Simulating Mirror Neurons

Kate Derosier and Jackie Kay

Abstract

Mirror neurons are motor cortical neurons found in primates and thought to be important for imitative learning. Multiple attempts have been made to model mirror neurons in a developmental robotics context. In this paper, we implement a system based on the article *Towards a robotic model of the mirror neuron system* by Rebrová, Pecháč and Farkaš [5]. Their system emphasizes biological plausibility by using a bidirectional learning algorithm to connect motor and visual modules in a manner inspired by primate brain architecture. In §1, we describe mirror neurons and previous developmental robotics work inspired by them. In §2, we explain the component parts to our system and how they work together. In §3 and §4, we perform experiments to test its performance on a supervised learning task: associating visual and motor data for three different hand grasp types. Finally, in §5, we suggest ways our implementation could be improved or expanded upon.

1 Background and related work

1.1 Mirror neurons

The mirror neuron system may provide a neural substrate for action understanding (inferring another's intentions by observing their actions) and for imitative learning [6]. Because action understanding and imitative learning are desirable skills for an adaptive agent to have, many developmental robiticists have attempted to implement a robotic version of a mirror neuron system.

Mirror neurons were discovered in area F5 in the premotor cortex of macaque monkeys. The distinguishing property of these neurons is that they respond to both visual and motor stimuli. Moreover, the visual and motor stimuli that drive a mirror neuron are in some sense the same: if a mirror neuron is active when the monkey performs a specific goal-directed action, such as a precision grasp to pick up an object, that same neuron will be active when the monkey views a human or another monkey performing the same action. The neuron will be active regardless of the object picked up, but it will not be active when viewing a visually similar but non-goal related arm movement. Note also that F5 mirror neuron activity is largely view-invariant, that is, neurons are equally active regardless of the distance or angle from which the subject monkey views the demonstrated action [6].

Another brain region important to the mirror neuron system is the superior temporal sulcus (STS). This area contains neurons with visual properties similar to F5 motor neurons, but which do not have motor responses. STS can be thought of as encoding both view-dependent and view-invariant representations of observed goal-directed actions, and sends output to premotor areas (including F5) via PF, a region of the rostral inferior parietal lobule that has both visual and motor response properties [6].

Although the circuit described above is based on the monkey brain, there is also evidence for a mirror neuron system in humans. Both fMRI experiements [6] and single neuron recordings from epilepsy patients [3] suggest that the human mirror neuron system exists and is architecturally similar to the monkey mirror neuron system.

1.2 HAMMER

Although we have chosen to focus on the system designed by Rebrová, Pecháč, and Farkǎs [5], in this section we will briefly describe the more widely known HAMMER system developed by Demiris and colleagues [1].

HAMMER, which stands for Hierarchical Attentive Multiple Models for Execution and Recognition, imitates the functionality of the mirror neuron system using a combination of inverse and forward models. An inverse model maps points in the behavior space to motor plans in the motor control space that would produce them, while a forward model maps points in the motor control space to the behaviors they result in. The HAMMER system has multiple inverse model/forward model pairs to represent different actions. During observation, each inverse model requests the information needed to predict what action the demonstrator is performing. Inverse models also pass on their confidence values. An arbitration module decides which inverse model(s) to listen to and uses the chosen requests to influence the saliency calculations of the bottom-up image processing models. The most salient part of the image is returned to the inverse model requesting it, and the inverse model generates a motor plan which its associated forward model turns into a prediction of the next state. Finally, the prediction error is calculated and used to update the confidence values. This loop allows the robot to learn from demonstration and develop an association between observed actions and its internal motor representations [1].

The foremost advantage of HAMMER is that it enables a physical robot to recognize when a human demonstrator is performing a known action. However, this learning is far from universal, and many aspects of the system are hard-coded in, including all the forward models [1]. Additionally, although the HAMMER system is inspired by the mirror neuron system, its structure and implementation are far from biologically plausible.

1.3 Self-perception

Another robotic system inspired by the mirror neuron system is the self-perception system designed by Saegusa and colleagues [7]. Like [5], this system uses the humanoid iCub robot. However, it is much more involved in visual processing and the development of an internal visual representation.

In these experiments, an iCub robot follows a humanlike developmental trajectory to learn self-perception, action perception, and action imitation. First, the robot learns to recognize its own body parts. It does this by identifying areas of motion that are correlated with its proprioception, or sense of its own movement. Then, through experimental but repetitive motions of each of its joints, it learns to perform action primitives such as fixation and grasping. At this stage, the robot is able to predict the outcomes of its own actions, and even to recognize the actions of a human. Finally, the human experimenters combine the action primitives into a more complex motion, which the robot observes and imitates. This series of stages is similar to the stages that infants are thought to go through, and indeed the calculations and algorithms used can almost be thought of as a model for perceptual development in humans.

The system in [7] is very focused on visual representations and does not utilize a complex motor representation. Although this may be a functional and useful system in real robots, it is unsatisfactory as a simulation of the mirror neuron system, since the mirror neuron system is inherently about the association between a motor representation and a vidual representation.

2 ARMS system

Our system, which we have called Adaptive Robotic Mirror System (ARMS), is essentially a Python implementation of the simulated mirror neuron system designed by Rebrová, Pecháč, and Farkăs [5], with some modifications. In this section, we first overview of ARMS and [5], and then give a more detailed description of merge self-organizing maps and bidirectional activation-based learning, two important components of the system.

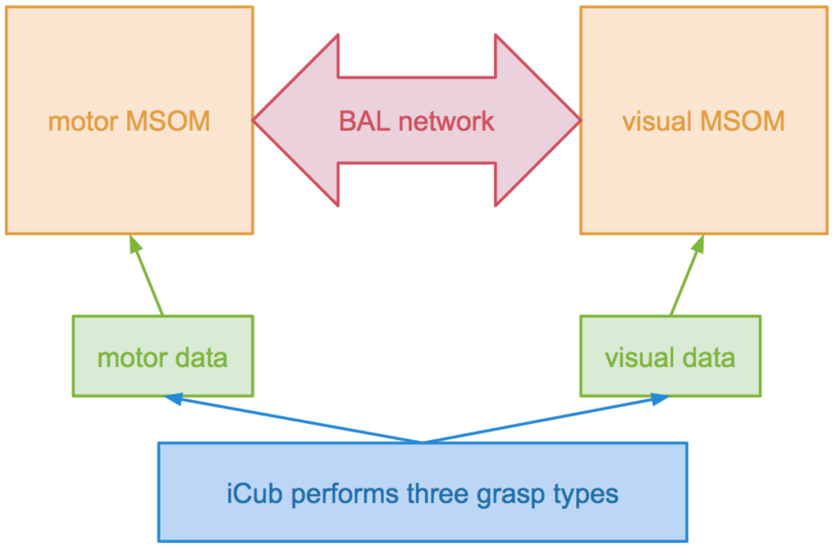

Figure 1: Schematic of the ARMS system.

ARMS and [5] share the same basic structure, which is illustrated in Figure 1. First, an iCub robot performs three different grasp types and gathers motor and visual data. Both ARMS and [5] use the standard iCub Simulator in place of an actual robot. The three grasp types are the power grasp, the precision grasp, and the side grasp, which are shown in Figure 2.

The motor and visual data from the iCub are then used to build merge self-organizing maps (MSOMs), a type of emergent representation for lowering the dimensionality of a data space. The motor MSOM constitutes the robot's internal representation of its movement, and can be thought of as a model for motor or sensorimotor cortex. The visual MSOM constitutes the robot's visual perception of its movement, and can be thought of as a model for visual cortex.

The MSOM activity for an instance of a grasp type is flattened and binarized (this process is described below), and used as input to a neural network that uses bidirectional activation-based learning (BAL) instead of traditional backpropogation. The BAL neural net can be run in two directions: motor to visual (forward), or visual to motor (backward). In the forward direction, the network is given a motor representation of a grasp and predicts the visual representation. In the backward direction, the network is given a visual representation and predicts the motor representation. Therefore, the BAL net is the part of the sysem that models the mirror neuron system.

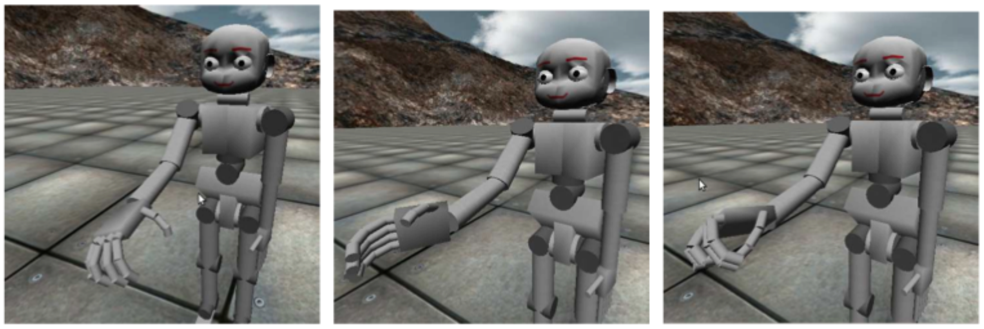

Figure 2: Simulated iCub performing a power grasp, a side grasp, and a precision grasp (L to R). Figure from [5].

There are several key differences between ARMS as presented here and the system designed by Rebrová, Pecháč, and Farkǎs [5]. First, [5] used a reinforcement learning algorithm to teach the grasps to the simulated iCub. Due to time constraints, we were unable to implement this part of the system, and so ARMS assumes that the three grasp types are hardcoded or previously learned.

Second, in [5], the training of MSOMs and the BAL neural net took place offline, using the data from the simulated iCub but not interacting with the robot in any other way. We found this to be unsatisfactory for two reasons. If a simulated mirror neuron system were to be implemented in an autonomous robot, it would clearly need to be part of the robot's control system. Also, it seems likely that, in primates, motor learning and the development of motor-visual associations takes place in conjunction. To this end, we had hoped to implement a system that alternated between stages of grasp learning and association learning. Unfortunately, time again prevented us from completely realizing this goal, so while ARMS has MSOM and BAL neural net training within the iCub controller, the association learning is still not truly online.

2.1 Merge Self-Organizing Maps

Both our system and [5] use merge self-organizing maps (MSOMs), which are based on self-organizing maps (SOMs). Like growing neural gas, SOMs learn a low-dimensional representation of a high-dimensional input distribution. However, unlike GNG, SOMs learn a fixed topology; typically a two or three-dimensional grid network. Each SOM neuron is associated with a weight vector in the high-dimensional input space which are initialized to small random values.

In the training stage, the SOM takes an example vector and individually computes its Euclidean distance from each of the neuron weight vectors. The most similar neuron vector and the weight vectors of all adjacent neurons are adjusted towards this example vector. A neighborhood function (typically Gaussian) weights the update to be inversely proportional to the distance from the best match neuron. Thus, the neurons of the SOM learn to respond to certain input vectors. Thus the neurons can potentially train a lower-dimensional representation of the input data, where each input example is represented by the response of the SOM's neurons to the example.

MSOMs (Merge Self-Organizing Maps) are a modification on SOMs that incorporate sequential information to the update rule. Instead of training on a set of input vectors, an MSOM trains on a set of input vector trajectories. In addition to a weight vector, each neuron keeps a context vector which is trained by combining the weight vector's response with the weight vector of the neuron that best matched the current input example in the previous timestep. Thus the response of each neuron varies in time, and the neurons learn to respond highly to inputs that are associated in time. [8]

Before MSOM training, each neuron is initialized to a small random value. In one iteration of training, each neuron undergoes the following update rules. Let w_v^i be the input weight vector for neuron v, and w_v^c be the context vector for neuron v. $u(t-1)$ denotes the best matching unit at time $t-1$. α and β are constant learning parameters, and $\Phi(u, v)$ is the neighborhood function between neurons u and v. The context descriptor for time t, $q(t)$, is

$$q(t) = (1-\beta)w^i_{u(t-1)}(t) + \beta w^c_{u(t-1)}(t)$$

Assume u is the best matching unit at time t, and γ is the learning rate. Then the update rules for the input vector and context vector are:

$$w_v^i(t) = w_v^i(t) + \gamma\Phi(u, v, t)(s(r) - w_v^i(t))$$

$$w_v^c(t) = w_v^c(t) + \gamma\Phi(u, v, t)(q(t) - w_v^c(t))$$

After the training stage, the scalar output of a neuron is calculated by combining that neuron's input vector and context vector. The output itself $y_v(t)$ is simply $y_v(t) = e^{-d_v(t)}$ where

$$d_v(t) = (1-\alpha||s(r) - w_v^i(t)||^2 + \alpha||q(t) - w_v^c(t)||^2.$$

To extract a useful representation of the input data from an MSOM after it has been trained, each input example is presented to the MSOM, and the response of each neuron over time is recorded. The responses can be flattened by adding up how many times each neuron responds to a particular input trajectory and saving that output map as the MSOM representation of that example. Since these outmaps will also be used as input to the BAL neural net, they must also be binarized, as described in §3.

2.2 Bidirectional activation-based learning

In the ARMS system and in [5], the association between motor and visual representations of an action is accomplished by a bidirectionally connected neural net, which learns using an algorithm called Bidirectional Activation-based Learning (BAL) [5, 2]. This algorithm is based on O'Reilly's Generalized Recirculation algorithm [4], which was designed to be a biologically plausible alternative to traditional error backpropagation. Error backpropagation is biologically implausible because it requires the computation of derivatives and uses global information not available at the synapse. Like Generalized Recirculation, BAL uses bidirectional activations and a two-phase update process to achieve error minimization while only using local variables and without computing error derivatives [2].

BAL uses a three layer network, which is fully connected in both the forward and backward directions. Instead of talking about input and output layers, we call the outer layers $\mathbf{x}$ and $\mathbf{y}$, since either can be used as the input. The hidden layer is called $\mathbf{h}$. In the forward phase, the stimulus $\mathbf{x}^F$ is given, and activation flows from the $\mathbf{x}$ layer to the $\mathbf{h}$ layer to the $\mathbf{y}$ layer. In the backward phase, the stimulus $\mathbf{y}^B$ is given, and activation flows from the $\mathbf{y}$ layer to the $\mathbf{h}$ layer to the $\mathbf{x}$ layer. The

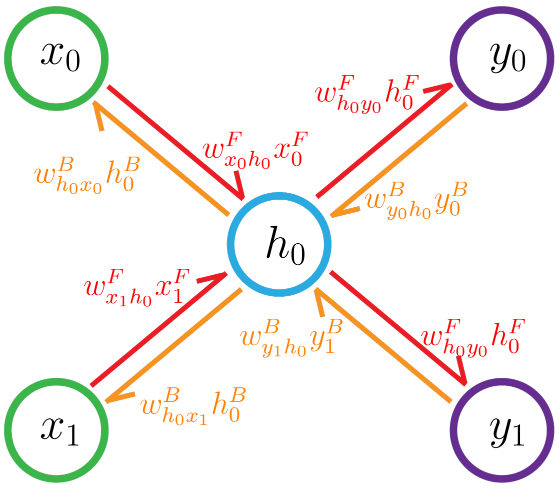

Figure 3: A sample BAL network, with forward activations shown in red and backward activations shown in orange. Each connection is labeled with the input the connection passes to the postsynaptic unit. In addition, each unit has a bias, which is omitted for simplicity.

forward and backward phases each have a set of weights, and each unit's activation is calculated by applying the sigmoid activation function to the unit's net input [2]. A sample network illustrating activation flow in the network is shown in Figure 3, and the net inputs and activations of each unit are summarized in Table 1.

Table 1: Net inputs and activations of units in each phase. Modified from [2].

Phase	Layer	Net Input	Activation
F	$\mathbf{x}$	-	x_i^F = stimulus
F	$\mathbf{h}$	$\eta_j^F = \sum_i w_{ij}^F x_i^F$	$h_j^F = \sigma(\eta_j^F)$
F	$\mathbf{y}$	$\eta_k^F = \sum_j w_{jk}^F h_j^F$	$y_k^F = \sigma(\eta_k^F)$
B	$\mathbf{y}$	-	y_i^B = stimulus
B	$\mathbf{h}$	$\eta_j^F = \sum_k w_{kj}^B y_k^B$	$h_j^B = \sigma(\eta_j^B)$
B	$\mathbf{x}$	$\eta_i^B = \sum_j w_{ji}^B h_j^B$	$x_i^B = \sigma(\eta_i^B)$

The network weights are updated using only local variables. Let a_p be the presynaptic unit activation and a_q be the post synaptic unit activation, with learning rate λ. Then the forward weights are updated according to the rule

$$\delta w_{pq}^F = \lambda a_p^F (a_q^B - a_q^F), \tag{1}$$

and the backward weights are updated according to the rule

$$\delta w_{pq}^B = \lambda a_p^B (a_q^F - a_q^B). \tag{2}$$

Each unit also has a bias, which is updated in the same way, but assuming a presynaptic input of 1. [2].

These update rules are similar to the weight update rules of General Recirculation - in fact, the forward weight update rule is identical - and a mathematical justification that the General Recirculation update rules minimize error is given in [4]. Experimental evidence that a BAL network can learn several basic neural network tasks is given in [2].

3 Experiments

We performed three variations on the same experiment. First, we generated ten variations on each of the three grasp types described in §2 (power, precision, side) and collected motor and visual data from the simulated iCub. Motor data was collected as the joint angles of the robot's arm at each time step. Visual data was collected as Cartesian coordinates of each joint with reference to the simulator.

These data were then used to train a motor MSOM and a visual MSOM. Both MSOMs were trained for 100 iterations. The motor MSOM had a learning rate of 0.08 and the visual MSOM had a learning rate of 0.05. The final representations of each input trajectory were then flattened and binarized. The binarization process worked as follows: for each grasp example, the maximum and minimum coordinate values in the flattened MSOM vector were identified, and the midpoint between them was calculated. Any coordinate less than this midpoint was set to 0, any coordinate above the midpoint was set to 1. This process corresponds to identifying a population code of selective neurons for each grasp.

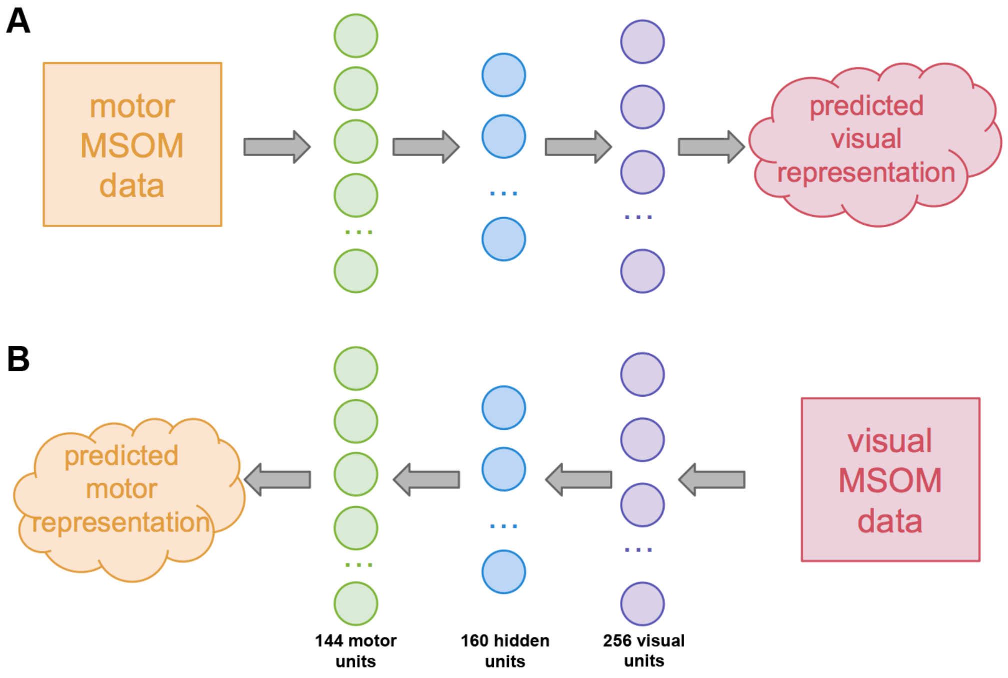

Figure 4: The testing phase of the experiments consists of (A) running the BAL network forward, to generate visual predictions, and (B) running the BAL network backward, to generate motor predictions.

These vectors were then used for training and testing the BAL network. For each grasp type, seven examples were used as training data and three were used for testing. After using the BAL algorithm to learn the testing data, the network was tested and evaluated as follows: for each grasp type, the binarized n-dimensional vectors representing all ten variations were used to calculate a sphere in n-dimensional space whose center was the mean of all ten examples and whose radius was the mean distance from that center to each vector. Each grasp in the test set was given as input to the BAL network, and the network's output was classified as a certain grasp type depending on which sphere it lay inside (if any) and which center it was closest to. Training and testing were both bidirectional processes, as shown in Figure 4.

The three experimental conditions differed in the size of MSOM used and the number of BAL training iterations performed. In the first condition, both motor and visual MSOMs were 3×3 and the BAL net had 9 motor units, 7 hidden units, and 9 visual units, and underwent 5,000 iterations of training. In the second condition, the motor MSOM was $12{\times}12$, the visual MSOM was 16×16, the BAL net had 144 motor units, 160 hidden units, and 256 visual units, and underwent 1,000 iterations of training. In the third condition, the motor MSOM had 144 motor units, 160 hidden units, and 256 visual units, and underwent 5,000 iterations of training.

4 Results

In all three experimental conditions, the ARMS system was unable to create robust associations between visual and motor representations of grasp types.

Table 2: Summary of results from the three experimental conditions. Each row represents a grasp type given as test input. Each column shows the classification of the network's output. Numbers in plain text indicate that those outputs lay within the sphere representing that grasp type; italicized numbers indicate that those outputs did not lie within any sphere but were closest to the center of the sphere corresponding with that grasp type.

		Forward (motor to visual)			Backward (visual to motor)		
		Power	Precision	Side	Power	Precision	Side
Small MSOMs, long training	Power	3 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(3)*
	Precision	3 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(3)*
	Side	3 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(0)*	0 *(3)*
Big MSOMs, short training	Power	0 *(0)*	0 *(0)*	3 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*
	Precision	0 *(0)*	0 *(0)*	3 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*
	Side	0 *(0)*	0 *(0)*	3 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*
Big MSOMs, long training	Power	0 *(3)*	0 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*	0 *(0)*
	Precision	0 *(3)*	0 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*	0 *(0)*
	Side	0 *(3)*	0 *(0)*	0 *(0)*	0 *(3)*	0 *(0)*	0 *(0)*

In the first condition, which used small ($3{\times}3$) MSOMs and 5,000 iterations of BAL training, the visual prediction output was identical for all nine test inputs in the forward direction, and motor prediction output was similar but not identical in the backward direction. In the forward direction, the constant output was classified as a power grasp. In the backward direction, the outputs did not lie within any of the spheres representing the three grasps, but were closest to the center of the side grasp sphere.

In the second condition, which used large MSOMs (motor $12{\times}12$; visual $16{\times}16$) and 1,000 iterations of BAL training, predicted outputs were nearly identical in both the forward and backward direction. In the forward direction, all outputs were classfied as side grasps. In the backward direction, outputs did not lie within the spheres representing any of the three grasp types, but were closest to the center of the precision grasp sphere.

In the third condition, which used large MSOMS and 5,000 iterations of BAL training, predicted outputs were nearly identical in both directions. Outputs did not lie within any of the spheres representing the grasp types in either direction. In both directions, outputs were closest to the center of the power sphere.

These results are summarized in Table 2. It is clear that ARMS failed to learn any association between visual and motor representations of grasps. Possible reasons for this failure are discussed in the next section.

5 Discussion and future directions

Although the ARMS system and our experiments were very closely based on [5], our results failed to show any learned associations between motor and visual representations of grasps. Some differences in results are to be expected due to our differing implementations, but that alone does not explain the magnitude of our failure.

We believe that much of our results can be explained by lack of time. The fact that the BAL network gives nearly the same output for any input suggests that it is under-trained. In preliminary experiments to test our BAL implementation (data not shown), we saw the same phenomenon in networks that simply hadn't been trained enough. However, running our third experimental condition, where the BAL net had 144 motor units, 160 hidden units, and 256 visual units, and underwent 5,000 iterations of training, already took over 48 hours. Training the large MSOMs also took significant amounts of time. Because of this, we were simply unable to perform enough experiments to determine the optimal number of hidden units or training iterations. We are hopeful that ARMS would perform better when given even more time for training both the MSOMs and the BAL neural net.

Another future direction we are interested in is modifying the system so that the development of mirror neuron associations and motor learning to take place in parallel, as likely happens in biological mirror neuron systems. We would like to see a system where phases of observation, imitation, and association-building happen in alternation, so that the robot builds the associations between motor and visual representations at the same time as it builds those motor and visual representations. We believe this would increase the biological plausibility of the model as well as its robustness and efficiency.

6 Acknowledgements

We are enormously grateful to Kristína Rebrová, Matej Pecháč and Igor Farkaš for sharing their code and explaining their work to us, without which our project wouldn't have been possible.

References

[1] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and autonomous systems*, 54(5):361–369, 2006.

[2] Igor Farkaš and Kristína Rebrová. Bidirectional activation-based neural network learning algorithm. In *Artificial Neural Networks and Machine Learning–ICANN 2013*, pages 154–161. Springer, 2013.

[3] Roy Mukamel, Arne D Ekstrom, Jonas Kaplan, Marco Iacoboni, and Itzhak Fried. Single-neuron responses in humans during execution and observation of actions. *Current biology*, 20(8):750–756, 2010.

[4] Randall C O'Reilly. Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural computation*, 8(5):895–938, 1996.

[5] Kristina Rebrova, Matej Pechac, and Igor Farkas. Towards a robotic model of the mirror neuron system. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pages 1–6. IEEE, 2013.

[6] Giacomo Rizzolatti and Laila Craighero. The mirror-neuron system. *Annu. Rev. Neurosci.*, 27:169–192, 2004.

[7] Ryo Saegusa, Giorgio Metta, Giulio Sandini, and Lorenzo Natale. Developmental perception of the self and action. 2013.

[8] Marc Strickert and Barbara Hammer. Merge som for temporal data. *Neurocomputing*, 64:39–71, 2005.