smoothing. When the smoothing strategy is used, the performance of all methods generally is improved, but the performance ranking of all methods remains almost invariant. The class imbalance issue is a main challenge for all methods. This can be seen from the confusion matrix generated by the proposed CNN method shown in Fig. 4. Due to the dominant `Null` class, all signals samples, except for the ones in class `closedrawer2`, tend to be classified into the `Null` class. Similar phenomena caused by the class imbalance issue exist in all methods but they are more severe for the other baseline methods.

The better performance of the CNN over DBN demonstrates that the supervised deep learning outperforms the unsupervised one for HAR. This observation has also been seen in other applications like image classification and speech recognition. Note that SVM and KNN use the raw instances in this paper while in Plätz et al. [2] they use the matrix-level instances whose amount is smaller than that of the raw instances. This may explain why DBN is a bit worse than SVM and KNN in our experiments while it is slightly better than SVM and KNN in the experiments shown in Plätz et al. [2]. The evidence

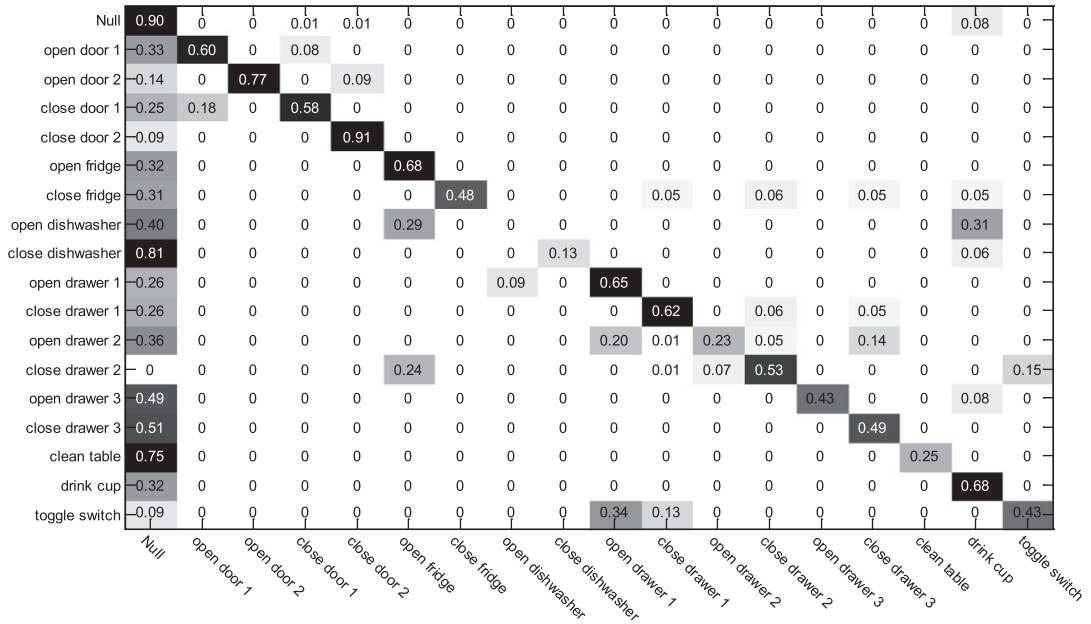

FIG. 4

Confusion matrix yielded by the proposed CNN method (without temporal smoothing) on the Opportunity Activity Recognition dataset for Subject 1 (the larger the value, the *darker the background*).

that the CNN has the better performance than SVM, KNN, and MV suggests that the CNN is closer to finding the nature of signals in terms of feature representation than the methods with shallowing learning architectures and heuristic feature designs for the HAR problem.

4.2 EXPERIMENT ON HAND GESTURE DATASET

The Hand Gesture dataset [1][4] is about different types of the human's hand movements. In this dataset, two subjects perform hand movements with eight gestures in daily living and with three gestures in playing tennis. In total, there are 12 classes in this hand gesture recognition problem.[5] Similar to the first dataset, the Null class refers to the periods with none of the predefined activities. The used body-worn sensors include a three-axis accelerometer and a two-axis gyroscope, and the sampling rate is 32 samples per second. Then, each record has 15 real-valued sensor readings in total. Every subject repeated all activities about 26 times. We randomly select one repetition as the testing data and the rest repetitions as the training data. The parameters for the CNN used for this dataset are obtained in a similar manner as the OAR dataset, and have similar values.

[4] https://github.com/andyknownasabu/ActRecTut.

[5] The 12 classes are `Null`, `open a window`, `close a window`, `water a plant`, `turn book page`, `drink from a bottle`, `cut with a knife`, `chop with a knife`, `stir in a bowl`, `forehand`, `backhand`, and `smash`.

Table 2 The AF, NF, and AC Results of the Proposed CNN Method and Four Baselines for the Hand Gesture Dataset

	Subject 1			Subject 2		
	AF	**NF**	**AC**	**AF**	**NF**	**AC**
	Without smoothing					
SVM	76.0	85.0	85.6	71.1	83.5	82.6
1NN	64.8	73.2	71.8	66.2	79.3	77.9
MV	87.5	91.3	91.2	84.1	90.1	89.3
DBN	71.8	82.1	82.8	69.0	81.4	80.1
CNN	**89.2**	**92.0**	**92.2**	**90.7**	**95.0**	**95.0**
	With smoothing					
SVM	85.1	89.2	89.6	86.0	89.3	88.5
1NN	**92.2**	93.3	93.2	86.1	89.8	89.2
MV	91.5	93.3	93.3	84.4	90.5	89.6
DBN	78.5	84.9	85.8	73.2	83.4	82.0
CNN	**92.2**	**93.9**	**94.1**	**87.0**	**95.5**	**96.0**

Similar to the results for the OAR dataset, we present the results on the Hand Gesture dataset in Table 2. Again, the CNN is able to consistently outperform the various baselines with Subject 2 showing a performance margin of about 5% in terms of accuracy both with and without smoothing.

We also conducted the experiments that the magnitudes of Fourier Transform of the raw data are taken as the inputs for all methods. However, no performance improvement can be made. Similar results have been observed in Cao et al. [22].

4.3 EXPERIMENT ON REALDISP DATASET

The REALDISP dataset [31, 32][6] is a HAR dataset that focuses on identifying a wide range of 33 different activities[7] across 17 different subjects. The data capture setup using 39 tri-axial accelerometers worn on the subject's body, under three different scenarios.

In the first scenario called the ideal displacement scenario, the dataset creators placed the various sensors precisely on the subjects' bodies. In the second scenario called the self-displacement scenario, the subjects were allowed to place the sensors themselves, after having been given general guidelines

[6] https://archive.ics.uci.edu/ml/datasets/REALDISP+Activity+Recognition+Dataset.

[7] The categories include `Walking, Jogging, Running, Jump Up, Jump Front & Back, Jump Sideways, Jump Legs/Arms open/closed, Jump Rope, Trunk Twist (Arms Outstretched), Trunk Twist (Elbows Bent), Waist Bends Forward, Waist Rotation, Waist Bends (Foot Reach with other Hand), Reach Heels Backward, Lateral Bend, Lateral Bend with Arm Up, Repetitive Forward Stretching, Upper Trunk and Lower body opposite twist, Lateral elevation of arms, Frontal elevation of arms, Frontal hand claps, Frontal crossing of arms, Shoulders high-amplitude rotation, Shoulders low-amplitude rotation, Arms inner rotation, Knees to the breast, Heels to the backside, Knees bending, Knees bending forward, Knee Rotation, Rowing, Elliptical bike, and Cycling`.

about which body part each sensor was supposed to be placed on. In the third scenario called the induced displacement scenario, the dataset creators placed multiple sensors with predefined displacements from their ideal settings in order to analyze performance in a controlled setting.

In this work, we use data only from the first two scenarios as they most closely correspond to a real-world usage example. Specifically, the algorithm is trained by using data from the ideal displacement scenario but tested on the self-displacement scenario. In other words, we demonstrate the impact on performance that moving away from an ideal laboratory setting has on various HAR algorithms.

The REALDISP dataset differs from the previous two datasets in that the NULL activity is not a label that is used for evaluating HAR performance. It is assumed by the dataset creators in their evaluation setup that the sensor data will be predivided into NULL and non-NULL sections. We follow the same methodology in our experiment and report the results for HAR using non-NULL activities. We compare our proposed experiment with the results from Baños et al. [32] using various methodologies:

- *Single sensor*: HAR performance is evaluated by using a single sensor at a time as the input to the system, and the reported performance is the average across the performance of all the sensors, taken one at a time.
- *Feature fusion*: HAR performance is evaluated by using a framework that unifies all the sensor inputs before passing the fused inputs to a classifier.
- *Decision fusion*: HAR performance is evaluated based on fusing together the decisions obtained from the Single Sensor scenario above, but using a hierarchical weighted classifier to assign weights to the decision of each sensor based on its performance on a validation set.

All three methodologies are evaluated using three widely used classifiers—C4.5 Decision Trees (DT) [33], KNN [34] with $k = 3$, and Naive Bayes (NB) [35]. The sensor signals are processed prior to being fed into the HAR framework. In general, the best performance is obtained when a relatively rich set of hand-crafted features consisting of the mean, standard deviation, maximum, minimum, and mean crossing rate (effective frequency) are used.

The work of Baños et al. [32] uses a 6-second sample window with no overlap in their evaluation. The length of the time window is chosen "to sufficiently capture cycles in activities" [36]. We too choose to use windows with no overlap; however, this presents a problem for using a CNN in the form of very limited data for training. On average, the REALDISP dataset contains about 13 minutes of non-NULL activity in the ideal displacement setting (training data) and about 14 minutes of such activity in the self-displacement setting (testing data) for each subject. Combined with missing data (due to sporadically faulty sensors), this translates to less than 2000 six-second windows for 33 activities for training, which results in overfitting for the CNN.

In order to circumvent this problem, we choose to use 1-second windows with no overlap to augment the data. Using a shorter-time window should result in several cycles not being captured in the data, which should make the HAR problem even more challenging for our proposed method. Indeed, several of the activities present in the dataset (such as `Elliptical Bike` and `Cycling`) can share many of the same basic movements. Despite this potential limitation, we show below that our method outperforms all the compared techniques.

A constant learning rate profile of 0.01 for 25 epochs is used for training the CNN used for this dataset, again determined by using a held-out validation set. While not directly comparable with the CNN trained on the OAR dataset (due to different dataset complexities), it can be seen that in general, a constant learning rate requires considerably more epochs than a variable learning rate profile.

Table 3 Accuracy of the Proposed CNN Method and Nine Baselines on the REALDISP Dataset for Self-Displaced Sensors

Method	DT	KNN	NB
Single sensor [32]	36.6	46.0	45.3
Feature fusion [32]	50.7	79.3	60.0
Decision fusion [32]	76.9	88.0	63.3
CNN		**90.1**	

Table 4 Accuracy of the Proposed CNN Method With Various Smoothing Methods

Smoothing Method	Accuracy
None	90.1
Blockwise	92.8
Median	91.4

The performance of our proposed method is shown in Table 3. It can be seen that the CNN-based HAR system outperforms the compared baseline methods, despite the limitation of working with shorter-time windows. This is presumably because the CNN is able to discover better feature representations than the hand-crafted features used by the compared baselines.

It is also interesting to note the effect of smoothing on the output of the proposed CNN. As NULL activities are not considered in this dataset, the smoothing strategy is slightly different from that explained earlier. Specifically, one may assume that every contiguous block of non-NULL activity corresponds to the same activity. This may be a reasonable assumption whenever there is a pause between activities. In such a case, the most frequently occurring activity in the block is assigned to all the data samples present in the block. Alternatively, one can consider each contiguous non-NULL block as composed of multiple non-NULL activities. In this case, median filtering can be applied within each block to remove singly misclassified observations. Note that an "observation" in this case corresponds to a 1-second interval. Since the smallest contiguous non-NULL block in the data is of 7 seconds, we use 7 as the size of the median filter. We refer to the above two cases as "Blockwise" and "Median" smoothing, respectively, in Table 4.

It can be seen that using median smoothing gives an improvement of about 1.3% in accuracy. On the other hand, using blockwise smoothing more than doubles the improvement to 2.7%. This indicates that instances of multiple labels for a single contiguous block do not occur very frequently in the data. Thus, having prior knowledge of how the data is obtained can help with choosing the correct smoothing strategy. In either case though, smoothing does help boost performance as with the previous two datasets.

4.4 COMPUTATIONAL REQUIREMENTS

All experiments are conducted on a PC, which has an Intel i5-2500 3.30 GHz CPU and 8 GB RAM. We report the timing results of the CNN on the OAR dataset for Subject 1 as this dataset is the largest one in all experiments. The training and testing raw samples for this dataset are 136,869 and 32,466,

respectively, and the input dimension is 107. The training time of the CNN is around 1 hour, while the testing time is 8 minutes. On average, within a second the CNN can predict 56 raw instances' labels. Thus, the efficiency of the CNN is good enough for online HAR.

4.5 FUTURE DIRECTIONS

The training and testing times for CNNs can be significantly reduced when the parallel computation of the CNN [26, 27] is implemented. This research topic will be fully investigated in our future work.

Additionally, it will be interesting to see if deep learning architectures for sequential data such as recurrent neural networks (RNNs) [37], long short-term memory networks (LSTMs) [38], etc., will be useful when applied to the HAR problem. These have been extensively used in other sequential tasks such as speech recognition, where it is important to label a group of sound samples with the right word or sound (phoneme). However, the HAR problem cannot be directly mapped to a sequential problem, since each sample (rather than only a well-defined group of samples) possesses a ground-truth activity label. Nevertheless, the use of such sequential networks holds some promise in bypassing the need for the sliding window approach, which is a heuristic parameter used in CNNs.

5 CONCLUSION

In this chapter, we have proposed a new method to automate feature extraction for the HAR task. The proposed method builds a novel deep architecture for the CNN to investigate the multichannel time series data. This deep architecture mainly employs the convolution and pooling operations to capture the salient patterns of the sensor signals at different time scales. All identified salient patterns are systematically unified among multiple channels and finally mapped into the different classes of human activities. The key advantages of the proposed method are: (i) feature extraction is performed in a task-appropriate and nonhand-crafted manner; (ii) extracted features have more discriminative power w.r.t. the classes of human activities; and (iii) feature extraction and classification are unified in one model so their performances are mutually enhanced. In the experiments, we demonstrate that the proposed CNN method outperforms other state-of-the-art methods, and we therefore believe that the proposed method can serve as a competitive tool of feature learning and classification for the HAR problems.

REFERENCES

[1] Bulling A, Blanke U, Schiele B. A tutorial on human activity recognition using body-worn inertial sensors. ACM Comput Surv 2014;46(3):33:1–3.
[2] Plätz T, Hammerla NY, Olivier P. Feature learning for activity recognition in ubiquitous computing. In: IJCAI; 2012.
[3] Reddy S, Mun M, Burke J, Estrin D, Hansen M, Srivastava M. Using mobile phones to determine transportation modes. ACM Trans Sens Netw 2010;6(2):13:1–13:27.
[4] Ji S, Xu W, Yang M, Yu K. 3D convolutional neural networks for human action recognition. In: ICML; 2010.
[5] Le QV, Zou WY, Yeung SY, Ng AY. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis; 2011. p. 3361–8.

[6] Cao H, Li XL, Woon YK, Ng SK. Structure preserving oversampling for imbalanced time series classification. In: Proceedings of IEEE 11th international conference on data mining; 2011.
[7] Cao H, Li XL, Woon YK, Ng SK. Integrated oversampling for imbalanced time series classification. IEEE Trans Knowl Data Eng 2013;12(25):2809–22.
[8] Huynh T, Schiele B. Analyzing features for activity recognition. In: Proceedings of the 2005 joint conference on smart objects and ambient intelligence: innovative context-aware services: usages and technologies; 2005. p. 159–63.
[9] Nguyen MN, Li XL, Ng SK. Positive unlabeled learning for time series classification. In: Proceedings of the 22nd international joint conference on artificial intelligence; 2011.
[10] Lin J, Keogh E, Lonardi S, Chiu B. A symbolic representation of time series, with implications for streaming algorithms. In: SIGMOD workshop on research issues in data mining and knowledge discovery; 2003.
[11] Bengio Y. Learning deep architectures for AI. Found Trends Mach Learn 2009;2(1):1–127.
[12] Deng L. A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Trans Signal Inf Process 2014;3:1–29.
[13] Yang JB, Nguyen MN, San PP, Li XL, Krishnaswamy SP. Deep convolutional neural networks on multichannel time series for human activity recognition. In: Proceedings of the 24th international joint conference on artificial intelligence; 2015.
[14] Hinton GE, Osindero S. A fast learning algorithm for deep belief nets. Neural Comput 2006;18(7):1527–54.
[15] Le Roux N, Bengio Y. Representational power of restricted Boltzmann machines and deep belief networks. Neural Comput 2008;20(6):1631–49.
[16] Tieleman T. Training restricted Boltzmann machines using approximations to the likelihood gradient. In: ICML; 2008. p. 1064–71.
[17] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: NIPS; 2012. p. 1097–105.
[18] Zeiler M, Fergus R. Visualizing and understanding convolutional networks; 2014.
[19] Deng L, Li J, Huang JT, Yao K, Yu D, Seide F, et al. Recent advances in deep learning for speech research at Microsoft. In: ICASSP; 2013.
[20] Zeng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, et al. Convolutional neural networks for human activity recognition using mobile sensors. In: MobiCASE; 2014.
[21] Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol Cybern 1980;36(4):193–202.
[22] Cao H, Nguyen MN, Phua C, Krishnaswamy S, Li XL. An integrated framework for human activity classific. In: ACM international conference on ubiquitous computing; 2012.
[23] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 2014;15(1):1929–58.
[24] Wan J, Wang D, Hoi SCH, Wu P, Zhu J, Zhang Y, et al. Deep learning for content-based image retrieval: a comprehensive study. In: ACM MM; 2014. p. 157–66.
[25] LeCun Y, Bottou L, Orr G, Muller K. Efficient BackProp. In: Orr G, Muller K, editors. Neural networks: tricks of the trade; Springer; 1998. p. 9–50.
[26] Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, et al. Caffe: convolutional architecture for fast feature embedding. In: ACM MM; 2014. p. 675–8.
[27] Donahue J, Jia Y, Vinyals O, Hoffman J, Zhang N, Tzeng E, et al. DeCAF: a deep convolutional activation feature for generic visual recognition. In: ICML; 2014.
[28] Sagha H, Digumarti ST, del R. Millán J. Chavarriaga R, Calatroni A, Roggen D, et al. Benchmarking classification techniques using the opportunity human activity dataset. In: IEEE international conference on systems, man, and cybernetics; 2011.

[29] Roggen D, Calatroni A, Rossi M, Holleczek T, Förster K, Tröster G, et al. Collecting complex activity data sets in highly rich networked sensor environments. In: Proceedings of the seventh international conference on networked sensing systems (INSS), Kassel, Germany; 2010.
[30] Keogh E, Kasetty S. On the need for time series data mining benchmarks: a survey and empirical demonstration. In: SIGKDD; 2002. p. 102–11.
[31] Baños O, Damas M, Pomares H, Rojas I, Tóth MA, Amft O. A benchmark dataset to evaluate sensor displacement in activity recognition. In: Proceedings of the 2012 ACM conference on ubiquitous computing. ACM; 2012. p. 1026–35.
[32] Baños O, Toth MA, Damas M, Pomares H, Rojas I. Dealing with the effects of sensor displacement in wearable activity recognition. Sensors 2014;14(6):9995–10023.
[33] Duda RO, Hart PE, Stork DG. Pattern classification. Hoboken, NJ: John Wiley & Sons; 2012.
[34] Cover TM, Hart PE. Nearest neighbor pattern classification. IEEE Trans Inf Theory 1967;13(1):21–7.
[35] Theodoridis S, Koutroumbas K. Pattern recognition and neural networks. Machine learning and its applications. Springer; 2001. p. 169–95.
[36] Bao L, Intille SS. Activity recognition from user-annotated acceleration data. Pervasive computing. Berlin, Heidelberg: Springer; 2004. p. 1–17.
[37] Graves A. Supervised sequence labelling. Berlin, Heidelberg: Springer; 2012.
[38] Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput 1997;9(8):1735–80.

GLOSSARY

Cost function Objective function to be minimized in the model learning process.
Feature map A representation of data that can be used to find salient information.
Feature representation learning A method to automatically learn useful representations of raw data without the need for hand-crafted features.
Free parameters Parameters of the model to be learned.
Hyper-parameters Parameters used to control the learning process of the model.
Ideal displacement Scenario for the REALDISP dataset where sensors are placed in their ideal positions.
Induced displacement Scenario for the REALDISP dataset where sensors are offset from ideal positions by a fixed amount.
Learning rate Hyper-parameter that controls the step size in the objective minimization process.
NULL class A class of human activities that is outside the scope of interest of a given task.
Overfitting A process occurs when a model is excessively complex, having too many parameters relative to the number of observations.
Self-displacement Scenario for the REALDISP dataset where sensors are offset from ideal positions by subject-determined random amounts.
Sigmoid function A function is a bounded differentiable real function that is defined for all real input values and has a positive derivative at each point.
Sliding window strategy A method to subset data into contiguous segments, possibly with some overlap.
Smoothing A postprocessing method to improve performance by filtering out short, spurious misclassifications.
Tri-axial accelerometer A sensor to record accelerations along three (typically, two horizontal and one vertical) axes.