ERA: Exoskeleton Robotic Arm
An exoskeleton robot arm for rehabilitation and injure recovery

Josep Fortuny Casablancas
Ignasi Escuder Olóndriz

July 9, 2019

Contents

1 Introduction

One of the biggest problems that Spain and many other countries are facing now or in a short future is the lack of doctors due to the growth of the old population and the cuts in health that the government is making to the public sanity [1].

In order to prevent this issue, in the recent years researchers have been studying and implementing different technologies that can simplify or complement the doctors work improving the patients quality of service and giving them more time to invest in attending efficiently more patients .

There are a large amount of topics in health on which we could have focused, but we decided to aim all of our effort in the rehabilitation topic due to the time that it requires from the doctor and the harm that can be done to the person its having the treatment if the exercises are made wrong.

In our case we decided to help in the rehabilitation process from Neuromuscular Disorders [2], which consists of a group of pathology's that usually affect the nerves that control the muscles the communication between them and the muscles itself. There can be different causes for these diseases. Many of them are genetic and some of them are autoimmune. Many of these diseases have no cure but treatments may improve symptoms, increase mobility, and lengthen life.

More specifically we decided to focused on the pathology's of the neuromuscular union that affects directly in the arm muscle. Our idea is to use a Exoskeletion robotic Arm and implement a program utilizing Matlab that can enable the doctor visualize the users input data from the arm muscles and create training exercises that can help the patience improve and see if they are performing the exercise correctly. This will allow the doctor to create exercises that can be practiced first in therapy but then in home, enabling the user get the feedback created in the exercise. Giving more time to the doctor to attend more patience's and still being necessary to make control therapy sessions to create more exercises depending on the users needs or the improvements made.

2 Goals

The main goal of this project is to create a tool that can monitor the patients that suffers from Neuromuscular Disorders in the upper extremity's actions and giving feedback to the doctor. To archive this goal we set certain objectives that we had to archive:

1. Implement a machine learning code that permits a user to create and implement automatically a classification of the movement without having any technological knowledge .
2. Create an easy utilization and understanding interface that the doctor or the patient can use with the following options:
    - Visualize the physical responses that the arm of the user gives so that the doctor can see the muscles answers from the users biceps, triceps, position and strength.
    - Enable the doctor create a personalized training method that applies automatically the machine learning code implemented. This option has to include the possibility to take samples of the users response and add or erase them in case of need. This will enable the doctor adapt the rehabilitation exercises and therapy for each patient and get the response to each user action.
    - Create predefined trained sets of algorithm in order to distinguish actions that can be helpful in the training process or use the just created program from the doctor in the therapy session or in the users home.
    - Create a generic workflow Matlab application that can simplify and minimize the users actions.

3 Contributions

- Easy-to-use, portable and cheap robot that will help the patients who cannot have multiple therapy sessions in a single week for various reasons, providing them a tool that can enable them to keep working out everyday whereas there is a therapist at that instant or not.
- The first contribution explained also benefits the doctors because this tool can help them in lifting some of the burden of their work. By providing the possibility of creating programs that classifies the movement and can give response to the doctor or user.

4 State-of-the-art

For hundreds of years, prostheses or mechanical instruments have been used to help people with reduced mobility or amputations [3]. And even today many of them are still used. These were driven by the same user with their functional limbs as it can be seen in the Figure 1.

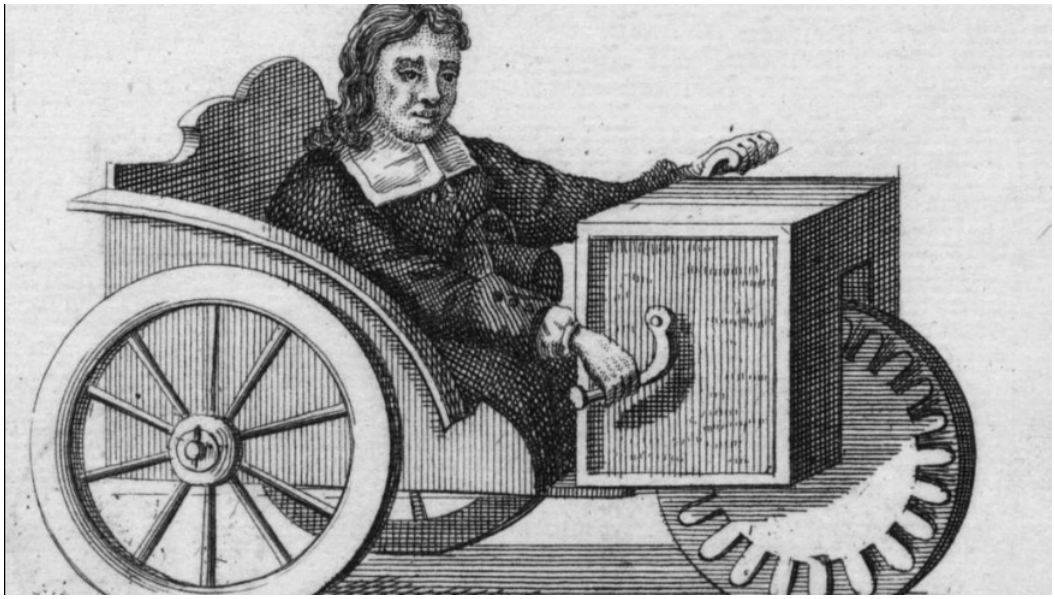

FIG 1: Stephen Farffler 1633-1699 first self-propel wheelchair

In the contemporary age and with the evolution of the technology, it has been possible to improve the help with the introduction of electronics and modern mechanics. In 1950 the first electric motorized wheelchair was created (Figure 2).

FIG 2: Stephen Farffler 1633-1699 First electric motorized wheelchair

Over time, medical and scientific research began to develop the first medical applications with myoelectric technology[4]. This new technology consist on

reading the muscle flexes or contracts with the use of sensors . In 1945 in Germany an inventor named Reinhold Reiter developed the first myoelectrical prosthesis with a simple mechanical design with a clamp and 3 possible positions (Figure 3). The logic was still not programmable yet and was based on simple electronic operational comparators that compared the voltage.

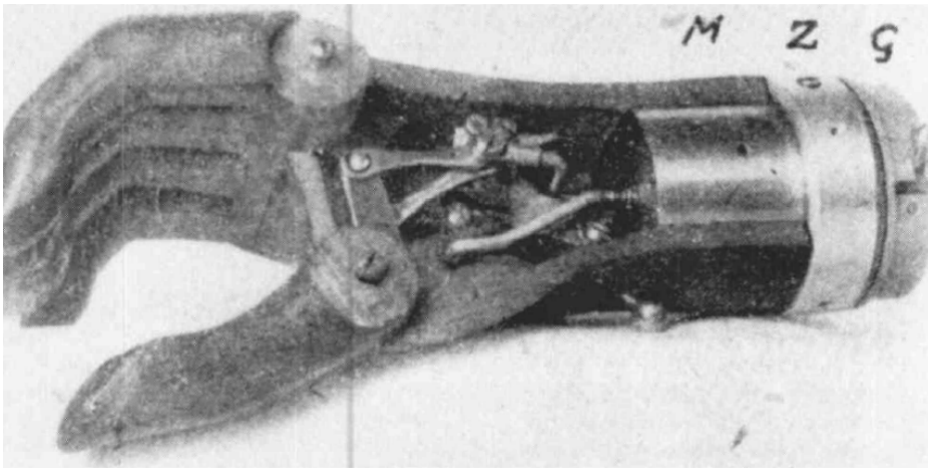

FIG 3: Reiter prosthesis

The next implementation depicted in Figure 4 was the hand of the USSR made in 1959 and designed for adults with amputations until colt. The great innovation that this project introduced was to introduce the frequency spectrum in the signal analysis, this was used to filter the signal gathered from the sensors and to process just the useful data.

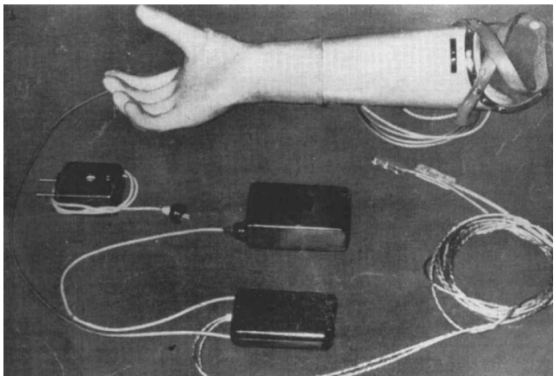

FIG 4: USSR hand - 1959

In 1973 in Sweden the firs prosthesis with 3 degrees of movement and 6 pairs of electrodes was developed (Figure 5).

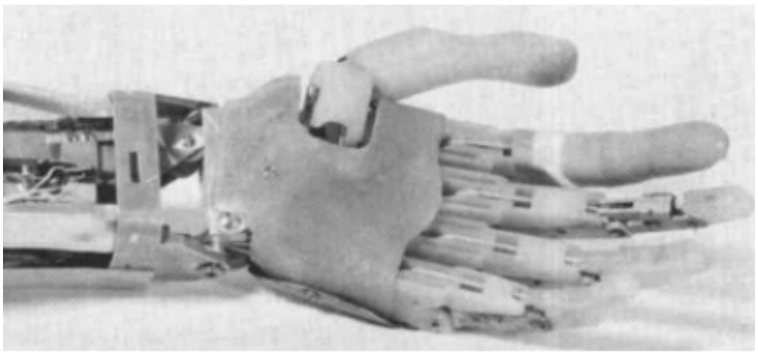

FIG 5: Hand Sven - 1973

In the 80's the evolution of the microprocessor industry made it much more cheap and easy to implement technological components into new or old projects. This characteristic enabled to create projects with much more complex software and extend its functionality.

Researcher's started applying mathematical signal processing software and implementing control software (PID's), which benefited the evolution of robotics by providing tools of control the signal given and returning a controlled action.

Nowadays, the technology called machine learning has archived a state of constant improvement and optimization due to the research made. This fact allows new projects to be implement the technology with smaller hardware and for a cheapest price. Machine learning specially took relevance in signal processing from the environment or from the users response.

One of the most modern examples currently applied these technologies for the rehabilitation of patients with degenerative diseases is the Atlas2020 project [5]. It is an exoskeleton of the lower trunk which objective is to improve the quality of life of children with muscular atrophy and re-establish deterioration (Figure 6). This project reads the residual force produced by the user, and with the help of an exoskeleton combined with machine learning algorithm's it help's the children movement by providing force to the legs.

FIG 6: Atlas2020 - 2018

Another very interesting and quite similar project to ours that is already being commercialized is the Armeo Project (Figure 7). This project, is oriented to users who have almost zero mobility in the arm, allowing them to accomplish specific exercises increasing the users arm force of specific movements and helping the doctors evaluate their response.

FIG 7: Armeo Power - 2006

5 Development

To make this section clear we will differentiate the hardware, the software and the interfaces implemented.

5.1 Hardware

For this project, we used a robotic exoskeleton kit called EduExo[6]that provided us the basic hardware components that we needed to develop this project for a low cost. This Kit consists in one degree-of-freedom elbow rigid exoskeleton that is made to combine it with off-the-shelf components. The components that will be used are servomotor, that will give information about the current position of the arm, a couple of electromyographic sensors and a force sensor. The sensors combined with an Arduino micro-controller will be acting as data gatherers. The force sensor is a standard strain gauge-based design that enables to measure the interaction force between the exoskeleton and human using a Wheatstone bridge. Because the signal of the force sensor was so weak, we had to implement a small circuit in order to amplify the data given by the sensor so the Arduino microcontroller can read it. The amplifier chip ones is connected, gives a reference voltage that has to be connected to a Wheatstone bridge, in our case, the force sensor. The output voltage from the bridge gets amplified by the chip and then its connected to the Arduino. The Figure 8 shows the schematics of the circuit implemented.

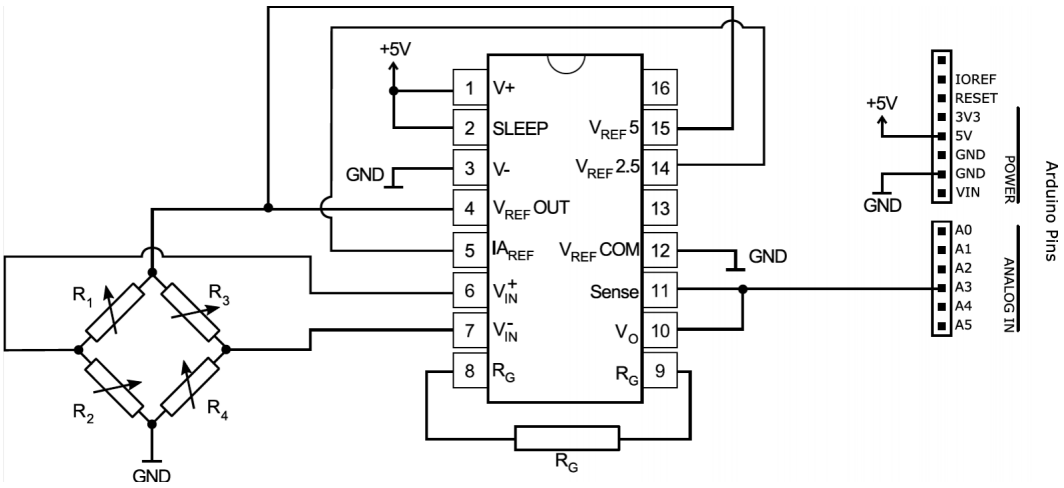

FIG 8: Schematics between force sensor, amplifier chip and arduino

Even we implemented the circuit perfectly, the voltage read from the amplifier wasn't enough to measure the interaction force between exoskeleton and human. These situation was because the force sensor is located in the same edge of the arm. So we decided to ignore the data gathered from the force sensor in the machine learning algorithm but still use this data in the sensors interface (Figure 13) visualization in order to enable the doctor to see the sensors output information.

5.1.1 Sensors

The sensors arm location in order to get valuable data for the machine learning algorithm have been the following:

- Electromyographic sensor 1: Biceps
- Electromyographic sensor 2: Triceps
- Position given by the servo: Elbow

In order to obtain and process the data given by the sensors we used an Arduino Uno, one of the cheapest microcontroller compatible with Matlab. This will enable to reduce the implementation cost in the real life. The components diagram of the final implementation are described in Figure 9.

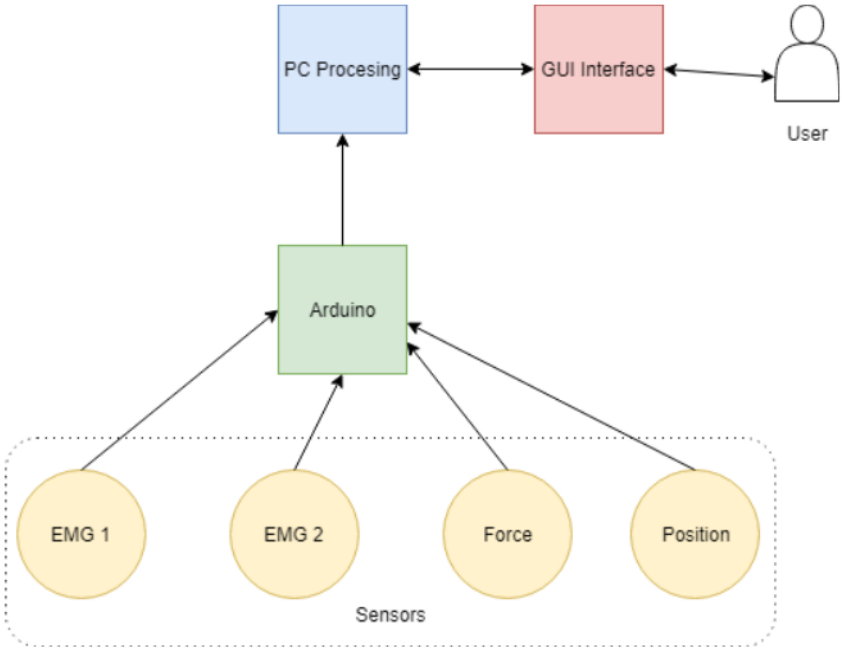

FIG 9: Components Diagram

5.2 Software

5.2.1 Matlab

Matlab libraries are compatible with Arduino's hardware and allows you to take the data with the serial port as if we were using the software provided by Arduino. So we decided to implement all the software with Matlab, which simplified the code and the transition problems that we could have had if we used both interfaces, and we can take advantage of the processing tools that Matlab Offers.

5.2.2 Processing Arduino Signals

We connected the Arduino with the computer via USB, and we implemented a code that automatically detects the Arduino port without introducing manually the port given. This implementation makes the program compatible with all computers and enables the user to use the program without worrying about external factors as the port the Arduino is connected. This connection will also allow the code automatically gather the data from the sensors.

The sensors output signal is analogical, and we found that sometimes there can be interferences produced by the signal coupling of the Arduino circuit or the sensors noise that may affect the classification algorithm. So we decided to create a filter that consist on the median from the last 3 samples taken in order to prevent them as much as possible. The number of samples used to do the median of the signal were decided after considering the number of total samples taking each training or classifying iteration, which were 10 and because two samples made the code not to filter properly the interferences.

In order to classify and train algorithm correctly, we needed to have a constant frequency data sampling. Due to the response time of the Arduino, and the code implemented in Matlab we realized that the maximum constant sample frequency we could archive was 5 HZ.

5.2.3 Machine learning

In order to train the algorithm we used a Matlab function named trainNetwork that is used to train a long short-term memory (LSTM) network for deep learning classification and regression problems [7]. Because our projects focuses in training the algorithm to classify users actions, this algorithm fits perfectly with our problem.

The characteristics of trainNetwork change depending on the options introduced or the algorithm applied. To use this algorithm, we had to process and modify the data obtained from int to a categorical cell array in order to store the data obtained and the options introduced. The trainNetwork function creates an interface (Figure 19) which can be personalized and depends on how you execute the deep learning algorithm to get a better result or view of the information given [8]:

- **Changing the Interface**: there are 3 options you can personalize in order to change the interface shown when the training algorithm is executing:

  Sequence Input: Number of different variable data taken, in our case 3 (2 EMG and 1 Positioning sensors).

  LayerAlgoritm: In our case LSTM with 100 hidden units that are the number of variables created by iteration for calculating next step, after trying several times we found out that the best results given were with 100 hidden units.

  Connected layer: Number of Classification variables given, in our case this variable depends on the doctors choose, so we make it variable.

- **Changing the algorithm**: there are 3 options you can personalize in order to change the algorithm execution :

Solver Name: It is used to specify the gradient and squared gradient done. In classification signal problems the Solver used is adam with the gradient 1 to give just possible classification for each sample iteration. The reason of choosing the adam solver was because is oriented for noised environments which is our case and for his optimization property which allows to have low memory requirements and is computationally efficient model[10].

MaxEpochs: Number of times the algorithm is repeated. We made several run tests to see the accuracy of the algorithm depending on this variable and we found out that with 800 Epoch it was perfect.

As we can see in the Figure 10 in the graph as we increase the MaxEpochs, the accuracy is also increasing, but after 800 it starts to saturate. And it is not profitable to increase more because of the relation improvement / price.

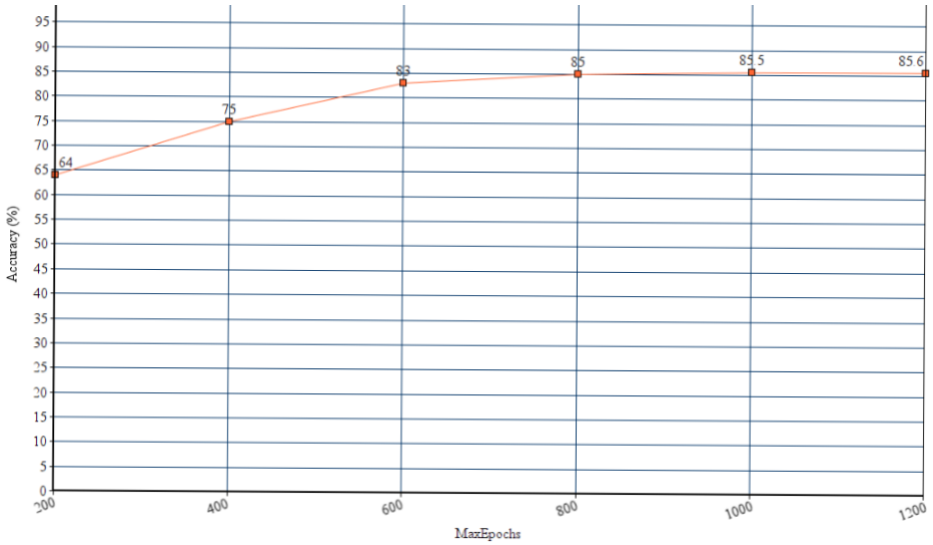

FIG 10: graph of the relationship between accuracy and maxepochs

MiniBatchSize:this characteristic controls the accuracy of the estimate of the error gradient when training neural networks. A after several tests we found out that the batch size that works better with our problem is 128[9]. With fixing max epochs we have been increasing the values of MiniBatchSize and we have observed in the Figure 11, in 128 the accuracy is stabilized

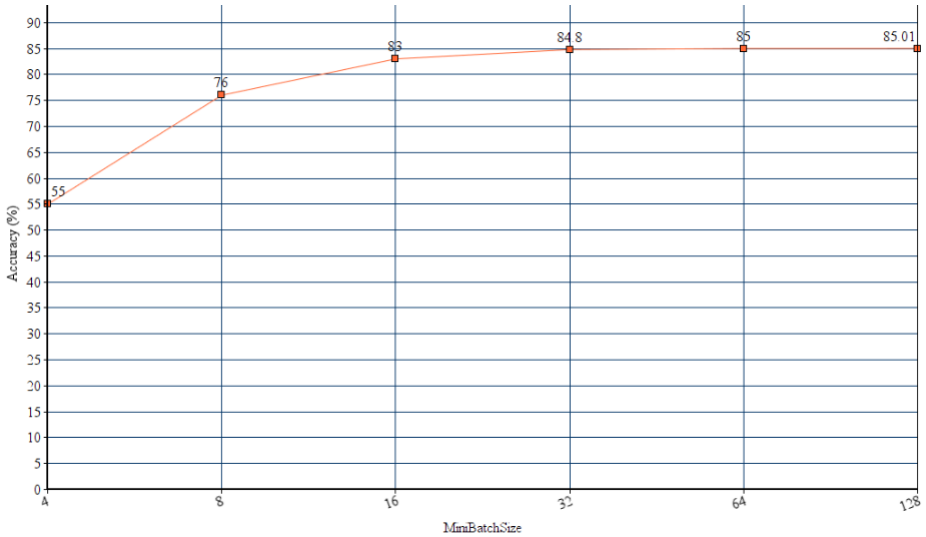

FIG 11: graph of the relationship between accuracy and MinBatchSize

Verbose: In order to validate the data given, to avoid problems it is recommended to leave it in 0.

5.3 Interfaces

As we said, one of the main goals was to create a generic workflow of the Matlab application created that can simplify the users actions. In order to archive this goal, we made the following interfaces:

5.3.1 Menu interface

As it can be seen in the Figure 10, the first interface consists on a menu interface where the user can select 4 different options, all of them except the "Exit the program" have a short explanation of what each button does:

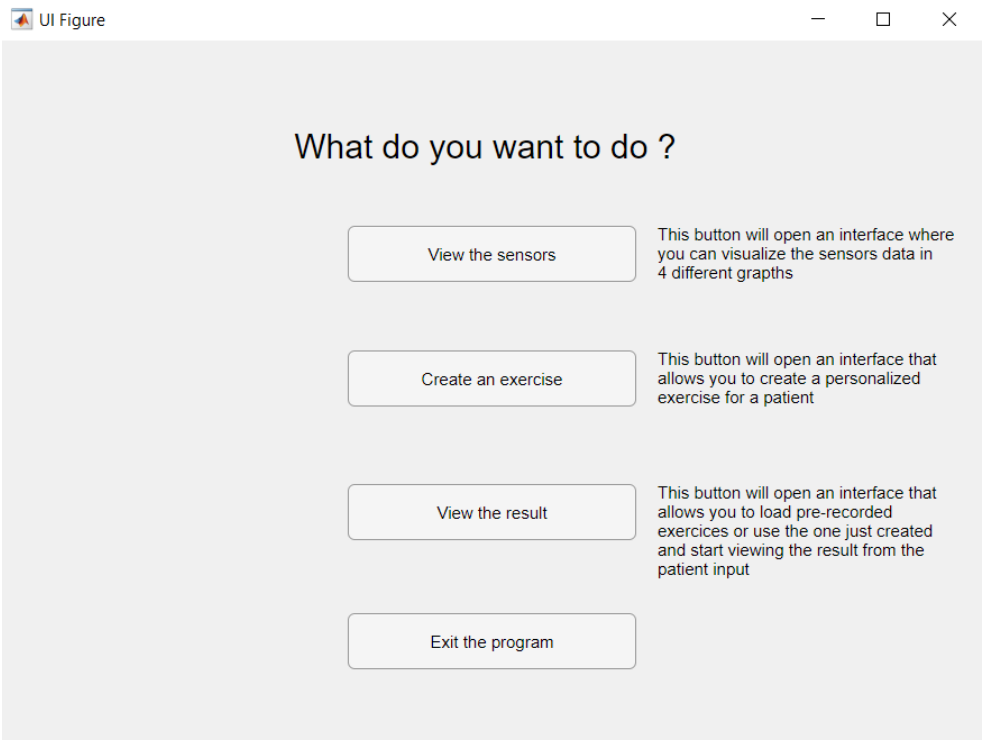

FIG 12: Menu Interface

Whenever the user press the "x" button or the "Exit the program" button the program will end.

5.3.2 Sensors interface

This options enables the user to view the data output from the 4 sensors explained earlier, and the medium of each result in separate plot metrics as it can bee seen in the figure 11. The graph moves according to the real time that the user is spending in this option and resets when he goes out of this interface.

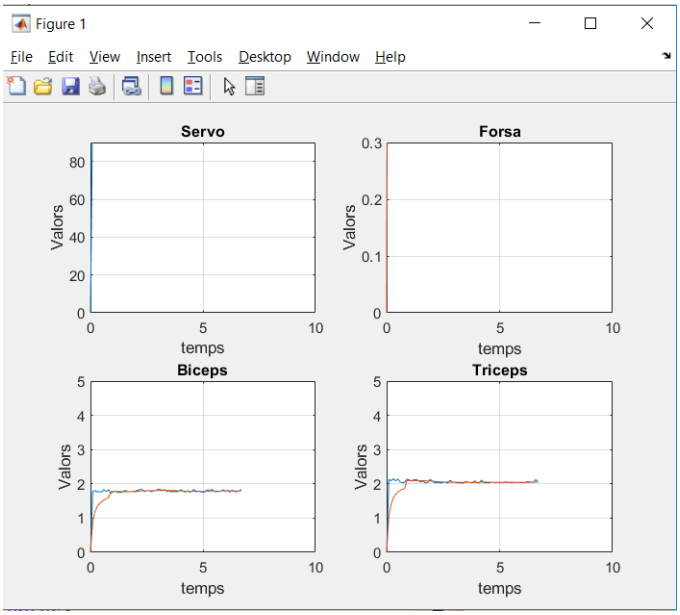

FIG 13: Sensors Interface

Whenever the user press the x button, this interface will close returning to the Menu interface.

5.3.3 Create an exercise interfaces

This interface is composed by 4 sub-interfaces and all of them are designed to prevent the algorithm code to from failing. The first one which consists of the Figure 12, is where the doctor should insert the options of the exercise that the doctor wants the patient to reproduce. To introduce an option the user have to write it in the first label and press the add option button to save the variable, whenever the doctors end in introducing the options that wants to classify, you have to press the start training button to move to take the sampling options. The options you selected will be showed in the listbox below of the label. If in any case you want to return to the menu press Go to Main Menu or the x button, but your saved variables will be erased.

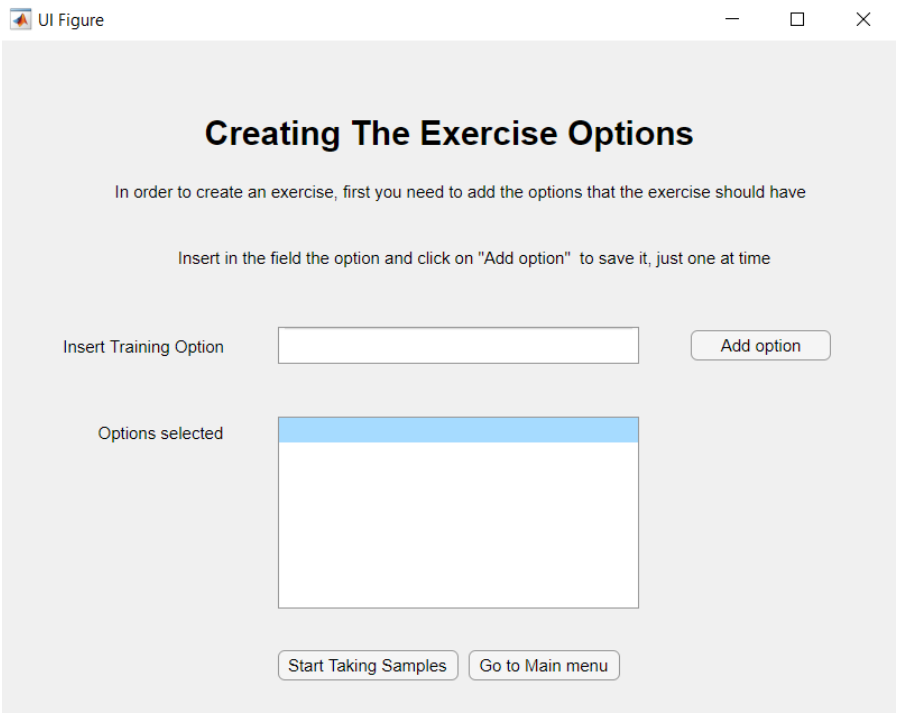

FIG 14: Choose Interface

If you press the start button without introducing any option there will be a warning message as it can be seen in Figure 13, making it impossible to the doctor to start the sample data acquisition of the exercise.

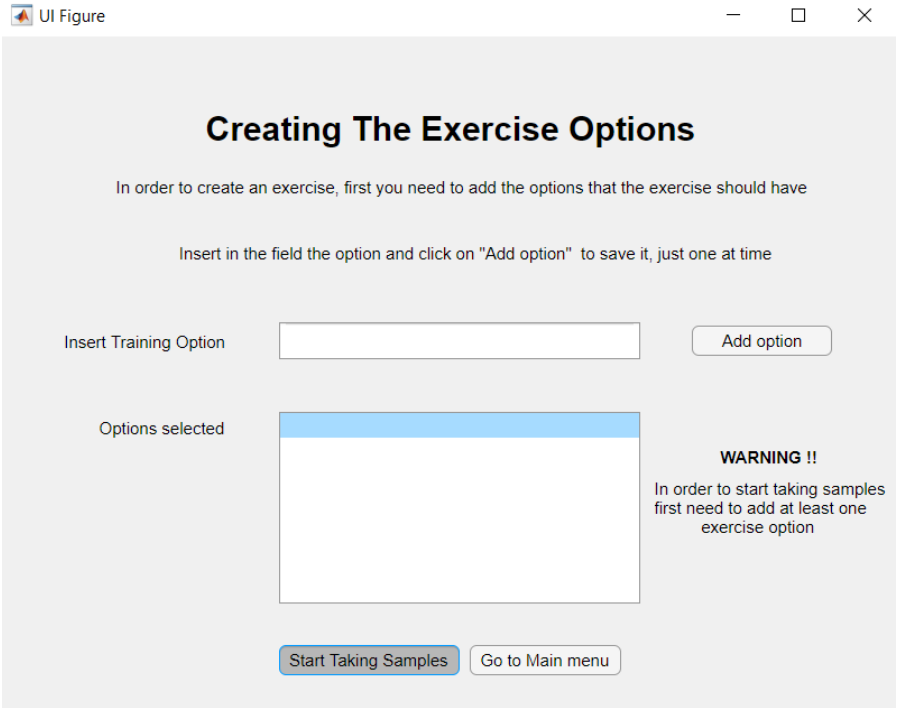

FIG 15: Warning in training Options

When the user presses "Start Taking Samples" button a new interface (Figure 14) will appear. In this interface there will be displayed the exercises options just created by the doctor, and will be used to take training samples from the user. To take a sample the doctor should select an exercise option created and press "Take Sample", then a plot interface will open showing the sensors data from 2 seconds duration since the button was pressed. This will enable the doctor see if the data acquired is correct.

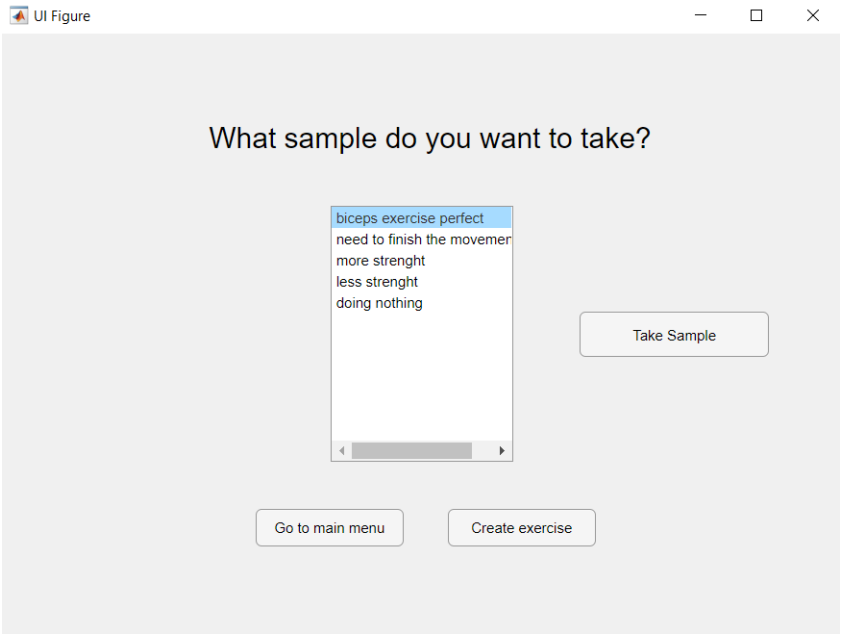

FIG 16: Training Classify Options

As it can be seen in the figure 15, after taking a sample the algorithm will show a message that enables to delete the last sample taken with a button of confirmation, if it is pressed, the code will erase the last sample taken. This is very useful to the doctor because he can visualize the result and just keep the good results improving the efficiency of the machine learning algorithm performance.

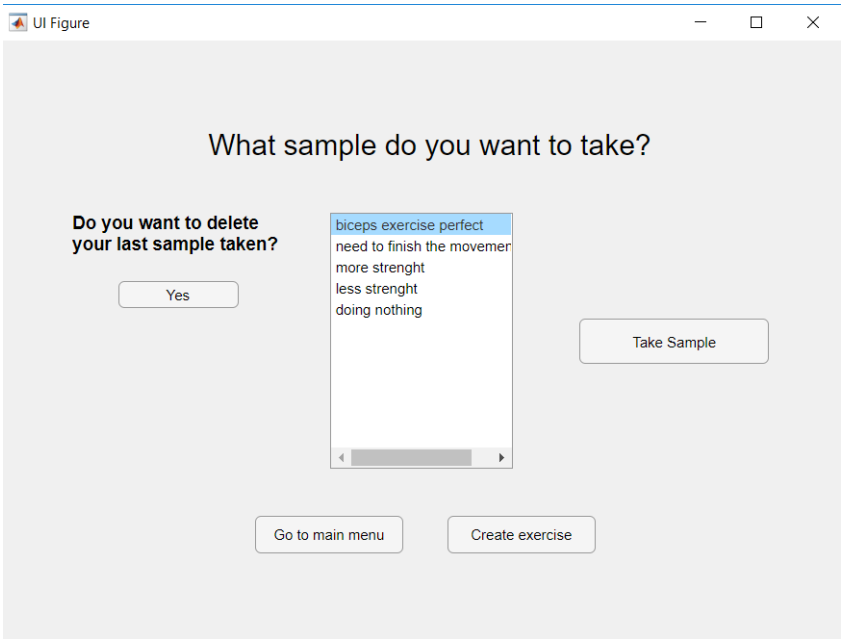

FIG 17: Delete Last Sample Taken

Whenever the user press the "x" button or the "Go to main menu" button the program will close the interface and return to the Menu interface, loosing all the data from the exercise created. If the user press the "Create exercise" option without having taken any sample, there will be a warning message showing up in the interface that can be seen in the figure 16.

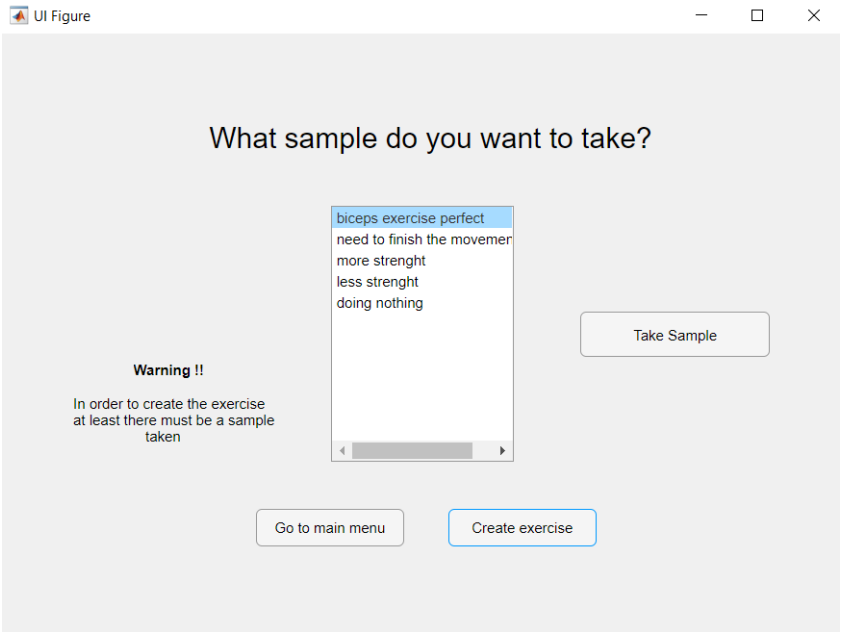

FIG 18: Warning Take Samples Options

If the user press the "Create exercise" option ones introduced at leas one sample, the training algorithm interface will start executing the machine learning code with the options previously mentioned (Figure 17).

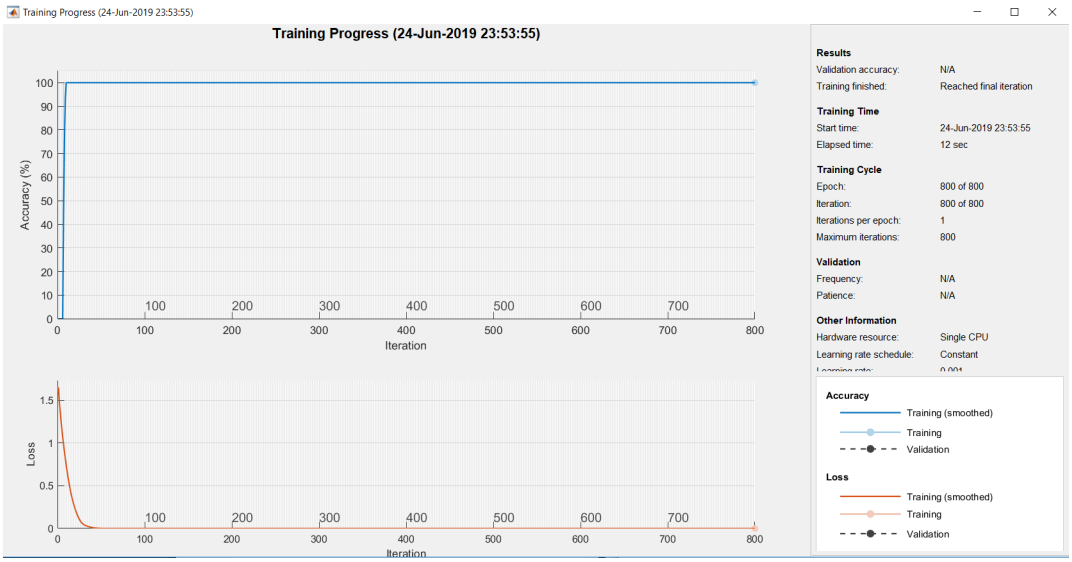

FIG 19: Train algorithm Options

At the end of this process the user will be redirected to the Menu interface.

5.3.4 View the result interfaces

When the user press the "View the result", It will appear the Figure 18 Interface where you can select from 3 already saved trained exercise algorithms, the one that the doctor just created.

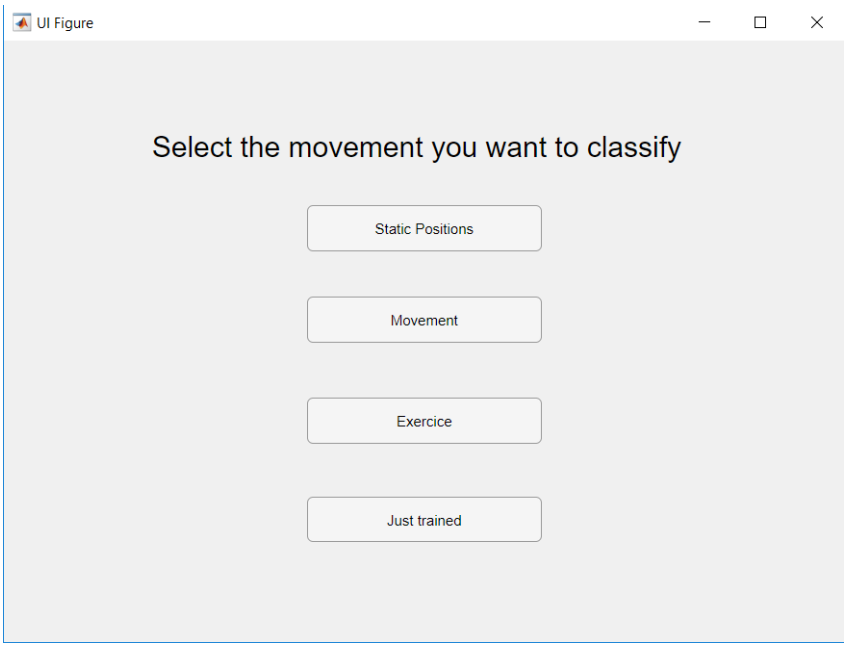

FIG 20: Result of Classify Options

Whenever the user press the x or the "Go to Main Menu" Button, this interface will close returning to the Menu interface. If you select the "Just Created Exercise" button but no exercise is created in this execution, a pop up interface (Figure 19) will show with a warning that will return to the previous interface (Figure 18) when the continue button is pressed.

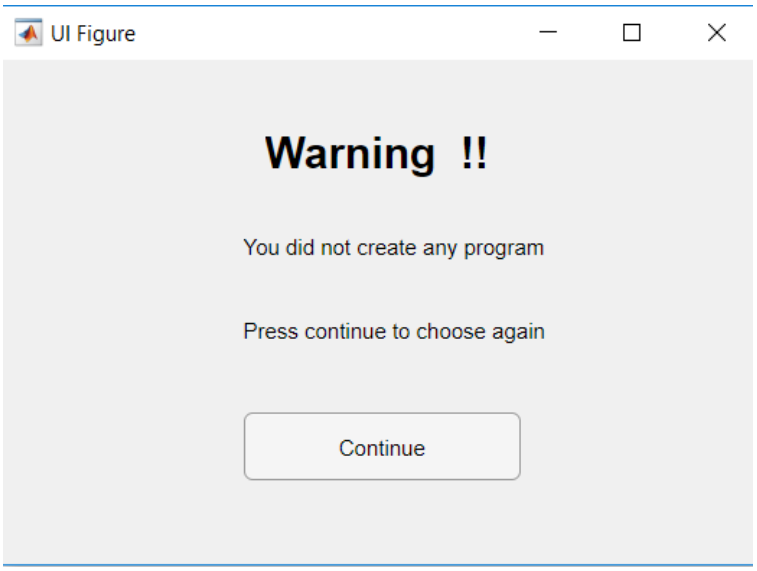

FIG 21: Warning in classify

If it is not the case there will be a plot interface (Figure 20) showing the sensors information and the result of the classification algorithm of the last 2 seconds

of samples, comparing this samples with the matrix of selection given by the trained code algorithm.

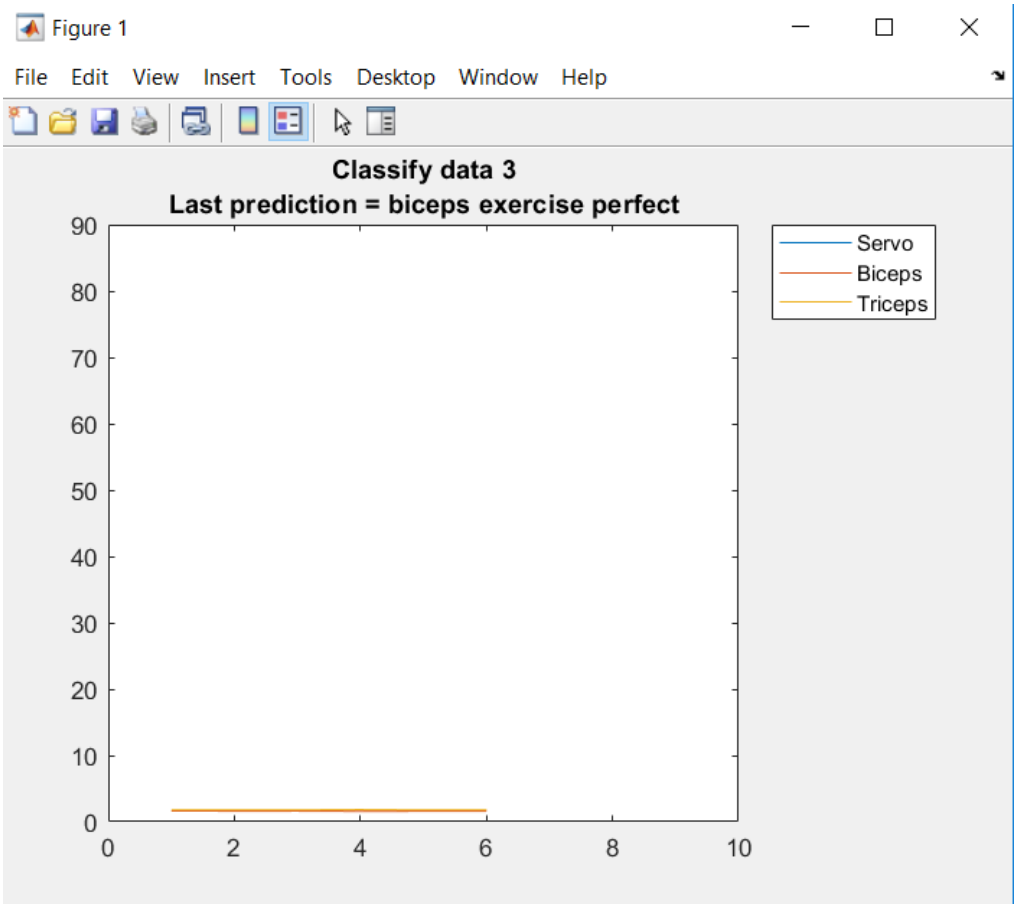

FIG 22: Plot with the code prediction

Whenever the user press the x button, this interface will close returning to the Menu interface.

6 Evaluation

First we started with a simply classification to test our implementation and and verify the correctly performance of our machine learning. That was only two components to classify, the first a movement ascendant of the arm, and the second a movement descendent. Once we had trained the machine we test the result of a direct movement classifying the data and we saw that the system worked without any problems or errors movement of the classification.

Once we have a basic functionality of the system we tried movements with more complexity and with more input variables (in the first test we only used the angular sensor), then we realized that the accuracy of our trained movements are directly proportional with this thinks:

- The quantity of data of each action to classify.
- The similarity between the exercises to classify.
- The relevance or proportional relation of the dates we extract with the final response of the arm.
- The quality and resolution of the data.

We tacked several tests with different exercises, now we are going to explain some of the most significant and indicate their accuracy and analyze each result.

Movements with and without dumbbell

We trained machine with differences movements and each of that movement we did with dumbell (of 2 kilograms) and without dumbell. The options to classify are:

- ascendancy movement (between 0^o to 30^o) without dumbell
- ascendancy movement (between 0^o to 30^o) with dumbell
- ascendancy movement (between 30^o to 60^o) without dumbell
- ascendancy movement (between 30^o to 60^o) with dumbell
- ascendancy movement (between 60^o to 90^o) without dumbell
- ascendancy movement (between 60^o to 90^o) with dumbell
- descendant movement (between 0^o to 30^o) without dumbell
- descendant movement (between 0^o to 30^o) with dumbell
- descendant movement (between 30^o to 60^o) without dumbell
- descendant movement (between 30^o to 60^o) with dumbell
- descendant movement (between 60^o to 90^o) without dumbell
- descendant movement (between 60^o to 90^o) with dumbell

In this case we obtained an a total accuracy of 85%. And if we separated the cases (only train with movement ascendent or descendent or only with dumbell or without) in the case of only ascendancy, the accuracy that we obtained are about 95% and only training the state of the weight that are keeping the arm we obtained an accuracy of 65%. With this data we have calculated that the direct relationship in percentage between the partial accuracy and the total accuracy is 0.66 on the part of the type of movement (ascendant or descendant) and 0.33% of the part of the weight that sustains the patient. This is because the quality of the data in the case of the movement (the encoder) is much better and precise that the data that we obtain with the myowares.

In this case also we realized that the previous medium filter that we applied increase reasonably the accuracy of the classifying, and that the force sensor are insignificant and this is because the quality of the data of the sensor force is deficient.

Execices of rehabilitation

We trained the machine with two type of musculature exercises, biceps dumbbells and triceps dumbbells, we obtain a great accuracy in this case about 92%. In this case we tried to implement also to introduce push-up's but we observed that the classification have many problems to differentiate the triceps dumbbells with the pull-up's. In our opinion we think that this could be because three concrete things:

- The similarity of the responses that we have between this two movement(in both we have a sinusoidal movement and we worked the same muscle)
- The few attributes of inputs that we have (may be with an accelerometer and more myowares we should increase the accuracy)
- And (again) the improvable quality of the myoware's and the laughable quality of the sensor force.

in the Figure 23 we can see the accuracy respect to the number of times trained for each classification.

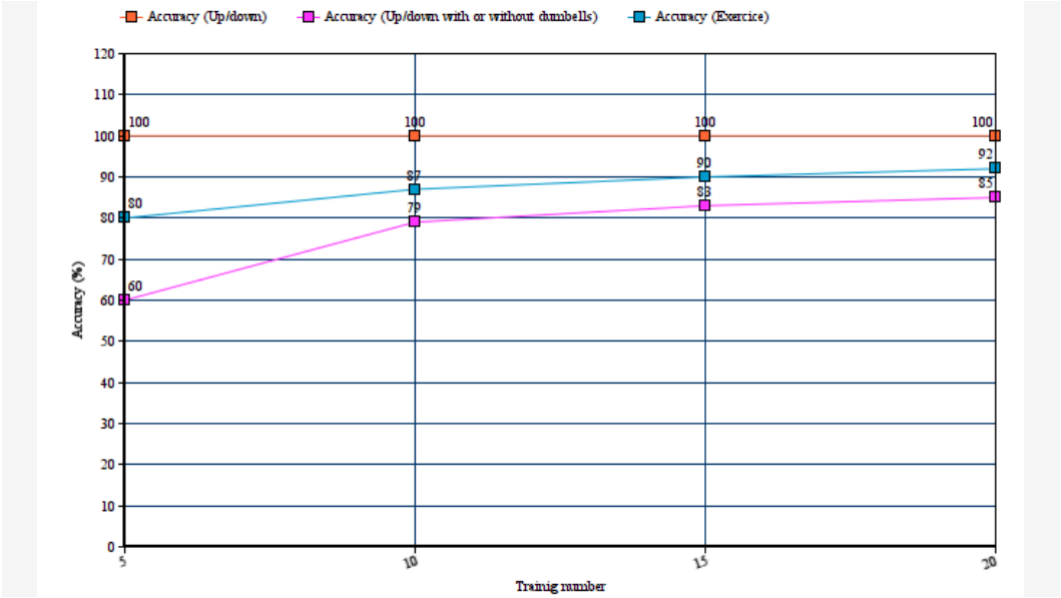

FIG 23: Graph with accuracy respect number of Trainded items

7 Conclusion

With this project we have been able to implement machine learning technology for the first time and we have seen its usefulness. We have also been introduced to the electromyographic sensors and how their work and its limitations. With the intention of creating a tool that could be useful and usable for the rehabilitation of people with reduced mobility.

We have faced a project that we did not know where to start, but little by little we have been solving the different challenges that we have been encountering in order to advise the objectives established for the project.

Our main objective was to help in some way to people with reduced mobility and the doctors who makes the therapy by creating an easy and cheap tool that can be useful in both sides by helping the patient carry out the rehabilitation exercises so that he does them correctly advising him during his performance depending on the doctors thoughts, strengthening his muscles and slowing down his possible muscular, neuronal and neuro-muscular disease.

In order to improve the classification in real life as much as possible, we would need some medical electrodes sensors to improve the data acquirement and move the strength sensor between the biceps and the forearm to gather data from it. Also we would like to create a users manual in other to explain to the doctor or the patient how it works, and also implement a new function to detect improvement over the time and the patient.

8 Bibliograpy

References

[1] https://elpais.com/sociedad/2019/01/23/actualidad/1548273989_380935.html

[2] https://medlineplus.gov/neuromusculardisorders.html

[3] https://marieandredestarac.wordpress.com/2016/09/07/exoesqueletos-comerciales-para-la-rehabilitacion-del-brazo/

[4] https://iie.fing.edu.uy/publicaciones/2015/BPV15/BPV15.pdf

[5] http://www.marsibionics.com/portfolio/atlas2020/

[6] https://www.eduexo.com/eduexo-kit/

[7] https://www.xataka.com/robotica-e-ia/machine-learning-y-deep-learning-como-entender-las-claves-del-presente-y-futuro-de-la-inteligencia-artificial

[8] https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html

[9] https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

[10] https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/