Deep Learning Models for Gesture Recognition with Automatic and Manual Feature Extraction

Luca Dal Zotto[†] , Giuliano Squarcina[‡]

***Abstract*—The capacity to recognize human activities has become increasingly important in last years and is going to be even more relevant for the future, due to the widespread of smart/IoT devices, whose utility consists in human activity recognition. In order to tackle this task various deep learning models, sensors selection, data segmentation and hand-crafted features were used in past researches, but always separately, without exploiting the synergy resulting from the combination of these techniques. We are going to fill this lack, combining together the best ideas found so far in the literature, with two main goals: find if it exists a model capable to significantly improve the best performances found so far (regardless of computational cost) and find a model capable to preserve good performances using the least amount of features. In both cases, we test the models also with a noisy test set, showing their robustness to artifacts. Regarding the first goal, we found a deep learning architecture in line with the best performances existing; regarding the second goal, we found that it is possible to reduce the number of features used by 60%, suffering a reduction in performance (F1-score) smaller than 2%. This could be useful for a wide range of devices which need to classify human gestures with a good reliability but with low energy consumption and data requirement.**

***Index Terms*—Gesture Recognition, Inertial Measurement Units, Orientation Invariant Coordinate System, Hand-Crafted Features, Convolutional Neural Networks, Recurrent Neural Networks.**

I. Introduction

The number of devices equipped with inertial motion sensors has incredibly increased during the last years. The technological progress allows to build smaller sensors with a better efficiency and precision. As a consequence, new products have been introduced into the medical field, where, for instance, they are extremely useful to monitor patients affected by motion-related diseases or to follow more effectively a path of rehabilitation. Thanks to the contained dimensions and to the affordable cost of some products, inertial sensors are present in a great variety of smart devices, such as smartphones but also sport-watches, fitness bands, and even earphones. However, in order to build reliable tools, this hardware progress should be accompanied by a software development, with programs that are accurate in different situations and robust to noise artifacts.

Clearly, the complexity of these devices changes depending on the context where they are applied. For smartphone applications, the target is to keep the energy consumption low, still guaranteeing a reasonable accuracy to compute basic information, such as daily steps or to automatically detect the sport activity the user is practicing. On the other hand, going back to the medical application, the main objective is the reliability of measurements, even if this implies to use a more obtrusive device, or possibly, more devices forming a system.

All these aspects constitute the motivation behind this piece of work: the goal is to explore the performances of some advanced deep learning architecture when combining two or more sensors, and finding out how much these models are affected by noise.

For this purpose, the Opportunity dataset [1] represents the ideal benchmarking dataset. In a nutshell, it consists of different recordings of five subjects when performing activity of daily living, such as opening and closing a door, a fridge, a dishwasher, drawers, moving a cup or cleaning a table. What makes this dataset extremely suitable for different analysis is that the subjects wore a large number of sensors, such as inertial measurements units (IMUs) placed on a jacket and on the shoes and other tri-axial accelerometers in different parts of the body. The task is to correctly predict the gesture performed by a person from these raw signals.

This great amount of data allowed us to perform a wide range of experiments. Firstly, we were able to explore how much the performance of a model is affected by selecting a subset of sensors instead of using all the available ones, or which is the best location for a sensor to recognize a specific kind of activity. For instance, the accelerometers and gyroscopes placed on the arms may be relevant for detecting the gestures listed above, while the IMUs located in the shoes may give significant information to identify the mode of locomotion of a person.

Secondly, feature engineering is considered one of the most important and critical factors in ML projects [2]. Therefore, for this project we have compared the effectiveness of manual feature extraction against automatic methods that rely on state-of-the-art architectures (Convolutional and Recurrent Neural Networks). The former often requires domain knowledge, while the latter exploits the hierarchical representation of features within the layers.

Since signals coming from the real world are prone to noise, in the original challenge, the authors of the dataset proposed a task focused on methods robust to artifacts. For this purpose, they extracted from the original dataset a second test set, to which rotational noise was manually added. Moreover, the participants of this challenging task were also asked to use only

[†]Department of Mathematics, University of Padova, email: luca.dalzotto.1@studenti.unipd.it

[‡]Department of Mathematics, University of Padova, email: giuliano.squarcina@studenti.unipd.it

the signals coming from the motion jacket sensors. Inspired by this task, the third important goal of our work consists in quantifying how much our different learning architectures are affected by rotational noise comparing their performances on the two test sets.

This report is structured as follows. In the next section, some related works are presented, including a couple of papers which inspired some ideas of this project. In Section 3, we will give an high level description of the processing workflow, which is explored in more technical details in Section 4 for what concerns the pre-processing steps and feature analysis and in Section 5 with a presentation of the learning architectures considered. In Section 6 the main results are reported and in Section 7 we will make some final remarks and considerations.

II. Related Work

Gesture recognition is a topic that engages a surprisingly high number of research fields, in particular Human Computer Interaction, Computer Vision, Machine/Deep Learning and Robotics. The possible applications are even more: gesture recognition can be usefully considered for environment control, sign language translation, augmented/virtual reality and most importantly in the healthcare field.

In a paper of 2010 [3], Wachs et al. considered a system that can interpret a user's gestures to manipulate objects within a medical visualization environment. In that case, the authors focused on a vision-based approach, since the video signal captured by some cameras was used for this purpose. Dynamic navigation gestures are then translated to commands based on their relative positions on the screen. The features were selected through a probabilistic neighborhood search and the algorithm used for classification is a soft clustering technique.

Beside computer vision, a completely different approach to gesture recognition is represented by sensors such as accelerometers and gyroscopes. In [4], Zhao et al. proposed *Gemote*, a smart wristband-based hardware/software platform for gesture recognition and remote control. The idea was to create a device with an affordable cost, appropriate for comfortable daily wear and open source for third party research. On the software side, it employs a continuous gesture segmentation and recognition algorithm which accurately and automatically separates hand movements into segments.

In 2010, Charrivaga et al. proposed '*Opportunity*', the first publicly available dataset on human activity recognition. The intention was to prevent the fact that a wide range of classification methods are tested on custom datasets acquired in very specific experimental setup. Therefore, the authors recorded a large number of realistic daily life activities in a sensor-rich environment, including body-worn sensors, object sensors and ambient sensors. Moreover, the dataset comprises different kinds of label: from low-level action classes (like 'move', 'release', 'open', 'reach') to high-level activity classes (such as 'relaxing', 'coffee time', 'sandwich time') besides the mid-level labels (e.g., 'open door', 'close fridge', 'drink from cup') and usual locomotion labels ('stand', 'sit', 'walk', 'lie'). For these reasons, this dataset is open to an extremely wide range of experiments. Even the original 'Opportunity Challenge' comprises different tasks, such as modes of locomotion recognition, automatic segmentation, gesture recognition. In the reference papers, the results obtained with four famous classification techniques (k-Nearest Neighbors, Nearest Centroid Classifier, Linear Discriminant Analysis, Quadratic Discriminant Analysis) are reported to be used as a baseline.

In [5], Valenti et al. used quaternions to translate a system of coordinate related to an object into an invariant frame of reference.

In [6], Hammerla et al. explored deep, convolutional and recurrent approaches across three datasets, including the Opportunity one. In that work, the authors described how to train recurrent networks introducing a novel regularization approach. They found out that bi-directional LSTMs outperform the current state-of-the-art results on Opportunity.

Besides differences in the final models architectures, in the literature we may also find different feature extraction approaches with the Opportunity dataset. For instance, in [7], Anguita et al. chose to define manually some features starting from raw signals and then trained a multiclass Support Vector Machine. On the other hand, in [8], Zeng et al. proposed a method to automatically extract discriminative features for activity recognition based on Convolutional Neural Networks (CNN).

Our contribution consists in combining together some of the relevant results previously listed to get out the best from the data. More in detail we:

- tackle the challenge presented in [1] with its raw data;
- remove gravity acceleration from from accelerometers and move into an orientation invariant frame as in [5];
- built hand-crafted features as in [7];
- segment the data and apply deep Neural Network architectures as in [6].

As far as we know, no previous research utilized all these methods together.

III. Processing Pipeline

The aim of this project is to correctly classify a wide variety of daily living gestures. In Fig. 1 the adopted workflow is outlined. Before building up the models, some specific pre-processing steps have to be carried out. The pipeline of this first part can be summarized in the following points:

1) data exploration;
2) missing values handling;
3) gravity filtering;
4) conversion into a coordinate invariant system;
5) data segmentation.

Exploring the data is crucial to understand in depth what we are dealing with. In this specific case, the opportunity dataset is split in six recordings for each subject: five of them are activity of daily living (ADL) runs consisting of actions usually carried out in the morning. For the remaining one,

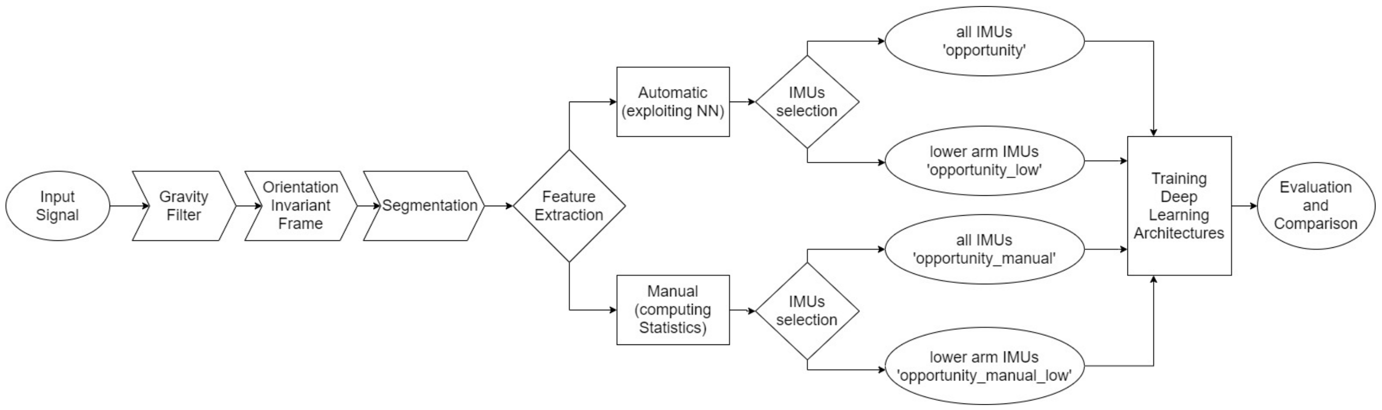
Input Signal
Gravity Filter
Orientation Invariant Frame
Segmentation
Feature Extraction
Automatic (exploiting NN)
Manual (computing Statistics)
IMUs selection
IMUs selection
all IMUs 'opportunity'
lower arm IMUs 'opportunity_low'
all IMUs 'opportunity_manual'
lower arm IMUs 'opportunity_manual_low'
Training Deep Learning Architectures
Evaluation and Comparison

Fig. 1: Project workflow.

called DRILL run, the subjects were asked to perform multiple times a specific sequence of actions, in order to record a large set of gesture instances.

After having combined these files, we checked the most relevant statistics of the dataset. For this project, we considered only the IMUs placed on the *motion jacket* (Fig. 2). We found out that the dataset presents a specific pattern of missing values: they are present only at the beginning and end of the runs. This is due to the fact that the recordings related to the shoes' IMUs start in advance and are stopped later with respect to the motion jacket activity. For this reason, we will not recover these values through imputation, but we will simply drop the recordings having missing values. Note that there are no missing values in the middle of a run, thanks to the fact that the sensors of the motion jacket do not use wireless technology, so there are no interruptions due to connection failures.

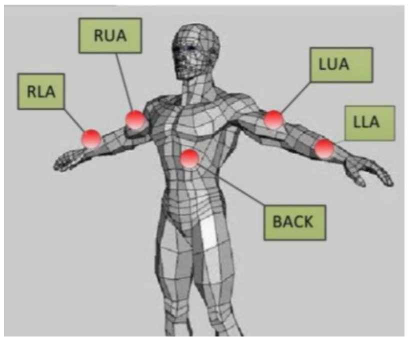

Fig. 2: Location of the IMUs of the motion jacket.

Then we filtered the acceleration of gravity from the accelerometers, so that the resulting measurements of those sensors are all due to human activity. Since the sensors are free to rotate in the space, their coordinate system is not aligned with the body frame or with a fixed frame. For this reason, we used the quaternions to move from the sensor frame into an orientation invariant frame. This has the advantage of increasing the interpretability of the variables: for example, aligning the measurements with a fixed coordinate system can be useful if we want to compute the horizontal or vertical component of acceleration of a sensor, or the angle of a rotation with respect to the floor. For instance, these advantages are extremely important in case we want to perform a manual feature extraction from raw signals.

Once the dataset is cleaned, since we are dealing with time series, we need also to segment it into time windows. We used half-overlapping windows of 30 samples (1 second). This duration seems reasonable knowing that the average duration of the actions is larger than 3 seconds.

After these pre-processing steps, we built the final four datasets to be fed into the deep learning models. Firstly, we created a version performing a manual feature extraction on top of the processed signals. Then, for each of these two copies we further saved two versions: one with all the IMUs of the motion jacket, the other just with the two IMUs placed on the lower arms. For all these datasets, we trained different learning architectures, and after selecting the best one, we made some considerations about the differences in term of performance of these approaches. In the next sections, all the steps will be described thoroughly.

IV. Signals and Features

The 'OPPORTUNITY Dataset for Human Activity Recognition from Wearable, Object, and Ambient Sensors' is a dataset devised to benchmark human activity recognition algorithms. For this project, we extracted a subset of this dataset, selecting only five body worn IMUs (those placed on the motion jacket). All these units provide readings of 3D acceleration, 3D rate of turn, 3D magnetic field, and orientation of the sensor with respect to a world coordinate system in quaternions. Since these values were acquired with a wired system, they provide no data loss. Moreover, the placement of the sensors is very reproducible among users and recording runs, as the sensors were integrated within a jacket.

In the official documentation of the dataset, some quality annotations are pointed out. Here we highlight just the following two:

- the boundaries of gestures are sometimes hard to be determined objectively: for instance the action 'open door' should start at some point when the person is approaching the door, but determining the exact instant when the opening gesture begins is extremely challenging. Rather a smooth transition between activities should be assumed;
- data is provided from different independent sensors which were aligned in post-processing. However, a jitter of the order of 100ms may be present between the channel of different systems.

To give some other useful technical details, the sampling rate is 30Hz and the acceleration measurement unit used is $\boldsymbol{g}$ ($9.81 m/s^2$).

When deciding how to merge the different runs, we have been consistent with the sets suggested in the reference paper of the dataset, where the reader may also find more details about data acquisition. For what concerns gesture recognition, the authors proposed two different tasks with two distinct test sets. The first one is a simple gesture classification task using the fourth and fifth runs of subject 2 and 3 as test set. On the contrary, the other task aims to evaluate the noise robustness of the models: indeed, the test set in this case consists of the last two runs of subject 4, but the signal has been manually modified adding rotational noise. For the evaluation of the models during the training phase, we opted for the hold-out set approach, since the cross-validation may be computationally expensive and not strictly necessary for such a large dataset. So, we extracted a validation set selecting the last two runs of subject 1. All the remaining runs gave shape to the training set (Table 1).

	Subject 1	**Subject 2**	**Subject 3**	**Subject 4**
ADL 1	train	train	train	train
ADL 2	train	train	train	train
ADL 3	train	train	train	train
ADL 4	*valid*	*test*	*test*	*test_noise*
ADL 5	*valid*	*test*	*test*	*test_noise*
DRILL	train	train	train	train

TABLE 1: Composition of training, validation and test sets reflecting those of the 'Opportunity challenge'. Here 'test_noise' is the second test set described above.

A. Pre-processing and data exploration

At this point, we decided to subtract the gravity from the accelerometer signals and to move into a rotation invariant frame of reference, taking inspiration from the work of Gadaleta and Rossi [9]. Even though the idea is the same, the strategy to reach the objective is different, since here we exploited the quaternions. Basically, they are an alternative to Euler angles that allows to represent any rotation in the 3D space using one axis and an angle. The advantage of this mathematical tool is that it is not affected by discontinuity issues (while Euler angles do when approaching 90 degrees angles) and from an algebraic point of view, it presents some interesting property. For more details, the reader may consult [10] or other related sources. Anyway, what is really of interest for this project is that from the quaternions we can reconstruct the rotation matrix $\boldsymbol{R}(\mathbf{q})$ with the formula

$$\begin{bmatrix} q_4^2+q_1^2-q_2^2-q_3^2 & 2q_2q_1-2q_3q_4 & 2q_2q_4+2q_3q_1 \\ 2q_3q_4+2q_1q_2 & q_4^2-q_1^2+q_2^2-q_3^2 & 2q_3q_2-2q_1q_4 \\ 2q_1q_3-2q_2q_4 & 2q_1q_4+2q_2q_3 & q_4^2-q_1^2-q_2^2+q_3^2 \end{bmatrix},$$

and use it to move from the sensor frame to a fixed frame and vice versa. If we denote with $\boldsymbol{z}$ a vector in the fixed frame and $\boldsymbol{z'}$ the same vector but in the object frame of reference, the following relations hold:

$$\boldsymbol{z} = \boldsymbol{R}(\mathbf{q})\boldsymbol{z}', \quad \boldsymbol{z}' = \boldsymbol{R}(\mathbf{q})^T\boldsymbol{z}.$$

Using these formulae, we can move the gravity acceleration into the sensor frame by setting $\boldsymbol{z} = (0,0,1)$ and also the procedure to move into a fixed reference system is quite easy: we just have to rotate also the $\boldsymbol{x} = (1,0,0)$ and $\boldsymbol{y} = (0,1,0)$ axes into the sensor frame, and project the original signal along these rotated vectors. Note that this operation is performed only for the accelerometers and gyroscopes. Its effectiveness can be observed just looking at the Fig. 3.

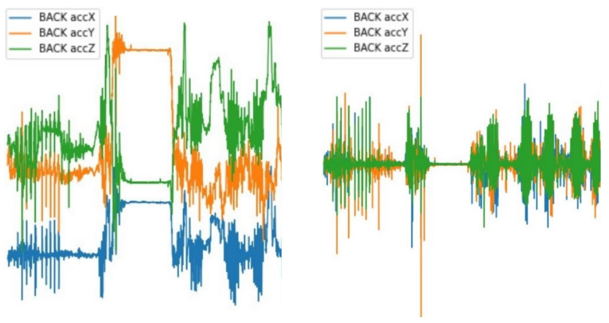

Fig. 3: Accelerometer signal before and after pre-processing.

After having processed the dataset, we computed some statistics in order to understand how to behave from here on. In particular, the following information should be pointed out:

1) the size of the dataset;
2) the range of values of the features;
3) the frequency of the different classes;
4) the average duration of each gesture.

The training set includes 622,511 recordings, the validation set 62,747. The test set without noise have 117,601 samples, while the test set with rotational noise 51,465. All this sets have 65 variables, but if we include only the accelerometer and gyroscope signals (discarding the magnetometer and quaternions), the variables are 30, since there are five IMUs in the jacket: one placed on the back, two on the lower arms and two on the upper arms.

Once the gravity acceleration has been removed, both accelerometer and gyroscope signals have mean equal to zero and almost all the values are in the range [-1,1]. Since the variables have approximately the same order of magnitude, it is not necessary to normalize them.

From the analysis of the class frequency, we can note that a large portion of the samples belongs to the Null class (represented with the label ‘Other’): in the two test sets they are more than 80%. This is reasonable since the gestures selected have a limited duration and are not performed continuously by a person. Having a highly unbalanced dataset implies some consequences, especially when it comes to evaluating a model: clearly the fraction of correctly predicted samples is not the most informative metric. In the reference paper, the author suggested to use two metrics that are less affected by class unbalanced: the weighted F-measure, defined as

$$F_1 = \sum_i 2 * w_i \frac{\text{precision}_i \cdot \text{recall}_i}{\text{precision}_i + \text{recall}_i},$$

where i is the class index and w_i is the proportion of samples of class i, and the weighted area under the ROC curve:

$$AUC_{\text{total}} = \sum_i w_i AUC\left(c_i\right).$$

Moreover, it will be useful to evaluate the models disregarding the samples belonging to the Null class. In Figure 4, we can observe the class frequency in all the datasets, excluding the label ‘Other’.

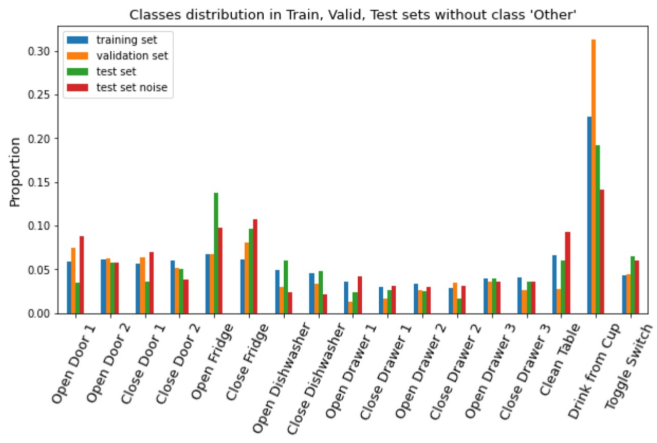

Fig. 4: Class frequency exluding ‘Other’.

Finally, we computed the average duration of each gesture in order to have an idea of the length of the time window to be selected. Some gestures are longer than others: for example, ‘Clean table’ on average involves 137 samples, or ‘Drink from cup’ 199 samples, but some gestures are shorter, like ‘Toggle Switch’, that on average has 49 samples. It should be noticed that the average duration of the class ‘Other’ is 228, so the average time between two consecutive labeled gestures is approx. 7.5 seconds. The global average duration of a gesture is 97.54 samples.

The above information is very useful when it comes to choose the windows size for segmentation. We found out that using half-overlapping windows of length between 24 and 30 samples guarantees good performances.

B. *Creating datasets*

Up to this point we have a dataset with signals coming from the five IMUs of the motion jacket. In what follows, we are going to consider only the accelerometer and gyroscope data, discarding the magnetometer, which we do not expect to be relevant, and the quaternions, which have already been used. The dataset obtained in this way (we will call it simply ‘opportunity’) has 30 variables.

Since we are dealing with gesture recognition, a first idea is to create a dataset which considers only the two IMUs located in the lower arms, which should be the more informative ones. Incidentally, we could mention the papers [8] and [11] where the authors faced the same task considering only the IMU in the right lower arm. We will call this dataset ‘opportunity_low’.

Secondly, we are also interested in comparing the effectiveness of manual vs automatic feature extraction. To better explain this point, recall that for generating the features we may create them by manually or we may rely on a Neural Network architecture (CNN is often very good for this task). For this reason, we made two copies of these two datasets, but in one of them we substituted the raw signal with some hand-crafted features. In the literature, it is possible to find tons of materials about this topic, however we focused on two papers, one from Anguita et al. [7] and the other from Frank et al. [12]. First of all, we computed the norm of each 3D signal, since this value quantifies the magnitude of movement (or rotation for the gyroscope) independently from the direction. Using this new vector, we computed some statistics in each time window: mean, standard deviation, minimum, maximum, first and third quartile (less sensitive to outliers compared to minimum and maximum), inter quartile range (iqr) and root mean square (rms).

Since we also wanted to keep the information about the direction of the gesture and the rotation axis, in each time window, we computed the mean of the original variables (tri-axial accelerometer and gyroscope) with respect to a fixed frame of reference. An alternative could be to explicitly compute the angle of the acceleration vector with respect to a fixed vector in the horizontal plane. Ideally, if we were able to compute the direction where a person is looking at, probably the movement with respect to that direction would have been more informative.

For this reason, we thought to use an approach similar to what has been done in [9], where basically the authors computed a body reference frame considering as forward direction the principal component of the accelerometer signal in the horizontal plane. However, in that case their work was focused on modes of locomotion, and while determining the direction of movement of a walking path is quite meaningful, this may be misleading in the case of gesture recognition (which is the main direction of the gesture ‘Open door’?). Therefore, we discarded this idea, keeping just the component of the signals with respect to a fixed frame. In Table 2 our hand-crafted features for each triplet of signals (e.g. for the

3D acceleration of an IMU) are listed.

Feature	Description
avg_z	mean vertical component
avg_x	mean horizontal components
avg_y	(x and y are two fixed directions)
mean	mean of the norm
std	standard deviation of the norm
min	minimum value of the norm
max	maximum value of the norm
q25	first quartile of the norm
q75	third quartile of the norm
iqr	inter quartile range of the norm
rms	root mean square of the norm

TABLE 2: Set of features extracted manually.

As anticipated, with these variables we built a variant of both the two previous datasets, and we called them 'opportunity_manual' and 'opportunity_manual_low', respectively with 90 and 36 variables.

Before describing the model implementation, let's recap the situation so far. We have saved these four datasets:

1) dataset 1: 'opportunity' with accelerometer and gyroscope of all IMUs;
2) dataset 2: 'opportunity_low' with accelerometer and gyroscope of RLA and LLA (Right/Left Lower Arm IMU);
3) dataset 3: 'opportunity_manual' with some hand-crafted features related to the accelerometer and gyroscope of all IMUs;
4) dataset 4: 'opportunity_manual_low' with some hand-crafted features related to accelerometer and gyroscope of RLA and LLA.

Now we are ready to train on top of these datasets some deep learning models. Recall that our intention is to perform three different experiments, evaluating:

- the effectiveness of manual vs automatic feature extraction (dataset 1 vs 3 and 2 vs 4);
- the accuracy reduction using just two IMUs (dataset 1 vs 2 and 3 vs 4);
- the noise robustness of the different models (performance on the test set vs test set with noise).

V. Learning Framework

The choice of the *Neural Network* model to be trained in each dataset is based on the following considerations [13]:

- hyperparameters (like n° *of layers* and n° *of units*) are expensive to be selected with a grid-search approach when the dataset's size is big. So it's often simpler and more efficient to pick a model with more layers and neurons than what actually needed and then use some *early stopping* strategy together with a *regularization technique* to prevent overfitting.
- large batch sizes often lead to training instabilities, especially at the beginning of training, and the resulting model may not generalize as well as a model trained with a small batch size. Default value largely accepted in common practice is 32, but we used 64 considering the datasets' size.
- the *activation functions* used are *Relu* and *Elu*. The first is the default choice, since it often guarantees good performances and speed of computation. When this is not the case, the second is used.

In the models, three types of *Neural Network architectures* are used:

- *Feed Forward Neural Network* (FFNN)
- *Convolutional Neural Network* (CNN)
- *Gated Recurrent Unit* (GRU)

CNN applies, at each convolutional layer, a *kernel*, with weights that are convolved along the input span. A kernel reuses the same weights and operates on a small portion of data. So the network connectivity structure is sparse and the computational cost is reduced with respect to fully connected NN. Moreover, CNNs have been proven to be an excellent feature extractor for motion data (see [9]).
GRUs are a gating mechanism that try to solve the vanishing gradient problem that affects standard recurrent neural networks. They managed to overcome it by using an *update gate* and a *reset gate*. The former controls information that flows into memory, and the latter controls the information that flows out of memory.

Now we present a detailed description of the models grouping together *dataset 1 & 2* and *dataset 3 & 4* because they use very similar models.

A. Dataset 1 & Dataset 2

Two different models are used:

- CNN followed by GRU and FFNN
- CNN followed by FFNN

We will refer to the first model as CNN+RNN and to the second as CNN.

CNN+RNN (Fig. 5)

In detail, we have (CL=Convolutional Layer, GRU=Gated Recurrent Unit, FL=Fully-connected Layer):

- **CL1** The first convolutional layer is 1D, so it implements one dimensional kernels (1x8 samples) performing a first filtering of the input and processing each input vector (rows of the dataset) separately. This means that at this stage we do not capture any correlation through time, but only correlation among accelerometer and gyroscope of different sensors. The activation functions are linear and the number of convolutional kernels is 128. *Batch Normalization* and *Dropout* are used to improve the performances and to avoid overfitting.
- **CL2** The second convolutional layer is again 1D but with smaller kernels (1x3) and the goal is to capture any form of correlation missed in CL1. *Batch Normalization* and *Dropout* are used as in the previous layer.

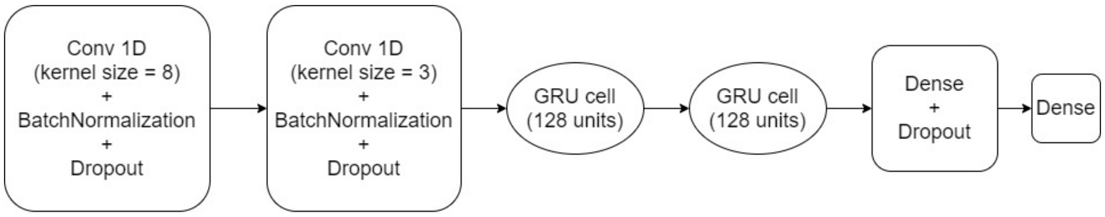

Fig. 5: CNN+RNN.

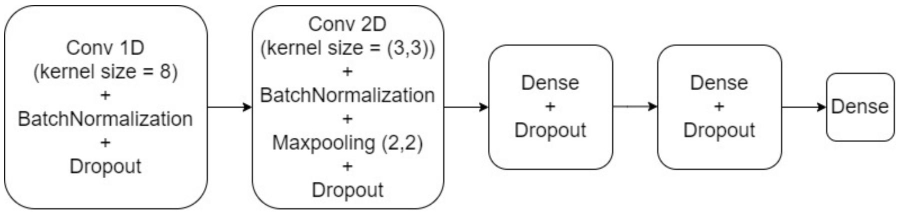

Fig. 6: CNN.

- **GRU1** The first GRUs layer uses 128 units to capture any correlation through time among accelerometer and gyroscope of different sensors.
- **GRU2** The second GRUs layer uses again 128 units to capture any correlation through time among accelerometer and gyroscope of different sensors missed previously.
- **FL1** Each output neuron of GRU2 is connected to all the 64 input neurons of this layer. A *Relu* activation function is used, followed by a Dropout.
- **FL2** Each output neuron of FL1 is connected to all the 18 input neurons of this layer (one for each class). A *softmax* activation function is used to convert each output into a probability.

The model is trained using the labels identifying the correct activity carried out in each time-frame and using as metric the *F1-score* instead of *accuracy*, because the latter is not a good metric when the dataset is unbalanced like in our case ($\approx 80\%$ of observations belong to class 0). Additionally the n° *of epochs* used is not a fixed number established in advance, but it is decided by an *early stopping* strategy, with consists in monitoring the value of F1-score through epochs and stopping if no improvement is recorded in last 8 ones.

CNN (Fig. 6)

Using same notation as above:

- **CL1** The first convolutional layer is 1D which implements one dimensional kernels (1x8 samples) performing a first filtering of the input and processing each input vector (rows of the dataset) separately. This means that at this stage we do not capture any correlation through time, but only correlation among accelerometer and gyroscope of different sensors. The activation function is *elu* and the number of convolutional kernels is 128. *Batch Normalization* and *Dropout* are used to improve the performances and to avoid overfitting.
- **CL2** The second convolutional layer is 2D with kernels (3x3) and the goal is to capture correlation through time among accelerometer and gyroscope of different sensors, since no GRU is going to be used later. *Batch Normalization* and *Dropout* are used as in the previous layer.
- **FL1** Each output neuron of CL2 is connected to all the 256 input neurons of this layer. A *elu* activation function is used, followed by a Dropout.
- **FL2** Each output neuron of FL1 is connected to all the 128 input neurons of this layer. Again a *elu* activation function is used.
- **FL3** Each output neuron of FL2 is connected to all the 18 input neurons of this layer (one for each class). A *softmax* activation function is used to convert each output into a probability.

The training procedure is the same as above, so we omit it here.

B. Dataset 3 & Dataset 4

The hand-crafted features are computed over time windows of 30 records, so the time evolution of the variables is already condensed in a single sample. For this reason only a CNN model is trained (Fig. 7). In detail:

- **CL1** The first convolutional layer is 1D which implements one dimensional kernels (1x9 samples) performing a first filtering of the input and processing each input vector (rows of the dataset) separately. This means that at this stage we do not capture any correlation through time, but only correlation among accelerometer and gyroscope of different sensors. The activation function is linear and the number of convolutional kernels is 128. *Batch Normalization* and *Dropout* are used to improve the performances and to avoid overfitting.
- **CL2** The second convolutional layer is again 1D but with smaller kernels (1x3) and the goal is to capture any form of correlation missed in CL1. *Batch Normalization* and *Dropout* are used as in the previous layer.

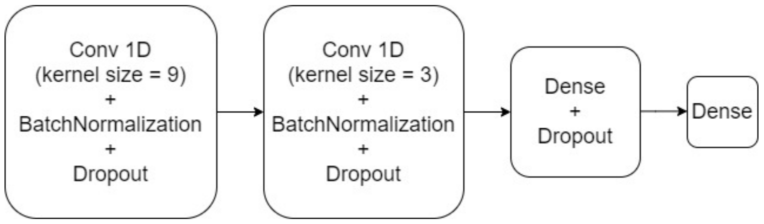

Fig. 7: CNN for Manual Features.

- **FL1** Each output neuron of CL2 is connected to all the 64 input neurons of this layer. A *Relu* activation function is used, followed by a Dropout.
- **FL2** Each output neuron of FL1 is connected to all the 18 input neurons of this layer (one for each class). A *softmax* activation function is used to convert each output into a probability.

The training procedure is the same as above, so we omit it here.

VI. Results

We start comparing the F1-score of CNN+RNN and CNN models for dataset 1 & 2 (for train, validation, test set and noisy test set defined in table 1). The main idea is to assess the effect of the GRU cells in the final performance.

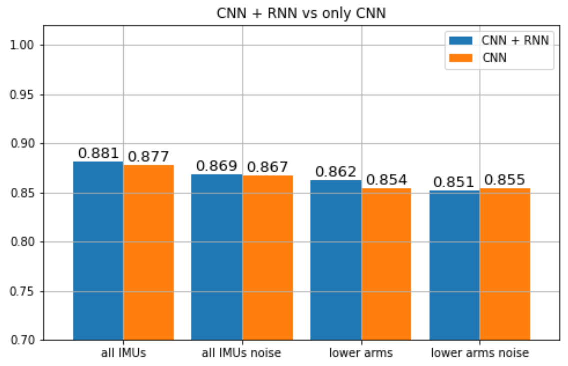

Fig. 8: F1-score for two competitive models *with* and *without* noise.

As shown in figure 8, the CNN+RNN model has slightly higher performances on both datasets, even in presence of noise (the only exception is dataset 2 with *noisy* test set). Even if the positive contribution of the RNN to the overall performance is mild, our goal here is to pick the best model found, hence the CNN model is discarded.
The next step is to compare the best model for each dataset in both the test sets. Looking at figure 9, all models show a reduction in F1-score when the *noisy* test set is used, which is expected. The crucial observation is that the reduction in the performance is limited, so all models are robust to noise, which is a very valuable feature for an *activity recognition* task with data coming from *unreliable* sensors.

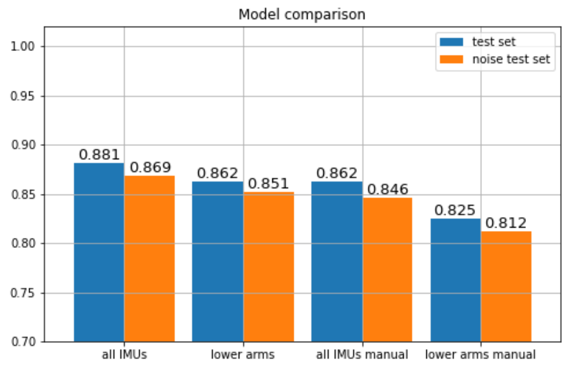

Fig. 9: F1-score in test set VS F1-score in *noisy* test set for each model.

	Without noise		**With noise**	
	Prec.	**Rec.**	**Prec.**	**Rec.**
all IMUs	0.882	0.887	0.873	0.872
lower arms	0.861	0.874	0.850	0.859
all IMUs manual	0.862	0.879	0.843	0.860
lower arms manual	0.832	0.855	0.813	0.843

TABLE 3: Precision and Recall for each dataset *with* and *without* noise.

From figure 9 it is also possible to compare the *automatic* VS *manual* feature extraction. The first is always better, but what is interesting is that the gap between the performances increases as the number of features decreases (i.e. when only *lower arms* sensors are used, the gap in F1-score is bigger). Then it is possible to compare *all IMUs* VS *lower arms* sensors, to evaluate the effect on performance of using just a subset of all jacket's sensors. When the features are automatically extracted, a reduction in their number has a limited negative impact of F1-score (reduction $< 2\%$). When the features are manually extracted, a reduction in their number has a bigger impact on F1-score (reduction $\approx 4\%$).

Finally, the models' performances are evaluated excluding the most frequent class, which is *class 0*, that represents $\approx 80\%$ of observations. The *precision* is mildly affected by the exclusion of class 0, which means that the proportion of gestures correctly classified is pretty robust. In other words, when an observation is classified with a given label, $\approx 80\%$ of times is correct. The *recall* is heavily affected by the

	Without class 0		
	Prec.	**Rec.**	**F1-score**
all IMUs	0.860	0.533	0.644
lower arms	0.833	0.440	0.559
all IMUs manual	0.823	0.408	0.531
lower arms manual	0.750	0.281	0.396

TABLE 4: Precision, Recall and F1-score for each dataset excluding the most frequent class 0 (and without noise).

exclusion of class 0, which means that the proportion of all the observations belonging to a class that are correctly identified as belonging to that class (recalled) drops. This happens because all observations which do not belong to class 0 but for which the models predict class 0, contribute to decrease the recall of the remaining classes by the increment of the False Negatives (which are kept in the datasets).

VII. Concluding Remarks

In this project, we tried to classify various daily-life activities using data from IMUs sensors. First we explored data and analyzed the missing values (NA) pattern for all features, showing that the jacket sensors have NAs exclusively at the end of each run, because they are stopped in advance. Second, we used quaternions to remove the gravity from the *acceleration* and *gyroscope* features obtaining an invariant coordinate system. Third, we created 4 different datasets to investigate the impact on performances of different features spaces in terms of number of variables (all IMUs VS lower arms) and type of variables (provided vs hand-crafted) and also the robustness of each one to noise. Finally, we trained the most appropriate model for each dataset and showed that best performances are achieved using all IMUs and automatic feature extraction.

The most relevant result of our analysis is that the use of the *lower arms* sensors requires many less features (and so data) guaranteeing a limited negative impact on F1-score, which drops by less than 2%. In fact, note that *all IMUS* in the jacket are 5 (BACK, RUA, RLA, LUA, LLA) and for each of them we took 6 variables (accX, accY, accZ, gyroX, gyroY, gyroZ), for a total of $5 \times 6 = 30$ features. While the *lower arms* sensors are 2 (RLA and LLA) and again for each of them we took the same 6 variables, for a total of $2 \times 6 = 12$, which is the 60% less in terms of feature space. This could be useful for many IoT devices, whose utility is strictly connected to the correct classification of the activity the user is performing (i.e. fitness tracker), but, at the same time, they need to be light, with small batteries and low energy consumption, which implies the use of the smallest amount of data to receive and process.

Future researches could explore the use of alternative deep learning architectures and hand-crafted features in order to find even better results in terms of absolute performances and relative performances (trade-off between n° of features and F1-score).

In conclusion, we learned that the most tricky parts when working with real data are:

- understanding how the dataset is structured (i.e. a series of `.txt` file) and, accordingly, the best way to import and aggregate the data in order to start the analysis;
- exploring carefully the data, to spot NAs or any further issue;
- cleaning the data consequently to what discovered at previous point (i.e. NAs handling, gravity acceleration filtering);
- further reshaping data for the model (segmentation of data in time windows of appropriate size to feed the CNN).

In a word, the *pre-processing* has been the most important and time-consuming activity and so the one in which we faced more difficulties, and, for this reason, also the one from which we learned more.

References

[1] R. Chavarriaga, H. Sagha, A. Calatroni, J. d. R. M. Sundara Tejaswi Digumarti, Gerhard Tröster, and D. Roggen, "The opportunity challenge: A benchmark database for on-body sensor-based activity recognition," *Pattern Recognition Letters*, 2013.

[2] T. B. Lisa C. Günther, Susann Kärcher, "Activity recognition in manual manufacturing: Detecting screwing process from sensor data," *ScienceDirect*, 2019.

[3] J. Wachs, H. Stern, Y. Edan, M. Gillam, C. Feied, M. Smith, and J. Handler, "A Real-Time Hand Gesture Interface for Medical Visualization Applications," *Advances in Intelligent and Soft Computing*, 2010.

[4] H. Zhao, S. Wang, G. Zhou, and D. Zhang, "Gesture-Enabled Remote Control for Healthcare," *arXiv*, 2017.

[5] I. D. Roberto G. Valenti and J. Xiao, "Keeping a Good Attitude: A Quaternion-Based Orientation Filter for IMUs and MARGs," *Sensors*, 2015.

[6] N. Y. Hammerla, S. Halloran, and T. Plötz, "Deep, Convolutional, and Recurrent Models for Human Activity Recognition," *IEEE*, 2016.

[7] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "A Public Domain Dataset for Human Activity Recognition Using Smartphones," *European Symposium on Artificial Neural Networks*, 2013.

[8] M. Zeng, L. T. Nguyen, B. Yu, O. J. Mengshoel, and J. Zhu, "Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors," Nov. 2014.

[9] M. Gadaleta and M. Rossi, "IDNet: Smartphone-based gait recognition with convolutional neural networks," *Elsevier*, 2017.

[10] M. Ben-Ari, "A Tutorial on Euler Angles and Quaternion," 2017.

[11] T. Plötz, N. Y. Hammerl, and P. Olivier, "Feature Learning for Activity Recognition in Ubiquitous Computing," *Twenty-Second International Joint Conference on Artificial Intelligence, School of Computing Science Newcastle University, UK* , 2011.

[12] K. Frank, M. J. V. Nadales, P. Robertson, and M. Angermann, "Reliable Real-Time Recognition of Motion Related Human Activities Using MEMS Inertial Sensors," 2017.

[13] *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly, 2019.