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

*Human Data Analytics*'s Project n° 3


Candidates:

*Luca Dal Zotto - Giuliano Squarcina*

Department of Mathematics "Tullio Levi-Civita"
Master Degree in Data Science

22nd September 2020


UNIVERSITÀ DEGLI STUDI DI PADOVA

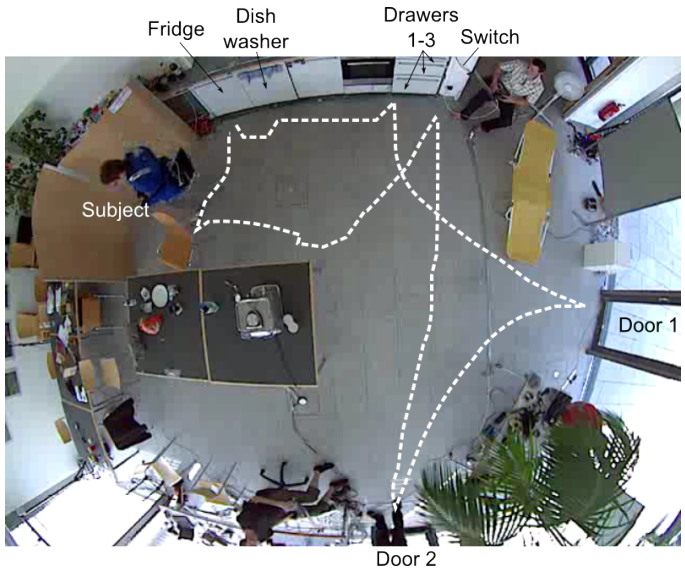
# Contents

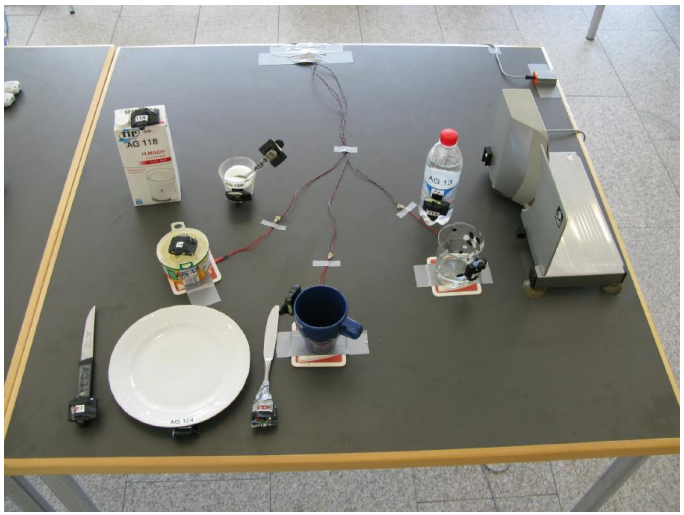
# Introduction to the task

## 'Opportunity' Dataset

Data collection:

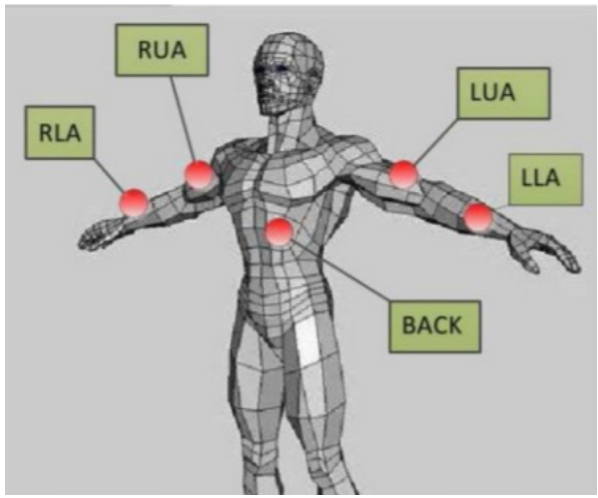
- four subjects performing activities of daily-life in a sensor-rich environment;
- six runs per subject: five ADL runs and one DRILL run;
- mid-level labels: open/close door (2), fridge, dishwasher, drawers (3), drink from cup, clean table and toggle switch.

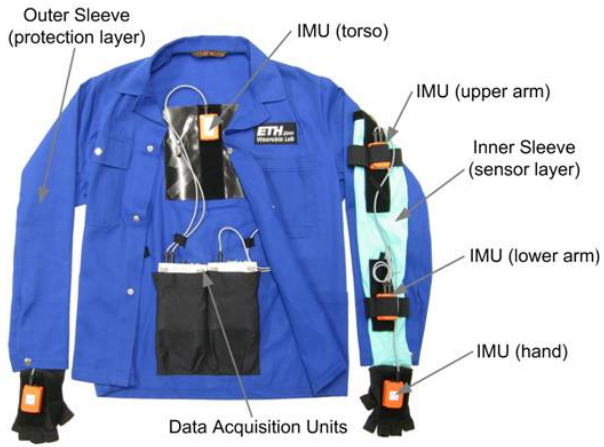


Data collection (cont.):

- 5 IMUs placed on a jacket, 2 on the shoes and other tri-axial accelerometers in different parts of the body;




- each IMU provides 3D acceleration, 3D rate of turn, 3D magnetic field, and orientation of the sensor with respect to a world coordinate system in quaternions.

First, we merged the runs consistently with the sets suggested in the reference paper. Then we followed this procedure:

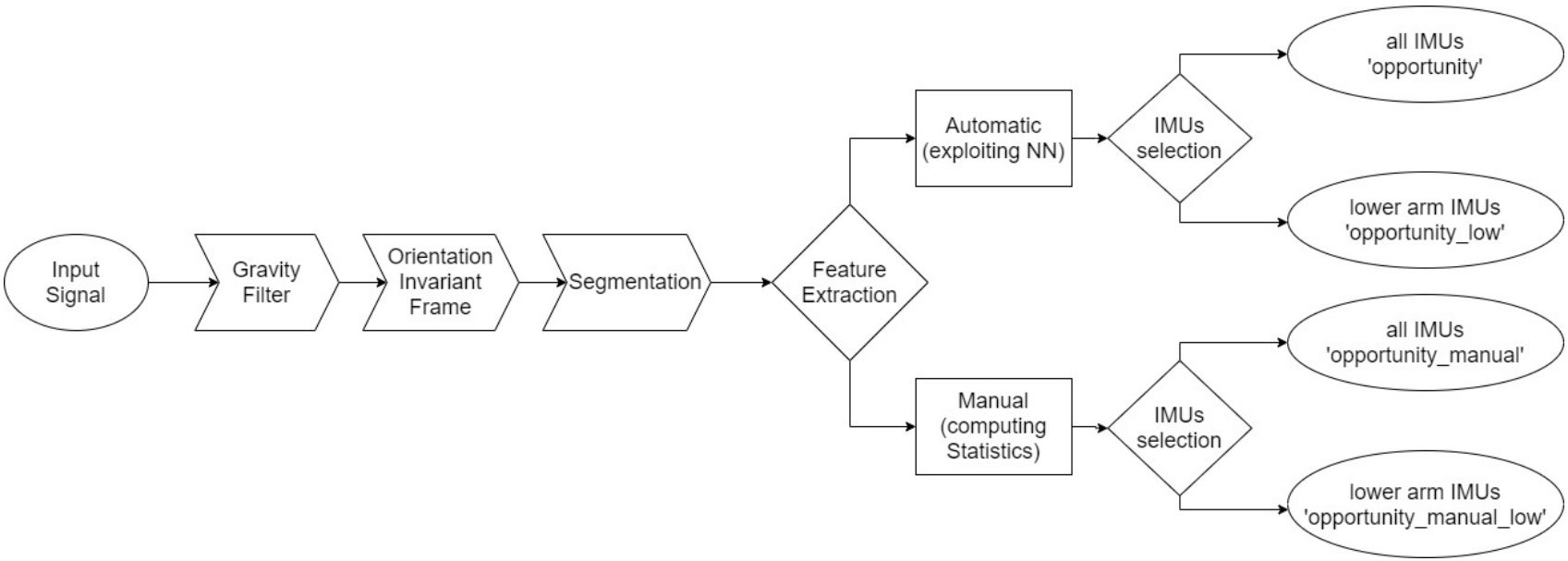



Figure: Project workflow.

Once having saved the final datasets, we trained the most appropriate model for each dataset and we compared their performances.

Missing values handling:

- specific pattern of missing values: they are present only at the beginning and end of the runs;
- recordings related to the shoes' IMUs start in advance and are stopped later with respect to the motion jacket activity;
- no missing values in the middle of a run, because a wired system is used (no interruptions due to connection failures);
- do not recover them through imputation;
- drop the recordings with missing values!

Use quaternions to reconstruct the rotation matrix

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} q_4^2 + q_1^2 - q_2^2 - q_3^2 & 2q_2q_1 - 2q_3q_4 & 2q_2q_4 + 2q_3q_1 \\ 2q_3q_4 + 2q_1q_2 & q_4^2 - q_1^2 + q_2^2 - q_3^2 & 2q_3q_2 - 2q_1q_4 \\ 2q_1q_3 - 2q_2q_4 & 2q_1q_4 + 2q_2q_3 & q_4^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}.$$

Let $\mathbf{z}$ be a vector in the fixed frame and $\mathbf{z}'$ the same vector but in the object frame of reference, then the following relations hold:

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

Idea: rotate $\mathbf{x} = (1, 0, 0)$, $\mathbf{y} = (0, 1, 0)$ and $\mathbf{z} = (0, 0, 1)$ and project the original signal along these rotated vectors.
Moreover, subtract the acceleration of gravity (the vector (0,0,1) in the world frame) to the accelerometer signal.

# Data Preprocessing

## Gravity Filtering and Orientation Independent Transformation

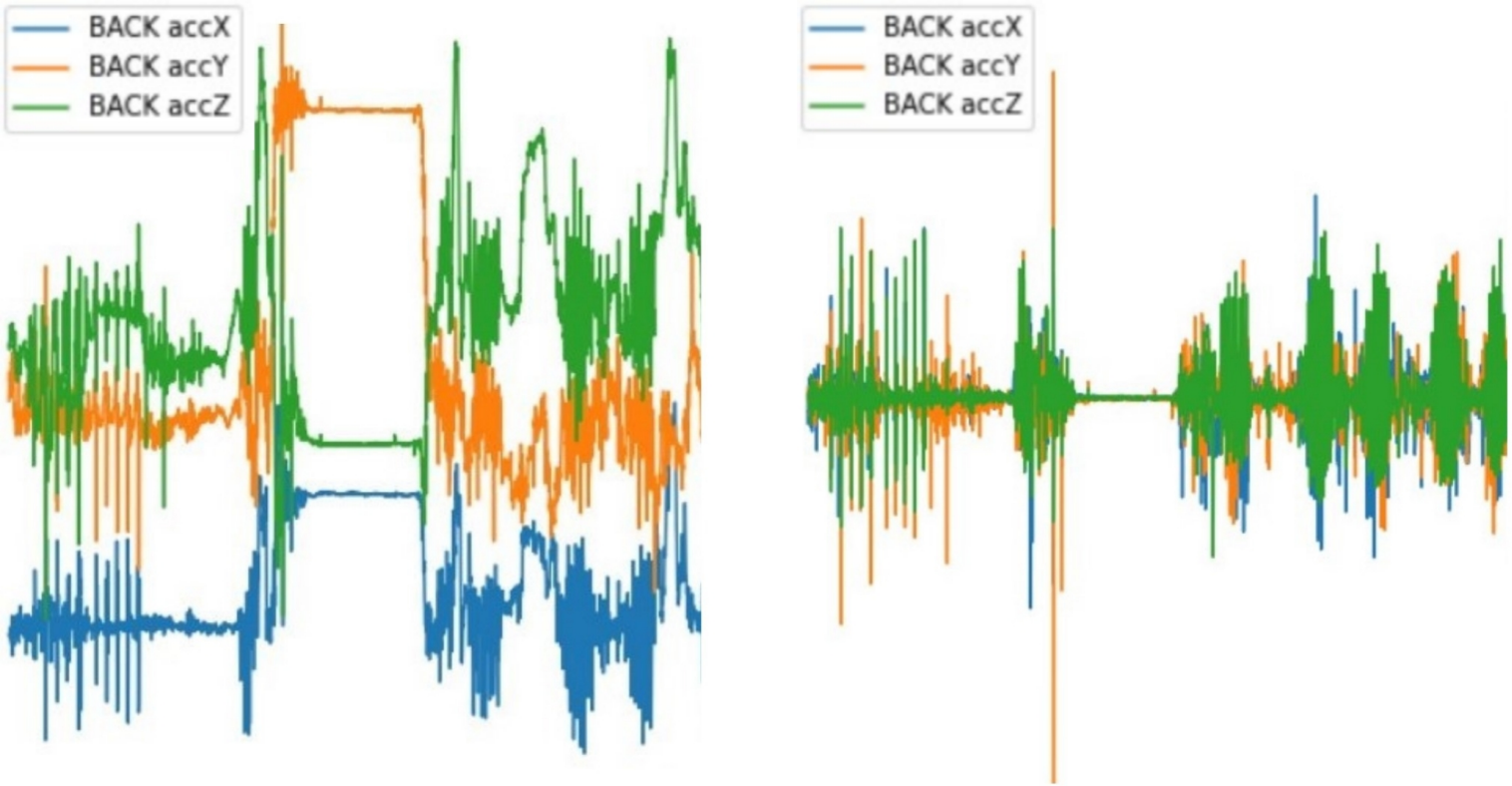



Figure: Accelerometer signal before and after preprocessing.

# Data Preprocessing

## Some statistics

Useful statistics:

- number of samples:
  - Training set: 622,511;
  - Validation set: 62,747;
  - Test set: 117,601;
  - Noisy Test set: 51,465;

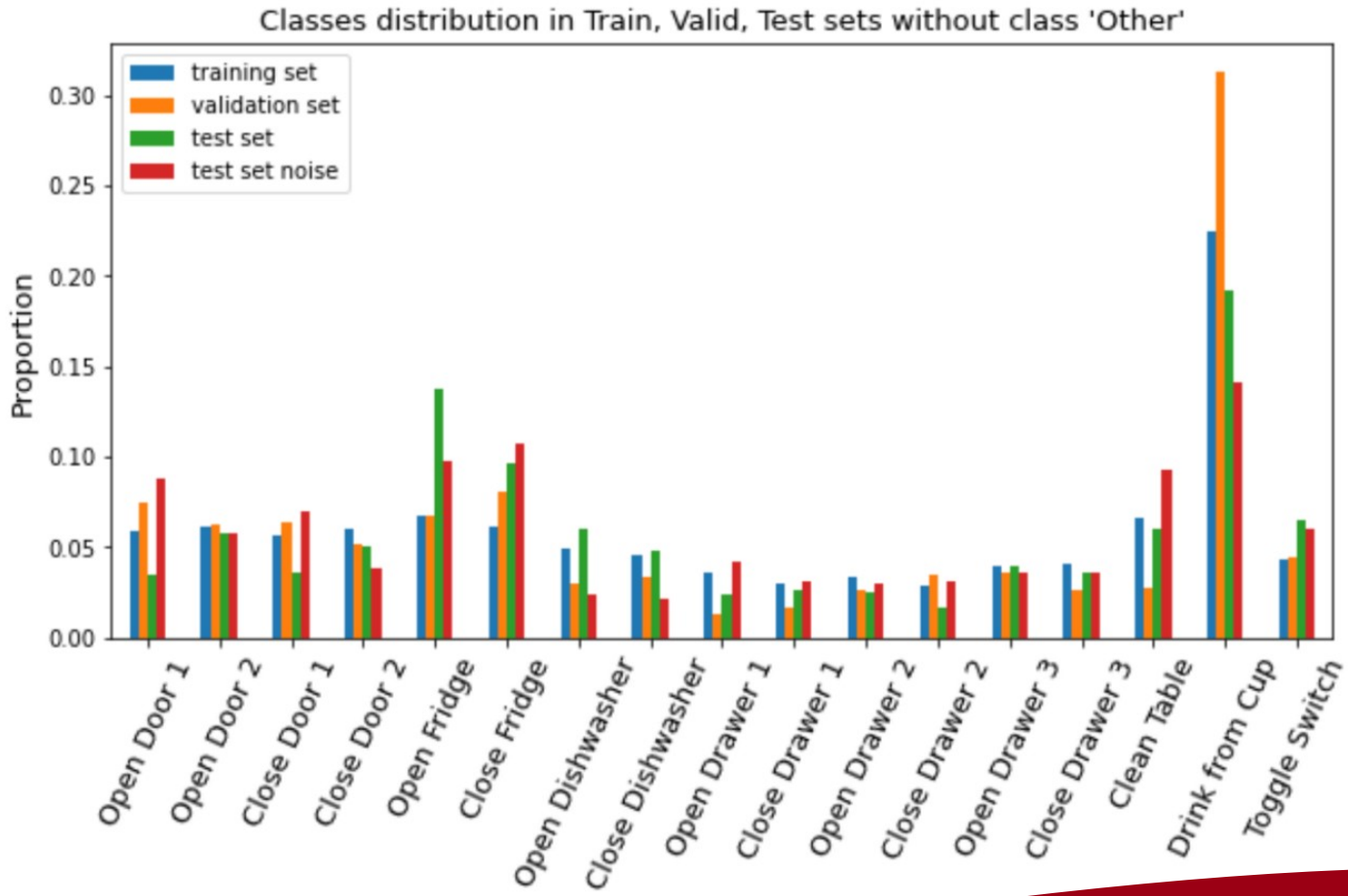
| | Subject 1 | Subject 2 | Subject 3 | Subject 4 |
|---|---|---|---|---|
| **ADL 1** | train | train | train | train |
| **ADL 2** | train | train | train | train |
| **ADL 3** | train | train | train | train |
| **ADL 4** | *valid* | *test* | *test* | *test_noise* |
| **ADL 5** | *valid* | *test* |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

- each IMU provides 13 variables (accelerometer (3), gyroscope (3), magnetometer (3) and quaternions (4)). Total = 65.
  From now on, we will consider only accelerometer and gyroscope, so 30 features;
- accelerometer and gyroscope signals have mean equal to zero and almost all the values are in the interval [-1,1].

# Data Preprocessing

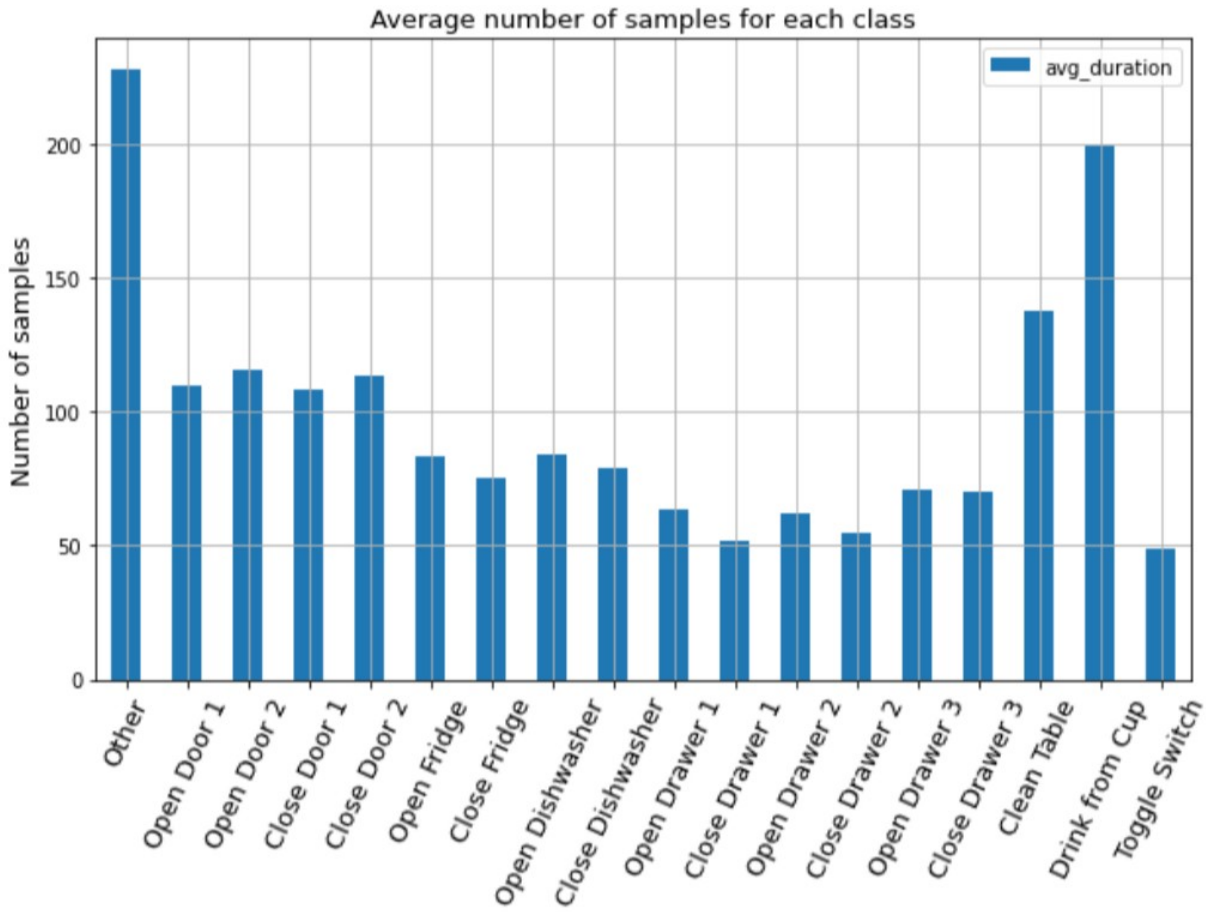
## Class frequency

Highly unbalanced dataset: 80% of recordings are labeled 'Other'.
Careful model evaluation using suitable metrics (weighted F1-score or weighted AUC), or ignoring the samples belonging to the majority class.



# Data Preprocessing

## Gesture average duration




Global average duration of a gesture = 97.54 samples.

Use half-overlapping windows of length 30 samples (1 second).

We have a dataset with 5 IMUs. Create some variants with:

- only the two IMUs located in the lower arms (RLA and LLA);
- some hand-crafted features.

| Feature | Description |
|---|---|
| *avg_z* | mean vertical component |
| *avg_x* | mean horizontal components (x and y are two fixed directions) |
| avg_y | |
| *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 |

Final datasets:

1. `dataset1`: acc and gyro of all IMUs;
2. `dataset2`: acc and gyro of RLA & LLA;
3. `dataset3`: handcrafted features from acc and gyro of all IMUs;
4. `dataset4`: handcrafted features from acc and gyro of RLA & LLA.

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).

The choice of the NN model is based on the following considerations:

- hyperparameters (n°layers, n°units) are choosen in excess, then *early stopping* and *regularization* prevent overfitting
- batch size is set to 64 considering the datasets' size
- default activation function is *Relu*, otherwise *Elu*. In the last layer *Softmax* is used.

3 types of NN architectures are used:

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

For dataset1 & dataset2 two competitive models are used:

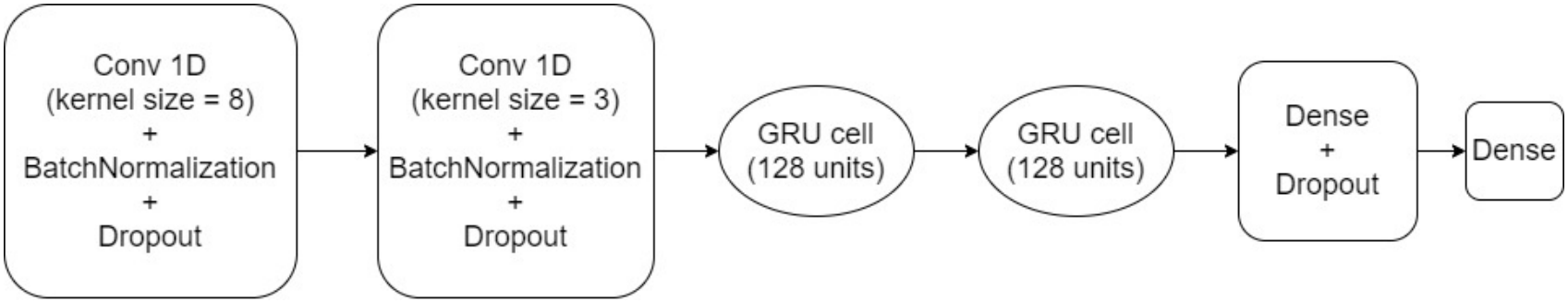



Figure: CNN+RNN

**Note:**

- Conv 1D captures correlation among features
- GRU captures correlation through time
- FFNN shrinks the n°units to 18 and Softmax converts each output in a probability

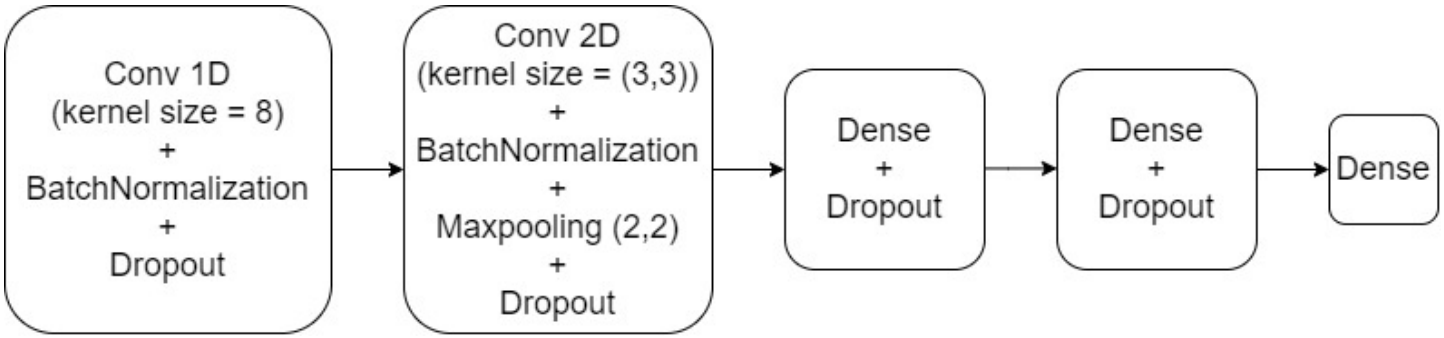



Figure: CNN

**Note:**

- `Conv 1D` captures correlation among features
- `Conv 2D` captures correlation through time (since `GRU` is not used)
- `FFNN` shrinks the n°units to 18 and `Softmax` converts each output in a probability

For dataset3 & dataset4 one model is used:

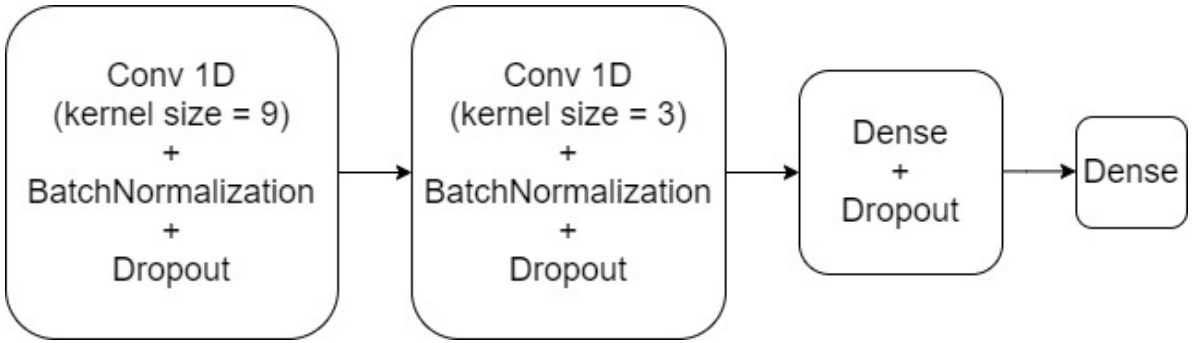



Figure: CNN for Manual Features

**Note:**

- *Manual 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
- FFNN shrinks the n°units to 18 and Softmax converts each output in a probability

The models' performances are computed using the following metrics:

$$\text{Precision} = \sum_i w_i \cdot \frac{TP_i}{TP_i + FP_i},$$

$$\text{Recall} = \sum_i w_i \cdot \frac{TP_i}{TP_i + FN_i},$$

$$\text{F1-score} = \sum_i w_i \cdot 2 \cdot \frac{\text{precision}_i \cdot \text{recall}_i}{\text{precision}_i + \text{recall}_i}.$$

# Results

## CNN+RNN vs CNN for dataset1 & dataset2

**Main idea**: evaluate the effect of GRU cells in final performance.

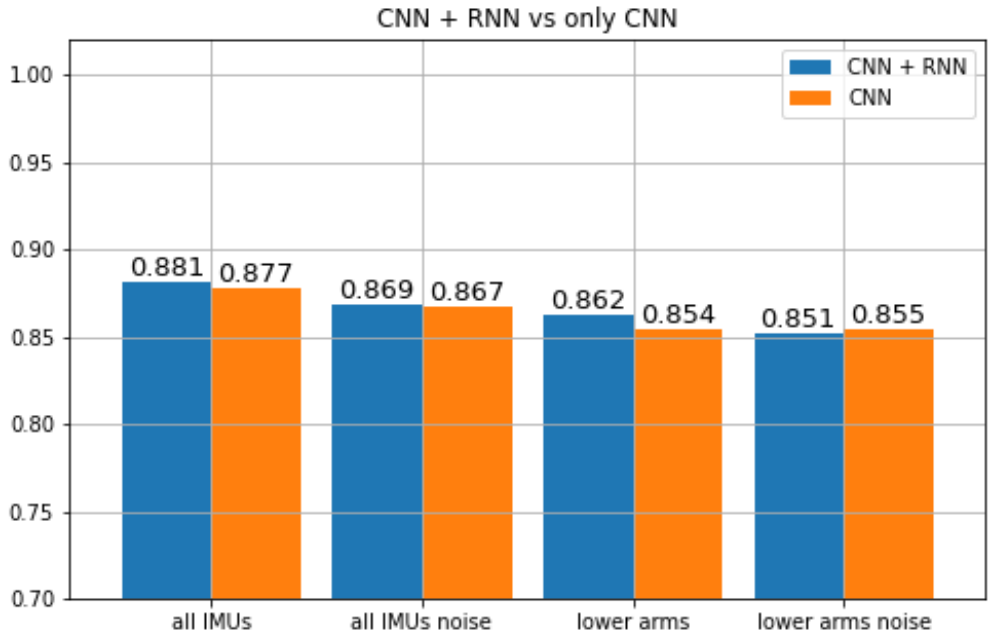



Figure: F1-score for CNN+RNN and CNN

**Results**: CNN+RNN model has slightly higher performances on both datasets, even in presence of noise (only exception is dataset2 with noisy test set). Our goal is to pick the best model, hence the CNN model is discarded.

# Results

## `F1-score` *with* and *without* noise

**Main idea**: compare the best model for each dataset in *both* the test-sets.

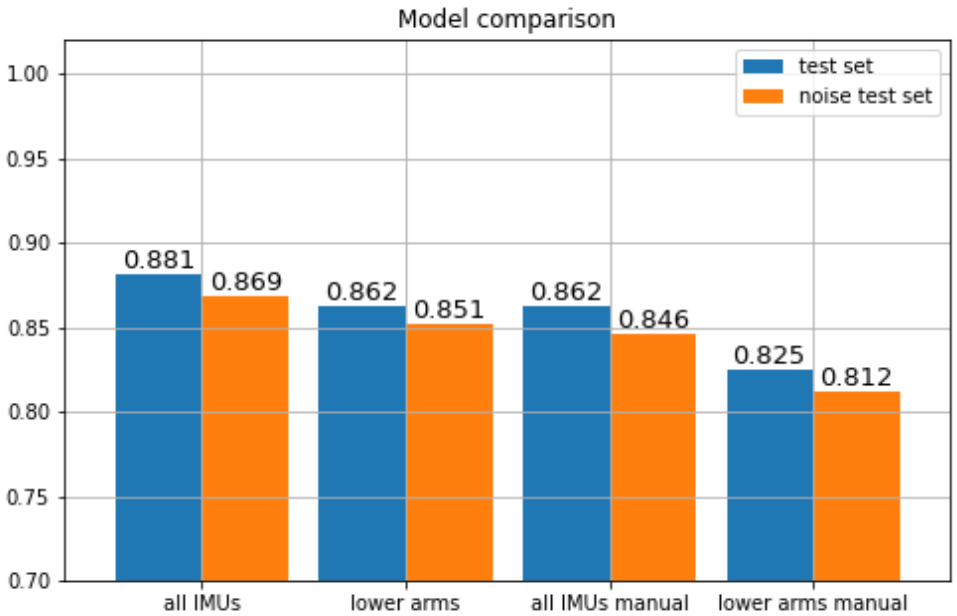



Figure: `F1-score` in test-set VS `F1-score` in *noisy* test-set

**Results**:

- all models are robust to noise
- gap between *Automatic* and *Manual* features increases as the n° features decreases
- reduction of n°features decreases `F1-score`:
  - $< 2\%$ for *Automatic* features
  - $\approx 4\%$ for *Manual* features

# Results

## F1-score, Precision, Recall *without* class 0

**Main idea**: evaluate the effect of removing class 0 in final performance.

| | With class 0 | | |
|---|---|---|---|
| | Prec. | Rec. | F1-score |
| all IMUs | 0.882 | 0.887 | 0.881 |
| lower arms | 0.861 | 0.874 | 0.862 |
| all IMUs manual | 0.862 | 0.879 | 0.862 |
| lower arms manual | 0.832 | 0.855 | 0.825 |

| | 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 |

**Results**:

- *Precision* slightly decreases: % records correctly classified is robust
- *Recall* strongly decreases: % records $\in$ $class_i$ $(C_i)$ correctly identified as $\in$ $C_i$ drops
  - Recall:= $\frac{TP}{TP+FN}$

# Conclusion

## Final remarks



The more relevant results of the analysis can be summarized in the following points:

- automatic feature extraction guarantees better performances than the handcrafted approach;
- variables reduction has a limited impact on F1-score: *lower arms* sensors requires 60% less variables than *all IMUS*, but when features are learned by the network, the reduction of the F1-score is $< 2\%$;
- all models show to be robust to noise artifacts.

### Comment

The last two points are of crucial importance for **real world applications**, which require good performances in a noisy environment with low energy consumption.



When the computational resources are limited, manual feature engineering is usually a better approach, since, in general, it is more memory efficient. However, a deep knowledge about the problem at handle is required: we need to know which information should be embedded into the features. For this reason, a potential development of this project consists in exploring further this topic.