Using forearm sEMG to control individual fingers of a robotic hand

Machine Learning Project II

Milo Imbeni, Stella Petronio, Samuele Caneschi
EPFL, Lausanne, Switzerland

December 17, 2020

***Abstract*—With the improvement in Machine Learning and in signal processing techniques, the EMG signals' relevance has progressively incremented in man-machine interaction. Hand prostheses with multiple degrees of freedom using sEMG for control have come up in the market in the last few years. Our project's focus was on a regression from sEMG recorded by a 64-channels acquisition array to predict the finger angles of six joints of the hand. Regression was made through a standard RBF-kernel-based Support Vector Machine and MultiLayer Perceptron on manually extracted features and with more complex deep learning algorithms such as 3D Circular Convolutional Neural Networks. These are not reliant on feature extraction but achieved results comparable to the best "traditional" model.**

I. Introduction

Transradial amputees have great difficulty to make use of the remaining limb. The absence of a grasping hand hugely hinders daily life quality, which can be recovered with a hand prosthesis. Traditional devices have only one or two degrees of freedom controlled with the shoulder, with movements that need to be re-learned. New approaches involve biomimetic robotic hands that can be controlled with the remaining forearm muscles. Prosthesis of increased complexity with multiple degrees of freedom allow different pre-programmed grasps, but solutions, that allow single finger movement control, have not yet reached the market: this is due to their not yet optimal robustness in real time [1]. The control approach usually makes use of external or internal electromiography (EMG) electrodes, which integrate the action potentials of the single motor units that compose the muscles of the forearm. The intensity and frequency of these pulses determine the contraction strength. Intramuscolar electrodes, which need to be physically implanted, hold stability and accuracy advantages over surface EMG (sEMG); nonetheless, due to the invasive procedure associated, their long term use might not be optimal. The goal of EMG decoding algorithms has traditionally been to classify between various types of pre-defined grasps to improve robustness. However, single finger proportional control would greatly improve the dexterity of the device, and it has proven successful in improving user experience [2]. Classification methods used with the features extracted from a small number (about 10) of accurately positioned sparse electrodes range from "traditional" shallow methods such as support vector machine models [3] to deep learning, with *Multilayer Perceptrons* and *Convolutional Neural Networks* [4]. The use of high density electrode arrays on the other hand solves the problem of a difficult consistent positioning of sparse electrodes, becoming invariant to the exact muscle placement while also collecting more information. Prosthesis control methods relying on HD-sEMG images have been proposed, achieving good accuracy in gesture classification [5] [6]. Spatial features contain valuable information: in a recent paper they have been manually extracted from HD-sEMG images and successfully used in gesture classification [7]. The advantage of Convolutional Neural Networks is then evident, as they are able to automatically discern spatial features. The goal of this project is to develop an ML approach able to regress single finger angles from short time windows of "medium definition" sEMG images, from a recently developed 64-electrodes array which wraps around the forearm. It is not as dense as true HD sEMG arrays but counts more electrodes than usual sparse recordings, being therefore called "medium definition". "Traditional" methods such as SVM Regression and MLP will be compared to a 3D CNN model, which uses as input a "video" composed of HD-sEMG images of the past 200 ms to allow real time control. A 3D Convolutional Neural Network with 3D kernels would be able to harness the temporal evolution of the EMG signal, which is stochastic and noisy. For example, in Chen, 2020 [8] this kind of architecture resulted in significantly better gesture classifying accuracy than a standard 2D CNN. We aim to make the best use of the cylindrical spatial positioning of the electrodes in our data by building a custom circular padding layer for 3D convolution, basing ourselves on the work in [9].

II. Data, Preprocessing & Feature Analysis

As stated in the introduction our dataset consisted of the EMG recording from a medium density array with 64 electrodes. The preprocessing was carried out in accordance with the general literature consensus using ad hoc functions developed by Vincent Mendez, our tutor from the Translational Engineering Lab at EPFL. The data regards an able bodied subject's right hand which followed *six times a sequence of movements*, alternating *flexion* and *extension* of various combinations of fingers, prompted by a video over around 10 minutes of recording with an EMG sampling frequency of 2400 Hz, resulting in a matrix with 64*1388107 points. This raw data was aligned with the labels, filtered, cut into a Train (3/5 of the total data), a Validation and a Test (each of which corresponded to 1/5 of the total data) datasets, standardized by means of the Train mean and standard deviation, and finally windowed by creating partially overlapping 200 ms windows. The overlap was set to 40 ms. The resulting datasets include then 8448 windows of 480 samples * 64 channels for the training and 2813 windows for the validation and testing of our models. This raw data was the input for the convolutional nets, while a further feature extraction step was needed for the computational approach. The Labels to regress consist in the **finger angles of 6 joints of the hand** (Figure 1) as determined by the simulation program used to generate the video. These angles go from 0 (fully extended) to 69 for the 4 fingers and 39 for the 2 thumb joints, fully flexed. These values have been resized from 0 to 10 to enhance evaluation of the performance. In particular, the joints examined are the 5 metacarpophalangeal joints, which are the articulations between the fingers and the palm, plus the carpometacarpal joint of the thumb.

There are many different criteria to find the right frequencies to define the most descriptive frequency range for the sEMG [10], but the majority of the information contained in the signal is between the frequencies of *5-15 Hz* and *400-500 Hz*. Filtering is obviously necessary to eliminate noise components present in the signal. In particular, a *notch filter* at 50 Hz was used to remove the AC power line interference and a band pass filter between 15 and 500 Hz was used to remove low and high frequency movement artefacts.

The EMG is the sum of the millions muscle potentials coming from motor neurons. It is a stochastic signal, meaning that it does not follow a specific template, whose variations can be studied and associated to the single specific biological inputs or causes. Due to this stochastic nature and to the huge quantity of data collected through the recordings,

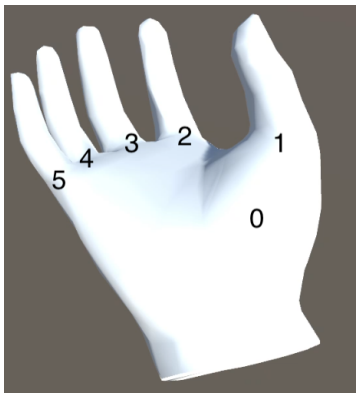

Fig. 1. Position and relative channel index of the angle labels.

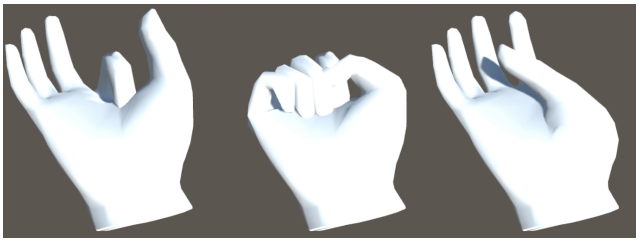

Fig. 2. Examples of movements which were performed during the s-EMG recording.

the raw signal cannot be studied as it is, at least when using a standard approach. An important part of the analysis of the signal is indeed the features extraction, i.e. the process of computing a set of meaningful features, able to describe the signal, starting from the raw EMG signal itself. Features associated to EMG and EEG are widely present in literature, since their analysis is one of the most effective methods to study those stochastic biological signals.

In particular, in this project, there are two categories of features that were extracted and used in the following phases:

- **Time-domain features**: Mean Absolute Value (*MAV*), Wavelength (*WL*), Maximum Absolute Value (*MaxAV*), Standard Deviation (*STD*), Zero Crossing (*ZC*), Slope Sign Change (*SSC*), Root Mean Square (*RMS*), Willison Amplitude (*WAMP*) and Log Detector (*LOG*);
- **Frequency-domain features**: Power Spectrum at different frequency ranges. We used only the most informative, between *150-400Hz*.

Since this was not the focus of our project, we are not going to describe in detail the formulae to compute these features, which can be found in the literature [11]. All of the above specified features were extracted for each window and normalized by a dedicated script, therefore obtaining a 10 features * 64 channels = 640 linear vector of features for all datapoints.

III. Methods and Models

All the following models were run on Google Colab and evaluated by calculating the averaged Mean Squared Error of all label channels. The hyper-parameter tuning was carried out either by *grid search* (SVM Regression) or with the support of a *genetic algorithm*. The genetic algorithm we employed from the geneal library [12] encodes the hyperparameters in the elements of a chromosome. A group of random models constitutes the initial generation: their performance (validation loss) is the fitness function: only the best models are retained. They either generate "offspring" with a combination of the traits of the "parents" or randomly mutate, leading to an efficient exploration of the hyperparameter space. We considered the architecture of the deep models as hyper-parameters: the tuning therefore includes, in addition to the regularization and optimization parameters, the number of layers, the activation function, the optimization algorithm and in convnets the pooling method to use. The complete parameter space can be found in Appendix A. The details of the best performing architectures will be detailed in the results section. Each NN training was evaluated with the MSE loss on the validation set, and run for maximum 50 epochs with early stopping in case of an improvement lower than 10e-4 over the last 5 epochs. The batch size was set to 64 for most trainings run on Colab due to RAM constraints, and increased to 256 during trainings on EPFL servers. The learning rate of the chosen algorithm was halved every ten epochs.

A. Support Vector Machine - Regression (SVR)

With the extracted features, we applied a "traditional" approach consisting of the *Support Vector Machine*, one of the most popular Machine Learning algorithms, implemented as a *multi target regression* method with the library Sklearn. The strategy undertaken consisted of fitting a separate regressor for each label channel. In order to expand the complexity of the model, *the Radial Basis Function* (RBF)

$$K(\mathbf{x}, \mathbf{x}') = exp(-\gamma \cdot \left\|\mathbf{x} - \mathbf{x}'\right\|)^2 \quad (1)$$

was chosen as a kernel function.

We applied a Grid-Search method to explore every combination of the parameters *C*, γ and ϵ, which briefly represent, respectively, the penalty factor used to set the tolerance of the model, a kernel function parameter which decides the curvature of the decision boundary and a parameter that controls the width of the insensitive zone, used to fit the training data.

In order to improve the accuracy achieved with the SVR model our focus moved on the implementation of more advanced methods such as the Multilayer Perceptron and the 3D Convolutional Neural Network.

B. Multilayer Perceptron

In theory, a deep and wide enough MLP would be able to learn to replicate whichever function with whatever number of parameters. Unfortunately the limits of our computational abilities restrict the architecture of these nets, which therefore benefit from a manual dimensionality reduction such as features extraction. The parameters that were selected with the Genetic algorithm are the number of *dense layers* (up to 5) and neurons, the *dropout rate*, the *regularization parameters* (L1 or L2), the optimization parameter *gamma* and the algorithm, between Adam, *SGD* and *RMSprop*, and finally the activation function, between *RELU*, *ELU* or *SELU*.

C. 3D Convolutional Neural Networks (3D-CNN)

The next step of our project was the training of a Convolutional Neural Network that could learn on its own the optimal features for our dataset, without the need to manually compute them as is needed with the other models.

As we reported in the Introduction, similar studies in literature ([6], [5]) underline the remarkable improvement in the labels prediction when taking the spatial information into consideration. Therefore, we decided to modify the structure of our data and tried to reflect the spacial disposition of the electrodes (Figure 3).

The first approach we decided to use was to build for each of the 480 instants in each window an *8-by-8 matrix* containing the 64 channels, then apply a traditional CNN with *zero padding*. That is simple from an implementation point of view, but does not take into account the real

circular spatial disposition of the electrodes, therefore deforming the real appearance of the EMG image. This is the 3D CNN 8*8 model.

To improve the performance of our model and get the most out of the spatial configuration of the array, we decided to reproduce the real physical disposition of the channels. The electrodes are not located in a rectangular array, as the rings of electrodes closer to the wrist are smaller. The EMG video used as input to the model however is composed of matrices, and hence must be rectangular. More in detail, the array does not have a constant number of columns: 3 electrodes in the first row, 7 from the 2nd to the 5th, 6 from the 6th to the 8th and 5 from the 9th to the 11th 3. Moreover, in the first row the electrodes are located only on the external side of the arm near the elbow. In the rest each row represents a circle around the forearm. The option of filling the holes with zeros implies neglecting the vicinity of electrodes physically close that end up being at the opposite side of the array. In fact, the reasoning behind our *circular padding* resides in the fact that the array of electrodes is wrapped all around the forearm, meaning that the electrodes of one side in its planar disposition are actually close to the electrodes of the other side, like in a panoramic 360° image. This is done by cropping and pasting the left and right slices on the opposite side, while at the top and bottom a regular zero padding is used, and repeating the process on each frame of the EMG video. This enables the use of spatial information of electrodes near the sides of the array. The models employing this padding are named CCNN, Circular Convolutional Neural Networks.

		3	2	1		
10	9	8	7	6	5	4
17	16	15	14	13	12	11
24	23	22	21	20	19	18
31	30	29	28	27	26	25
37	36	35	34	33	32	
43	42	41	40	39	38	
49	48	47	46	45	44	
	54	53	52	51	50	
59	58	57	56	55		
	64	63	62	61	60	

Fig. 3. On the left, spatial disposition of the electrodes with the associated numeration. On the right, reconstruction of the electrodes array in a rectangular matrix; in orange are reported the positions of the "missing" electrodes.

We came up with two distinct solutions to reshape the array, and both would then allow us to implement circular padding on the newly rectangular input image. The first strategy is to resample specific rows so that each has 7 channels. This means extrapolating the trend of the voltage in a specific row from the information available from the electrodes physically present, and then *resample* that function to get 7 samples, that could be seen as "virtual" denser electrodes. That was achieved with trigonometric interpolation, taking into account the periodic nature[1] of the function to regress. Using this method, we decided to exclude the first three-elements row, because the information encoded in the three channels was not enough to describe the whole ring around the arm. In doing so, the linear array of dimension 64 was transformed into a 10-by-7 matrix, containing the spatially resampled information.

In the second strategy, we decided to take into consideration the three electrodes of the first row as well, because they contain useful information that we can exploit to predict the labels. This time, to fill in the gaps representing the "missing" electrodes (Figure 3) we decided to compute the mean of the 4 closest electrodes in its neighbourhood, i.e. the electrodes located in the Swiss-cross centered around the one we are considering.

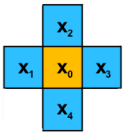

Fig. 4. Swiss-cross element used for the estimation of the missing electrodes with the second approach.

For example, if we want to compute the value to associate to the electrode in the location x_0 (Figure 4), we compute it as

$$x_0 = \frac{x_1 + x_2 + x_3 + x_4}{4}. \quad (2)$$

When considering the electrodes located on the first or the last rows, we simply compute the mean of the 3 closest electrodes. The first and last columns' electrodes values are computed by considering the wrapped array. More than one missing electrode's information depends on other missing electrodes, so to find those values we solve two independent linear systems for each one of the matrices we want to build. With this method, the 64-long array with the data becomes an 11-by-7 matrix, containing the correct spatial relationship between electrodes.

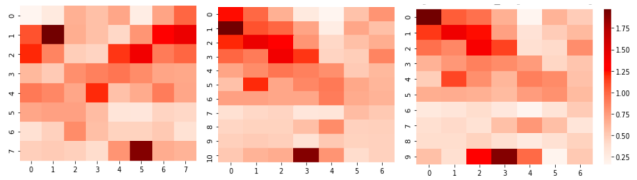

Fig. 5. A sample EMG image reshaped with the three approaches. From left to right: 8*8, mean, resample. First row elbow, last wrist. Mean and resample mantain the spatial patterns.

The above described three convnet models, CNN 8*8, CCNN "mean" and "resample" were trained with the genetic algorithm, which selected the best individual with a 17-elements chromosome(details in Appendix A).

IV. Results and Discussion

In order to better comprehend the following MSE values, it is important to know that by guessing the mean of the Test set channels in each point the error is 16.27. Every value lower than this then corresponds to better performance than guessing the mean.

A. Support Vector Machine Regression

Adopting the approach just described, the table below shows the results we obtained and the choice of hyper-parameters which gave us the minimum Mean Square Error (MSE) on the validation set.

SVR Results			
C	γ	ϵ	*MSE*
200	$3\text{x}10^{-5}$	0.7	**4.4383**

[1]The periodicity of the regression of each row of electrodes is due to the spatial periodicity: each row of electrodes is a ring that surrounds the forearm, so if a ring is made of 7 channels - hence 7 samples - the 8-th channel will contain the same information of the 1-st one

B. Multilayer Perceptron

The best performing architecture has an input layer 640 elements wide, four densely connected layers of respectively 16, 32, 32 and 128 neurons and an output of 6 elements. It relies on a 0.2 dropout rate to avoid overfitting, but no L1 or L2 regularization. The activation function is the Exponential Linear Unit which is not affected by vanishing gradients. The RMSprop optimization algorithm with an initial learning rate 10e-3 was selected above SGD and Adam algorithms. The best MSE obtained on validation data was 3.037.

C. 3D Convolutional Networks

The main result of our project was that the **Convnets with Circular padding** applied performed better than the reference 3D CNN with zero-padding and spatial deformation [2]. The parameter space over which the hyper-parameter tuning was carried out is detailed in the Appendix A. In particular, the 3D CNN with 8*8 input images best performed with the following architecture: input data 480*8*8, 2 Conv layers with respectively 32 and 8 filters and Max-pooling after every layer with pool size (2, 1, 1), three densely connected layers with 16,64 and 16 neurons, ELU activation, a kernel size of 3*3*5, dropout rate 0.2, L2 activation with lambda 0.1, RMSprop with learning rate 0.001. The final MSE on Val data is 8.923.

Here is the best performing CCNN architecture with "resample" data. Input sized 480*10*7, 4 Conv layers, with respectively 32,8,32 and 16 filters, each followed by an average pooling layer with pool size (2,1,1) plus an additional average pooling (2,2,2) to reduce tensor size, three dense layers with 32, 32 and 64 neurons, sELU activation, kernel dimension of 1*1*3, dropout rate 0.2 and no L2 regularization, Adam optimization algorithm with initial learning rate 0.01. The Val MSE after 50 epochs is 3.523.

Using the same tuning algorithm with "mean" reshaped data, the best performing architecture is: input sized 480*11*7, 4 Convnet layers with respectively 16, 64, 16, 64 filters with dimensions 3*3*3, each followed by an average pooling layer with pool size (2,1,1) plus an additional average pooling (2,2,2) to reduce tensor size, two dense layers with 32 and 16 neurons, dropout rate 0.2 and no L2 regularization, sELU activation and RMSprop optimizer algorithm with initial learning rate 0.001. The MSE on Val data is 4.644.

D. Comparison on Test data

There was a discrepancy between results on Validation and Test data. [3]. In the following table each finger angle MSE is shown. It is evident that the basal joint of the thumb proved harder to decode from EMG data for all models, while the middle and ring fingers gave the best results.

Label Channels	0	1	2	3	4	5	Avg
SVMR	7.953	8.938	4.124	6.909	8.417	5.458	6.966
MLP	8.382	6.436	3.665	2.917	3.739	3.543	4.836
CNN 8*8	11.169	8.883	6.796	4.255	5.602	7.431	7.356
CCNN resample	11.945	11.477	6.227	3.530	2.603	7.481	7.211
CCNN mean	7.991	8.290	5.515	4.092	4.347	5.230	5.911

The above Test MSE results are not considered totally reliable, as the difference between Test and Val set was not predicted during data preparation. The results will then be presented with the Val

[2]It must be noted that the tuning process was carried out on EPFL servers for CCNN models and on Colab for the normal CNN

[3]The distributions of labels in the Test and Val datasets were not exactly equal due to the different length of the gesture routine in the dataset with respect to the split ratio

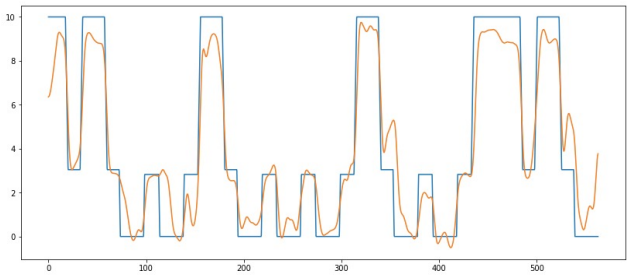

Fig. 6. Output (smoothed with a gaussian filter) vs real angles (Test dataset) of the ring finger, CCNN resample method.

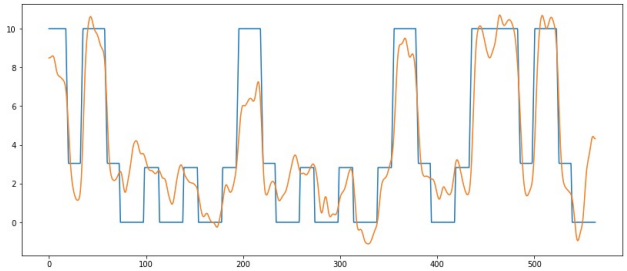

Fig. 7. Output (smoothed with a gaussian filter) vs real angles (Test dataset) of the middle finger, CCNN mean method.

MSE in mind as well. The above results are nonetheless promising: raw data can be used to regress finger angles by 3D CCNN with circular padding achieving comparable accuracy as traditional models that need pre extracted features. In particular, the "resample" approach achieved lower error on Val data but "mean" performed better on the Test dataset. The results are then inconclusive about which is better. There are some interesting common features between the best performing models: average pooling seems to be better than max pooling, maybe because of the diffuse nature of EMG signals. ELU and sELU outperform the traditional rELU activation. RMSprop and Adam optimization algorithms performed better than SGD. The 8*8 CNN has bigger filter size than the other CCNN models: because of the spatial deformation useful patterns might tend to appear further away. However, the statistical nature of the genetic algorithm hyper-parameter tuning prevents to have a clear view of the individual impact of each parameter. A bigger or more varied dataset might give a better representation of the performances of our models, which then could be selected with speed for real time control of the prosthesis in mind.

V. Conclusion

We built various valid models intending to compare their performance on our dataset. The first two, SVM and MLP, are based on manually extracted features, representing a "traditional" approach, while the following three 3D convnets are trained on raw data and differ in the reshaping of the EMG images. A rectangular reshape that preserves the spatial relationship between electrodes also allows the use of Circular padding, which means that even information on the side electrodes is used. This resulted in a better regression than with spatially distorted input, achieving comparable accuracy on raw data with respect to extracted features.

References

[1] Purushothaman Geethanjali. "Myoelectric control of prosthetic hands: state-of-the-art review". In: *Medical Devices (Auckland, N.Z.)* 9 (July 2016), pp. 247–255. ISSN: 1179-1470. DOI: 10.2147/MDER.S91102.

[2] Katie Z. Zhuang et al. "Shared human–robot proportional control of a dexterous myoelectric prosthesis". In: *Nature Machine Intelligence* 1.9 (Sept. 2019), pp. 400–411. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0093-5.

[3] Nayan M. Kakoty and Shyamanta M. Hazarika. "Recognition of grasp types through principal components of DWT based EMG features". In: *IEEE ... International Conference on Rehabilitation Robotics: [proceedings]* 2011 (2011), p. 5975398. ISSN: 1945-7901. DOI: 10.1109/ICORR.2011.5975398.

[4] Manfredo Atzori, Matteo Cognolato, and Henning Müller. "Deep Learning with Convolutional Neural Networks Applied to Electromyography Data: A Resource for the Classification of Movements for Prosthetic Hands". In: *Frontiers in Neurorobotics* 10 (2016). ISSN: 1662-5218. DOI: 10.3389/fnbot.2016.00009. URL: https://www.frontiersin.org/articles/10.3389/fnbot.2016.00009/full.

[5] Weidong Geng et al. "Gesture recognition by instantaneous surface EMG images". In: *Scientific Reports* 6.1 (Nov. 2016), p. 36571. ISSN: 2045-2322. DOI: 10.1038/srep36571.

[6] Alexander E. Olsson et al. "Extraction of Multi-Labelled Movement Information from the Raw HD-sEMG Image with Time-Domain Depth". In: *Scientific Reports* 9.1 (May 2019), p. 7244. ISSN: 2045-2322. DOI: 10.1038/s41598-019-43676-8.

[7] F. Nougarou et al. "Pattern recognition based on HD-sEMG spatial features extraction for an efficient proportional control of a robotic arm". In: *Biomedical Signal Processing and Control* 53 (Aug. 2019), p. 101550. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2019.04.027.

[8] Jiangcheng Chen et al. "High-Density Surface EMG-Based Gesture Recognition Using a 3D Convolutional Neural Network". In: *Sensors* 20.4 (Jan. 2020), p. 1201. DOI: 10.3390/s20041201.

[9] S. Schubert et al. "Circular Convolutional Neural Networks for Panoramic Images and Laser Data". In: (June 2019), pp. 653–660. ISSN: 2642-7214. DOI: 10.1109/IVS.2019.8813862.

[10] Carlo J. De Luca et al. "Filtering the surface EMG signal: Movement artifact and baseline noise contamination". In: *Journal of Biomechanics* 43.8 (May 2010), pp. 1573–1579. ISSN: 00219290. DOI: 10.1016/j.jbiomech.2010.01.027.

[11] Cemil Altın and Orhan Er. "Comparison of Different Time and Frequency Domain Feature Extraction Methods on Elbow Gesture's EMG". In: *European Journal of Interdisciplinary Studies* 2.3 (Aug. 2016), p. 35. ISSN: 2411-4138, 2411-958X. DOI: 10.26417/ejis.v2i3.p35-44.

[12] Diogo Matos Chaves. *Introducing GeneAl: a Genetic Algorithm Python Library*. June 2020. URL: https://towardsdatascience.com/introducing-geneal-a-genetic-algorithm-python-library-db69abfc212c.

Appendix A
CNN hyper-parameters selected with the Genetic algorithm

This appendix contains all the possible choices of the parameters selected with the Genetic algorithm:

- The number of *hidden-layers* for the Convolutional Neural Network can be selected between 2, 3 and 4;
- The number of *filter* for each layer can be 8,16,32 and 64;
- The value of the *drop rate* can be chosen between 0, 0.2 and 0.5;
- The number of nodes in the *first* hidden layer can be 16, 32 and 64;
- The number of nodes in the *second* hidden layer can be 0, 16, 32 and 64;
- The number of nodes in the *third* hidden layer can be 0, 16, 32 and 64;
- The value of the parameter *lambda* can be selected among 0.1, 10_{-3} and 0.1;
- The value of the parameter *gamma* can be picked out among 0.1, 0.01 and 0.001;
- The *size* of each filter can be considered among 1, 3 and 5;
- The choice of the *algorithm* can be done among Adam, SGD and RMSprop;
- The *activation function* can be selected among ReLU, ELU and SeLU;
- The *pooling* can be chosen between the average pooling and the max pooling.