Prediction of reaching movements with target information towards trans-humeral prosthesis control using Reservoir Computing and LSTMs

Paul Bernard, Frédéric Alexandre, Xavier Hinaut

Prediction of reaching movements with target information towards trans-humeral prosthesis control using Reservoir Computing and LSTMs

Paul Bernard[1,2,3][0009−0009−3384−9821], Frédéric Alexandre[1,2,3][0000−0002−6113−1878], and Xavier Hinaut[1,2,3][0000−0002−1924−1184]

[1] Inria Center of Bordeaux University, France
[2] IMN, Université de Bordeaux, CNRS, UMR 5293, France
[3] LaBRI, Bordeaux INP, CNRS, UMR 5800, France

Abstract. Controlling a prosthetic upper limb requires the reconstruction of multiple distal articulations. Moreover, the higher the amputation level, the more joints need to be reconstructed, and the less kinetic information is available in the residual limb. By exploiting contextual information, such as the position and orientation of a target in a reaching task, we aim to reconstruct the natural dynamics of the distal joints using recurrent neural networks. We compare performances of two models, an Echo State Network (ESN) and an LSTM, on two conditions: training on individual subjects, and training on a 5-fold CV on 15 subjects. We explored hyperparameters on both models: the ESN shows better performances on the single-subject task, and the LSTM shows better performances on the multiple-subject task. When looking qualitatively at the predictions, we observe that even if networks don't have the same MSE errors, they perform the task well and are able to reach the targets most of the time. We further analyze the performance of the models on the multi-subject task and report different kinds of generalizations.

Keywords: Arm prosthesis · Reservoir computing · recurrent neural networks · virtual reality · movement prediction

1 Introduction

1.1 Current challenges in upper limb prostheses

Active upper limb prostheses aim to restore all or parts of the functions of the amputated limb. The activation of the active joints of the prosthesis is controlled by different bio-signals depending on the prosthesis. In the case of upper limb amputation, electromyographic (EMG)-controlled prostheses are among the most popular solutions. EMG prostheses interpret the electrical signals produced by the muscle contractions of the residual limb of the amputee. The signal is received by electrodes placed on the skin [2]. The decoded signal, processed through a myoelectric pattern recognition algorithm, controls the prosthesis motors. However, current prostheses face a high rejection rate [13]. Functional limitations are

among the most frequently cited reasons for abandonment. EMG-based prostheses in particular suffer from a dimensionality problem: as the amputation level gets closer to the shoulder, the number of controllable joints increases, but the number of available residual muscles to control these joints decreases, doubling the challenge of achieving natural movements. In the case of trans-humeral (between the shoulder and the elbow) amputation, five joints have to be reconstructed, while only two remain (see Fig. 1).

1.2 Objectives

Redundancies. Even though we count 7 degrees of freedom (DoF) in the upper limb, the position and orientation of the hand in space only need 6 DoF (3 for the position and 3 for the orientation). For a single placement of the hand, an infinity of arm configurations are possible, as the elbow can still rotate around the shoulder-hand axis. This is only one of the many redundancies of the upper limb [8,16]. Those redundancies can be a way to deal with constrained environments or injuries. Despite the overabundance of configurations, the upper limb presents some regularities in its movement at multiple levels, which are not physical constraints but rather preferences of the central nervous system [14].

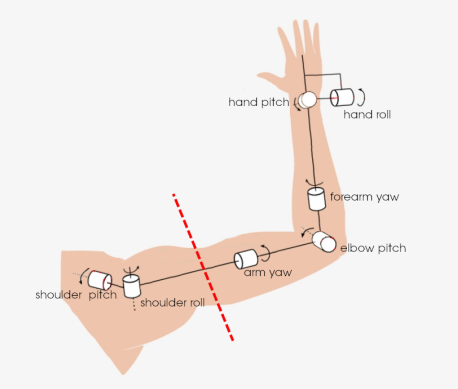


Fig. 1. Illustration of the seven degrees of freedom of the upper limb. **Shoulder Pitch**: shoulder flexion/extension, **Shoulder Roll**: shoulder adduction/abduction, **Arm Yaw**: humeral rotation, **Elbow Pitch**: elbow flexion/extension, **Forearm Yaw**: forearm pronation/supination, **Hand Roll**: wrist flexion/extension, **Hand Pitch**: wrist ulnar/radial deviation. The dashed red line represents the trans-humeral amputation level.

Principle. In this paper, we focus on a reaching task, with a movement-based control, in which the movements of the residual limb are measured and used to control the movements of the distal joints. In the case of trans-humeral robotic

prostheses, the goal is to control 5 degrees of freedom (arm yaw, elbow pitch, forearm yaw, hand roll and hand pitch) from the 2 degrees of freedom of the residual limb (shoulder pitch and shoulder roll). To achieve this goal, contextual information is added: the position and orientation to be reached. This adds new dimensions to the input of the problem. Instead of predicting the angles of the distal joints, we focus on the generation of the hand position and orientation. It is then possible to infer the distal joint angles by applying a simple inverse kinematic model.

Using a recurrent neural network. The reservoir computing paradigm is a well-suited choice for this task, due to its inherent ability to process time series. Furthermore, one big advantage of reservoir computing is its ability to be integrated in an embedded system. It requires very low energy, as the reservoir part can take multiple physical forms [17]. Because of their energy efficiency, such networks have previously been used in bionic arms, successfully improving the decoding of myoelectric signals [12,11]. The objective of this paper is to evaluate the performances of both Reservoir Computing (the particular instance of Echo State Networks - ESNs [6]) and Long Short-Term Memory networks - LSTMs [5] in mimicking the natural upper limb movement in a reaching task in which the position and the orientation of the target are known.

2 Methods

The code and supplementary material is publicly available on GitHub (https://github.com/PAUL-BERNARD/Bernard2024_ICANN_prosthesis_reservoir).

2.1 The experiment and the data

The subjects. The experiment was conducted on 17 right-handed and able-bodied subjects, with normal or corrected vision, aged from 21 to 33.

Experimental setup. The experiment takes place in a virtual environment. Participants are seated on a chair, wearing a virtual reality headset (HTC Vive™ Pro). Four body trackers (HTC Vive™ Tracker) are placed on the trunk and on the upper arm, forearm and hand of the right upper limb, as shown in Fig. 2.

Virtual environment. The virtual environment consists of an empty and white room. The subject's right arm is reproduced, thanks to the trackers described above. Inside the simulated hand, a transparent bottle is represented as if the hand was holding it. A bottle-like object is placed randomly in front of the subject. A bottle can fully be described with 5 parameters: its position in space on the X, Y and Z axes, and two orientations: pitch and roll. As the bottle is invariant by rotation along its main axis, the yaw parameter is not used.

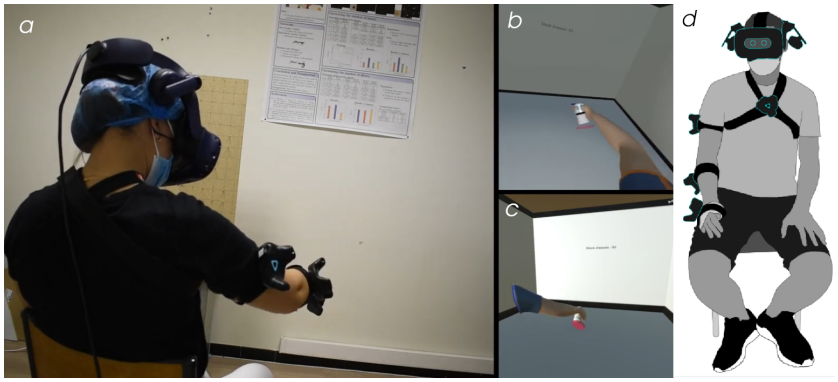


Fig. 2. Simulation setup. **(a)** Participant during the experiment. **(b)** Virtual environment from the participant point of view. **(c)** Virtual environment from another point of view. **(d)** Illustration of the participant equipment. A sensor is placed on the center of the trunk, one is placed at the top of the arm, one is placed right before the elbow and the last one is placed on the wrist. Taken from [14]

The task. The participants are tasked with moving their hand towards the bottle: the palm of the simulated hand must contain the transparent bottle in the simulation. We consider that the bottle has been reached when the distance between the hand and the target is below 4cm and the rotation difference between the hand and the target is below 10°. When the hand is close enough to the target, the bottle turns red, and the participant can validate the task by pressing a space bar using his left hand. Once the target has been validated, a new bottle instantly appears in another place of the target space. If the target has not been validated within 5 seconds, a new target appears. Participants are tasked with validating 300 targets in a single session. They are allowed to take a break.

The data. All measures are sampled at 90Hz. The constructed dataset consists of 17 timeseries, each containing 253 ± 43 targets, for a total of 4300 targets. Each timestep contains the measures from the four trackers (the orientation in space as quaternions, and the position as X, Y and Z values), as well as a subject identifier, a target identifier, and the position and orientation of the target. The dataset can be found at [15].

2.2 Preprocessing

When the subject decides to take a break, or if the subject doesn't reach the target before 5 seconds, the target sequence is removed from the dataset. In total, 304 out of 4609 targets were removed from the dataset. Two subjects out of the seventeen had orientation parameters spanning over a full circle. As those angles are stored as a numerical value in $[-\pi; \pi]$, this leads to an underflow/overflow

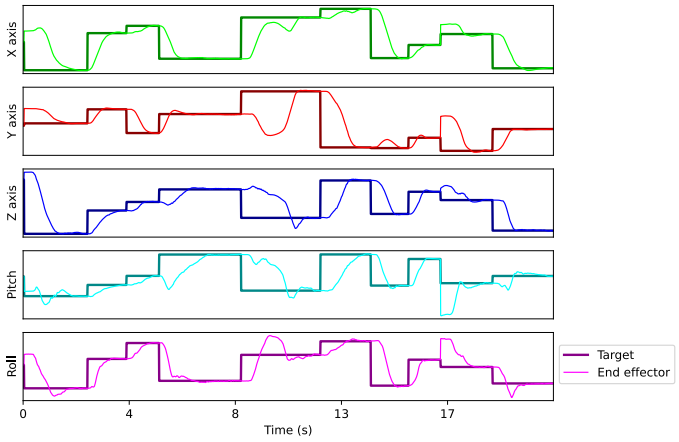


Fig. 3. Sample view from the dataset. The thicker lines represent the position of the bottle to be reached over time on all three position dimensions and the two orientation dimensions. The thinner lines represent the moving hand of the subject. The hand has to reach the *Target*, i.e. a virtual bottle, with the correct orientation. When the task is validated, a new bottle appears at a random location: this can be seen by discontinuous changes of *Target* position.

that cannot be easily wrapped back to $[-\pi; \pi]$ by offsetting the values. Those subjects are removed from the dataset.

The complete upper limb configuration (the 7 angles described in Fig. 1) is computed from the position and orientation of the trackers. As participants have different upper limb measurements, the position space, the orientation space and target state space are different for every subject. In order to have a consistent dataset with the same position and orientation space for every subject, we reconstruct the position and orientation of the hand from the configuration of the limb's joints, with an arbitrary forearm length of 28cm and an arm length of 25cm. The new state of the end effector (the subject's hand) is computed using the Python library `ikpy` [9]. The new target position and orientation in space are then considered to be the position and orientation of the subject's hand at the last timestep before a target change. The success condition ensures that the real target position is very close to the hand at the end of the target sequence. Finally, all parameters used in the model are normalized between -1 and +1 (min-max scaling).

2.3 The models

From the shoulder angles and knowing the position and orientation of the target to be reached, the goal is to generate the position and orientation of the hand

at every timesteps (its trajectory). Figure 4 represents a general view of the two models.

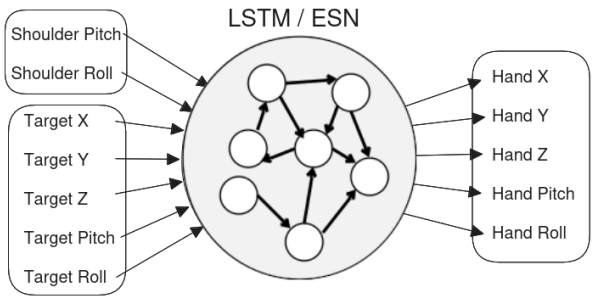


Fig. 4. Schematic view of the models. The recurrent neural network (the reservoir or the LSTM) takes the shoulder angles as an input, along with the target position and orientation. It is connected to a fully-connected layer, which outputs the position and orientation of the hand.

Echo State Network (ESN). We used the *ReservoirPy* [18] Python library for the implementation of the Echo State Network model. The ESN contains sparsely-connected leaky integrator neurons, with a sigmoid activation function:

$$\mathbf{x}[t+1] = (1-\alpha)\mathbf{x}[t] + \alpha f(\mathbf{W}_{in}\mathbf{u}[t+1] + \mathbf{W}\mathbf{x}[t])$$

The read-out layer is a feed-forward, densely-connected layer trained using a simple linear regression, with Tikhonov regularization:

$$\mathbf{y}[t] = \hat{\mathbf{W}}_{out}\mathbf{y}[t]$$

where:

$$\hat{\mathbf{W}}_{out} = \mathbf{Y}\mathbf{X}^{\top}\ (\mathbf{X}\mathbf{X}^{\top} + \lambda\mathbf{I})^{-1}$$

In this paper, we consider echo state networks with a connectivity degree of 3 on average[4]. Reservoir connections with non-zero weights in W follow a normal distribution. The role of the feedback has been studied (the study can be found on the GitHub repository), but preliminary results indicate that the best scores are obtained when the feedback scaling is negligible. As such, no feedback from the read-out to the reservoir is used in this study. The trainable weights of an ESN are the weights of the read-out matrix, with a bias. Following the recommendations in [4], we explore the impact of the spectral radius, the leak rate, the input scaling and the ridge regularization coefficient on the performances.

[4] This was used to speed up the computations compared to a 10 or 20% connectivity that is usually used. In preliminary testings, we didn't see significant changes in performance of the reservoir connectivity for this task.

LSTM. The LSTM model consists of a recurrent layer with LSTM cells[5], followed by a densely-connected layer, using Keras v2.15.0 [1]. The models are trained using the Adam optimization method [7].

3 Results

In each experiment, multiple instances of the same model with different initializations were evaluated. For the ESN, 30 instances are randomly initialized. For the LSTM network, 5 different seeds are used. This difference comes from the computational cost of training an LSTM network, which is much higher than for an ESN. We can also mention that, as an LSTM is fully trained, it is expected to be less dependent on its initialization. We evaluate our models using mean squared error (MSE).

3.1 Single-subject task

In this section, we focus on single subjects. The goal is to evaluate the performances of an ESN and an LSTM on movement production on smaller datasets before evaluating their ability to generalize over different subjects. For all the subjects, the model is trained on the first 80% of the timeseries, and is tested on the last 20%. The MSE loss obtained for each model and each subject can be found in Fig. 5. We observe that the LSTM network frequently falls behind the ESN one. The ESN forecasts show an average MSE of 0.0271 with a standard deviation of 0.00658. On the other hand, the LSTM predictions have an average MSE of 0.0308 and a standard deviation of MSE equal to 0.00733. The average score and the standard deviation for all subjects and for both models can be found in Table 1 A representation of the real and predicted timeseries on all five output dimensions can be found in Fig. 6.

3.2 Multi-subject task

In a prosthetic application, training and testing on the same person isn't feasible, as the distal joints and the hand position and orientation are missing. A model used as a prosthesis control would be trained on able-bodied people, and used once trained on trans-humeral prostheses. This approach may suffer from inter-individual variability. In this section, we compare the generalization performance of both networks using all 15 available subjects.

All performances displayed here are evaluated using a 5-fold cross-validation on the subjects (training with 12 subjects, testing on the 3 remaining subjects). As shown in the second line of Table 2, LSTM models seem to perform better in generalizing the task to multiple subjects, as their performances drastically improve from the previous task, from an average MSE of 0.0303 to an MSE of

[5] Each LSTM recurrent cell has 4 parameters: cell state, input gate, output gate and forget gate.

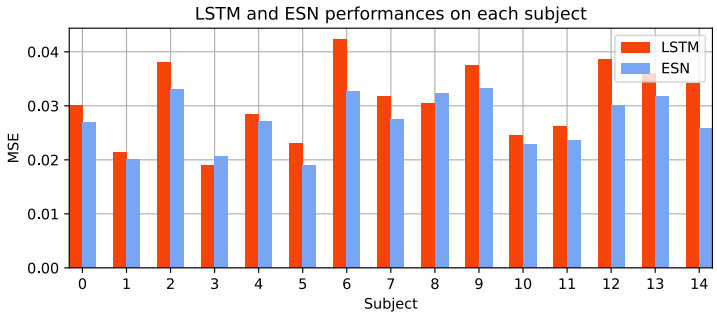


Fig. 5. ESN and LSTM errors on single-subject task. Each subject is trained on the first 80% of the timeseries, and is tested on the last 20% independently. ESNs perform better than LSTMs on all subjects. ESNs averaged on 30 instances; LSTM averaged on 5 instances.

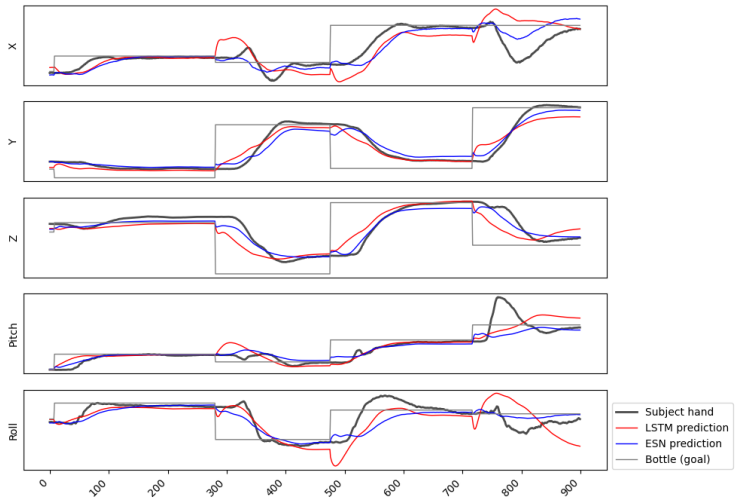


Fig. 6. Single-subject task predictions. Comparison of the predictions between the best LSTM and ESN networks on the first subject. The participant moves his hand (in black) towards the bottle to be reached (in gray). When the bottle has been reached by the participant, a new target appears at a random location, thus the sudden change in position and orientation. Both the ESN (in blue) and the LSTM (in red) predictions are shown. The closer the prediction is to the subject hand, the better.

Table 1. Average MSE and standard deviation for each model and each subject on the single-subject task.

Subject	ESN	LSTM
1	0.027 (0.0054)	0.030 (0.0016)
2	0.020 (0.0026)	0.021 (0.0011)
3	0.033 (0.0042)	0.038 (0.0023)
4	0.021 (0.0053)	0.019 (0.0003)
5	0.027 (0.0051)	0.028 (0.0020)
6	0.019 (0.0033)	0.023 (0.0023)
7	0.033 (0.0054)	0.042 (0.0061)
8	0.027 (0.0045)	0.032 (0.0052)
9	0.032 (0.0046)	0.030 (0.0015)
10	0.033 (0.0058)	0.038 (0.0026)
11	0.023 (0.0042)	0.025 (0.0012)
12	0.024 (0.0040)	0.026 (0.0020)
13	0.030 (0.0023)	0.039 (0.0009)
14	0.032 (0.0044)	0.036 (0.0043)
15	0.026 (0.0037)	0.034 (0.0019)

0.0111. In contrast, ESN instances do not benefit as much from the enlargement of the dataset, improving from an average MSE of 0.0233 to 0.0169. Again, we observe the higher stability of a fully trained neural network over a randomly generated one. From Figure 7, we observe that the perturbations when a new bottle spawns are greatly reduced.

Table 2. Average MSE and standard deviation of the models on multi-subject task.

	ESN	LSTM
Mean (std.)	0.0197 (0.0077)	**0.0111** (0.0013)
Best	0.0146	**0.0106**

Impact of the model size on performances. To be embedded in a prosthetic system, models must be fast [3] and energy efficient. In this section, we evaluate both the network speed (on training and predicting) and MSE score when varying the size of the recurrent network. For the ESN model, we explore the number of neurons between 50 and 5 000, on a logarithmic scale. The other hyperparameters (leak rate, spectral radius and input scaling) are set to the best values obtained in a hyperparameter exploration with reservoirs of 500 neurons. For the LSTM model, we explore the number of neurons between 10 and 300. The results of these explorations can be found in Fig. 8.

As expected, the computational time of the Echo State Network model increases significantly with its size. In contrast, the loss obtained for the best models doesn't improve as much. More surprisingly, the computational time of an LSTM model is important both for small and large networks. This is ex-

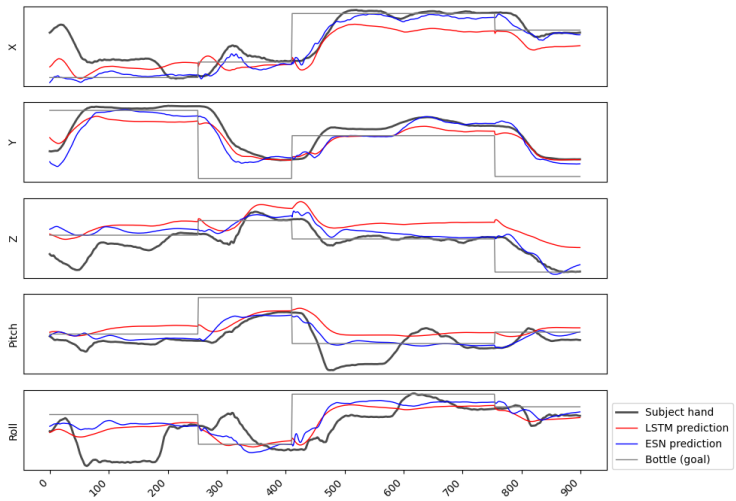


Fig. 7. Multi-subject task: Comparison of the predictions between the best LSTM and ESN networks. The participant moves his hand (in black) towards the bottle to be reached (in gray). When the bottle has been reached by the participant, a new target appears at a random location, thus the sudden change in position and orientation. Both the ESN (in blue) and the LSTM (in red) predictions are shown. The closer the prediction is to the subject hand, the better.

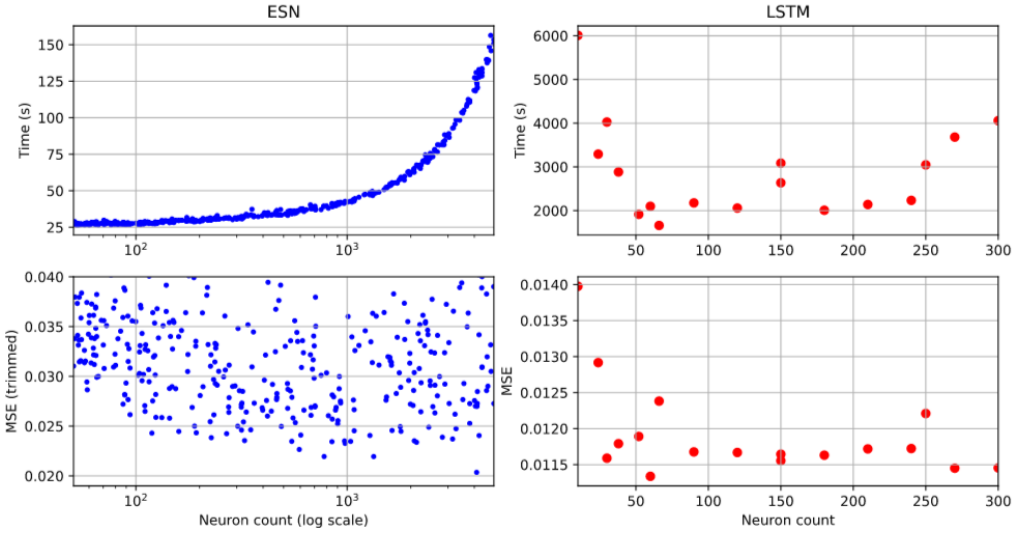


Fig. 8. Impact of network sizes on MSE error and computation time (train and test time).

plained by the duality between convergence speed and the execution time of a single epoch: smaller models take more time to converge, and larger models converge faster but take more time to run on the dataset. The size of an LSTM model appears to have a small impact on the fitness of the prediction. Regardless of the size of the networks, the LSTM model outperforms the ESN model on the multi-subject task.

Reaching the bottle. In addition to closely following the hand movement, the goal of the models is to reach the target at the end of each sequence. To evaluate the performances on this task, we compute the Euclidean distance (with the orientation) between the bottle and the predicted position of the best models, right before target change. A histogram of distances at the end of all the sequences in the testing dataset can be found at Fig. 9. We observe that while both the ESN and the LSTM models tend to end farther from the bottle than the real hand, they end up close enough to the bottle.

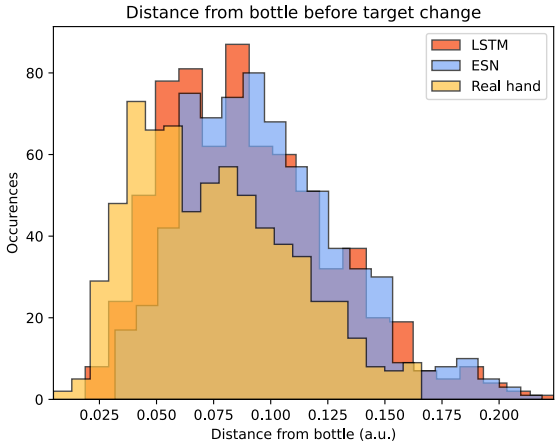


Fig. 9. Distance between the target bottle and the hand position in the dataset (in yellow), the prediction from the best LSTM model (in red), and the prediction from the best ESN model (in blue).

4 Discussion

One of the main reasons for abandoning an arm prosthesis is the lack of natural control and the cognitive cost of using it. Current prosthetic solutions rely on electromyographic signals, which are difficult to measure and process. Adding a contextual information – the target position and orientation –, and using the

remaining limb kinematics have been shown to be an effective way to overcome the limitations of EMG prostheses in previous studies [10].

In this paper, we propose an ESN-based control of the missing distal joints in such a context. The ability of an ESN-based solution to be embedded in a system and its fast generalization ability make it a powerful candidate. We compared its performance with a LSTM network, with hyperparameter optimization on both models. We first evaluated the performances of both networks on the task on each subject, and then evaluated the generalization capabilities of the networks on all 15 subjects.

Considering quantitative results, we showed that, on the single-subject task, the ESN model shows systematically better performances than the LSTM model – both on average and for the best instance – which indicates that the ESN is able to perform well when presented with little data. In contrast, the LSTM model's performance increases significantly on the multi-subject task. An exploration of the size of recurrent neural networks indicates a low impact on their effectiveness, but it substantially affects the time required for training and executing.

Considering the qualitative results, it is notable that both models correctly reach the bottle at the end of the reaching task. On Figure 6 the predictions obtained from ESNs and LSTMs on a single subject show some particular features of this task. First, we see that the subject has a non-negligible reaction delay compared to the ESN/LSTM – for which the activity changes quickly after the input *Target* has changed –. Second, we see that the LSTM tends to "over-react": the change of target could produce a big change in the activity of the network. It is not clear why the LSTM behaves like this. Is it because it has "over-learned" to detect the beginning of a sequence thanks to its gating units? The problem here is that it prevents the LSTM from having an adequate behavior for a small sample size (one subject).

When now looking at the predictions for multi-subject tasks in Figure 7, we see that the LSTM has learned to "refrain" this over-reactive behavior, even if it is still present. Looking at the curves, it seems that the ESN predictions are often in-between the curves of the human hand and of the LSTM, even if sometimes "over-correcting" when the subject starts to go away from the target and then goes back towards the target: indeed, the ESN can "cross" the line of the target because it goes "ahead" of the subject in terms of distance to the target. The LSTM seems to find a "trick" to minimize the MSE better than the ESN by averaging its predictions, which may not be so close from the target.

Further investigations could enable us to understand several interesting points:

(1) What is the nature of the discrepancy between ESNs and the LSTMs for single-subject results (i.e. why are ESNs better and LSTMs have "over-reactive" behavior when the target changes)?

(2) Why good ESN performances on single-subject data extend in a limited fashion to multi-subject data (and conversely, why do LSTMs have a higher gain in performances when more data are available)? Qualitatively, on the multi-subjects task, the ESN seems to do well, even "better" than the subject sometimes as it approaches faster to the target (i.e. reaching the virtual bottle), as one

can see in Figure 7. Conversely, the LSTM may have a smaller MSE, but from a look at its behavior, it seems that it is just providing an average of the expected target, and often does not even reach the target. In other words, the apparent "bit worse" result of the ESN for multi-subjects task may be due to the "dual nature" of the problem to be solved. Indeed, on the one hand, the task of the human subject is to reach the virtual target (i.e., the bottle); the human may use indirect movement to reach the target and not the shortest movement that one would obtain with a machine learning method. On the other hand, the ESN and the LSTM have to imitate the hand movement of the human at each time step t given (a) the shoulder position (still of the human) and (b) the global target to be reached (i.e., the virtual bottle). This is a difficult task, as each subject may not behave in a fully predictive fashion and provide variable movements for similar targets. In the results we obtain, we see that this dual task is difficult for the networks, and could lead to different kinds of generalization, or more precisely, to a gradient of generalization between the first and the second tasks described above. Here, the ESN seems to generalize more towards "reaching the virtual target" task than the LSTM: this is not a bad thing and is probably a desirable behavior, but in terms of minimizing the MSE, the LSTM does a better job at minimizing the average human hand behavior. Supplementary material is available on the GitHub to investigate further this discussion. Implementing a different objective cost function would probably lead to different performances for the ESN and LSTM, and may help to reach a more adequate consensus on both networks.

Finally, further experiments with humans in the loop using the predictions of the neural networks in real time (in the virtual reality scenario) would give new insights on how to go further toward using such models in actual prostheses. Subjects could qualitatively evaluate the behavior of the two models, thus giving another point of comparison to evaluate the two models. In this perspective, preliminary experiments have been performed with users in the loop using ESNs: they provide promising results that validate the approach.

5 Acknowledgment

This work was funded by the ANR-DGA-ASTRID grant CoBioPro (ANR-20-ASTR-0012-1). We thank Aymar De Rugy, Effie Segas, Vincent Leconte and Bianca Lento from the INCIA HYBRID team for sharing data and collaborating on this project.

References

1. Chollet, F., et al.: Keras. https://keras.io (2015)
2. Cimolato, A., Driessen, J.J., Mattos, L.S., De Momi, E., Laffranchi, M., De Michieli, L.: Emg-driven control in lower limb prostheses: A topic-based systematic review. Journal of NeuroEngineering and Rehabilitation **19**(1) (May 2022). https://doi.org/10.1186/s12984-022-01019-1

3. Farina, D., Jiang, N., Rehbaum, H., Holobar, A., Graimann, B., Dietl, H., Aszmann, O.C.: The extraction of neural information from the surface emg for the control of upper-limb prostheses: emerging avenues and challenges. IEEE Transactions on Neural Systems and Rehabilitation Engineering **22**(4), 797–809 (2014)
4. Hinaut, X., Trouvain, N.: Which Hype for my New Task? Hints and Random Search for Reservoir Computing Hyperparameters. In: ICANN 2021 - 30th International Conference on Artificial Neural Networks. Bratislava, Slovakia (Sep 2021), https://inria.hal.science/hal-03203318
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
6. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148**(34), 13 (2001)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2017), https://arxiv.org/abs/1412.6980
8. Latash, M.L., Scholz, J.P., Schöner, G.: Toward a new theory of motor synergies. Motor control **11**(3), 276–308 (2007)
9. Manceron, P.: Ikpy: An inverse kinematics library aiming performance and modularity (v3. 3.3). Phylliade/ikpy (2022)
10. Mick, S., Segas, E., Dure, L., Halgand, C., Benois-Pineau, J., Loeb, G.E., Cattaert, D., de Rugy, A.: Shoulder kinematics plus contextual target information enable control of multiple distal joints of a simulated prosthetic arm and hand. Journal of NeuroEngineering and Rehabilitation **18**, 1–17 (2021)
11. Prahm, C., Schulz, A., Paaßen, B., Aszmann, O., Hammer, B., Dorffner, G.: Echo state networks as novel approach for low-cost myoelectric control. In: Conference on Artificial Intelligence in Medicine in Europe. pp. 338–342. Springer (2017)
12. Saleh, Q.M.: Design of a Neuromemristive Echo State Network Architecture. Rochester Institute of Technology (2015)
13. Salminger, S., Stino, H., Pichler, L.H., Gstoettner, C., Sturma, A., Mayer, J.A., Szivak, M., Aszmann, O.C.: Current rates of prosthetic usage in upper-limb amputees – have innovations had an impact on device acceptance? Disability and Rehabilitation **44**(14), 3708–3713 (2022). https://doi.org/10.1080/09638288.2020.1866684, pMID: 33377803
14. Ségas, E.: Contrôle biomimétique de prothèses à partir des mouvements résiduels et d'informations contextuelles. Theses, Université de Bordeaux (Mar 2023), https://theses.hal.science/tel-04164931
15. Segas, E., Mick, S., Leconte, V., Klotz, R., Cattaert, D., de Rugy, A.: Data and code for intuitive movement-based prosthesis control in virtual reality (Oct 2022). https://doi.org/10.5281/zenodo.7187851, https://doi.org/10.5281/zenodo.7187851
16. Soechting, J.F., Lacquaniti, F.: Invariant characteristics of a pointing movement in man. Journal of Neuroscience **1**(7), 710–720 (1981)
17. Tanaka, G., Yamane, T., Héroux, J.B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., Hirose, A.: Recent advances in physical reservoir computing: A review. Neural Networks **115**, 100–123 (2019)
18. Trouvain, N., Hinaut, X.: reservoirpy: A Simple and Flexible Reservoir Computing Tool in Python (Jun 2022), https://inria.hal.science/hal-03699931

Bernard2024_ICANN_prosthesis_reservoir / supplementary /

Add file

PAUL-BERNARD Supplementary materials d8ee811 · 2 months ago

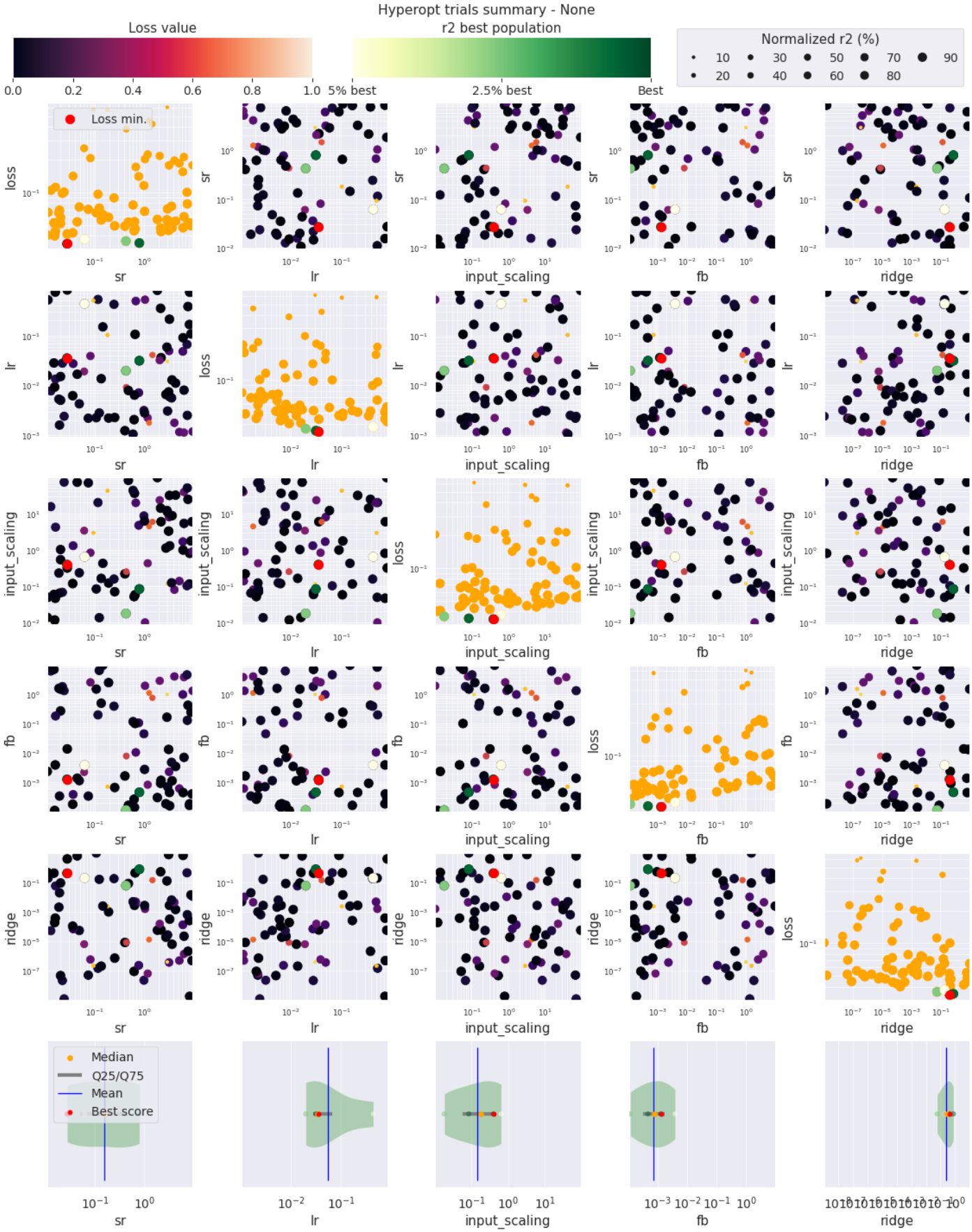
Name	Name	Last commit date
..		
README.md	Supplementary materials	2 months ago
feedback_importance.png	Supplementary materials	2 months ago
target_hand_distances.p...	Supplementary materials	2 months ago

README.md

Supplementary materials

Impact of feedback on the ESN model

This has been evaluated by exploring the feedback scaling hyper-parameter, along with other classical hyper-parameters: the spectral radius (`sr`), leak rate (`lr`), an input scaling, and the regularization parameter of the readout part (`ridge`).



This is the report generated by ReservoirPy on the hyper-parameter exploration. Every tested reservoir had 500 neurons. The spectral radius is explored between 0.01 and 10 (log-uniformly). The leak rate is explored between 0.001 and 1 (log-uniformly). The input scaling is explored between 10^-2 and 10^2 (log-uniformly). The regularization is explored between 10^-9 and 10 (log-uniformly). The feedback scaling is explored between 10^-4 and 10 (log-uniformly).

The diagonal plots displays the impact of each parameter individually on the used score. In the feedback scaling column, we observe that the best scores are obtained when the feedback is negligible.

Distance to hand vs distance to target

For each time step, we record the distance between the model prediction and effective hand position, and the distance between the model prediction and the bottle position.