Learning Personalized ADL Recognition Models from Few Raw Data

Paul Compagnon, Grégoire Lefebvre, Stefan Duffner, Christophe Garcia

Learning Personalized ADL Recognition Models from Few Raw Data

Paul Compagnon[a,b], Grégoire Lefebvre[a], Stefan Duffner[b], Christophe Garcia[b]

[a] *Orange Labs, Grenoble, France*
[b] *LIRIS, UMR 5205 CNRS, INSA Lyon, France*

Abstract

Recognition of Activities of Daily Living (ADL) is an essential component of assisted living systems based on actigraphy. This task can nowadays be performed by machine learning models which are able to automatically extract and learn relevant features but, most of time, need to be trained with large amounts of data collected on several users. In this paper, we propose an approach to learn personalized ADL recognition models from few raw data based on a specific type of neural network called *matching network*. The interest of this few-shot learning approach is three-fold. Firstly, people perform activities their own way and general models may average out important individual characteristics unlike personalized models that could thus achieve better performance. Secondly, gathering large quantities of annotated data from one user is time-consuming and threatens privacy in a medical context. Thirdly, matching networks are by nature weakly dependent on the classes they are trained on and can generalize easily to new activities without needing extra training, thus making them very versatile for real applications. Our results show the effectiveness of the proposed approach compared to general neural network models, even in situations with few training data.

Keywords: Few-Shot learning, Matching Networks, Activity of Daily Living, eHealth, Inertial Measurement Unit, Gated Recurrent Units
2010 MSC: 68T05, 62H30

1. INTRODUCTION

As life expectancy increases, more and more elderly people show difficulties in their every day life, and allowing them to stay at home is a social and public health issue[1]. Many people even struggle performing basic need activities. They are also particularly exposed to chronic diseases: diabetes, cancer, psychological and cognitive disorders, heart diseases, Parkinson and Alzheimer, etc. They are finally vulnerable in simple daily life activities where they could fall, make a

[1] `www.who.int/ageing/publications/world-report-2015`

wrong move or loose attention. Furthermore, 15% of the world population lives with a form of handicap, between 2 and 4% suffering severe disabilities[2]. All these persons could benefit from eHealth services and particularly activity monitoring [1]. The process of recording the every day life activities of a subject using sensors (inertial sensors, for instance) is called actigraphy. Instead of organizing regular visits at the hospital, the patient can be monitored in his/her house with several upsides: it improves the quality of life of the patient and shortens hospital stays while facilitating the diagnosis as important data are collected in the usual environment of the patient. Of course, clinical visits are indispensable but can only take a snapshot of the patient's condition and may occur too late during the disease development [2].

Inertial data are particularly interesting and nowadays easy to record with smartphone sensors (or smart watches, clothes, etc.) which can be carried without stigmatizing the person. They are judged less intrusive at home and more respectful of privacy. They also allow for a continuous monitoring of the person whereas home sensors only work where they are installed and do not target a specific inhabitant. These data can then be used to provide eHealth services regarding the recorded level of activity and the distribution of those activities during the day. Traditionally, the level of autonomy has been evaluated thanks to several criteria related to the Activities of Daily Living (ADL, e.g. having lunch, watching television etc.). This evaluation appears as a pertinent factor for the clinical evaluation of elderly people [3]. It is possible to observe, with the help of actigraphy systems, changes in a person's behavior and so the possible loss of autonomy. Data obtained this way allow to perform ADL or posture classification and prediction and to automatically detect falls. Early approaches consisted in (manually) defining expert rules and thresholds for the sensor values leading to posture recognition and then activities, nowadays machine learning models can automatically learn to classify these data. Moreover, neural network models are able to automatically extract the relevant features to process and are able to adapt easily to new activities and new users [4]. To be able to eventually equip people with such systems, high classification accuracy is required, particularly for critical events such as falls. Moreover, compliance with legal regulations, such as the General Data Protection Regulation[3] (GDPR), and privacy preservation are a necessity.

We tackle both of these issues in this extended version of our paper [5] by proposing a model able to perform personalized ADL classification from few raw data. Contrary to a common practice [6], we advocate for personalized models instead of general models: better performances can be achieved with personalized models since each user has his/her own way of doing his/her activities. It is for example, possible to recognize a person by analyzing his/her gait [7]. In broader perspectives, these personalized models may be more compact as well as easier and faster to train, adapted to smartphones and embedded wear-

[2] www.who.int/disabilities/world_report/2011/report

[3] https://eugdpr.org/the-regulation/

able devices with less energy consumption [8] in the critical context of climate change. However, most of the time, due to privacy concerns and the time needed to annotate each sample, we have very few data coming from a single user in order to effectively train supervised activity recognition models. To overcome this issue, we propose an ADL recognition model based on the matching network architecture [9], and performing few-shot learning, that is, a model able to recognize classes from just one or few samples. Matching networks are by design weakly dependent on the classes they are trained on and therefore can adapt with only one new annotated sample to any new activity class performed by the user, making them very versatile and suited for real environments. Exploiting this property, we demonstrate that the performance can be improved by using another inertial dataset that contains different classes to pretrain the encoding part of the network and that further acts as a validation set to prevent overfitting. The final results show that our approach called SSMN (Sequence-to-Sequence Matching Networks) achieves comparable performances with classical neural network approaches trained on a whole dataset and further obtains over 90% accuracy on one-shot fall classification.

The paper is organized as follows. We summarize in Section 2 previous work on personalized activity recognition and few-shot learning. We then describe our approach for few-shot personalized activity recognition based on matching networks in Section 3. In the Section 4, we report the results of several experiments on the *MobiAct* V2 Dataset [10] and the UCI HAR dataset [11] and assess the utility of the different components of the model and its capacity to predict classes that have not been used for training. Finally, conclusions and perspectives are drawn in Section 5.

2. RELATED WORK

Human Activity Recognition is a very broad computer science field which aims to recognize what a person is doing by analyzing data related to this person recorded from various sensors or instruments. It has numerous applications: from crime detection on video surveillance images to gesture recognition when performing a physical activity. It can be performed in several contexts (Lara et al. [12] listed seven types: ambulation, transportation, phone usage, exercise/fitness, military, upper body gestures, for instance involved in human-computer interactions), using different types of data (e.g. photos, videos, sounds but also smart home data, smart phone communications data or inertial data) and with different approaches *i.e.* time series analyses, rule-based models, statistical models, machine learning and especially neural networks. The question of personalized models is essential to achieve the best performances: indeed, most of the time, people perform activities in a very personal way. In 2004, Bao et al. [13] already observed that user-specific models could lead to better results. They used a decision tree to classify 20 activities including daily household activities with data coming from five accelerometers placed at different body locations. The activities were represented as vectors composed of several extracted features such as mean, energy, etc. Since then, several other

papers came to the same conclusion that personalized models are indeed more accurate. Sun et al. [14] proposed an approach based on multitask learning to learn online personalized activity recognition models while taking advantage of the high resemblance of each task. Another approach by Longstaff et al. [15] proposes to improve the training of already deployed models with specific-user data to obtain a more adapted model. They succeeded in improving the performance by at least 10% with various methods of active learning or co-learning, though active learning requires a large amount of additional user interaction. This issue of not having sufficient data from a single user was not really tackled by previous works, and few-shot learning provides in this context a promising approach. For example, Nguyen et al. [16] built a model able, once operational, to learn to detect never seen activities from very few wearable-sensor data. The model consists in a feature-based part associated by decision fusion with rules connecting the new activity to the others already learned by the model. This way, the model is able to prevent the degradation of performances resulting from adding a class but also not to overfit too much by learning from only 2 or 3 sequences. In another work, Wu et al. [17] presented an approach for one-shot classification of gestures from RGBD videos. They used classical video features and tried to match test samples with one unique training example using the maximum correlation coefficient. Matching networks [9] (see Section 3.2 for details) are also based of this idea of matching samples with references but use metric learning instead [18]. Following this idea, Sani et al. [19] proposed to use matching nets to produce personalized models from accelerometer data. Their method achieved 79% of F-measure on 9 classes, outperforming most standard approaches trained on every user data. However, our work differs from theirs in several points. First, they seem not to have used the full-context embedding (see 3.2) which can provide slight improvements. Then, they trained and tested the model on every class at the same time whereas matching networks are conceived to work independently of the classes they are trained on. Finally, they used discrete cosine transform coefficients as features whereas we trained a sequence-to-sequence model on a different dataset to get a good encoder without having to select handcrafted features.

All previous works tested their approach on different datasets and the lack of reference datasets is a known issue among the activity recognition community. It is due to the variety of contexts in which it can be performed and also the variety of data that can be used. We chose to concentrate on two recent datasets on which results have already been published in order to be able to perform comparisons. The first one is *MobiAct* V2 [10] which contains 12 ADL and 4 falls. The authors of the dataset used it for activity and fall recognition with the goal to develop the most effective pipeline in terms of accuracy. To do so, the authors conducted an exhaustive study to find the best features (for example, they found that the spectral centroid was quite essential). They achieved 97% accuracy score with an instance-based k-nearest neighbor classifier. On the contrary, Di Pietro et al. [20] proposed a new neural network model called MIxed hiSTory Recurrent Neural Networks (MIST) which they tested on

MobiAct V2, among others. They did not select features and learn directly from the raw data. They separated users into fixed train, validation and test groups and repeated the same experiment 50 times and kept the 5 best results. Their approach requires less computation and produces better results than a Long Short-Term Memory (LSTM) Neural Network by achieving 71% of accuracy. Finally, Tsinganos [21] proposed a threshold-based approach combined with a nearest neighbor algorithm and a mechanism of adaptation to the user of the feature vectors to classify falls. They achieved above 90% of accuracy of fall detection on *MobiAct* V2 dataset. Contrary to those approaches, we aim at building personalized models on the *MobiAct* V2 but each user has at most around 50 sequences, which is not enough to train a classical neural network model able to automatically extract and process the relevant features. We thus propose to use matching networks instead to build our architecture.

We also experimented on a second dataset called UCI HAR [11] containing 30 users and 6 activities. The authors notably achieved 96% accuracy with a multiclass SVM. Numerous results have since been published for this dataset. Zhao et al. [22] proposed to use a residual bidirectionnal long-short term memory recurrent neural network to classify the sequences of the dataset and obtained an accuracy of 91.1%. Jiang et al. [23] proposed an approach based on convolutional neural networks (CNN). They designed an architecture able to achieve high accuracy with a low computational cost and got around 95% test accuracy. CNN notably require a large quantity of data to be properly trained which is more difficult to obtain in these personalized settings. San-Segundo et al. [24] proposed to use Hidden Markov models (HMM) but with user adaptation which resulted in an error rate of 2% and a recall of 95.3%. Their model is first trained on the data of every users before being tuned for one specific user with Bayesian adaptation. Though being personalized and using finetuning, our approach presents several differences with theirs. First, their approach actually needs to be first trained with the whole dataset before being finetuned for one user whereas the proposed Matching Networks-based approach can be finetuned with any inertial dataset from the literature. Then, their model is composed of 6 HMM, one for every class which makes the addition of a new class more difficult to set up whereas, once trained, our approach can recognize a new class using just one sequence.

3. LEARNING PERSONALIZED MODELS FOR ADL CLASSIFICATION

3.1. Sequence Processing with Recurrent Neural Networks

Postures, ADL or falls exhibit a dynamic characteristic signature (*e.g.* walking, running, going upstairs) or are, on the contrary, more static (*i.e.* lying, sitting, standing, *etc*). Considering the whole sequence of raw data to classify a posture is thus pertinent [25]. A classical approach when working with sequences is to extract several signal feature vectors from subsequences of the signal in order to build a classifier. This approach is efficient in numerous cases

but, as the window size is limited, it cannot exploit long term dependencies. Recurrent Neural Networks (RNN) possess recurrent connections which give the ability to map an input sequence to an output sequence while at each step taking the information of previous steps into account. This enables the network to not only extract inter-signal but also intra-signal correlations and thus to detect more complex patterns. Let x be the sequence of inputs of the network and x_t be the input at time t. A recurrent network computes $\hat{y}$, the sequence of outputs following this equation:

$$\hat{y}_t = f(W_x x_t b_x + W_h h_{t-1} + b_h), \tag{1}$$

where W_x and W_h are respectively the input and hidden synaptic weight matrices, b_x and b_h the biases and h_{t-1}, the hidden state of the previous step. f is an activation function, generally *tanh*. It is a well-known issue that these networks struggle with long-term dependencies, *i.e.* when they need to learn to retain information during a long time [26]. This is the so-called vanishing gradient problem which has been partly solved by the LSTM model [27]. Gated Recurrent Units (GRU) [28] can be viewed as a simplified version of the LSTM approach, while showing similar performance on most sequential data analysis tasks [29], e.g. speech and music modeling. A GRU computes the vector h_t as follows:

$$h_t = (1 - z_t) * \tilde{h}_t + z_t * h_{t-1}. \tag{2}$$

The hidden state h_t is updated by forgetting the old content and directly adding some new. z_t is called the update gate and is computed according to the following equation:

$$z_t = \sigma(W_{iz} x_t + b_{iz} + W_{hz} h_{t-1} + b_{hz}), \tag{3}$$

where σ is the logistic sigmoid function. We have the following relation for $\tilde{h}_t$ named the new gate:

$$\tilde{h}_t = \tanh(W_{i\tilde{h}} x_t + b_{i\tilde{h}} + r_t(W_{h\tilde{h}} h_{t-1} + b_{h\tilde{h}})), \tag{4}$$

where finally r_t is the reset gate computed similarly as z_t:

$$r_t = \sigma(W_{ir} x_t + b_{ir} + W_{hr} h_{t-1} + b_{hr}). \tag{5}$$

One interesting way of using RNN and GRU is to learn vector representations in a similar fashion as auto-encoders [30] by connecting two RNN: an encoder and a decoder. So-called Sequence-to-Sequence models [31] learn how to reconstruct the input sequence from the last output of the encoder. In that way, a robust representation of the sequence can be learned. Adding noise (often putting 30 or 50% of the values of the sequence to zero) to the input is a good way to improve the model's generalization capacities [32]. The decoder is therefore trained to reconstruct the non-noisy original sequence, forcing the encoder to produce more robust features. The learned representations can then be used by other neural network architectures such as matching networks, which we will describe in the following.

3.2. Few-shot Learning with Matching Networks

Neural networks typically need a large quantity of annotated data to be trained properly, this requirement is even harder to fulfill for personalized models. A way around that would be to train a GRU on data from several users and then to perform a fine tuning of the last layer with user-specific data. This typically works well for image classification [33] and allows to reuse classical CNN architectures trained on large datasets (e.g. Imagenet[4]) on more specific tasks. These types of models make use of several layers of representations, the first ones extracting low-level features common to many images, the last more high-level and specific patterns.

Another way is to train models specifically designed to work with few training samples. Vinyals et al. [9] developed a model called matching networks based on metric learning and attention to efficiently learn to perform this task. This model is not actually trained to classify but rather to match samples with other examples that are part of a support set called S: it learns to produce a nearest-neighbor classifier. It allows the model to work at test time with some classes never seen during training and therefore to perform few-shot and even one-shot learning. S contains N labeled support examples, one or several for each of the C classes. The model is described in Figure 1 and is composed of 4 parts. The first is an encoder, a neural network (in this case, a GRU) trained to produce a vector representation y of the example x to be matched and of each sequence in the support set thus called S_{emb}. The following two parts are called the context embedding and are used to adapt the representations y and S_{emb} relatively to each other. Thus, the second part, the bidirectional GRU, will produce S'_{emb}, representations of each element in S_{emb} according to each other element of the set. This component processes the sequence in both directions and aggregates the results according to the following equation:

$$S'_{\text{emb}} = \overrightarrow{h} + \overleftarrow{h} + S_{\text{emb}}, \tag{6}$$

where $\overrightarrow{h}$ and $\overleftarrow{h}$ are the sequences of hidden states produced by the bidirectional GRU which therefore contain the same number of vectors than S_{emb}. The aggregation is an element-wise addition. Another level of sophistication is the third part which consists in transforming the embedding y according to S'_{emb}, that is, make it closer in the latent space to the embedding of the support elements it could match. This is done thanks to an Attention GRU model, detailed in Algorithm 1, in order to avoid the new representation y' depending on the order of the vectors in S'_{emb} [34]. The parameter p, the number of processing steps, is the number of times y' will be passed through the GRU with r, as input hidden state. Once representations are produced, the last part computes the distances between y' and each vector in S'_{emb} with, for instance, the Euclidean distance or the cosine distance. A softmax function allows then to get probabilities of matching for each member of S which correspond to the

[4] `http://www.image-net.org/`

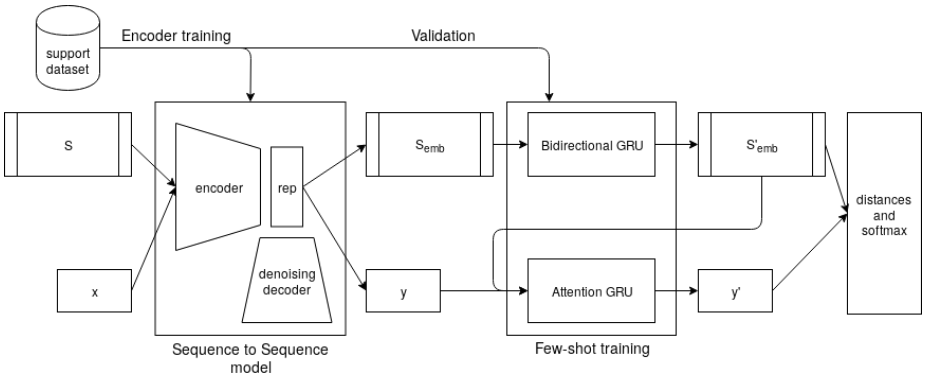


Figure 1: Overview of the SSMN architecture adapted to few-shot sequence classification

Algorithm 1 GRU with read attention [34]

procedure ATTENTIONGRU(y', S'_{emb}, p)
  $h \leftarrow 0_{1,n}$
  $h \leftarrow \text{GRU}(y', h)$
  $i \leftarrow 0$
  for $i = (2..p)$ **do**
    $h \leftarrow h + y'$
    $a \leftarrow 0_{1,N}$
    for $j = (0..N-1)$ **do**e
      $a_{1,j} \leftarrow hS'_{emb,j}$
    $a \leftarrow \text{Softmax}(a)$
    $r \leftarrow \sum_{n=0}^{N-1} a_{1,n} S'_{emb,n}$
    $h \leftarrow \text{GRU}(h, r)$
    $i \leftarrow i + 1$
  return $h \leftarrow h + y'$

probability for x to belong to the same class as the member. Finally, the highest probability determines the class of the input sample.

3.3. Few-shot Learning for Personalized ADL Classification

Matching networks, which are able to generalize new classes from just one example, present several attracting properties in the context of activity recognition. They are particularly adapted to build personalized models and can easily adapt to new activities performed by the user. This is very interesting in a real context where people do a large variety of activities every day and sometimes completely new ones which would thus be recognizable without retraining the whole model. The general architecture described in the previous section can be used with any type of data by choosing the right encoder for those data. In their original paper, Vinyals et al. [9] used classical convolutional models pre-trained on large datasets as encoders (such as VGG [35] or Inceptions [36] for

image processing tasks) then they perform or not a finetuning of this encoder at training time on the specific small datasets. Here, we propose to train a GRU encoder for sequences as a sequence-to-sequence model [37] from which we kept only the encoder part. We leverage the properties of matching nets previously exposed by proposing to use another dataset to train the sequence-to-sequence model, and we therefore call our approach SSMN (Sequence-to-Sequence Matching Networks). This dataset must contain very similar data (*i.e.* inertial data) to the dataset we are trying learn but can be taken from the literature and can have completely different classes. This could be assimilated to some form of transfer learning [38] but the data actually stay very similar. Another way to exploit this property is to also use this similar dataset as a validation set to avoid overfitting during the training of the matching nets which is done with very few data (2 or 5 sequences for each activity plus one in the support set). Thus more data can be kept to train and test the model. We call this other dataset, a support dataset: it is not strictly necessary but can largely increase the performanc of SSMN.

In the next section, we experimentally verify the benefit of using a support dataset together with the property of matching networks to be independent of the classes they are trained on by presenting only test results on classes not used for training and recognized from just one support example (*i.e.* performing one-shot learning).

4. EXPERIMENTS

4.1. Preliminary Experiment: Personalized Postures Classification

We first propose a preliminary experiment on a dataset called *Postures* where personalized models are learned on inertial data with a standard GRU only. This dataset will be used afterwards as support dataset as explained in section 3.3. The *Postures* dataset was created by Quach [39]. The data has been acquired using a 9-axes Inertial Measurement Unit (accelerometer, gyroscopes and magnetometer, IMU) on 9 subjects executing the same sequence several times. Each user produced 5 sequences apart from user 2 who did 10. The sequences are composed of five postures, repeated several times: walking, sitting, lying, standing and transfer and have been conceived to reflect a daily routine of 24h on a 12 minutes activity sequence. "Transfer" represents the transition between two postures. Overall, there are about 358k labeled 9D vectors and 1938 segments of sequence related to one activity (see Table 1 for details). The sampling frequency is about 10 Hz. The used sensor was a *Shake SK6* [40] with the following range and precision for each sensor. The range of the triple axes accelerometer is at most $\pm$6g with a precision of 1mg. The range of the triple axes gyroscope is of $\pm$500 °/s with a precision of 0.1 deg/second. The triple axes magnetometer has a range of $\pm$2 Gauss and a precision of 1 mGauss.

The personalized posture GRU-based models were trained on four sequences (*i.e.* nine for user #2) during 150 epochs and were tested on 1 randomly chosen

	walking	sitting	lying	standing	transfer	Total
# segments	454	280	138	280	751	1938
# vectors	63863	118802	149602	19555	28425	380247

Table 1: Summary of *Postures* dataset activity sequences

Table 2: Results for posture classification on Postures with GRU. (SD: Standard Deviation)

Method	**F-measure**	**SD**	**Accuracy**	**SD**
Makni et al [43]	-	-	0.807	0.024
GRU [8,8] (all users)	0.553	0.068	0.742	0.056
GRU [8,8] (user 2)	0.705	0.059	**0.874**	0.049

sequence. Based on preliminary experiments on the full dataset, we use personalized GRU models with two hidden layers of size 8 and found that this shallow architecture was sufficient to achieve good performance. The process was reproduced excluding a different sequence each time (5 times for each user except for user 2, 10 times). We thus performed a k-folds leave-one-out test of our architecture where k is the number of sequences associated to one user. Moreover, regularization is introduced in the training by using dropout and weight decay. Dropout [41] randomly drops units with a probability 0.5. Weight decay adds a small penalty to the loss function for the magnitude of the weights, improving generalization [42]. Finally, each training starts with a learning rate of 0.01 which decreases by a factor 10 if the loss does not diminish during 10 epochs.

In a previous paper using this dataset, Makni et al. [43] compared two attitude device estimation algorithms and used expert rules and Kalman filters in order to estimate the individual postures. Our first experiment consists in reproducing these experiments using only machine learning and no expert *a priori* rules. A comparison is presented in Table 2. On average, the GRU accuracy of 0.742 is lower than the 0.807 achieved in [43]. Nevertheless, our model is not tuned for each user and no specific expert rule is applied. We used only one GRU architecture which learns on few examples and generalizes well to unknown user sequences. This is particularly encouraging when we focus on the user #2 performances which achieved the best accuracy of 0.874. This is mainly due to the fact that the user provides twice as much data as other users. Consequently, asking people to collect only 10 sequences for building a shallow GRU model shows promising results with an acceptable user effort, in practice.

This preliminary experiment shows the benefit to learn personalized models with GRU even from only 5 continuous sequences of activities. We will now push this approach further on a larger dataset with more users and activities but less data per activities. Based on the results of this preliminary experiment, we will test our approach SSMN by performing a pretraining of the encoder as a Sequence-to-Sequence model on *Postures*.

4.2. The MobiAct V2 Dataset

The *MobiAct* V2 dataset [10] is an inertial dataset created to support research in ADL recognition. It includes 15 different activity labels: 4 falls and 11 ADL. The activities were recorded following a realistic scenario: a typical day of work by 67 subjects. In total, the dataset is constituted of about 3200 trial sequences. Data were acquired using a smartphone which the user could place anywhere. The IMU is composed of a *LSM330DLC* itself composed of a tri-axes gyroscope and a tri-axes accelerometer. The measurement range of the accelerometer can be selected between ±2g, ±4g, ±8g or ±16g. The measurement range of the gyroscope can be selected between ±250°/s, ±500°/s or ±2000°/s. The orientation sensor combines data from the accelerometer and the magnetometer. We considered only the 19 users[5] for our experiments who have performed the 15 activities with at least two trials for 12 activities out of the 15 and one trial for "walking", "sitting" and "standing". Each user has around 53/54 trial sequences in total. No preprocessing was applied to the trial sequences apart from a resampling to a size of 50. Some trial sequences contain only one activity, others have several (for example, standing and lying before and after a fall). We treated each trial as one activity, the one mentioned in the name of the file containing the trial, and we labeled the resampled sequence. We did the same on the *Postures* dataset, after segmenting the dataset to train the encoder. In a real environment, this segmentation could be replaced by resampled sliding windows of the signal.

The *Postures* and *MobiAct* datasets have three postures in common: walking, standing and sitting. The lying posture is not independent and always concatenated with a fall. The posture "transfer" in the Postures dataset can be assimilated to "stand to sit" ADL in the *MobiAct* V2 dataset. However, "transfer" is very diverse and less characterized and we did not consider those two as strictly similar. Moreover, the activities "walking", "sitting" and "standing" only have one trial sequence available per user which is not enough to test our algorithm: we need at least one sequence to be the support example and another to match with it. Thus, we removed "walking", "sitting" and "standing" from the *MobiAct* dataset, to have both datasets having strictly disjoint sets of activities[6]. Features in both datasets are globally the same apart from the magnetometers which is rather an orientation computed from the other features in *MobiAct*. The learned encoder is therefore not exactly adapted to the features of *MobiAct*. We hereafter name this modified version of *MobiAct* dataset *MiniMobiAct*.

[5] The selected users are the following : 1, 2, 3, 5, 6, 12, 20, 45, 53, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67.

[6] Actually, there remain some "standing" or "lying" parts in some sequences, for example in fall sequences. However, they are not labeled as such and thus will not be learned as such by the model. Potentially, this could be a factor of confusion for the model, but, in view of the experimental results, our approach still focuses on the relevant parts of the sequence.

4.3. Training Strategies and Protocol Details

The protocol to train matching networks differs significantly from a classical machine learning protocol. It is not trained to classify but to produce a one-shot learning classifier at test time by matching test samples with one support example per class. The training of such algorithm needs therefore to be independent from the classes and from the support examples: thus, classes and not instances are sampled. Nine classes and their corresponding sequences are randomly chosen to be part of the training set, the remaining three are part of the test set. During the training, the batches are composed of disjoint classes and not of a specific number of sequences. We chose a batch size of three to match the number of classes to predict at test time. In a batch of three classes, one instance of each class is randomly selected to be a support example. After each batch, the parameters of the model are updated by computing the log likelihood loss on the remaining sequences of the batch classified with the support examples. At test time, one support example from each test classes is chosen to be a support example, the other instances are classified. Therefore, at test time, SSMN is a one-shot classifier (one labeled example per class) which does not require any more training and every validation or test results subsequently presented are one-shot learning results on classes never seen at training.

To train this model, we used a learning rate of 0.001 which was divided by 2 every 10 epochs without decreasing of the training loss. The value of the gradient was also clipped to 6 to avoid exploding gradient in accordance with [44]. The GRU were initialized with orthogonal weight matrices scaled to have a spectral radius of 1.1 in accordance with [45]. We used the *Postures* dataset as a support dataset to improve the performance of one-shot activity classification. A sequence-to-sequence model is trained on it and only the encoding part is kept. For this model, we used a learning rate of 0.01 and a batch size of 10. To improve generalization capacity, we trained the decoding part as a denoiser by randomly setting to 0 the values in the input sequence with probability of 0.3 [32] which produces more robust features. A similar initialization as mentioned above is also done. In the validation phase, three classes of *Postures* are randomly selected to match the batch size.

We report in the following the accuracy and F-measure[7] by concatenating the results of several train-then-test phases and computing a global score after 40 iterations (thus, on about 400 classifications, depending on the sampling for *MiniMobiAct*). The evaluation scores are computed in two different ways. The 3-way scores are computed from the results of a 3-class classification: the model is given one new support example (one-shot learning) of three never-seen classes and tries to classify test instances into these three classes. In the 12-way setting, the model is given support examples for the three new classes and also the 9 classes used at training (randomly selected, here again) and tries to classify the same test instances but with 12 choices instead of 3. Nevertheless,

[7]Accuracy corresponds to F-measure micro-averaged for multiclass classification so we report F-measure macro-averaged, which does not take into account class imbalance.

the presented scores are always for the 12 classes due to the 40 test iterations being concatenated and the selected test classes being different each time.

4.4. Validation of SSMN Components

We subsequently validate each component or the SSMN architecture on the first user of *MiniMobiAct*. We first decide if fine-tuning is effective and the number of processing steps of the Attention GRU to be performed. We begin with only one processing step and keep the best configuration after each stage. Then, we validate the interest of using the *Postures* dataset as a support dataset.

4.4.1. Finetuning Evaluation

First, we evaluate the interest of finetuning a linear layer after the output of the pretrained encoder of our SSMN architecture. The encoder provides representations of size 100: this size was chosen according to the reconstruction error achieved on the *Postures* dataset during the pretraining. When using this encoder, the representation can either be used as is or passed through a linear layer which will be finetuned with the training data coming from the user we are learning a model for. We tested two output hidden layer sizes, 20 (a small version) and 100 (the same output size). The results are presented in Table 3a. We observe that the architectures labeled `a1` for cosine distance with 0.856% of 3-way accuracy and `a2` for euclidean distance with 0.825% of 3-way accuracy, got the best results and we therefore select them for the following experiments. We observe that in both cases, the small size got the worst results with significant degradation indicating that this representation size does not seem to preserve enough information.

4.4.2. Number of Attention GRU Processing Steps

Next, we investigate the parameter p, *i.e.* the number of processing steps of the attention GRU. We tested four values of processing steps between 1 and 10, the largest value experimented in [34]. The results are presented in Table 4. We observe that for the cosine distance 0.858% of 3-way accuracy could be achieved (`b1`) with 10 processing steps and that for the euclidean distance, a maximum of 0.839% could be achieved (`b2`). We globally notice that better values are achieved with more processing steps as in [34]. Also as in [9], this is a very slight improvement. The batch size used here is only three, and a more important impact should be expected when trying to work with more classes. We kept these parameters for the remaining experiments.

4.4.3. Impact of Pretraining

Now that components of the architectures and their parameters have been validated, we aim at measuring the exact impact of using the *Postures* dataset to pretrain our model. We thus propose two experiments. First, the same GRU encoder is trained by only using the *MiniMobiAct* training data (so not as a sequence-to-sequence model). The results are shown in Table 3b. The proposed architectures correspond to the ones experimented for the finetuning where a

Distance	Finetuned layer size	3-way accuracy	3-way F-measure
cosine	no	0.812	0.79
cosine	20	0.819	0.763
cosine (a1)	100	**0.856**	**0.823**
euclidean (a2)	no	**0.825**	**0.81**
euclidean	20	0.750	0714
euclidean	100	0.793	0.726

(a) With pretraining on the *Postures* dataset and different finetuned layer sizes.

Distance	GRU network Architecture	3-way accuracy	3-way F-measure
cosine	[100]	0.745	0.696
cosine	[100, 20]	0.694	0.676
cosine	[100, 100]	0.733	0.709
euclidean	[100]	0.8	0.758
euclidean	[100, 20]	0.763	0.736
euclidean	[100, 100]	0.784	0.754

(b) Without pretraining on the *Postures* dataset and different architectures.

Table 3: Classification accuracy on *MiniMobiAct*, user1, with different matching network architectures.

layer of size 100 had already been learned and frozen. We observe that none of those architectures could outperform the results obtained by the best ones for each metric (b1 and b2). The difference is more flagrant for the cosine metric where a more than 10% improvement could be achieved with pretraining on the Postures dataset. These results show the benefit of pretraining an encoder as a sequence-to-sequence model, on a different dataset containing similar inertial data even if both have no activities in common.

The other interesting property of matching networks is that they can recognize new classes using just one new sequence. In situations of very few available training data, there may not even be enough data to perform a proper valida-

Distance	Processing steps	3-way accuracy	3-way F-measure
SSMN cosine (a1)	1	0.856	0.823
SSMN cosine	3	0.793	0.802
SSMN cosine	5	0.779	0.783
SSMN cosine (b1)	10	**0.858**	**0.826**
SSMN euclidean (a2)	1	0.825	**0.81**
SSMN euclidean	3	0.777	0.759
SSMN euclidean (b2)	5	**0.839**	0.805
SSMN euclidean	10	0.814	0.809

Table 4: Classification accuracy of matching network variants on *MiniMobiAct*, user 1, with different numbers of processing steps.

Distance	Epochs	3-way accuracy	3-way F-measure
SSMN `b1`	20	0.845	0.795
SSMN `b1`	50	0.791	0.762
SSMN `b1`	100	0.855	0.833
SSMN `b2`	20	0.705	0.668
SSMN `b2`	50	0.753	0.709
SSMN `b2`	100	0.645	0.606

Table 5: SSMN early stopping comparison on *MiniMobiAct*, user 1.

tion or "early stopping" to prevent overfitting. In those conditions, with SSMN, another dataset can be used as validation set, here the *Postures* dataset. We trained several models (using the same parameters as `b1` and `b2`) with fixed number of epochs to compare the results with those previously obtained with early stopping based on the performance on the *Postures* dataset. The results are presented in Table 5.

While the advantage for the cosine metric seems not significant, even non-existent, we remark that the models using the Euclidean distance clearly overfit and results are worse than those obtained with early stopping and decreasing over 50 epochs. The use of the *Postures* dataset as a validation set improves the training in this case.

4.5. Test Results on All Users

We now apply `b1` and `b2` using a pretrained encoder on Postures dataset and early stopping of the training also on Postures dataset to every other users of *MiniMobiAct* V2 having more than 50 sequences (18). To recall some results from the state of the art, Chatzaki et al. [10] achieved 97% accuracy with an instance-based k-nearest neighbor classifier and heavily relying on signal processing techniques and manual feature selection. DiPietro et al. [20] achieved 71% accuracy with their MIST-recurrent neural network approach. It is not possible to directly compare these results to those obtained with matching nets since the protocols are completely different: at test time, matching net learns in one shot to predict three new classes whereas the other models were trained on every class. Thus, these numbers are only a rough indications. The test results on 18 users are presented in Table 6.

Algorithms	3-way accuracy	12-way accuracy
SSMN `b1`	**0.755±0.084**	**0.533±0.103**
SSMN `b2`	0.742±0.085	0.505±0.099
	3-way F-measure	12-way F-measure
SSMN `b1`	**0.741±0.087**	**0.522±0.099**
SSMN `b2`	0.734±0.081	0.494±0.094

Table 6: Test scores on *MiniMobiAct*, average of 18 users.

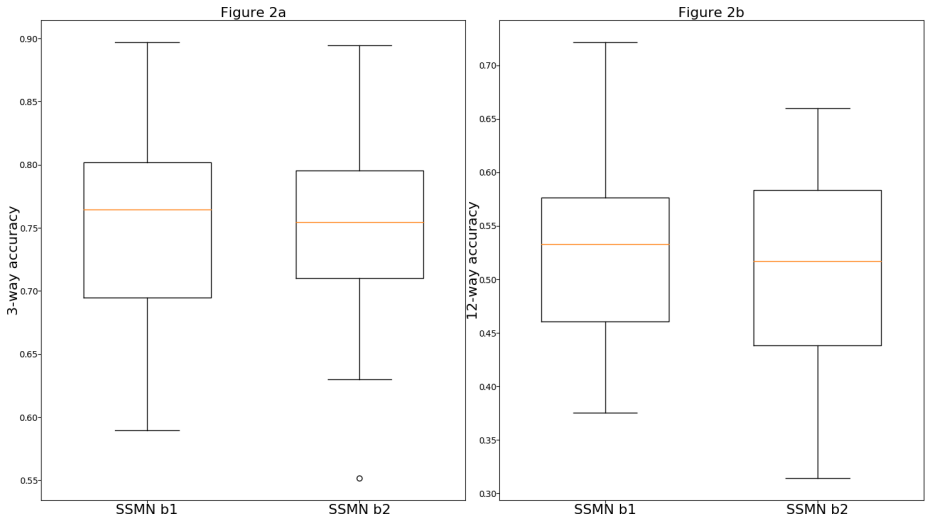


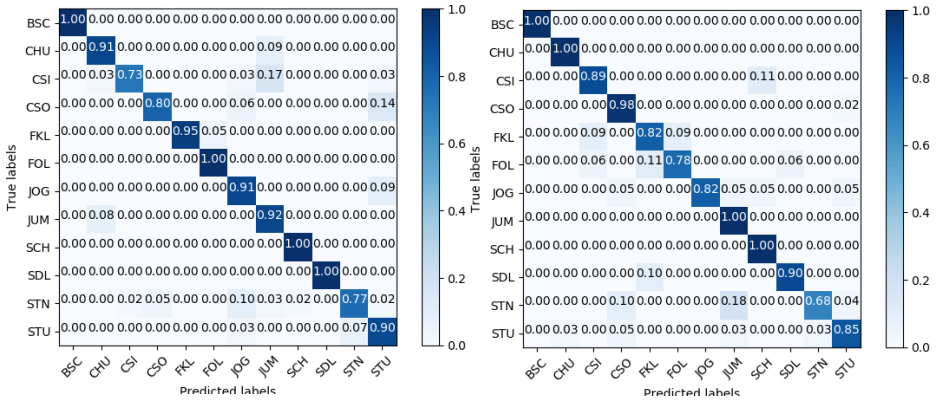
Figure 2: Result per user dispersion for all activities for 3-way accuracy (a) and 12-way accuracy (b)

The best results are achieved by the cosine metric with 75.5% 3-way accuracy even if the gap with the Euclidean distance can be considered as not significant regarding the standard deviations. To make comparisons with the 71% achieved by DiPietro et al. [20], our model achieved superior results in the 3-way scenario.

We see on Figure 2a that about 25% of the users having more than 80% accuracy and about 50% more than 75%. On the 12-way classification, our model achieved more mixed results as could be expected. However, we see on Figure 2b that two users achieved at least 70% 12-way accuracy with the cosine metric. These results demonstrate the capacity of SSMN to efficiently classify new classes from only examples. The worst performing user for the Euclidean distance is actually always the same across the figures. These performances seem inherent to the quality of data gathered for this user which indicates that, although SSMN can work with few data, it still requires good quality data. Concerning the difference between cosine metric and Euclidean distance, the results seem in contradiction with what was observed by Snell et al. [46] [8] which they explained by Euclidean distance being a Bregman divergence [47] contrary to the cosine metric. However, this is coherent with what we observed with other metric learning architectures making use of GRU [48] where cosine metric systematically outperformed the Euclidean distance.

On Figure 3, we show the confusion matrices obtained by the best users for

[8] Snell et al. also proposed a model for few-shot learning called prototypical networks. Actually, their approach coincides with matching networks in the one-shot scenario which we performed here.



(a) SSMN `b1`, User 5 with 89.7% 3-way accuracy (b) SSMN `b2`, User 6 with 89.4% 3-way accuracy

Figure 3: Confusion matrix of best users for both metrics.

Algorithms	3-way fall accuracy	12-way fall accuracy
SSMN `b1`	0.928±0.053	**0.922±0.057**
SSMN `b2`	0.927±0.049	0.896±0.0876
	3-way fall F-measure	12-way fall F-measure
SSMN `b1`	0.891±0.071	**0.879±0.076**
SSMN `b2`	0.885±0.071	0.848±0.109

Table 7: Test scores for fall detection on *MiniMobiAct*, average of 18 users.

each metric. We observe difficulties for similar classes[9] namely STN, JOG or CSI but the errors are not the same. For example, STN is confused with JOG or JUM. In the case of the cosine metric, 9 classes out of 12 achieved more than 90% 3-way prediction accuracy.

Finally, we propose to analyze the results as a binary classification of falls vs. non falls similarly as in [5] where 0.808 of F-measure and 0.878 of Accuracy could be achieved on personalized fall detection. The results are presented in Table 7. We observe gains for both models on both metrics, the most important being the cosine metric with with a gain of 3% accuracy but especially 11% of F-measure (macro) which means a great improvement of the recognition of falls, even in a one-shot learning setting. We can also compare those results to the work from Tsinganos et al. [21] who achieved a recall of 97.53% and a specificity of 94.89%. Our one-shot learning approach reached only slightly inferior results compared to this general approach.

[9] Class legend, JOG: Jogging ; JUM: Jumping ; STU: Stairs up ; STN: Stairs down ; SCH: Stand to sit ; CHU: Sit to stand ; CSI: Car-step in ; CSO: Car-step out ; FOL: Forward-lying ; FKL: Front-knees-lying ; BSC: Back-sitting-chair ; SDL: Sideward-lying

Distances	Finetuning	# Training epochs
Cosine (`c1`)	20	100
Euclidean (`c2`)	20	50

Table 8: Parameters validated on User 1 of UCI HAR dataset.

Algorithms	2-way accuracy	6-way accuracy
SSMN `c1`	0.755±0.096	0.669±0.07
SSMN `c2`	**0.80±0.081**	**0.706±0.65**
	2-way F-measure	6-way F-measure
SSMN `c1`	0.725±0.093	0.63±0.08
SSMN `c2`	**0.789±0.072**	**0.665±0.077**

Table 9: Test scores on UCI HAR, average of 29 users.

4.6. Complementary Experiments on UCI HAR Dataset

We conducted experiments on another dataset called UCI HAR [11] which provides the data of 30 users who performed 6 activities : "walking", "walking upstairs", "walking downstairs", "sitting", "standing" and "lying". It contains 10299 sequences of size 128 which are fixed-width sliding windows of 2.56 sec with a 50% overlap. The sensors are a tri-axes accelerometer and a tri-axes gyroscope recording at 50 Hz. We employed a similar protocol as on *MiniMobiAct*. Four activities out of the 6 are sampled to train the model, 2 are left for testing. We used a batch size of 2 to match the number of test classes. We validated the parameters reported in Table 8 on the data of user 1 for both distances. These architectures are hereafter named `c1` and `c2`. AttentionGRU (cf. Algorithm 1) was not used for this dataset since it lead to weaker results. This may be due to the low number of vectors in the support set with a batch size of 2 which provokes overfitting since we observed slightly better results when trying to classify one element in the 6 classes (*i.e.* using a support set of size 6). Similarly, early stopping based on the *Postures* dataset leads to slightly inferior results for user 1 and was therefore replaced by a fixed number of training epochs. We report, in Table 9, the test results averaged over the 29 other users.

We observe that the Euclidean distance got slightly better results than the cosine metric, SSMN achieved an average 2-way accuracy of 0.8 with 16 users out of the 29 over 0.8 and one user over 0.9. For the 6-way scores, they culminate at 0.706. Best state-of-the-art approaches achieve around 95% accuracy on the complete dataset. San-Segundo et al. [24] achieved this with a user adaptation approach. Our approach obtained lower results but the model is only trained with the data of one user and is much more flexible. This dataset may actually bring two obstacles for our approach. Firstly, the sequences are short (128 points, less than 3 seconds) which decreases the quantity of discriminant information in each sequence and therefore makes matching more difficult. Secondly, some classes can be very difficult for SSMN to learn to differentiate

from just one or few sequences as the data may look very similar: e.g. "walking upstairs" and "walking downstairs", "sitting" and "lying". This is particularly visible on the 6-way score values.

5. CONCLUSIONS AND PERSPECTIVES

We presented in this paper an approach for personalized ADL classification based on matching networks combined with sequence-to-sequence pretraining (SSMN). This approach presents two major advantages which make it very relevant to be implemented for real applications. First, it addresses the problem of limited training data that is encountered when learning personalized models by being able to learn from just a few examples. Second, it is very versatile regarding each new activity a user could perform by being able to learn it just from one example. When properly trained, SSMN is indeed independent from the classes it was trained on and only relies on the provided support set. In this way, its performance can be boosted with any dataset from the literature which possesses similar characteristics even if the features are not exactly the same and the activities different. With this model, we achieved over 75% of 3-way accuracy, a performance comparable to those obtained by classical neural network models trained on the whole *MiniMobiAct* dataset whereas our test results were systematically obtained from classes never seen at training. Those results were particularly good for "fall" classes with over 90% 3-way accuracy and 12-way accuracy, meaning SSMN is a relevant approach to detect any kind of falls. With a second experiment, we showed that this approach could be extended to another dataset but also that it presents some limits despite its great flexibility.

A further step of our work would be not to use any annotated data and work in a pure unsupervised manner to achieve complete flexibility regarding activities and adaptation to any user since he/she would have nothing to do but wear the sensor. We started tackling this objective by studying recurrent daily behaviors (routines) with metric learning algorithms adapted to sequential data [48]. Metric learning algorithms typically need a notion of similarity between samples instead of class labels to be trained and we have been using the recording periods of data as such. We aim now at defining a more subtle notion of similarity for routines and at building more powerful sequence metric learning algorithms.

References

[1] H. Pigot, B. Lefebvre, J.-G. Meunier, B. Kerhervé, A. Mayers, S. Giroux, The role of intelligent habitats in upholding elders in residence, in: 5th international conference on Simulations in Biomedicine, 2003, pp. 497–506.

[2] A. Avci, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, P. Havinga, Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey, in: ARCS, VDE, 2010, pp. 1–10.

[3] S. Katz, A. B. Ford, R. W. Moskowitz, B. A. Jackson, M. W. Jaffe, Studies of illness in the aged: the index of adl: a standardized measure of biological and psychosocial function, JAMA 185 (12) (1963) 914–919.

[4] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, C. Liu, A survey on deep transfer learning, in: International conference on artificial neural networks, Springer, 2018, pp. 270–279.

[5] P. Compagnon, G. Lefebvre, S. Duffner, C. Garcia, Personalized posture and fall classification with shallow gated recurrent units, in: 2019 IEEE 32nd International Symposium on Computer-Based Medical Systems (CBMS), 2019, pp. 114–119.

[6] C. Debes, A. Merentitis, S. Sukhanov, M. Niessen, N. Frangiadakis, A. Bauer, Monitoring activities of daily living in smart homes: Understanding human behavior, IEEE Signal Processing Magazine 33 (2) (2016) 81–94.

[7] L. Wang, T. Tan, H. Ning, W. Hu, Silhouette analysis-based gait recognition for human identification, IEEE PAMI 25 (12) (2003) 1505–1518.

[8] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in nlp, arXiv preprint arXiv:1906.02243 (2019).

[9] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.

[10] C. Chatzaki, M. Pediaditis, G. Vavoulas, M. Tsiknakis, Human daily activity and fall recognition using a smartphone's acceleration sensor, in: ICT4AWE, Springer, 2016, pp. 100–118.

[11] D. Anguita, A. Ghio, L. Oneto, X. Parra, J. L. Reyes-Ortiz, A public domain dataset for human activity recognition using smartphones., in: ESANN, 2013.

[12] O. D. Lara, M. A. Labrador, A survey on human activity recognition using wearable sensors., IEEE Communications Surveys and Tutorials 15 (3) (2013) 1192–1209.

[13] L. Bao, S. S. Intille, Activity recognition from user-annotated acceleration data, in: International Conference on Pervasive Computing, Springer, 2004, pp. 1–17.

[14] X. Sun, H. Kashima, N. Ueda, Large-scale personalized human activity recognition using online multitask learning, IEEE Transactions on Knowledge and Data Engineering 25 (11) (2012) 2551–2563.

[15] B. Longstaff, S. Reddy, D. Estrin, Improving activity classification for health applications on mobile devices using active and semi-supervised learning, in: 2010 4th International Conference on Pervasive Computing Technologies for Healthcare, IEEE, 2010, pp. 1–7.

[16] L. T. Nguyen, M. Zeng, P. Tague, J. Zhang, Recognizing new activities with limited training data, in: Proceedings of the 2015 ACM International Symposium on Wearable Computers, ACM, 2015, pp. 67–74.

[17] D. Wu, F. Zhu, L. Shao, One shot learning gesture recognition from RGBD images, in: Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE, 2012, pp. 7–12.

[18] A. Bellet, A. Habrard, M. Sebban, Metric learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 9 (1) (2015) 1–151.

[19] S. Sani, N. Wiratunga, S. Massie, K. Cooper, Personalised human activity recognition using matching networks, in: International Conference on Case-Based Reasoning, Springer, 2018, pp. 339–353.

[20] R. DiPietro, C. Rupprecht, N. Navab, G. D. Hager, Analyzing and exploiting NARX recurrent neural networks for long-term dependencies, Workshop track - ICLR (2017).

[21] P. Tsinganos, A. Skodras, A smartphone-based fall detection system for the elderly, in: Proceedings of the 10th International Symposium on Image and Signal Processing and Analysis, IEEE, 2017, pp. 53–58.

[22] Y. Zhao, R. Yang, G. Chevalier, X. Xu, Z. Zhang, Deep residual bidir-lstm for human activity recognition using wearable sensors, Mathematical Problems in Engineering 2018 (2018).

[23] W. Jiang, Z. Yin, Human activity recognition using wearable sensors by deep convolutional neural networks, in: Proceedings of the 23rd ACM international conference on Multimedia, 2015, pp. 1307–1310.

[24] R. San-Segundo, J. M. Montero, J. Moreno-Pimentel, J. M. Pardo, Hmm adaptation for improving a human activity recognition system, Algorithms 9 (3) (2016) 60.

[25] G. Lefebvre, S. Berlemont, F. Mamalet, C. Garcia, BLSTM-RNN based 3D gesture classification, in: ICANN, Springer, 2013, pp. 381–388.

[26] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE transactions on neural networks 5 (2) (1994) 157–166.

[27] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (8) (1997) 1735–1780.

[28] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv:1406.1078 (2014).

[29] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, arXiv preprint arXiv:1412.3555 (2014).

[30] M. A. Kramer, Nonlinear principal component analysis using autoassociative neural networks, AIChE journal 37 (2) (1991) 233–243.

[31] I. Sutskever, O. Vinyals, Q. V. Le, Sequence to sequence learning with neural networks, in: Advances in neural information processing systems, 2014, pp. 3104–3112.

[32] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, Journal of machine learning research 11 (Dec) (2010) 3371–3408.

[33] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, J. Liang, Convolutional neural networks for medical image analysis: Full training or fine tuning?, IEEE transactions on medical imaging 35 (5) (2016) 1299–1312.

[34] O. Vinyals, S. Bengio, M. Kudlur, Order matters: Sequence to sequence for sets, arXiv preprint arXiv:1511.06391 (2015).

[35] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[37] I. Sutskever, Training recurrent neural networks, University of Toronto, Toronto, Ont., Canada (2013).

[38] S. J. Pan, Q. Yang, A survey on transfer learning, IEEE Transactions on knowledge and data engineering 22 (10) (2009) 1345–1359.

[39] K. A. Quach, Extraction de caracteristiques de l'activite ambulatoire du patient par fusion d'informations de centrales inertielles, Ph.D. thesis, UCBL1 (2012).

[40] J. Williamson, R. Murray-Smith, S. Hughes, Shoogle: excitatory multimodal interaction on mobile devices, in: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM, 2007, pp. 121–124.

[41] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, JMLR 15 (1) (2014) 1929–1958.

[42] A. Krogh, J. A. Hertz, A simple weight decay can improve generalization, in: NIPS, 1992, pp. 950–957.

[43] A. Makni, G. Lefebvre, Attitude estimation for posture detection in ehealth services, in: IEEE CBMS, 2018, pp. 310–315.

[44] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, 2013, pp. 1310–1318.

[45] I. Sutskever, J. Martens, G. Dahl, G. Hinton, On the importance of initialization and momentum in deep learning, in: Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28, ICML'13, JMLR.org, 2013.

[46] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.

[47] A. Banerjee, S. Merugu, I. S. Dhillon, J. Ghosh, Clustering with bregman divergences, Journal of machine learning research 6 (Oct) (2005) 1705–1749.

[48] P. Compagnon, G. Lefebvre, S. Duffner, C. Garcia, Routine modeling with time series metric learning, in: Artificial Neural Networks and Machine Learning – ICANN 2019: Deep Learning, Springer International Publishing, 2019, pp. 579–592.