Deep Learning Techniques for Gesture Recognition: Where to Split the Complexity

Matteo Drago, Riccardo Lincetto†

***Abstract*—With the increasing interest in deep learning techniques and its applications, also Human Activity Recognition (HAR) saw significant improvements; before neural networks were put into practice, most of the research activities on the field relied on hand-crafted features which, however, couldn't represent nor distinguish well enough complex and articulated movements. Moreover, the use of smart devices and wearable sensors brought the challenge to another level: dealing with high-dimensional and noisy time series while assuring optimal performances requires a detailed study and, most of all, a considerable computational effort.**

In our paper we present the design of an HAR architecture which implements convolutional layers in order to extract significant features from windows of samples, along with Long-Short Term Memory (LSTM) layers, suitable to exploit time dependencies among consecutive samples. For our study, we tried to minimize the collection of layers per network and thus the amount of parameters to train, which could be of great advantage in real time applications. In addition, we also decided to study how performances change if we split the process into two distinct phases: the first one that performs *activity detection* while the last one *activity classification*. The dataset that we used to assess the efficiency of our architectures is the OPPORTUNITY dataset.

***Index Terms*—Human Activity Recognition, Machine Learning, Neural Networks, Motion Detection.**

I. Introduction

During the past decade, time series classification has captured growing interest thanks to the introduction of deep learning mechanisms, such as neural networks. These tools are indeed capable of identifying and learning signal features, which are then exploited for classification, without the need of human domain-knowledge: this is a huge step forward considering that features were traditionally hand-crafted. Human Activity Recognition (HAR) in particular has been fostered by the spread of powerful, efficient and affordable sensors, which nowadays are commonly found in mobile phones and wearable devices, with multiple applications, ranging from health care to gaming and virtual reality [**?**]. Wearable sensors allow us to collect and process a huge amount of signals, which are essential for deep neural networks (DNN) to work properly: in fact, in order for them to learn and for being accurate enough to be preferred over standard machine learning approaches, we need the input training set to be heterogeneous, meaningful and representative of the problem. For these reasons, HAR is not an easy classification problem: when dealing with on-body sensors, system performances heavily depends on human behaviour, which is a source of high variability; moreover, data collected from sensors is typically high-dimensional, multi-modal and subjected to noise, making the problem even more difficult from a machine learning perspective.

In recent years, several ways of performing activity detection and classification have been proposed: in the literature there's no shortage of models. The trend has been to expand the power of networks, adding more and more layers: this resulted in more accurate models, that had to face though an increasing computational complexity. The computational power however is limited in real applications and excessive complexity might translate into impracticability. Moreover, as pointed out in [**?**] and [**?**], despite the proliferation of models to perform activity detection and classification, the lack of common benchmarking data, and of detailed descriptions of the proposed models, prevented a fair comparison between different solutions.

Considering also that the activity recognition problem has been already widely addressed by many authors, we decided to present a systematic comparison between two different commonly proposed types of pipeline. The two differ on how inactivity is handled: the first one tries to learn a representation of the signals where no action is performed, adding a *Null Class* to the other activities; the second one instead splits the classification into two tasks, first deploying an activity detector that filters out inactivity signals and then classifying the remaining activities. Our study is meant to provide a baseline for future work, giving an idea on which system could be more appealing. In order to assess the efficiency of our models, that have been designed against the trend trying to minimize the number of trainable parameters, we used the OPPORTUNITY dataset [**?**], [**?**] which will be described in details in the following sections.

In conclusion, the contributions of this paper are:

- overview of the latest progresses of the state of the art;
- implementation of those solutions;
- comparison of two different approaches.

The paper is organized as follows: section II provides a summary of the latest and more important works related to our studies; in section III we start delving into the details of how we organized our HAR architecture, step by step; section IV is dedicated to the description of the dataset and to the decisions we made in the preprocessing phase; finally in section V we are ready to describe meticulously the learning framework, while sections VI and VII are for discussion of results and for drawing our conclusions.

†Department of Information Engineering, email: {matteo.drago,riccardo.lincetto}@studenti.unipd.it

II. Related Work

The OPPORTUNITY activity recognition dataset has been introduced in [**?**] to overcome the lack of an evaluation setup, to compare different classification systems and to provide a more exhaustive dataset compared to the others which, as they report in the paper, "are not sufficiently rich to investigate opportunistic activity recognition, where a high number of sensors is required on the body, in objects and in the environment, with a high number of activity instances". As pointed out in [**?**] in fact, previously, several datasets were related to the activities which were to be classified: in those cases researchers acquired signals only from sensors positioned in specific locations, according to the task of interest. To overcome this drawback, the OPPORTUNITY dataset has been gathered from a monitored, sensor rich environment: objects from the scene were connected to acquisition sensors, while people participating to the session were equipped with on-body sensors; signals collected from the latter type of measurement units will be described in section IV. This particular dataset has been fundamental over the past years, since it provided an heterogeneous and complete set of time series, perfectly suitable for different studies in the HAR domain. In [**?**] they presented it as a *benchmarking dataset*: as a demonstration, they provided the results obtained with four classification techniques (*k-nearest neighbours, nearest centroid, linear discriminant analysis, quadratic discriminant analysis*) and they compared them with some other contributed models. Since our analysis is based on this dataset, to make the discussion consistent we introduce a collection of interesting works that use the same dataset for their evaluation.

The authors in [**?**] propose an exhaustive framework which, besides the standard preprocessing on the activity data sequence, e.g. filling of the gaps via interpolation and data normalization, presents also a solution for the well-known class imbalance problem [**?**]. Moreover, they also include a post-processing procedure after classification, consisting of a smoothing operation along the temporal axis and of a strategic fusion procedure to integrate prediction sequences from different classifiers, in order to reduce the risk of making an erroneous classification. The classifiers used in this work consisted in a 1-layer neural network (1NN) and a Support Vector Machine (SVM, complete overview of this tool in [**?**]).

We can see an example of CNN applied to HAR in [**?**] where, in order to evaluate their model, they use also the Hand Gesture dataset [**?**]. In this configuration the authors designed a network with three consecutive convolutional blocks; the first two are constituted by a convolutional layer, a rectified linear unit layer (ReLU) and a max pooling layer. The latter instead is constituted of a convolutional layer followed by a ReLU and a normalization layer. The reason behind this collection of layers is that, while the first ones identify *basic* movements in human activity, higher layers characterize the combination of these basic movements. At the end of these core blocks, two fully-connected layers are added in order to complete the architecture. It's important to notice also that here a sliding window strategy has been adopted to segment the time series: in this way the prediction is not focused on a single sample but is associated to a temporal matrix; the corresponding label is determined by the most-frequent label among the matrix of samples.

As a form of regularization, in [**?**] the authors add to the CNN also a dropout layer (more on this technique in [**?**]); moreover, they also put together a recurrent neural network in order to exploit time dependencies between different samples. In particular, they built two flavours of LSTM networks: one that contains multiple layers of recurrent units connected forward in time, and another which exploits dependencies either backward and forward with respect to the time-step of interest. This last configuration in particular gives the best results when applied to the OPPORTUNITY dataset in terms of F_1 measure (described in details in section VI).

In conclusion, recent work in [**?**] provides a complete comparison among different features extraction and classification techniques; first, they demonstrate how the method of hand-crafted features gives poor results with respect to deep learning mechanism. Then, they make a step forward with respect to the work in [**?**] as they create an hybrid architecture, comprehensive of both convolutional and a recurrent LSTM layers. In this way they exploit correlation among samples of a single window (as in [**?**]), searching for significant features; also, with LSTM they exploit correlation in time among independent windows. This combination results in a slight improvement with respect to those configurations where CNN and LSTM layers are not implemented jointly.

In our work we implement this hybrid model, introduced in [**?**], and some others, varying the number of layers. These models are then tested in different types of tasks and compared.

III. Processing Pipeline

We start off our analysis by preprocessing the collected signals within the MATLAB environment: we chose this framework because we find it is easier to operate with matrices. In this first step we import the data collected by sensors, which are given as *'.dat'* files, then we select the signals from on-body sensors and discard the others, so we replace the missing values by means of interpolation and, at last, we store them as *'.mat'* files.

Then, this time with python, we import the preprocessed data and make the dataset suitable for the classification task: this consists for example of segmenting data into windows, scaling and normalizing raw signals.

Finally, after the last step of preprocessing, we define and train a suitable learning model. This is respectively done for both the locomotion activity and gestures recognition, i.e. with two different sets of labels. The model, which is forced to learn also the null class together with the actual movements, is then compared to a different system where two stages are carried out; in that case, the first one has the purpose of detecting activity while the second one classifies the type of movement, if detected. Figure 1 shows a schematic depiction

of the comprehensive pipeline, which we are going to present in the following sections.

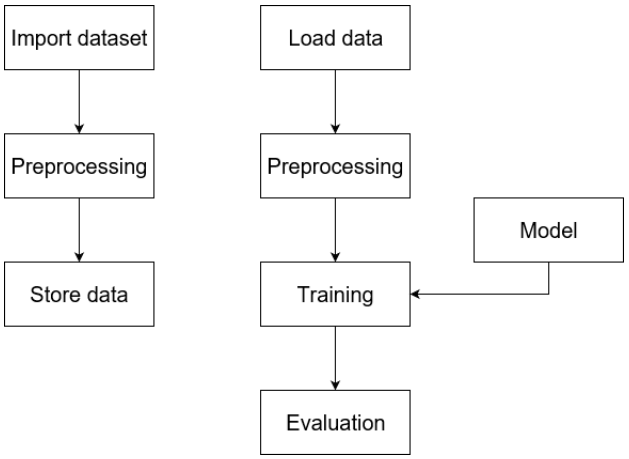

Fig. 1: Framework Pipeline. On the left, the 3 block represent the process followed within the MATLAB framework; on the right instead, the pipeline for our python code.

IV. Signals and Features

The OPPORTUNITY dataset, succinctly introduced in section II, has been collected from four subjects accomplishing different Activities of Daily Life (ADLs). As highlighted before, both the environment they moved in and the participants were meticulously monitored. The process of acquisition consisted in 5 consecutive runs (named ADL_1 to ADL_5) that followed a predetermined script, plus a sixth run consisting of 20 repetitions of each activity present in the script (named *Drill* run). Each vector of samples corresponding to a single time-step is assigned with a distinct set of labels: in the following we'll refer to Task A when we consider high-level modes of locomotion (*Standing, Walking, Sitting, Lying*), while we'll refer to Task B for more specific arm gestures (17 in total). To these tasks we need of course to add the *Null Class* that we mentioned earlier in this paper: specifically, this label represents the state in which the participant does nothing (or something that is not considered among the listed classes).

The wireless sensors worn by the subjects (IMU - Inertial Measurement Unit) provide acceleration among the three-axes, rate of turn, magnetic field and orientation information; in addition, 12 accelerometers were placed on the subjects' parts of the body sensible to movements (arms, back, hips and feet). Figure 2 shows how these sensors were placed on the body. All these devices acquired together a total of 145 distinct channels but, for the purposes of our work, we base the analysis only on on-body sensor signals, considering just 113 channels.

Since sensors were wireless, the collected signals could present some missing data, which are indicated with NaN values. These manifest mostly as bursts and thus, in the preprocessing phase, we performed spline interpolation with a cubic polynomial; however, this type of interpolation ends up

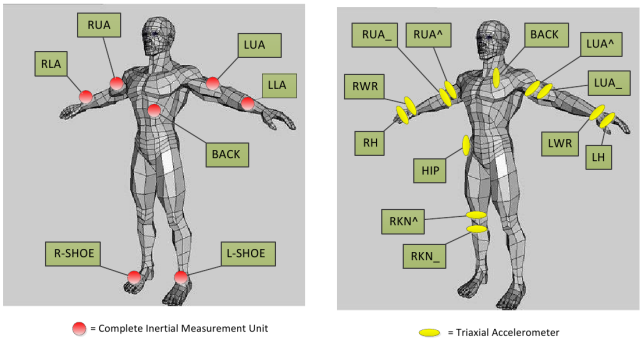

Fig. 2: Sensor Locations. The two images show the locations of on-body sensors for the measurements of the OPPORTUNITY dataset. The first image shows complete *Inertial Measurement Units*, while the second shows *Triaxial Accelerometers*.

meaningless if more than the 30% of data is missing. For this reason we had to discard all the three columns corresponding to one of the physical devices: this led us to work with 110 channels. Since we noticed that the head and tail of the measurement sessions correspond to a transient where most of the sensors are turned-off, we decided to discard them. In this way we ensure that the interpolation phase provides consistent results; the subsequent step consists in normalizing each column with respect to mean and variance. After that, in order to instruct the framework on one subject, we stack sessions from ADL_1 to ADL_3 and *Drill* as a first step to create our training set; then, we assemble also ADL_4 and ADL_5 to build the test set.

To conclude the preprocessing phase, finally we decided to follow the same procedure as in several works we cited earlier: we apply in fact the *sliding window* technique on the datasets, obtaining a tensor of windows constituted of 15 samples (500 ms), using a stride of length 5. In our case, we decided to assign to each window the most frequent label: this doesn't constitute a problem per se, even when changing the size of the sliding window, as long as it is kept short enough for being representative of a movement.

We found in fact that the choice of window size and stride is fundamental in order to obtain good results: we observed that a window too short can't represent a single gesture with fair precision and, on the contrary, if the window is too large we could include two distinct movements (or modes of locomotion), leading to non significant labelling. The choice of the size of the displacement separating two consecutive windows can guarantee a good trade-off between windows diversity and dataset population.

In the first phase of our project we also tried to create a reduced dataset of features: since each accelerometer returns the acceleration value in the direction of all the three axes, we tried to substitute these three values with a unique value representing the *mean acceleration* measured by the sensors. In our first experiments, however, this led to poor results: with respect to the simulations where all features were fed

Model name	Convolutional	Recurrent	Fully connected
Conv	3	0	2
Conv1DRecurrent	1	2	2
Conv2DRecurrent	1	2	2
ConvDeepRecurrent	3	2	2

TABLE 1: Model architectures. The number of different layers stacked in each models is here reported. 'Conv' stands for 'Convolutional'.

to the model, the configuration using the reduced dataset was affected by almost a 5% loss in terms of accuracy.

V. Learning Framework

One of the main problems in Human Activity Recognition is handling inactivity. Since we segmented our signals with a sliding window approach, we resulted in having portions of signals associated to a unique label: this label is null when the subject is idle during most of the time window. Thinking of a real-time recognition system, we can implement two different strategies to deal with this class, which are reported below in V-A and V-B.

A. One Shot Classification

This classification technique uses just one model to predict whether there is activity and of which type: e.g. an n-class classification in this case becomes a (n+1)-class classification, because even the *Null Class* is considered as an activity. We think that this process fits particularly simpler classification tasks, because the number of operations is constant during each time window: this means that even when the subject is idle, the system has to perform the same operations, with the same energy consumption.

B. Two Steps Classification

As an alternative, the process can be split into two steps: one to detect activity and one to recognize it when present. In practice the first step receives signal windows and decides if the subject is active or inactive, performing thus a binary classification; then, only if it's labelled as active, the signal is passed to a second model which tries to recognize the activity. This approach could be particularly appealing in real-time systems, especially if the detection phase has a low number of operations: considering that in daily life recordings one is mostly idle, this leads to a decreased energy consumption. A major drawback though is that the final accuracy of the model is affected by the errors made by both the models: even having a good classification model would be inefficient if it were passed, erroneously, mostly inactivity signals. This suggests that there's a trade-off in activity detection task between accuracy and energy consumption. Luckily enough this phase could also be effective exploiting non deep learning techniques, which anyway aren't explored in our work.

To test our pipelines we implemented 4 different models, named *Convolutional, Convolutional1DRecurrent, Convolutional2DRecurrent* and *ConvolutionalDeepRecurrent*. The four of them, reported by increasing complexity, use some convolutional layers and, exception made for the first one, a couple of recurrent layers with LSTM units. A superficial comparison between their architectures is given in table 1, where we specified the number of layers per type. These models are tested and compared on different tasks so, to use them properly, the last fully connected layer of each one has the number of units required by the specific task, which depends on the pipeline selected. The final choice of all the hyper-parameters (size of convolutional kernels, size of pooling windows, number of cells per LSTM layers, etc) has been done via fine-tuning and consecutive trials: other options could have been grid search or Bayesian learning, however we didn't investigate them in this paper.

Among many experiments, we also tried to add an SVM layer in order to boost our performances: however, considering that we obtained just a slight improvement, the additional computational effort needed to evaluate the features wasn't worth it.

VI. Results

As in most of the works mentioned in section II, besides accuracy, we used F_1 measure to estimate the goodness of our models. Defining precision and recall as:

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN} \tag{1}$$

the F_1 measure is evaluated as the harmonic average between the two (in the previous formula we have *TP = true positive*, *FP = false positive*, *FN = false negative*). In particular, since we deal with a multi-class problem we need to add a measure of weights to the F_1 equation:

$$F_1 = \sum_i 2w_i \frac{p_i \cdot r_i}{p_i + r_i} \tag{2}$$

where the weights are defined as the number of samples of a particular class divided by the total number of samples $w_i = \frac{n_i}{N}$. The weighted measure can help also with the class imbalance problem that we outlined in the previous section; we must highlight however that our models are still trained on an imbalanced dataset, so in our opinion this could provide only a minor improvement.

In Figures 3 and 4 we present the results regarding task A (modes of locomotion, high level movements) for each participant of the experiment: in those configurations we wanted to see if there was one architecture that evidently outperforms the others. As we can see, unfortunately that is not the case since among all the frameworks that we tested the differences in terms of weighted F_1 measure is negligible. The variation that we can clearly see is among distinct subjects, since they are all studied separately: S_1 for example performs better in both the configurations, while S_2 is typically the worst. Considering that we consistently performed the same procedure independently from the subject, in this case this discrepancy could be due to a problem occurred during data collection.

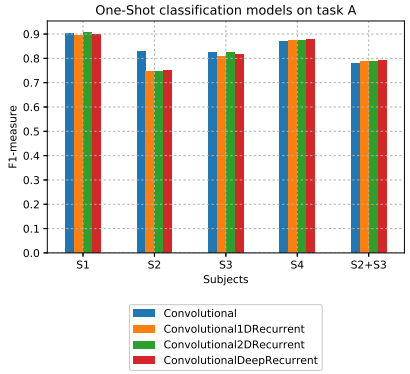

Fig. 3: Task A : One-Shot Classification

With an F_1 value of ~ 0.91 for the *one-shot* scenario and ~ 0.94 in the *two-steps* (if we just consider S_1), we can definitely say that the results presented in [**?**] are outperformed by these more powerful deep learning models; with respect to the other subjects, we can say that we obtained state of the art results. In particular it's interesting to see that, when we consider the *Null Class*, the best model with S_2 is represented by the neural network without recurrent layers (Figure 3); when the *Null Class* is discarded, instead, the hybrid model that combines convolutional layers and LSTM provides the best results (Figure 4). In our opinion the reason is that in certain cases the *Null Class* has to be intended as noise so, when the windows assigned to that class are removed, the LSTM can exploit and reveal the time correlation among different samples more clearly. In addition, we tried to train our architectures on S_2 and S_3 jointly, using ADL_4 and ADL_5 of both subjects as test set: as is shown on the last column on the right in Fig 3 and 4, the results are not brilliant. In fact, we obtained a kind of average performance between the results of S_2 and S_3, in line again with what presented in [**?**]. Finally, as we can see, for task A the use of the *two-step* approach definitely improved performances for all users.

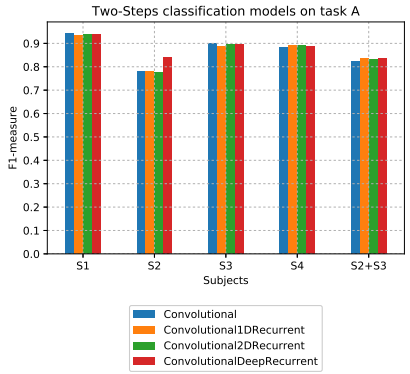

Fig. 4: Task A : Two Steps Classification

Despite the promising results on high-level movement classification, when we used our models to do the more precise and low-level classification of task B, the *two-steps* configuration wasn't effective as we hoped - Figure 6. Still, in both architectures, S_2 is the one that performs the worst, while with S_1 we are able to extract the best results; moreover, even in this case we haven't noticed considerable differences among the different neural network set-ups that we used: this suggests us, as we briefly outlined in previous sections, increasing the complexity of the architecture in order to obtain astonishing results, isn't the right path to follow.

We can see in Figure 5 that implementing one convolutional layer jointly with 2 LSTM followed by 2 fully connected layers, contributed positively on the performances of *one-shot* model, for all subjects considered. The considerations done for Task A, regarding the behaviour of the models fed with S_2 and S_3 together, remains valid also for Task B.

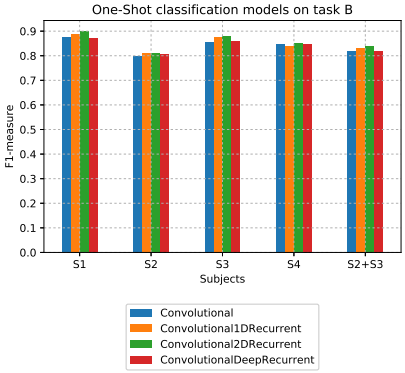

Fig. 5: Task B : One-Shot Classification

In general however, despite S_2, the results for the configuration in Figure 6 are better with respect to what presented by Chavarriaga in [**?**]; regarding what shown in Figure 5 we can say that the results are compatible with the state of art, reaching a value of $\sim 90\%$ for S_1.

In conclusion we found out that, even with powerful deep learning models, if we discard the *Null Class* the results are not appealing when we have to deal with many classes (in addition, affected by the imbalance problem); this, however, shouldn't discourage researchers to go down this path. Considering that for Task B we reached almost ~ 0.92 in the detection step for S_1, this suggests us that building two distinct models for detection and classification respectively could lead to promising results: the design of a specific model for each step could focus the effort on a single purpose, in this way leading to ad-hoc and less complex networks. In our work however we wanted to see how the problem could be tackled with the solutions proposed by the state of the art, still trying to reduce computational effort. Just to make some numeric comparison, while the Hybrid model proposed in [**?**] counts 17 millions of trainable parameters, our most complex model reduced this number more than an half.

VII. Concluding Remarks

In our project we realized from scratch two different pipelines to perform activity recognition. A comparison between the two is carried out using different models, which are inspired to the best ones in literature. This helped us

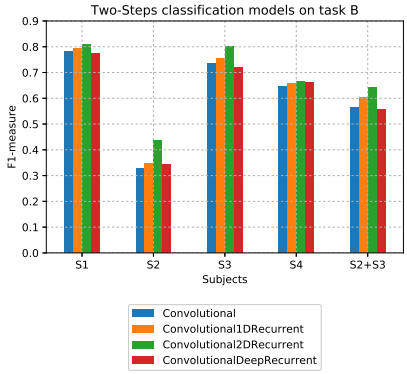

Fig. 6: Task B : Two Steps Classification

understanding that there isn't a clear best choice from an accuracy point of view; in fact, in locomotion classification, results were slightly worse when the *Null Class* was considered, while in gesture recognition it was the opposite, with a sensible decrease in accuracy when inactivity wasn't considered. Despite our deep study, still one must examine case by case when to use one model or the other, with respect to the discussion that we developed in section IV.

Future works could try to implement effectively the Two-Steps classification pipeline, by placing the classification model after the detection one: we tried ourselves, but results weren't satisfactory enough to be reported here. A good approach to the problem could be trying to understand how the accuracies of the two single models add up when in cascade. In this paper in fact, when comparing the two pipelines, we didn't keep into account the precision of the detection model in the Two-Step architecture; we considered instead only the classification performances without the null class to provide an estimate of the accuracies achieved using only deep learning models.

Another thing that is left to be done, is to try to solve the class imbalance problem; a good proposal could be trying to replicate what has been done in [**?**]. In our work we only dealt with it for evaluation purposes, but something could be done also to improve the training phase.

This project was very helpful to us because we actually learned to code in python, we had to face different obstacles and learned that results are never what one would expect: we hoped in better results, for the advanced techniques used, and expected similar networks to have almost equal performances, which wasn't the case throughout our work.