A Spiking Neuronal Model Learning a Motor Control Task by Reinforcement Learning and Structural Synaptic Plasticity

Martin Spüler, Sebastian Nagel and Wolfgang Rosenstiel
Department of Computer Engineering
University of Tübingen
Tübingen, Germany
Email: spueler@informatik.uni-tuebingen.de

***Abstract*—In this paper, we present a spiking neuronal model that learns to perform a motor control task. Since the long-term goal of this project is the application of such a neuronal model to study the mutual adaptation between a Brain-Computer Interface (BCI) and its user, neurobiological plausibility of the model is a key aspect. Therefore, the model was trained using reinforcement learning similar to that of the dopamine system, in which a global reward and punishment signal controlled spike-timing dependent plasticity (STDP). Based on this method, the majority of the randomly generated models were able to learn the motor control task. Although the models were only trained on two targets, they were able to reach arbitrary targets after learning. By introducing structural synaptic plasticity (SSP), which dynamically restructures the connections between neurons, the number of models that successfully learned the task could be significantly improved.**

I. Introduction

In the past, spiking neural networks (SNNs) [1] have been used to model different aspects of motor control. Especially, when it comes to robot control, there are many studies in which SNNs are used for controlling robots' motor behavior [2], [3], [4]. While modeling of the spiking neurons is inspired by its biological original, the methods used for learning do not aim to be biologically plausible in most studies.

However, some works used spike timing dependent plasticity (STDP) [5] as a biologically motivated learning method to drive a robotic arm [6], [7], [8]. In this paper, we extend the work by Chadderdon et al. [8] who have presented a biologically realistic model of motor cortex that learns to reach a target angle by reinforcement learning. In a later work, this model was extended for control of a two-joint virtual model [9]. While the model is able to learn to reach the target position, it is not able to learn a stable neural representation that allows to navigate to arbitrary positions. In this paper, we will show how we extended the model to be able to do so.

While Dura-Bernal and colleagues [10] have argued that such a biomimetic neuronal model could be used for Brain-Computer Interfaces (BCIs) to translate brain activity for control of a robotic limb, we want to use this model in the opposite direction. Adaptive decoding methods, that adapt to the users brain signals, have recently been shown to improve BCI performance [11], [12], [13]. The currently fastest non-invasive BCI uses unsupervised learning and error-related potentials [14] for adaptation to reach information transfer rates of more than 140 bit/min which allows to write more than 20 error-free characters per minute [15], showing that adaptation is a key aspect in BCI.

However, there is a mutual interaction since the user is also adapting to the BCI and it is yet unclear if this mutual interaction (also termed co-adaptation) is always beneficial or how adaptive algorithms need to be designed to improve co-adaptivity. Previous approaches to study co-adaptivity either use human subjects to simulate the mutual interaction [16] or model the co-adaptation as control problem [17], [18]. To have a more biologically realistic simulation, we modeled a user learning BCI control by reinforcement learning to investigate co-adaptation [19]. However, there are some aspects of BCI control (e.g. use of BCI in stroke rehabilitation) that can not be modeled using that previous approach, which is why we want to use spiking neurons in future modeling. The first step towards that goal is a working spiking neuronal model that learns motor control by reinforcement learning.

In the following, we show how the work by Chadderdon et al. [8] was extended, so that the model learns to reach arbitrary target angles by reinforcement. We further show how we introduced structural synaptic plasticity as biologically motivated method to improve learning. In the end, we discuss how the results can be used for investigating co-adaptivity in BCI.

II. Methods

A general outline of the key parts of the simulation is shown in figure 1. On the one side, there is a spiking neuronal model with different inter-connected areas. On the other side, there is a simulated arm with extensor and flexor muscles which control the angle of the arm. Both muscles receive input from the excitatory motor (EM) neurons of the neuron model. Proprioceptive feedback about the current angle of the arm $\theta_{current}$ is given into the proprioceptive (P) neurons. Information about the target angle θ_{target} that the arm has to reach, is given to the target (T) neurons. Depending on the error of the movement, some of the connections of the neuronal model are changed by reinforcement learning. In the following subsections, the different aspects of the model will be described in more detail.

978-1-4799-1959-8/15/$31.00 ©2015 IEEE

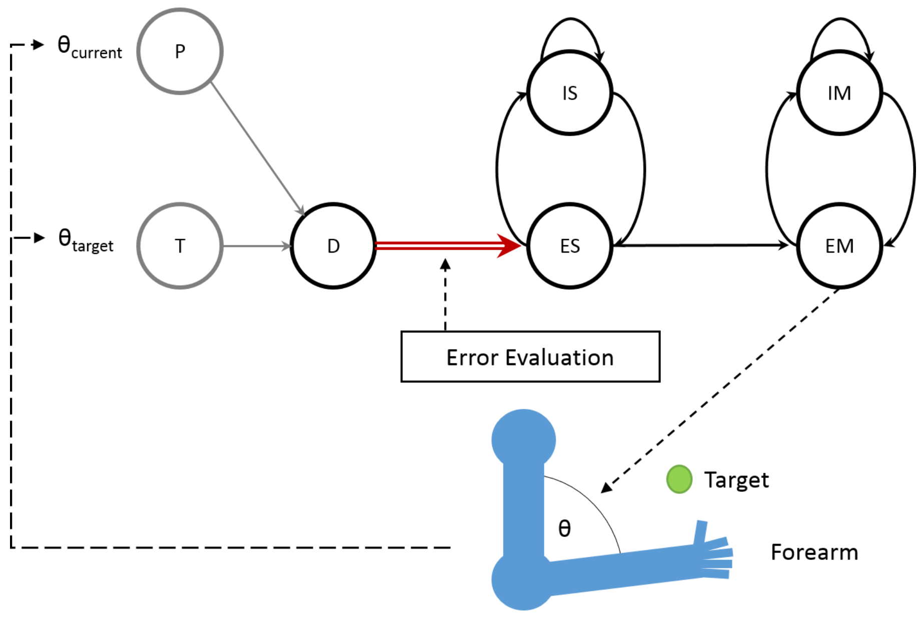

Fig. 1. Overview of the model: The upper part depicts the neural network with the different areas (circles), representing proprioceptive neurons (P), target neurons (T), difference neurons (D), inhibitory sensory neurons (IS), excitatory sensory neurons (ES), inhibitory motor neurons (IM) and excitatory motor neurons (EM). Neurons that are modeled implicitly by only modeling their effect on other areas are shown in gray. Static connections are shown in black. Connections shown in red are affected by reinforcement learning. The lower part shows the virtual arm and the target. The virtual arm is driven by the activity of EM neurons. Information about the current arm angle $\theta_{current}$ and the target angle θ_{target} is given back to the neuronal model.

A. *Neuron Model*

To simulate the individual neurons, we used Izhikevich's "simple model" [20], which was also used by Chadderdon et al. [8]. This model provides a biologically realistic behavior similar to the Hodgkin-Huxley model [21] and on the other side has a similar computational efficiency as the integrate-and-fire model [22].

Thereby, one neuron is modeled by the following two time-dependent differential equations:

$$v' = 0.04v^2 + 5v + 140 - u + I \tag{1a}$$

$$u' = a(bv - u) \tag{1b}$$

and an after-spike resetting

$$\text{if } v \geq 30 \text{ mV, then} \begin{cases} v \leftarrow c \\ u \leftarrow u + d \end{cases} \tag{2}$$

where v represents the membrane potential and u represents the membrane recovery variable. The latter provides negative feedback to v and stands for both the activation of K^+ and the inactivation of Na^+ ionic currents. The variable I defines the synaptic input currents. The choice of dimensionless parameters a, b, c, and d leads to various intrinsic firing patterns [20].

Each individual neuron is modeled as a dynamic unit and is labeled as excitatory or inhibitory. Each cell has a membrane potential variable (V_m) describing the current voltage state, a resting membrane potential (V_r) describing the baseline voltage, a spiking threshold (V_t) describing the voltage at which the cell fires an action potential, and a noise input variable (V_n) which is needed for the motor babble.

At initiation of the model, V_m is set to V_r for each cell. The underlying step-wise processing is done by the simple spiking model equations (1) and (2) whereas $v = V_m$. The parameter c is equivalent to V_r, and for the values for parameters a, b, and d we used the default values recommend by Izhikevich [20], which are also shown in table I. To achieve heterogeneity, in order that different neurons have different dynamics, a uniformly distributed random variable r_i is introduced for each neuron i. If a neuron i is excitatory than $r_i = 0$ corresponds to regular spiking and $r_i = 1$ leads to a chattering behavior. Otherwise, if a neuron i is inhibitory than different values for r_i shifts the neuron's dynamic between fast spiking and low-threshold spiking.

TABLE I. Parameters that were used for the simple spiking model separated for excitatory and inhibitory neurons.

Parameter	Excitatory	Inhibitory
a	0.02	$0.02 + 0.08r_i$
b	0.2	$0.25 - 0.05r_i$
c	-65	-63
d	$8 - 6 \cdot r_i^2$	2

If a neuron receives additional noise input, the noise input V_n is modeled similar to the AMPA noise stimulation used by Chadderdon et al. [8], so that noise stimulation rates are about $f_n(\text{AMPA}) = 300$ Hz. The calculation of synaptic noise input V_n depends on the current membrane potential V_m.

$$V_n' = w_n \left(1 - \frac{V_m}{E_n}\right) \tag{3}$$

B. *Neural Network Architecture*

The architecture of the neural network is similar to the one used by Chadderdon et al. [8] but has some subtle differences that ultimately enables the model to learn arbitrary target angles.

TABLE II. CONNECTION WEIGHTS AND PROBABILITIES BETWEEN THE DIFFERENT NEURON TYPES. A NEGATIVE WEIGHT DENOTES INHIBITORY CONNECTIONS.

Connection	Weight	Probability
EM $\rightarrow$ IM	1.9	0.43
IM $\rightarrow$ EM	-4.5	0.44
IM $\rightarrow$ IM	-4.5	0.62
ES $\rightarrow$ EM	5.28	0.08
ES $\rightarrow$ IS	1.9	0.43
IS $\rightarrow$ ES	-4.5	0.44
IS $\rightarrow$ IS	-4.5	0.62
D $\rightarrow$ ES	8.27	0.2

The neural network consists of populations with different neuron types: excitatory sensory (ES) neurons, inhibitory sensory (IS) neurons, excitatory motor neurons (EM), inhibitory motor (IM) neurons and a neuron population coding the difference (D) between current angle and target angle. The D neurons receive input from proprioceptive (P) neurons encoding the current angle and target (T) neurons encoding the target angle. However, T and P populations were not modeled directly, but were modeled implicitly by only modeling their effect on D, which will be explained in subsection II-D in more detail. In total, the network consisted of 48 EM neurons, 32 IM neurons, 96 ES neurons, 32 IS neurons and 96 D neurons.

Which neuron populations were connected is show in figure 1. For two connected populations P_1 and P_2, there is a probability with which a neuron in population P_1 is connected to a neuron from population P_2. Table II shows the connection probabilities for the different connections as well as the different synaptic weights. A negative weight denotes an inhibitory connection.

The EM, IM, IS neurons had additional noise input that was differently weighted by 2.2 for EM, 1.5 for IS and IM. These weights were adjusted to have an average spiking rates f_a (representing motor babble) for the individual neuron populations of $f_a(ES) = 0.4$ Hz, $f_a(IS) = 4.4$ Hz, $f_a(EM) = 0.5$ Hz, $f_a(IM) = 4.3$ Hz in the absence of learning.

To train the model, the connections D $\rightarrow$ ES changed due to the reinforcment learning, which will be explained in detail in section II-E. All other connections were kept constant during the whole simulation. The EM neurons were used as input to control the extensor and flexor muscle activities, while proprioceptive information about the current forearm angle was used as input for P neurons and information about the target angle was encoded in the T neurons.

C. System Design

As previously explained, the spiking neural network controls a virtual arm. The virtual arm's current joint angle $\theta \in [0, 135]$ varies on its interval, whereas $\theta = 0°$ stands for a straight arm position and $\theta = 135°$ stands for fully flexed, respectively. The angular position of the arm is changed every 50 ms. The angular change $\Delta\theta$ is determined by the number of spikes in the EM neuron population integrated over an 50 ms interval that started 100 ms before the current time point, resulting in a delay of 50 ms, which represents the peripheral and sub-cortical processing delay.

The first half of the EM neurons provides input to the extensor muscle, while the second half provides input to the flexor muscle. The absolute difference in the spike counts between the EM extensor neurons and the EM flexor neurons provides the absolute angular change $\Delta\theta$, with one spike corresponding to the change of $1°$. Depending, if the extensor or the flexor neurons have the higher spike count, the arm performs either an extension or a flexion.

Proprioceptive input about the current arm angle $\theta_{current}$ is encoded in the P neurons, while the current target angle θ_{target} is encoded in the T neurons. In this simulation P and T neurons are not modeled directly, but only their effect on the D neurons is modeled as explained in the next section.

D. Difference Coding

The difference between θ_{target} and $\theta_{current}$ is an important information to correctly control the virtual arm. In our simulation, the encoding of the difference between θ_{target} and $\theta_{current}$ was modeled implicitly, in a way that P and T cells as well as the connections P $\rightarrow$ D and T $\rightarrow$ D were not modeled directly, but only the resulting activation pattern on D was modeled. Since the connections P $\rightarrow$ D and T $\rightarrow$ D were fixed and non-random for each model, it was faster in terms of implementation and computational efficiency to use this implicit modeling of P and T by only model the resulting activation pattern in D. Activity from P neurons and from T neurons stimulated the D neurons in a way that D neurons were stimulated with both proprioceptive information and target angle at the same time or rather in the same time window by using different neural coding methods for proprioceptive and target information. Each of two encoding schemes (for proprioceptive information or target angle) should slightly stimulate the corresponding D cells, in order that no action potential occurs, only those cells which were stimulated by both encodings fire an action potential.

An exemplary activity of the D cells for 4 different combinations θ_{target} and $\theta_{current}$ is shown in figure 2. To encode the target angle θ_{target}, a temporal coding scheme is used, in which all D neurons are stimulated at a time point τ

A temporal coding scheme is used for target angles θ_{target} and a mixture of both population coding and temporal coding is used for proprioceptive information $\theta_{current}$. A well defined time-window τ_{coding} = 50 ms is used for both encodings. The temporal coding rule for the θ_{target} is given by:

$$\theta_{norm} = \frac{\theta_{target}}{\theta_{max} - \theta_{min}} \tag{4a}$$

$$t_{fire} = round((\tau_{coding} - 1) \cdot \theta_{norm}) + 1 \tag{4b}$$

where $\theta_{max} = 135°$, $\theta_{min} = 0°$ and t_{fire} is the exact time-point (in ms) during the time-window τ_{coding} at which all D neurons receive input from the T cells. Because of the *ms* time-scale, t_{fire} lies within [1, 50], whereas $t_{fire} = 1$ ms means a target angle of $\theta_{target} = 0°$ and $t_{fire} = 50$ ms means $\theta_{target} = 135°$. In figure 2, the two upper subplots (A and B) show a $t_{fire} = 12$ ms (vertical bar at 12 ms) representing a $\theta_{target} = 30°$, while the lower two subplots (C and D)

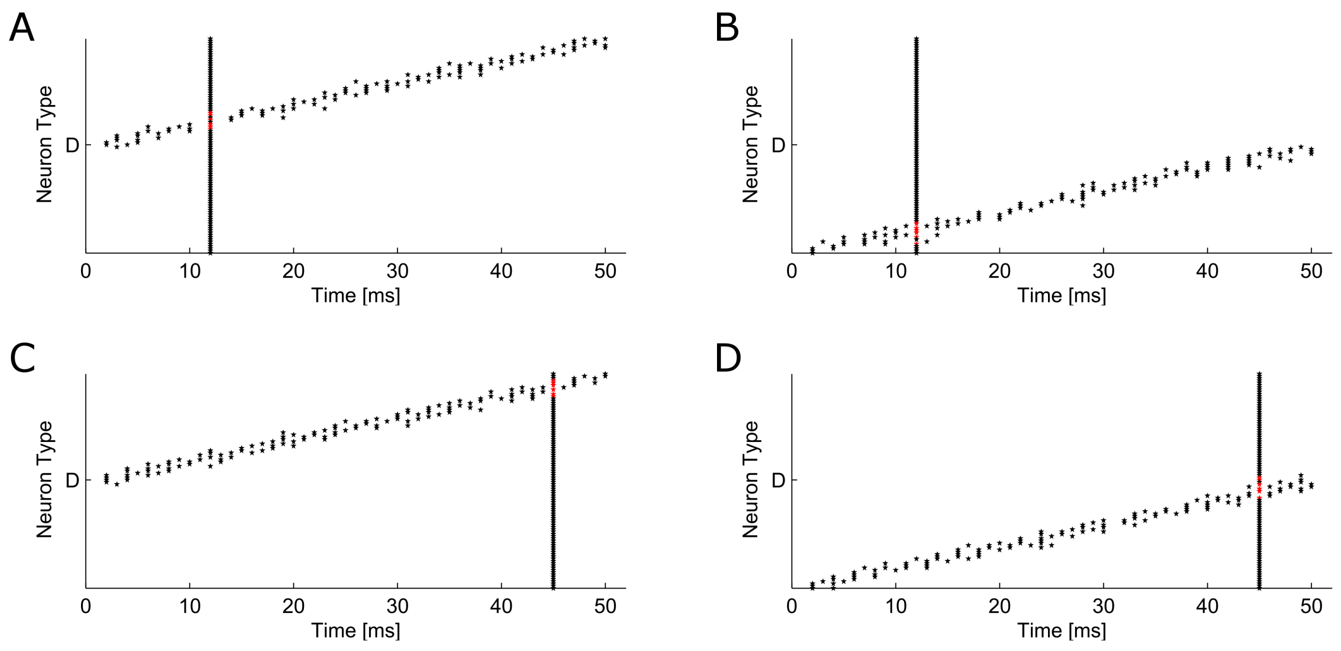

Fig. 2. Activity of distance (D) neurons for different combinations of proprioceptive angle and target angle. Black dots denote a stimulation of the corresponding neuron without eliciting an action potential, while a red dot denotes an action potential. The vertical line corresponds to the activity caused by the T neurons, encoding the target angle θ_{target}, while the diagonal line represents the activity caused by the P neurons an encodes the proprioceptive information $\theta_{current}$. A: $\theta_{target} = 30°$, $\theta_{current} = 0°$ B: $\theta_{target} = 30°$, $\theta_{current} = 135°$ C: $\theta_{target} = 120°$, $\theta_{current} = 0°$ D: $\theta_{target} = 120°$, $\theta_{current} = 135°$

show a $t_{fire} = 45$ ms (vertical bar at 45 ms) representing a $\theta_{target} = 120°$.

The coding of the proprioceptive information is a bit more complex, because it is a mixture of population coding and temporal coding. First of all, at each 1 ms step within τ_{coding} some of the D neurons were stimulated. The coding depends on both the current angle $\theta_{current}$ and the current time $t_{current}$ within τ_{coding}

$$t_{norm} = \frac{\tau_{coding} - t_{current}}{\tau_{coding} - 1} \tag{5a}$$

$$\theta_{shift} = (\theta_{max} - \theta_{current}) - t_{norm} \cdot (\theta_{max} - \theta_{min}) \tag{5b}$$

where t_{norm} is the normalized time relative to the time-window and θ_{shift} is an angle in the interval $[-135°, 135°]$, whereas this angle will be increased by increasing $t_{current}$ and decreased by increasing $\theta_{current}$, and vice versa. Therefore, if $\theta_{current} = 135°$ and $t_{current} = 1$ ms then $\theta_{shift} = -135°$, otherwise if $\theta_{current} = 0°$ and $t_{current} = 50$ ms then $\theta_{shift} = 135°$. At each time-point $t_{current} \in [1, \tau_{coding}]$ within each time-windows τ_{coding}, D neurons were stimulated using a probabilistic model based on θ_{shift}, where each neuron has a center and the stimulation probability was modeled using a radial-basis function based on the distance to the center. Patterns for $\theta_{current} = 0°$ and $\theta_{current} = 135°$ are shown as "diagonals" in figure 2, the patterns for each other $\theta_{current}$ shifts linearly between these.

E. Reinforcement Learning

Reinforcement learning methods do not, contrary to teacher-supervised learning methods, require a known desired output representation to match against the models current output, however, they do offer some feedback regarding fitness of the behavior [8]. The Perception-Action-Reward Cycle (PARC) [23] is a method for motor reinforcement learning. This system is based on an actor-critic method, whereas the actor maps the signals (perceptions) to actions, and the critic evaluates those actions and sends reward/punishment feedback to the actor.

In our model, the neural network serves as the actor, controlling the motor system. A critic was implemented by comparing the current angle $\theta_{current}$ with the θ_{target} and for each time-point t (in 50 ms steps) calculating the difference $\theta_{d,t} = \theta_{target} - \theta_{current}$. Based on $\theta_{d,t}$, the critic generates a global reward signal if $\theta_{d,t} < \theta_{d,t-1}$ or a global punishment signal if $\theta_{d,t} > \theta_{d,t-1}$, which means that the current behavior of the neural network is either rewarded or punished based on whether the arm is moving towards or away from θ_{target}.

Spike timing dependent plasticity (STDP), was used to change the connection weights of the D $\rightarrow$ ES connections. We used a spike-timing dependent rule to trigger eligibility traces [5] to solve the credit assignment problem. The eligibility traces were turned on for a synapse if a postsynaptic spike followed a presynaptic within a time-windows of 100 ms. After that, the eligibility weight decayed linearly from 1 to 0 in the next 100 ms. Eligibility tagged synapses were potentiated when a reward was delivered (long-term potentiation; LTP) or depressed (long-term depression; LTD) when punishment was delivered.

Weights of the synapses $w(t)$ were updated using a weight scale factor w_s:

$$w(t) = w_0 w_s(t) \tag{6}$$

$$w_s(t+1) = w_s(t) + \Delta w_s \quad (7)$$

$$\Delta w_s = \begin{cases} 1 - w_s(t)/w_s^{max} & \text{for LTP reward} \\ w_s(t)/w_s^{max} & \text{for LTD punish} \end{cases} \quad (8)$$

where $w_s^{max} = 5$ is the maximum weight scale factor and w_0 is the initial synaptic weight. w_s is initialized to 1 and varies between 0 and w_s^{max}.

F. Structural Synaptic Plasticity (SSP)

To further improve the model, we added Structural Synaptic Plasticity (SSP), which is a known mechanism in the mammalian brain [24]. SSP describes the ability of synaptic connections to emerge and disappear over time and thereby restructure the neuronal connections. Recent experimental work provides compelling evidence that SSP is a crucial component of learning [25]. Therefore, the reinforcement learning process was extended by this feature in the following manner.

The criteria for determining whether a reconnection has to occur or not is made by the weight scale factor w_s. If w_s of a single synapse drops below 0.2 then the synapses will be disconnect and randomly reconnected to a new not yet connected neuron of the same type as the previously connected neuron. This ensures that the number of connections between two neuronal populations remains the same. The weight scale factor for a freshly connected synapse is set to $w_s = 1.0$. By improving reinforcement learning through SSP, this should help to increase the number of working models and speed up the learning process.

G. Simulation Procedure

For this study, we trained a total of 1000 models with randomly initialized connections. In 500 of the models, the model was trained with reinforcement learning based on STDP only. In the other 500 models, we additionally enabled SSP. The procedure for each simulation consisted of one training run and if the model was successfully trained, the learning was disabled and the quality of the model (regarding angular error) was evaluated.

For the training, the model was only trained with two angles (0° and 135°). At the beginning of the simulation, the current arm angle was set to $\theta_{start} = 135°$ and the first target was $\theta_{target1} = 0°$. Once the arm reached the target angle, the target was changed to $\theta_{target2} = 135°$. If the this target angle was also reached, the learning was turned off (weights were kept static from now on) to see if learning was successful. Therefore, the target angle was set to $\theta_{target} = 0°$ for 30 seconds and to $\theta_{target} = 135°$ for another 30 seconds. If both targets were reached during the corresponding 30 seconds interval, the model was labeled as successfully trained.

To test how well the models were trained, a testing run was performed for each model that was labeled as successfully trained. For this testing run, 6 different target angles $\theta_{target} \in [0°, 30°, 60°, 90°, 120°, 135°]$ were presented for 30 seconds each, making the total duration of the testing run 180 seconds.

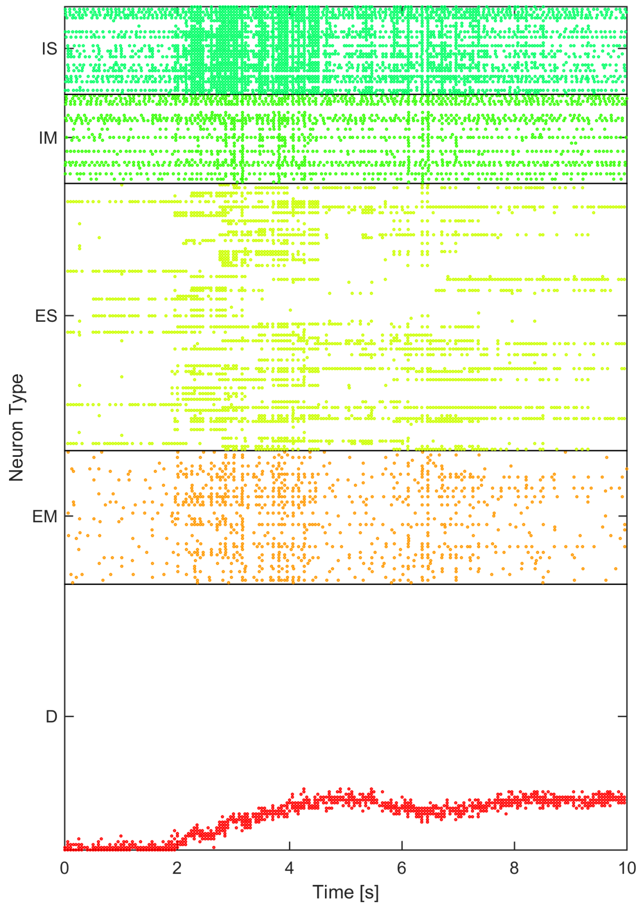

Fig. 3. Exemplary activity of an untrained model during the first 10 seconds of simulation. Each dot represents a firing of a neuron, the different neuron types are marked by different colors.

The root mean squared distance (RMSD) was used to calculate the performance of the trained model. Given the current arm angle $\theta_{current,t}$ at time t and the target angle $\theta_{target,t}$ at time t, the RMSD was computed with the following equation:

$$RMSD = \sqrt{\frac{1}{n}\sum_{t=1}^{n}\left(\theta_{current,t} - \theta_{target,t}\right)^2} \quad (9)$$

To obtain separate RMSD values for each angle, RMSD was calculated for each angle on the last 20 seconds.

The whole simulation (of the neural network and the simulation of the virtual arm) were implemented as MATLAB code.

III. Results

Figure 3 shows 10 seconds of exemplary activity for an untrained network. The angular change over time, as well as the target angles during a training run, can be seen for one exemplary model in figure 5, while the results of the testing run for the best model is shown in figure 6.

To investigate the benefit of the SSP implementation, we trained 500 models with SSP and 500 models without. As

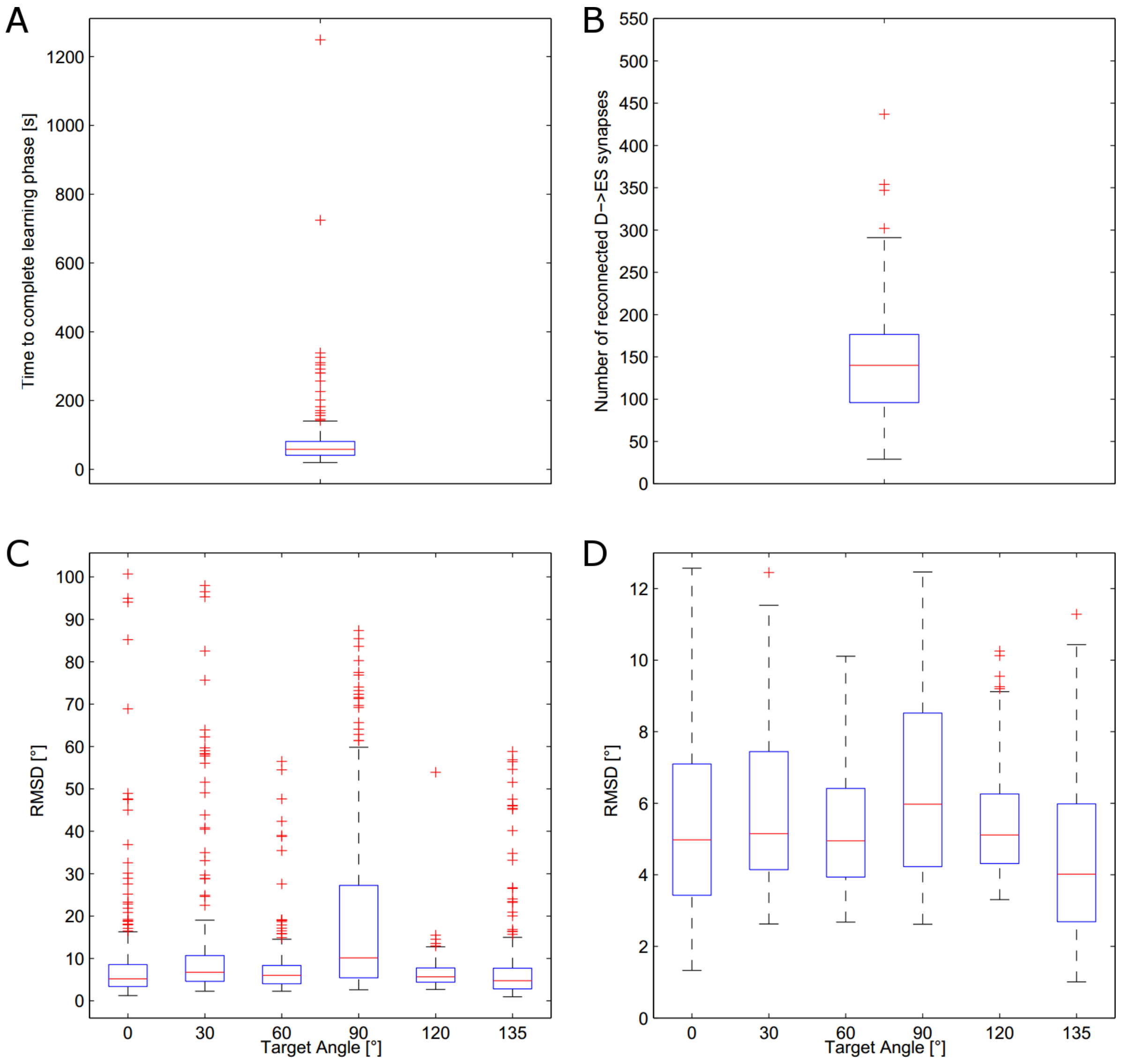

Fig. 4. Performance of the 257 models that were successfully trained with SSP. The red line represents the mean and the box represents the interquartile range (IQR) limited by the first (q_1) and third quartile q_3. The whiskers extend to $q_3 + 1.5 \cdot \text{IQR}$ and $q_1 - 1.5 \cdot \text{IQR}$, respectively. The red crosses show the outliers. (A) Variations of the needed learning duration. (B) Number of reconnected D $\rightarrow$ ES synapses caused by SSP. (C) RMSDs of the 257 models: starting at θ_{max}, the task was to reach different angles in this sequence: 30°, 90°, 0°, 60°, 135°, and 120°, whereas targets were changed each 30 seconds (in total 180 seconds). (D) RMSDs of the best 100 models.

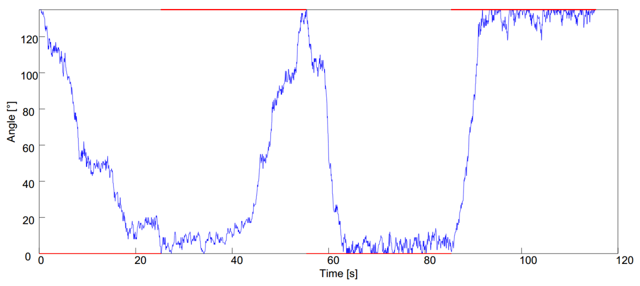

Fig. 5. Movement trajectory during the training of the model. The blue line represent the angle of the simulated arm, while the red lines (at 0° and 135°) are the target angles, the arm has to reach. The learning was enabled for the first two targets, while learning was disabled for the last two targets. If the last two targets were reached in time, the model was labeled as successfully learned.

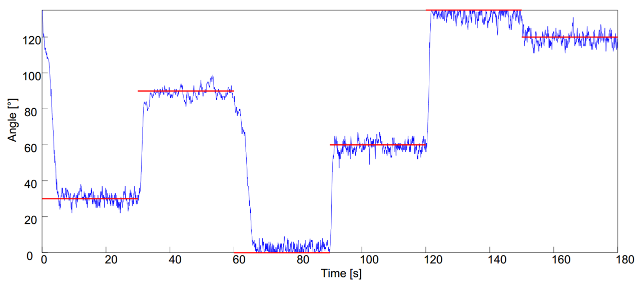

Fig. 6. Movement trajectory of a successfully trained model. During the simulation, the connections were kept static and no reinforcement learning was applied. The blue line represent the angle of the simulated arm, while the red lines are the target angles.

a result, we obtained 257 successfully trained models with SSP and 198 successfully trained models without SSP, showing that SSP leads to a significantly higher number of successfully trained models ($p < 0.001$, Wilcoxon ranksum test).

Regarding the performance of the models trained with SSP, figure 4A shows the required simulation time to train the models. On average it took 79 seconds (of simulated time) until the learning phase has been completed. Interestingly, the shortest learning duration amounts only 19.6 seconds, whereas the longest learning phase required 1249 seconds to complete. While learning, the SSP rule reconnected on average 142 D $\rightarrow$ ES synapses (figure 4B), whereas the average number of D $\rightarrow$ ES synapses amounts to 1847, meaning that about 7 % of the synapses were reconnected.

For each of the 257 models that were successfully trained using SSP, we calculated the RMSD to check the performance of the models. The results can be seen in figure 4C for all 257 models, as well as for the best 100 models (figure 4D). Separated by target angles, $\theta_{target} = 90°$ was the angle that on average had the highest deviation with an RMSD of about $10°$, while RMSD was between $4.5°$ and $7°$ for the other angles. Considering only the 100 best models, the average RMSD was between between $4°$ and $6°$ for all angles.

Regarding the computational time to run the simulation, simulating 1000 ms took on average 1027 ms of calculation time, while learning is enabled. After a model was trained, the simulation of 1000 ms took on average 987 ms on an Intel (R) Core TM i5-3210M (2x 2.5 GHz, 8 GB RAM). This shows that (using a slightly faster CPU) a real-time simulation is possible.

IV. Discussion

With this work we aimed at implementing a biologically plausible model of motor learning based on spiking neural networks. We based the initial design on the work by Chadderdon et al. [8], but extended it to be able to reach arbitrary target angles. The model by Chadderdon et al. did not use information about the target angle as input for the neural network, but only used the difference to the target angle in their error evaluation function that was used for the reinforcement learning. Thereby their model was trained to hold one specific target angle and once trained, was only able to hold this angle. When the model was provided with a new target angle, the model had to re-learn to hold the new target angle.

Based on the work by Chadderdon et al. we extended the model in a way that information about current angle and target angle are put directly into the neural network and that the network learns one stable representation, which allows it to reach arbitrary target angles. It is noteworthy that a training on two target angles ($0°$, $135°$) is sufficient for the model to be able to reach arbitrary target angles with averages errors below $10°$.

We further introduced structural synaptic plasticity (SSP) into our model, which allows elimination of old synaptic connections of no importance, as well as the formation of new synaptic connections between two previously unconnected neurons. By introducing SSP, another aspect that is close to its biological original, is modeled in our simulation. Further, it significantly improves the number of successfully trained models by about 30 %.

Based on this model learning motor control, we will extend it in future work to investigate the mutual learning effects in adaptive Brain-Computer Interfacing. As previous simulations still rely on human subjects [16], use simulation methods that are not biologically motivated [17], or are biologically motivated but not applicable for all BCI scenarios (e.g. BCI for stroke rehabilitation) [19], we believe that using spiking neurons offers greater possibilities to investigate the mutual learning effects between an adaptive BCI and a learning user. Especially, to investigate co-adaptivity for BCIs in stroke rehabilitation [26], the use of this model seems beneficial. Once the model is trained, the effects of stroke (damage to motor-related parts of the brain) could be simulated by removing specific neuronal areas or removing connections in our model, which should result in loss of motor control. Based on the generated neuronal activity of our model, a BCI can be trained and the effects of BCI on rehabilitation (learning time till motor control is regained) can be evaluated and different adaptive methods can be tested to optimize co-adaptivity.

V. Conclusion

In this study, we extended the work by Chadderdon et al. to train a model to reach arbitrary target angles using reinforcement learning. Although our model is only trained on 2 target angles, the trained model can reach arbitrary target angles after learning. Once the model has learned a stable neuronal mapping, no further learning is required when the model is presented with a new target angle to reach. Further, we could show that the use of structural synaptic plasticity (SSP) leads to a significantly higher number of successfully trained models.

Acknowledgment

The work was supported by the *Baden-Württemberg Stiftung* (GRUENS), the *Deutsche Forschungsgemeinschaft* (DFG; KOMEG) and the Bernstein Center for Computational Neuroscience, Tübingen, which is funded by the German Federal Ministry of Education and Research (BMBF; FKZ: 01GQ1002).

References

[1] W. Maass, “Networks of spiking neurons: the third generation of neural network models,” *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[2] L. I. Helgadottir, J. Haenicke, T. Landgraf, R. Rojas, and M. P. Nawrot, “Conditioned behavior in a robot controlled by a spiking neural network,” in *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*. IEEE, 2013, pp. 891–894.

[3] H. Burgsteiner, “Training networks of biological realistic spiking neurons for real-time robot control,” in *Proceedings of the 9th international conference on engineering applications of neural networks, Lille, France*, 2005, pp. 129–136.

[4] D. Gamez, A. K. Fidjeland, and E. Lazdins, “ispike: a spiking neural interface for the icub robot,” *Bioinspiration & biomimetics*, vol. 7, no. 2, p. 025008, 2012.

[5] E. M. Izhikevich, “Solving the distal reward problem through linkage of STDP and dopamine signaling,” *Cerebral cortex*, vol. 17, no. 10, pp. 2443–2452, 2007.

[6] R. R. Carrillo, E. Ros, C. Boucheny, and J.-M. C. Olivier, “A real-time spiking cerebellum model for learning robot control,” *Biosystems*, vol. 94, no. 1, pp. 18–27, 2008.

[7] A. Bouganis and M. Shanahan, "Training a spiking neural network to control a 4-dof robotic arm based on spike timing-dependent plasticity," in *Neural Networks (IJCNN), The 2010 International Joint Conference on*. IEEE, 2010, pp. 1–8.

[8] G. L. Chadderdon, S. A. Neymotin, C. C. Kerr, and W. W. Lytton, "Reinforcement learning of targeted movement in a spiking neuronal model of motor cortex," *PloS one*, vol. 7, no. 10, p. e47251, 2012.

[9] S. A. Neymotin, G. L. Chadderdon, C. C. Kerr, J. T. Francis, and W. W. Lytton, "Reinforcement learning of two-joint virtual arm reaching in a computer model of sensorimotor cortex," *Neural computation*, vol. 25, no. 12, pp. 3263–3293, 2013.

[10] S. Dura-Bernal, G. L. Chadderdon, S. A. Neymotin, J. T. Francis, and W. W. Lytton, "Towards a real-time interface between a biomimetic model of sensorimotor cortex and a robotic arm," *Pattern Recognition Letters*, vol. 36, pp. 204–212, 2014.

[11] M. Spüler, W. Rosenstiel, and M. Bogdan, "Adaptive SVM-based classification increases performance of a MEG-based Brain-Computer Interface (BCI)," in *Artificial Neural Networks and Machine Learning–ICANN 2012*. Springer, 2012, pp. 669–676.

[12] C. Vidaurre, C. Sannelli, K.-R. Müller, and B. Blankertz, "Machine-learning-based coadaptive calibration for brain-computer interfaces," *Neural computation*, vol. 23, no. 3, pp. 791–816, 2011.

[13] M. Spüler, W. Rosenstiel, and M. Bogdan, "Principal component based covariate shift adaption to reduce non-stationarity in a MEG-based brain-computer interface," *EURASIP Journal on Advances in Signal Processing*, vol. 2012, no. 1, pp. 1–7, 2012.

[14] M. Spüler, M. Bensch, S. Kleih, W. Rosenstiel, M. Bogdan, and A. Kübler, "Online use of error-related potentials in healthy users and people with severe motor impairment increases performance of a P300-BCI," *Clinical Neurophysiology*, vol. 123, no. 7, pp. 1328–1337, 2012.

[15] M. Spüler, W. Rosenstiel, and M. Bogdan, "Online adaptation of a c-VEP Brain-Computer Interface (BCI) based on Error-related potentials and unsupervised learning," *Plos One*, vol. 7, no. 12, p. e51077, 2012, doi: 10.1371/journal.pone.0051077.

[16] J. P. Cunningham, P. Nuyujukian, V. Gilja, C. A. Chestek, S. I. Ryu, and K. V. Shenoy, "A closed-loop human simulator for investigating the role of feedback control in brain-machine interfaces," *Journal of Neurophysiology*, vol. 105, no. 4, pp. 1932–1949, 2011.

[17] J. S. Merel, R. Fox, T. Jebara, and L. Paninski, "A multi-agent control framework for co-adaptation in brain-computer interfaces," in *Advances in Neural Information Processing Systems*, 2013, pp. 2841–2849.

[18] M. Lagang and L. Srinivasan, "Stochastic optimal control as a theory of brain-machine interface operation," *Neural computation*, vol. 25, no. 2, pp. 374–417, 2013.

[19] M. Spüler, W. Rosenstiel, and M. Bogdan, "Co-adaptivity in Unsupervised Adaptive Brain-Computer Interfacing: a Simulation Approach," in *COGNITIVE 2012, The Fourth International Conference on Advanced Cognitive Technologies and Applications*, 2012, pp. 115–121.

[20] E. M. Izhikevich *et al.*, "Simple model of spiking neurons," *IEEE Transactions on neural networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[21] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of physiology*, vol. 117, no. 4, p. 500, 1952.

[22] E. M. Izhikevich, "Which model to use for cortical spiking neurons?" *IEEE transactions on neural networks*, vol. 15, no. 5, pp. 1063–1070, 2004.

[23] J. Sanchez, A. Tarigoppula, J. Choi, B. Marsh, P. Chhatbar, B. Mahmoudi, and J. Francis, "Control of a center-out reaching task using a reinforcement learning brain-machine interface," in *Neural Engineering (NER), 2011 5th International IEEE/EMBS Conference on*, April 2011, pp. 525–528.

[24] P. Caroni, F. Donato, and D. Muller, "Structural plasticity upon learning: regulation and functions," *Nature Reviews Neuroscience*, vol. 13, no. 7, pp. 478–490, 2012.

[25] A. Holtmaat and K. Svoboda, "Experience-dependent structural synaptic plasticity in the mammalian brain," *Nature Reviews Neuroscience*, vol. 10, no. 9, pp. 647–658, 2009.

[26] S. Silvoni, A. Ramos-Murguialday, M. Cavinato, C. Volpato, G. Cisotto, A. Turolla, F. Piccione, and N. Birbaumer, "Brain-computer interface in stroke: a review of progress," *Clinical EEG and Neuroscience*, vol. 42, no. 4, pp. 245–252, 2011.