UNIVERSITÉ LIBRE DE BRUXELLES ULB

FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

# INFO-F-422 - Statistical foundations of machine learning Project 2024-2025

Gianluca Bontempi, Cédric Simar, Pascal Tribel

April 10, 2025

The project counts for 50% of your final grade (i.e. 10/20). This project has to be developed by a team of 3 students registered for the class. Projects submitted by teams with a different number of students will not be considered unless prior authorization is explicitly granted. The project shall be completed independently and it shall represent the sole efforts of the team submitting the assignment. The result of another team's efforts, or the copy of another team's efforts (current, or past, semester(s)), is considered academic dishonesty and will be punished accordingly.

## 1 Background

Myoelectric prostheses have become increasingly important for restoring upper limb functionality, relying on *surface electromyography* (sEMG) to measure the electrical potentials generated by muscle contractions. Historically, many control strategies have classified discrete gestures (e.g., "open hand," "pinch," "fist") from the EMG signal to actuate prostheses. While effective for simple commands, such approaches often fail to capture the complexity of natural hand movements in daily life.

In contrast, regression-based methods aim to predict continuous joint angles from EMG signals, enabling fine-grained control across multiple degrees of freedom (DOFs) in real time. Recent advances in machine learning have made these continuous decoding methods increasingly viable. Ultimately, researchers hope to develop prostheses with more natural and intuitive control.

## 2 Dataset

### 2.1 Participants and data collection protocol

Fourteen healthy volunteers participated in a protocol designed by the *Machine Learning Group*. **However, this project only uses data from a single participant**, consisting of raw sEMG signals and corresponding hand poses for this individual.

Each participant performed two types of exercises:

**Guided gestures:** They repeated five predefined hand postures: an "open hand" resting pose and four single-finger extension gestures. A VR interface presented each target gesture and provided feedback when it was recognized. Each session included multiple repetitions of the five poses.

**Free gestures:** Participants interacted naturally with various virtual 3D objects (e.g., hammers, keyboards, cups, rings) in six distinct VR scenes—one empty scene plus five interactive scenes—eliciting a broad range of realistic hand and finger movements.

In total, participants completed 5 guided and 6 free-gesture sessions, each lasting about 5 minutes.

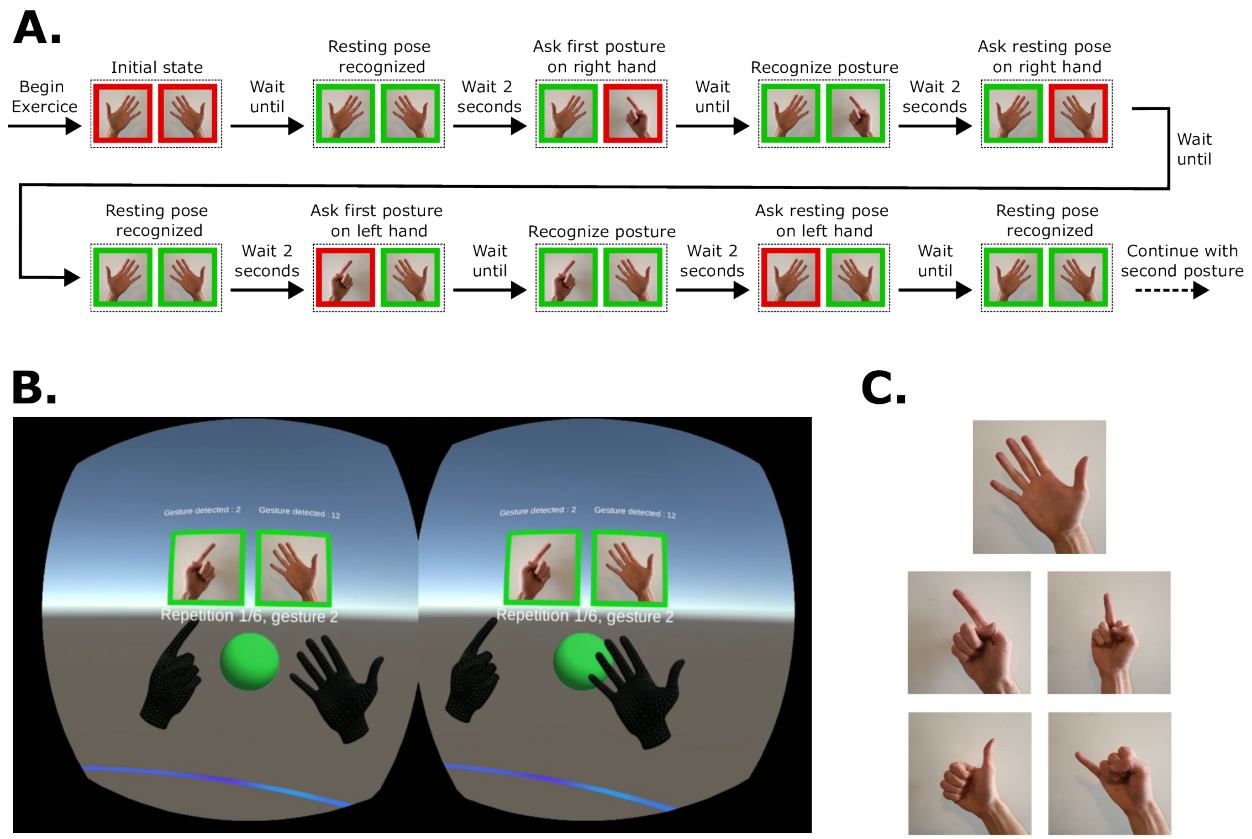



FIGURE 1 – **A.** Experimental protocol for predefined gestures **B.** VR interface from the Oculus Quest **C.** The 5 guided gestures, including the open hand resting pose.

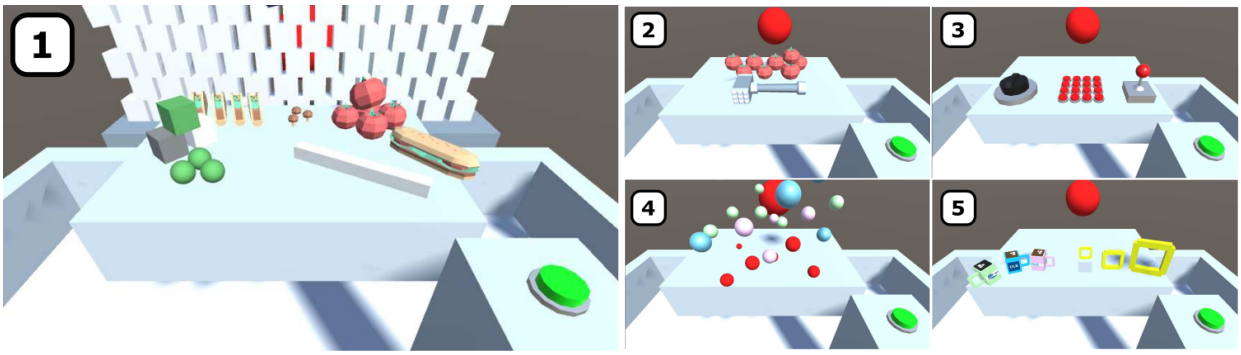



FIGURE 2 – Visual representation of the 5 interactive scenes in VR

## 2.2 EMG acquisition

Eight wireless EMG sensors were placed on the participant's forearm (as illustrated in Figure 3), capturing sEMG at 1024 Hz (so 1024 samples correspond to 1 second of data). The high sampling rate preserves crucial temporal information for decoding continuous hand movements.

## 2.3 Motion capture

An Oculus Quest VR headset recorded hand and finger movements at approximately 50 Hz. Each motion capture sample consists of **51 continuous joint-angle values** (3 rotation angles for each of 17 joints). The motion-capture data was then resampled to 1024 Hz to match the EMG sampling frequency. Synchronizing the two datasets through timestamps resulted in high-frequency EMG signals paired with continuous joint-angle measurements, as illustrated in Figure 4.

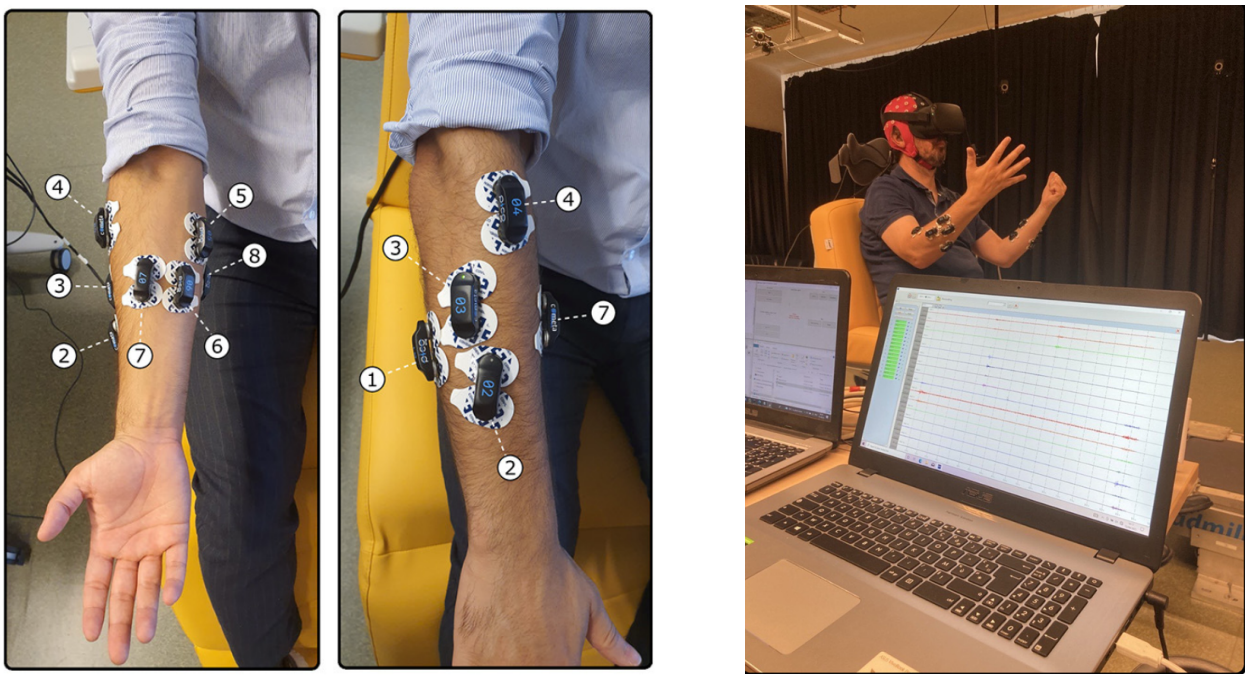


FIGURE 3 – Electrode placement on the forearm (left) and a participant during recording (right)

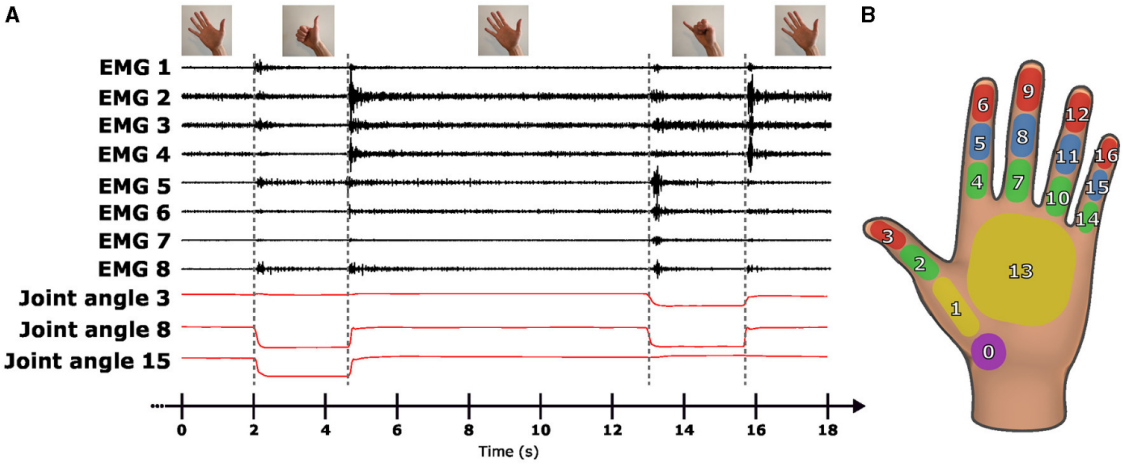



FIGURE 4 – (A) Example of raw EMG (black) and three joint angles (red) during a guided exercise. (B) Mechanical hand model from the Oculus Quest.

## 2.4 Data files

The data files for the project can be downloaded using the following link. In practice, the teams receive two datasets. Each dataset includes a training/validation set with the corresponding hand pose estimations and a test set for which teams must predict hand pose estimations as part of the competition:

1. The *guided gestures dataset* contains continuous sEMG signals (8 electrodes) and the corresponding hand pose estimation (51 joint-angles) of 5 sessions during which the participant performed repetitions of five predefined hand postures. Considering the well-defined and repetitive nature of the gestures in this dataset, you should obtain good pose estimation results. This will be your baseline before venturing into the second dataset. Concretely the *guided gestures* dataset is composed of the following files:

    - `guided_dataset_X.npy`: contains the training/validation data in the form of a Numpy array of shape `(5, 8, 230000)` for `(session, electrode, time)`
    - `guided_dataset_y.npy`: contains the corresponding training/validation hand pose estimation (the target to predict) in the form of a Numpy array of shape `(5, 51, 230000)`
    - `guided_testset_X.npy`: contains the test data in the form of a Numpy array of shape `(5, 332, 8, 500)` `(session, window, electrode, time)`. For the competition, you have to predict the corresponding hand pose estimation (51 values) for each window (so $5 * 332$ predictions of 51 values in total).

2. The *free gestures dataset* contains continuous sEMG signals (8 electrodes) and the corresponding hand pose estimation (51 joint-angles) of 5 sessions during which the participant freely performed a broad range of realistic hand and finger motions. As you will see, this task is much more complex than the previous one but we trust you will be able to achieve great results! Concretely the *free gestures* dataset is composed of the following files:

    - `freemoves_dataset_X.npy`: contains the training/validation data in the form of a Numpy array of shape `(5, 8, 270000)` for `(session, electrode, time)`
    - `freemoves_dataset_y.npy`: contains the corresponding training/validation hand pose estimation (the target to predict) in the form of a Numpy array of shape `(5, 51, 270000)`
    - `freemoves_testset_X.npy`: contains the test data in the form of a Numpy array of shape `(5, 308, 8, 500)` `(session, window, electrode, time)`. For the competition, you have to predict the corresponding hand pose estimation (51 values) for each window (so $5 * 308$ predictions of 51 values in total).

# 3 sEMG features extraction and regression

This section introduces three widely used approaches for extracting meaningful features from sEMG signals, which will then serve as input for your machine learning models to predict the joint-angle values.

## 3.1 Common features in the time domain

Extracting features from EMG signals in the time domain is a widely used and straightforward approach for characterizing muscle activity. Time-domain features are popular due to their computational simplicity, interpretability, and effectiveness in capturing the underlying physiological properties of muscular signals. Here is a table of six commonly used time-domain features used for EMG-based regression models. Students may implement more features to further improve the performance of your model (although it goes beyond the scope of this project). With $K \in \mathbb{N}$ the size of the signal window, $x \in \mathbb{R}^K$ the signal window of size K from one electrode, and $\bar{x}$ the average of $x$:

| Feature | Description | Formula |
|---|---|---|
| **Mean Absolute Value (MAV)** | Average of the absolute value of the signal. | $\text{MAV} = \frac{1}{K}\sum_{i=1}^{K}\lvert x_i\rvert$ |
| **Root Mean Square (RMS)** | Square root of the average of squared samples, indicating signal amplitude. | $\text{RMS} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}x_i^2}$ |
| **Variance** | Measures the spread of the data points around their mean. | $\text{Var} = \frac{1}{K-1}\sum_{i=1}^{K}\left(x_i-\bar{x}\right)^2$ |
| **Standard Deviation (STD)** | Square root of the variance, capturing signal dispersion. | $\text{STD} = \sqrt{\frac{1}{K-1}\sum_{i=1}^{K}\left(x_i-\bar{x}\right)^2}$ |
| **Zero Crossing (ZC)** | Counts how many times the signal changes sign. | $\text{ZC} = \sum_{i=1}^{K-1}\mathbf{1}\left(x_i\,x_{i+1}<0\right)$ |
| **Myopulse Percentage Rate (MPR)** | Counts how often $\lvert x_i\rvert$ exceeds $\sigma$. | $\text{MPR} = \frac{1}{K}\sum_{i=1}^{K}\mathbf{1}\left(\lvert x_i\rvert>\sigma\right)$ |

Table 1 – Common features extracted from EMG signals.

The *indicator function* of a set A is defined as $\mathbf{1}_A(x) = 1$ if $x \in A$, and $\mathbf{1}_A(x) = 0$ otherwise. For example, in the case of the MPR feature, if we consider the set $A = \{x \mid |x| > \sigma\}$ for some threshold $\sigma \in \mathbb{R}$, we have:

$$\mathbf{1}_{\{|\cdot|>\sigma\}}(x) = \begin{cases} 1, & \text{if } |x| > \sigma, \\ 0, & \text{otherwise.} \end{cases}$$

## 3.2 Covariance matrices

Another robust approach involves extracting covariance matrices, representing the spatial correlations across EMG channels. These covariance matrices are symmetric positive-definite (SPD) and thus lie on a Riemannian manifold, necessitating specialized analytical methods. `PyRiemann`, a Python library, provides tools tailored for processing SPD matrices through Riemannian geometry. The source code, library installation procedures, and practical examples can be found here.

The typical workflow consists of the following steps:

- **Covariance estimation**: covariance matrices can be robustly estimated from multichannel EMG signals using `PyRiemann`'s `Covariances` class with the Oracle Approximating Shrinkage (OAS) method.
- **Tangent Space Projection**: maps SPD matrices onto their tangent space, transforming them into Euclidean vectors of dimension $m(m+1)/2$ (where $m$ is the matrix dimension). This operation locally approximates the SPD manifold, preserving its inner geometry, while enabling standard vector-based machine learning methods. The projection requires a reference matrix, typically the geometric (Riemannian) mean of the covariance matrices from the training set. More detailed information can be found here
- **Regression Algorithm**: After tangent space projection, traditional regression algorithms (e.g., Ridge regression, Support Vector Regression) can be used as the final step of the machine learning pipelines.

# 4 Performance metrics

We are predicting the continuous hand pose estimation composed of 51 joint angles. In this project, each $y_i$ or $\hat{y}_i$ is a 51-dimensional vector of joint angles. Thus, we evaluate the quality of the predictions using the following adapted metrics:



1. The *root mean squared error* (RMSE):

$$\text{RMSE} = \sqrt{\frac{1}{\text{N}_{ts} \cdot \text{D}} \sum_{i=1}^{\text{N}_{ts}} \sum_{d=1}^{\text{D}} (y_{i,d} - \hat{y}_{i,d})^2}$$

2. The *normalized mean squared error* (NMSE):

$$\text{NMSE} = \frac{\sum_{i=1}^{\text{N}_{ts}} \sum_{d=1}^{\text{D}} (y_{i,d} - \hat{y}_{i,d})^2}{\sum_{i=1}^{\text{N}_{ts}} \sum_{d=1}^{\text{D}} (y_{i,d} - \bar{y}_d)^2}$$

where:

- $\text{D} = 51$ is the dimensionality (number of joint angles in each vector),
- $y_i = (y_{i,1}, \ldots, y_{i,\text{D}})$ is the measured 51-dimensional joint-angle vector of the $i$-th test observation,
- $\hat{y}_i = (\hat{y}_{i,1}, \ldots, \hat{y}_{i,\text{D}})$ is the predicted 51-dimensional joint-angle vector of the $i$-th test observation,
- $\text{N}_{ts}$ is the number of test observations,
- $\bar{y}_d = \frac{1}{\text{N}_{ts}} \sum_{i=1}^{\text{N}_{ts}} y_{i,d}$ is the mean of the $d$-th joint angle across all test observations

# 5 Objectives

Given the described dataset, the main overall objective of the project is to find the regression model that best predicts continuous hand articulation angles at time $t$ using sEMG signals collected over a window $[t-k, t]$. Specifically, the project involves the following tasks:

1. **(Optional) Signal filtering**: if you plan to filter your sEMG signals, it is recommended to perform this preprocessing step directly on the continuous raw data prior to window extraction or feature computation. Note that this step is completely optional but may improve your results.
2. **(0.5 point) Dataset preparation and augmentation through overlapping windows**: you should first segment your sEMG signals into smaller **windows of fixed size** $k = 500$. These windows should be created with a chosen degree of overlap, which you can adjust based on the computational and memory resources available to you. Keep in mind that a larger overlap results in a greater number of samples and thus a larger dataset to train your models but to the cost of increasing computational demands.

   *Illustrative example of overlapping windows*: consider a sEMG recording with 2,000 samples and windows of length $k = 500$ samples. With an overlap of 50%, each new window starts 250 samples after the previous one, resulting in:

   - **Window 1**: samples 1–500
   - **Window 2**: samples 251–750
   - **Window 3**: samples 501–1000
   - and so forth...

   If you increase the overlap to 75%, each new window starts 125 samples after the previous one, thus increasing the dataset size:

   - **Window 1**: samples 1–500
   - **Window 2**: samples 126–625
   - **Window 3**: samples 251–750
   - and so forth...

3. **(1 point) Cross-validation strategy:** Determine and implement an adequate cross-validation strategy to validate your regression models, specifying how you organized your data partitions for training and validation. Provide a detailed justification showing that your validation sets remain completely independent from the training set. Include reasoning or evidence demonstrating explicitly that your chosen partitioning strategy prevents data leakage or bias, ensuring the reliability and generalizability of your model performance estimates.

4. **(3 points) Baseline approach:** Create a custom class inheriting from scikit-learn's `BaseEstimator` and `TransformerMixin` that implements the extraction of common time-domain features described in section 3.1. Note that the features described in Section 3.1 represent the minimal required set. We encourage you to include additional features or preprocessing steps if you would like to further improve your model performances. Select at least two different regression models, compare their cross-validated performance, and evaluate their feature importances. For both models, perform feature selection to determine the optimal subset of features minimizing the Root Mean Squared Error (RMSE). Clearly document this process in your notebook, discussing the outcomes in detail. Finally, create a scikit-learn `Pipeline` that integrates your custom feature extraction class, the optimal feature selection step, and the best-performing regression model identified from your cross-validation results. Using visualizations and tables to illustrate your findings, and employing formulas or pseudo-code to explain the feature selection procedure, is strongly encouraged. Note that one-third of the score will depend on the quality and clarity of your documentation.

5. **(2 points) More sophisticated approach:** Implement a regression model based on either the covariance matrices (3.2) pipeline described above or a neural network approach using the PyTorch library. **Optionally**, you can implement both approaches. Assess its cross-validated performance with respect to the baseline approach and discuss the results. The use of figures, formulas, tables and pseudo-code to describe this approach is again strongly encouraged. Note that one-third of the score will depend on the quality and clarity of your documentation.

6. **(3 points)** Design and implement two ensembling strategies that combine the predictions of all individual regression models you implemented, each independently trained on distinct feature representations of the EMG signals. The two required ensembling approaches are the following:

   - Compute the average of the predicted values, such that the final prediction is given by

     $$\hat{y}_{\text{ensemble}} = \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \hat{y}_m,$$

     where $\hat{y}_m$ denotes the prediction of the $m$-th model out of M total models.
   - Use a meta-learner strategy (often referred to as stacking), in which a separate model—trained on the outputs of the base learners—generates the ensemble prediction. For example, you might concatenate predictions $\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\mathrm{M}}$ as input features to a secondary regressor $f_{\text{meta}}$, resulting in

     $$\hat{y}_{\text{ensemble}} = f_{\text{meta}}(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{\mathrm{M}}).$$

   For each ensemble strategy, **compare and document** the ensemble's regression performance against each of its constituent base models using consistent regression metrics. Additionally, for the meta-learner strategy, **determine and discuss** the relative contribution of each base model to the final ensemble prediction. **Discuss** how the bias-variance tradeoff relates to the observed (or expected) evolution of performance.

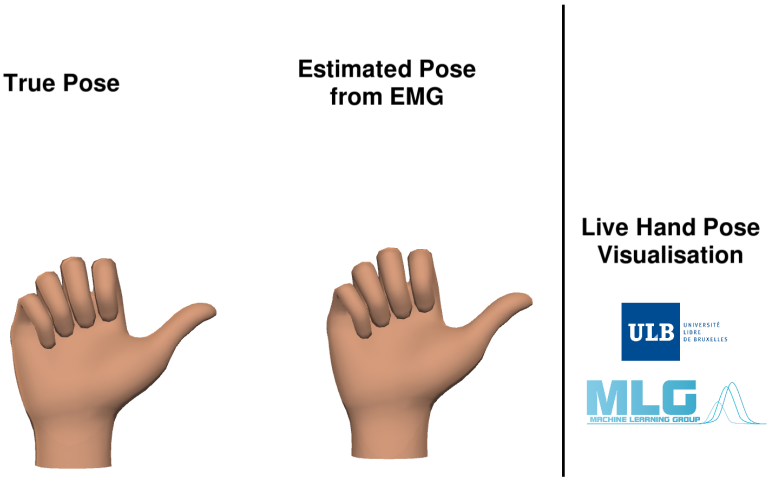
7. **(0.5 point)** For each protocol (**guided gestures** and **free gestures**), **select** the regression model or ensemble you believe will generalize best to the test set. You may choose the same or a different model for each protocol. **Clearly explain** why you selected this particular model(s) or ensemble(s) over the alternatives, providing specific reasons to support your choice.

   Using your selected model or ensemble, predict the vector of hand pose estimations (51 values per window) for each window in the file `guided_testset_X.npy`, starting from the first window of session 1 and ending with the last window of session 5. This will produce a prediction array of shape (1660, 51). Repeat the procedure for the file `freemoves_testset_X.npy`, resulting in a second prediction array of shape (1540, 51).



Finally, concatenate the two arrays in the following order: guided predictions first, then freemoves predictions (for example using `np.vstack((guided_predictions, freemoves_predictions))`). This yields a final array of shape (3200, 51). Save this array to a `.csv` file without a header (for example by converting your array into a pandas dataframe `df` and calling `df.to_csv('team_submission.csv', index=False)`. Finally, **submit** the resulting `.csv` file through the competition website. The leaderboard and submission portal can be found here.

8. **(bonus 0.5 point)** Predictions visualization: sometimes, a high performance metric can be misleading if your model is overfitting common poses (e.g., an open, stationary hand) rather than accurately capturing movements. However, what we truly want is a model that excels at predicting hand movements. To better visualize your model's predictions and assess its actual performance, we developed an application that displays two hands (representing the the ground-truth pose and the estimated pose) in real time, allowing you to stream your model's predictions directly into the visualization. For more information, and working examples, visit the dedicated *Live Hand Pose Visualisation* GitHub repository.




# 6 The team

A project team has to be composed of exactly three students registered for the class. Team composition must be submitted **by one of the group members** through a dedicated Google Form, available **here**. The team composition has to be finalized no later than **11PM of the 17th of April 2025**. (Hard deadline!)

# 7 Competition

We have created a dedicated competition website where each team can submit predictions from their best models and compete with other teams on the leaderboard. At the end of the competition, the top three teams on the leaderboard will receive special recognition.

To submit your best model's predictions, first prepare your submission as described in Objective 7. Then, visit the competition website, click on the `Submit` button, authenticate using the team's credentials (which you will have received beforehand), and upload your `.csv` file.

# 8 Github

The submission of the deliverables for the project is made on a group repository on the GitHub Classroom platform. **The access link required to setup your repository on the platform will be provided once all the teams are formed.** Once you click the link, you shall either create and name your group (i.e. "Group [group number]") or join your existing group if it has already been created by your group partner.

The version control system offered by Git allows you to easily keep track of the different versions of your code and go back to a previous version of your code in case you have any problems. In order to do so, you will need to regularly instruct the system to keep track of the different versions of your code (i.e. `commit` in the Git terminology). However, a `commit` only keeps track of a version of your code locally (on your PC). In order to transfer a copy of your code to a remote server (like Github) you will need to perform a `push` operation. We encourage you to regularly perform `commit` and `push` operations in order to have up-to-date versions of your code both locally and remotely. More information concerning the Git workflow and how to manage your repositories can be found at `https://docs.github.com/en/repositories`. A short kick-start demonstration of Git and Github Classroom will also take place during the practical session introducing the project.

For each group of students, the final submission of your code that will be evaluated is the last uploaded version of the deliverables on your assigned Github repository before the deadline time and date. If you wish to submit your final deliverables between the 11PM the 15th of May 2025 deadline and the 11PM the 16th of May 2025 (penalized by one point), you need to contact your teaching assistants to make sure we evaluate the correct deliverables.

# 9 Students participation

The Git version control system will also be used to assess students' participation in the group effort. Students with little to no visible participation in the project will be penalized accordingly. Thus, we encourage every member of each group to regularly perform `commit` and `push` operations so that we can reliably appreciate the input of each member with respect to the final deliverables.

# 10 Plagiarism and code generation

Considering that new code generation tools were recently made publicly available, we would like to especially draw your attention to the fact that this project **shall be completed as a team and it shall represent your sole efforts**. Each project will be scanned for text and code plagiarism or automatic generation. The result or the copy of another team's efforts (current or past semester(s)) of a publicly available notebook, either in full or in part, or the use of an automatic code/text generation tool, is considered academic dishonesty. Plagiarism, in the sense of copy-pasting from existing or generated reports or code, is a serious issue and will lead to disciplinary measures being applied to the whole group.

# 11 Deliverables

The student team will deliver its assigned Github repository:

1. the documented implementation (in a Python Jupyter notebook format) of all the objectives described in this document, with a clear justification of each choice.

2. the predictions submitted to the competition portal (in .csv format).



## Rules for project submission

*To be read carefully!*

1. The assignment should be made by teams of **exactly** three students. The team composition has to be finalized no later than **11PM the 13th of April 2025**.
2. The assignment will be graded on the notebook containing your documented implementation
3. The code should be **commented** and each choice of implementation should be justified.
4. The assignment will be submitted through the Github Classroom platform and the repository should include all deliverables listed in the deliverables section.
5. The deadline for the submission of your project is: **11PM the 15th of May 2025**.
6. All the projects submitted after the deadline will be:
   - Penalized by one point if submitted **before 11PM the 16th of May 2025**.
   - Not considered **after 11PM the 16th of May 2025**
7. Sharing of code is not allowed (you may, however, verbally discuss ideas on how to tackle the project).
8. This project counts for 50% of your grade (10 points). This project **shall be completed as a team and it shall represent your sole efforts**. The result of the copy of another team's efforts (current or past semester(s)) or automatic code generation is considered academic dishonesty. Plagiarism, in the sense of copy-pasting from existing reports or code, is a serious issue.
9. Each project producing any error during its execution will receive a grade of 0/10.