Deep learning for hand prosthesis piloting based on sEMG signals

Benady Antoine
Master MVA
École normale supérieure Paris-Saclay
antoine.benady@ens-paris-saclay.fr

Reme Raphaël
Master MVA
Institut Polytechnique de Paris
raphael.reme@telecom-paris.fr

Abstract

The development of comfortable hand prosthesis is still an open challenge. The question of retrieving the user's intentions is limiting in this design. Over the past few years, research has shown that the exploitation of electromyographic signals (sEMG) is promising. Recent advances in deep learning have opened up new perspectives for predicting user intent from sEMG signals. Our work focuses on the use of recursive neural networks (RNN) to predict the position of each hand angle from sEMG signals. We implemented architecture inspired from literature based on LSTM cells (Long-Short Terms Memory). We also take the initiative to extend these architectures to GRU (Gated recurrent unit) and to introduce new loss functions.

1. Introduction

Active, handy and cheap hand prosthesis conception is currently a challenge. It's limited by the complexity of the movements of the human hands, and by the interpretation of the user's will. This challenge can have important repercussions on the lives of amputees of superior limb.

In the context of upper limb prosthesis design, one main difficulty remains the implementation of the human-machine interface. For years, the state-of-the-art has focused on the search for a solution to acquire the user's intention. The problem is often stated as a classification problem [10], where the goal is to find the movement desired by the user among a set of gestures and then execute it. But this formulation of the problem is really limited by the number of gestures considered, and does not grant an intuitive control over the prosthesis. Some recent results seem to show that, with the recent breakthroughs in deep learning, the regression problem could be addressed [6] (guessing directly all the relative positions of the hand from the user's will). It brings a much more intuitive control over the prosthesis and is not limited by a set of pre-defined gestures, which would significantly improve the comfort of people with prostheses.

To date, the most promising technique remains the processing of electromyographic signals. Surface electromyographic electrodes (sEMG) measuring a set of muscle-activating action potentials are widely used in research. They have the advantage of being non-invasive, easy to set up, and they have made it possible to develop many motion classification algorithms.

Here, we focused on the regression problem. The problem can be addressed as the following : given a signal $\boldsymbol{x}_t \in \mathbb{R}^d$, where d is the number of electromyographic sensors, we would like to predict, the more accurate as possible, the 22 [1] hand joint coordinates signal $\hat{\boldsymbol{y}}_t \in \mathbb{R}^{22}$. We have several databases from the Ninapro projects [2] providing us $\boldsymbol{x}_t$ and the ground truth $\boldsymbol{y}_t$ hand motion. The Ninapro databases gather several subjects that have been asked to reproduce a

[1] or 18, depending on the database used

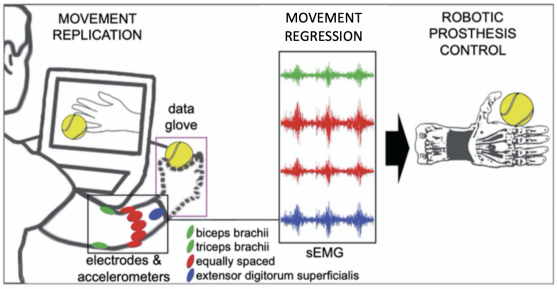

Figure 1. General framework of the project. This article focus on the movement regression task

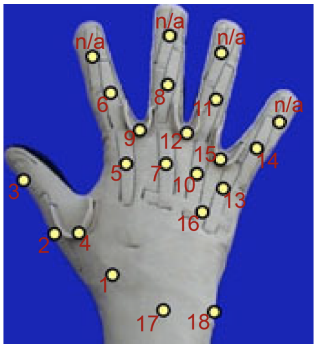

Figure 2. Position of the hand joint coordinates (for database VII). Credit [7]

motion projected on a screen. The electromyographic (EMG) sensors and a glove provide respectively the EMG and the 22 hand joint coordinates. The goal is to minimize the mean absolute error (MAE). The general framework is illustrated figure 1, and figure 2 shows the position of the hand joint coordinates.

2. Related work.

The question of predicting the user's intention was initially stated as a classification problem. The goal was to predict the intention of the user among predefined postures. Before the emergence of deep learning, a challenge was to extract relevant features for classification [8]. Phinyomark et al. distinguish four main types of method for features extraction : energy and complexity, frequency, prediction model, and time-dependence.

Then, deep learning based methods achieve similar results [4, 5] as the previous methods. Indeed, feature extraction is included in the learning in the first layers of the networks.

However, the classification problem has a main inconvenient: the output motion is limited among a set of postures, which is restrictive for amputees daily life. With the growth of databases size and improvement of deep learning techniques, regression problem seems now achievable [3, 6, 9]. Both Hoch et al. and Quivra et al. achieve promising results for this problem. [3, 6, 9] use recurrent neural networks to tackle the regression problem.

In this article, we focus on the regression problem with recurrent neural networks as well.

3. Methodology

This section present our methodology. In the first place, information on the database are provided. Secondly, the splitting strategy used. Thirdly preprocessing techniques such as sub-sampling, normalization and standardization are introduced. Then, we present our model which is based on LSTM. In the last part, we present the training process.

3.1. Database

The first step of our project was to find a good database. A quick literature review told us that the Ninapro databases are the references in the field. We had to choose among the different databases from the Ninapro project. A PhD student told us that the recent databases are more accurate than the first ones. Then we read in [6], that the best results were obtained using the database VII so that's the one we chose. Before the pre-processing, we have access to the Ninapro database VII defined in [7]. Table 1 summarizes information on these databases.

3.2. Database splitting

For our experiments, we have split our dataset in subsets in order to validate our method. The idea is to train the network on the "train subset", validate our different trainings on the "validation subset" and test our final model on the "test subset".

We have split the dataset along subjects. The goal is to train on some subjects and evaluate the model

	Ninapro DB VII
Number of subjects	20⋆
Number of EMG sensors	12
Number of joint coordinates	18
Sampling frequency	2kHz
Size (Gb)	33

Table 1. Information on the database, before pre-processing.
⋆ *We haven't not use the 2 amputees as the hand movements are not recorded.*

on other subjects. In the end we would like to have a single model able to handle any new subject.

This is particularly useful for amputees as we cannot train the model on them as we do not have the regression data.

In order to reduce bias we have split the 20 subjects of the database in three subsets keeping the left handed ratio as constant as possible (We used the meta information provided by Ninapro).

Split sequences A ENLEVER!! In this process we train on all the users of the database. But only on some parts of their exercise. And we validate and test on other parts. It should be an easier task but it's much less useful for our goal.

We made sure not to use overlapping sequences on training and testing, and in order to avoid to add bias to the subsets, we split all the exercise of each subject in the different subsets.

3.3. Pre-processing

Table 1 highlights the important size of the database. In order to allow fast computations, sub-sampling is needed to deal with the quantity of data. A trade-off has to be found between fast computation and loss of information. A sub-sampling at 400Hz was carried out, the results of which can be seen in figure 3.

The ranges of variation of sEMG measurements vary greatly from one sensor to another, which can be a problem during training with LSTM (long-short term memory) cells. To address this issue, normalization or standardization are used on the each input $\boldsymbol{x}_i$. The normalization operation can be stated as :

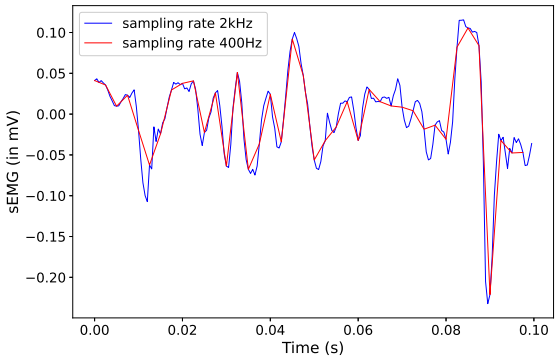

Figure 3. Influence of the sub-sampling at 400Hz on one sEMG signal

$$\forall t \; x_{t,i,norm} = \frac{x_{t,i} - x_{i,min}}{x_{i,max} - y_{i,min}}$$

where $x_{i,min}$ and $x_{i,max}$ are respectively the minimum and the maximum of each input on the whole database.

The standardization operation can be stated as :

$$\forall t \; x_{t,i,stand} = \frac{x_{t,i} - \bar{x}_i}{\sigma_i}$$

where $\bar{x}_i$ and σ_i are respectively the mean and the standard deviation of each input on the whole database. In our project, the standardization process gave better results so this is the one we kept for the following.

3.4. Model

We used two types of recurrent neural network (RNN) architectures. The first one is based on LSTM cell and the second one is based on GRU (gated recurrent unit) cell. We describe these cells in the following paragraphs.

LSTM cell

Figure 4 shows the general principle of a LSTM cell. With $\boldsymbol{x}$ one element of an input sequence, $\boldsymbol{h}$ / $\boldsymbol{c}$ the corresponding output / hidden sate. The update of the

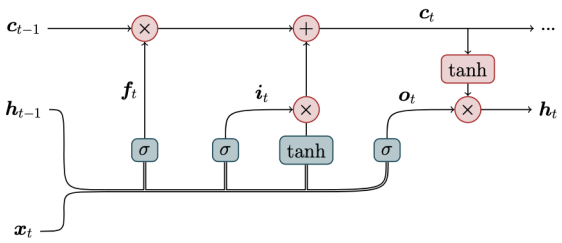

Figure 4. Illustration of the principle of a LSTM cell. Credit [6]

cell is given by :

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)$$

With $\odot$ being the Hadamard product, with the input gate :

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)$$

and the forget gate :

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$$

The output of the LSTM cell is described by :

$$\boldsymbol{h}_t = (\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \odot \tanh(\boldsymbol{c}_t)$$

All the $\boldsymbol{W}$ are trainable weights and $\boldsymbol{b}$ bias terms .

GRU cell

Figure 5 shows the general principle of a GRU cell. The output of the cell is given by :

$$\boldsymbol{h}_t = (1 - \boldsymbol{z}_t) \odot \boldsymbol{h}_{t-1} + \boldsymbol{z}_t \odot \tilde{\boldsymbol{h}}_t$$

With :

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t])$$
$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_z \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t])$$
$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}[\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1}), \boldsymbol{x}_t])$$

Where $\boldsymbol{W}$ are all training weights.

Architectures

We implemented several architectures based on LSTM and GRU. Table 2 details the best architectures we used for LSTM and GRU cells.

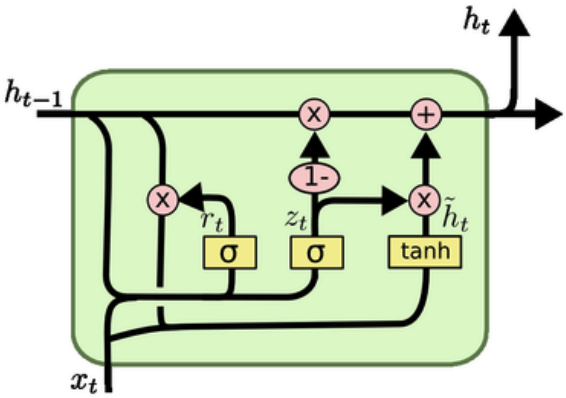

Figure 5. Illustration of the principle of a GRU cell. Credit [1]

	Architecture 1	Architecture 2
Cell type	"GRU"	"LSTM"
Input dimension	12	12
Hidden dimension	128	128
Number of layers	1	1
Fully-connected	128 x 18	128 x 18

Table 2. Descriptions of the architectures used.

3.5. Training process

At training time, we extract all the sub-sequences $\{(x_k, y_k)_{i \leq k < i+L}, i\}$ of fixed length L in our training subset and train over each of them in order to leverage all the information of the dataset. We used $L = 400$ in order have 1s length sequences as in [6].

We use RNN over these sequences in a sequence to sequence way, in order to predict all the targets of each sequence. At each epoch we validate the model as describe in 4.1.

Our main contribution compared to the articles we read is the work on the training loss function. First we started by using the Mean Square Error (MSE) function over the whole sequence:

$$\mathcal{L}(Y, \hat{Y}) = \frac{1}{n}\sum(Y - \hat{Y})^2$$

Where n is the number of joint angles multiply by the length of a sequence. Yet, one particularity of the dataset is that during the exercises the subjects are "resting" most of the times, *i.e.* there is no movement.

Therefore, during the process our network learnt to predict a constant around the mean for each joint coordinates and seems to struggle to go beyond (It requires more epochs to do so.)

In order to improve things we worked on the loss function. First rather than using a MSE loss over the whole sequence we tried to use a MSE loss only over the last target of the sequence. (Using MSE loss over the whole sequence is probably to hard, as the network has not enough information to predict the angles at the beginning of the sequence.).

We also designed a new loss in order to focus on what we want to learn: Movement. This loss will perform a weighted MSE loss over the sequence. First it will apply a decay factor, the idea is to allow the first predictions to be wrong, and to mainly focus on the end of the sequence. And second, it will also apply a multiplicative factor according to the current derivative of the angle movements at this time step.

Our loss function can be stated as the following :

$$\mathcal{L}(Y, \hat{Y}) = \sum_{t=1}^{L} \alpha_t \beta_t(Y)(Y_t - \hat{Y}_t)^2$$

Where α_t is an increasing function of t. We typically used $\alpha_t = \frac{t^p}{L}$ with p a parameter to tune.

And $\beta_t(Y)$ is an increasing function the absolute difference between Y_t and Y_{t-1} (The movement at time t). We used $\beta_t(Y) \propto \epsilon + |Y_t - Y_{t-1}|^q$ where q and ϵ are parameters to tune.

In order to keep meaningful losses we averaged all the losses by the batch size, sequence length and positions numbers, with these settings we use a batch size of 256, with Adam optimizer and a learning rate of 0.001.

4. Evaluation and results

4.1. Evaluation

We decided to evaluate the performance of our model like the papers [6, 3, 9] to enable comparison, although the database are not always the same. We used the mean absolute error (MAE):

$$MAE(Y, \hat{Y}) = \frac{1}{n} \sum |Y - \hat{Y}|$$

	Arch. 1	Arch. 2
Duration of 1 training epoch	1208s	1290s
Validation loss after 1 epoch	16,24	14,95

Table 3. Performances during training of architecture 1 and 2 (defined table 2)

After each epoch of training, we evaluate the performances over the validation set with the MAE loss. At validation time, we use each exercise as a single very long sequence: The MAE over a real long sequence seems a better metric than the MAE over the artificial sub-sequences.

Moreover as we noticed, the network is struggling to predict anything but an approximation of the mean. We have therefore decided to create a baseline predictor: The mean predictor. We have computed the mean of the joint angles over the training set. And predict for the validation/test sets these means. The computation of the associated MAE loss will be our baseline. And our goal is to improve this prediction, and as we will saw this is not easy at all.

4.2. Results

Influence of the architecture

First, we focused on the choice of the architecture. AS GRU cells are relatively news compared to LSTM, all the articles [6, 9, 3] we read were based on LSTM. Therefore we compared GRU and LSTM based architectures. GRU are reputed to increase the speed of the training. We confirmed this reputation but yet the training was less performing when we look closer at the validation loss as we can see table 3

Influence of the loss function

We did a lot of experiments to find out how to learn efficiently. We never really managed to do better on the validation set than the mean baseline. And it seems that the same problem occurs in the literature: Those who have tried to validate the method on a new subject have similar results [3].

Nonetheless the training converge much quicker with our newly designed loss, or with the MSE loss on the last target. But as figure 6 show, the MSE loss on the last target do not generalized well at all.

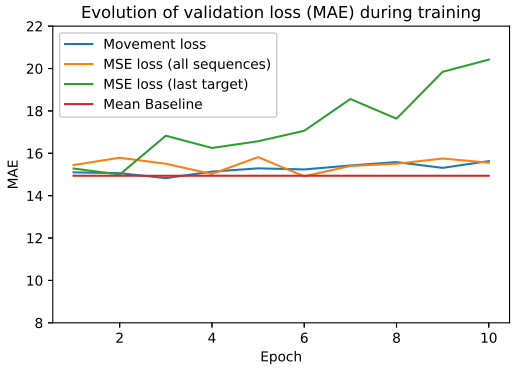

Figure 6. Comparison of the different losses and the mean baseline

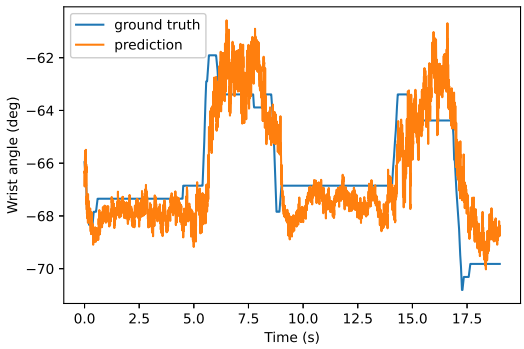

Figure 7. Prediction of the wrist angle on the validation set

This results might seem disappointing. Yet, the MAE is a mean and if we look closer we have some really good predictions for some articulations. For instance, figure 7 shows the prediction of the wrist angle on the validation set. The prediction of certain articulations lower considerably our results, such as the prediction of the articulation shown in 8

Due to lack of time we did not manage to train and test a network on the same subjects but on different moments, but we expect this to improves the results and match those of [6]. We try to finetune a model on a single subject by splitting its exercises. It improves the results up a MAE of 12.3. It is beyond the scope of our work, but it explains some of the results of the literature.

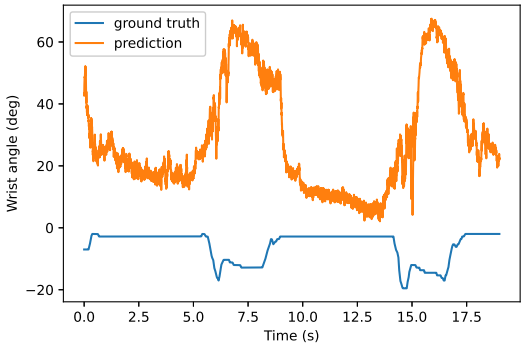

Figure 8. Prediction of a thumb angle on the validation set

5. Conclusion

In this project we have implemented the regression approach from predicting hand gestures from sEMG signals. A literature review encourage us to implement recurrent neural networks to solve this task, using the Ninapro database. We develop a methodology with the following steps :

- Database splitting
- Sub-sampling
- Standardization
- Training neural networks :

  - with different architecture (LSTM, GRU)

  - with different loss we specifically design for this application

- Visualization and validation of the results

For each step listed, we implemented several experimentations (some have not been discussed here, only the more relevant have been kept). The conclusion of our work is that indeed this issue can be addressed with recurrent neural networks but methodology need to be improved to match state-of-the art approaches. For instance we strongly advise to compare results to the mean baseline and to try to understand what we are learning. As we show we struggled to reach this baseline on new subjects. It seems that it is hard to

generalize the predictions. Thus fine-tuning on specific subject seems interesting but would be hard to do for amputee subjects.

References

[1] Abien Fred Agarap. A neural network architecture combining gated recurrent unit (gru) and support vector machine (svm) for intrusion detection in network traffic data, 10 2017. 4

[2] M. Atzori, A. Gijsberts, S. Heynen, A. M. Hager, O. Deriaz, P. van der Smagt, C. Castellini, B. Caputo, and H. Muller. Building the ninapro database: A resource for the biorobotics community. In *2012 4th IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1258–1265, 2012. 1

[3] S. Kirchhofer T. Chateau C. Bouzgarrou. Estimation des coordonnees articulaires de la main a partir de signaux electromyographiques. 10, 2019. 2, 5

[4] Ulysse Cote-Allard, Cheikh Latyr Fall, Alexandre Drouin, Alexandre Campeau-Lecours, Clement Gosselin, Kyrre Glette, Francois Laviolette, and Benoit Gosselin. Deep learning for electromyographic hand gesture signal classification using transfer learning, 2019. 2

[5] Zhen Ding, Chifu Yang, Zhihong Tian, Chunzhi Yi, Yunsheng Fu, and Feng Jiang. semg-based gesture recognition with convolution neural networks. *Sustainability*, 10(6):1865, Jun 2018. 2

[6] P. Koch, M. Dreier, A. Larsen, T. J. Parbs, M. Maass, H. Phan, and A. Mertins. Regression of hand movements from semg data with recurrent neural networks. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, pages 3783–3787, 2020. 1, 2, 4, 5, 6

[7] Agamemnon Krasoulis, Iris Kyranou, Mustapha Suphi Erden, Kianoush Nazarpour, and Sethu Vijayakumar. Improved prosthetic hand control with concurrent use of myoelectric and inertial measurements. *Journal of NeuroEngineering and Rehabilitation*, 14(1), July 2017. 2

[8] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. Feature reduction and selection for emg signal classification. *Expert Systems with Applications*, 39(8):7420 – 7431, 2012. 2

[9] F. Quivira, T. Koike-Akino, Y. Wang, and D. Erdogmus. Translating semg signals to continuous hand poses using recurrent neural networks. In *2018 IEEE EMBS International Conference on Biomedical Health Informatics (BHI)*, pages 166–169, 2018. 2, 5

[10] N. Tsagkas, P. Tsinganos, and A. Skodras. On the use of deeper cnns in hand gesture recognition based on semg signals. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–4, 2019. 1