Acquiring Social Interaction Behaviours for Telepresence Robots via Deep Learning from Demonstration

Kyriacos Shiarlis[1], João Messias[1], and Shimon Whiteson[2]

***Abstract*— As robots begin to inhabit public and social spaces, it is increasingly important to ensure that they behave in a socially appropriate way. However, manually coding social behaviours is prohibitively difficult since social norms are hard to quantify. Therefore, *learning from demonstration* (LfD), wherein control policies are inferred from demonstrations of correct behaviour, is a powerful tool for helping robots acquire social intelligence. In this paper, we propose a deep learning approach to learning social behaviours from demonstration. We apply this method to two challenging social tasks for a semi-autonomous telepresence robot. Our results show that our approach outperforms *gradient boosting regression* and performs well against a hard-coded controller. Furthermore, ablation experiments confirm that each element of our method is essential to its success.**

I. Introduction

In recent years, robots have been migrating out of conventional, constrained, industrial environments and into more social and less structured ones such as museums [1], airports [2], restaurants [3] and care centres [4]. This shift poses countless challenges for robotics, from hardware to high-level behaviour design. A particularly vexing challenge is to imbue robots with sufficient *social intelligence*, i.e., to ensure that they respect social norms when interacting with humans or completing tasks in the presence of humans. For example, a robot facing a group of people must maintain an appropriate distance, orientation, level of eye contact, etc.

Because socially acceptable behaviour is difficult to describe procedurally, it is typically infeasible to manually code social intelligence. Furthermore, even traditional approaches to learning behaviour, such as *reinforcement learning*, are not practical because social norms are too difficult to quantify in the form of a reward or cost function.

A compelling alternative is *learning from demonstration* (LfD) [5], wherein control policies are inferred from example demonstrations of correct behaviour. Because demonstrating socially appropriate behaviour is typically straightforward for humans, obtaining the data needed to learn social intelligence from demonstration is highly feasible. In addition, many different behaviours can be learned with the same LfD algorithm, allowing non-roboticists to program personalised robotic behaviours simply by demonstrating the task.

However, to be practical for real robots, LfD must overcome some challenges of its own. First, it must learn control policies that are robust to inevitable errors in the robot's perception system. For example, a social robot must be able to detect and localise people in its environment. If errors in such perception occur while demonstration data is being collected, then the LfD algorithm may misinterpret the demonstration, e.g., by inferring that the robot was avoiding a person when actually no person was present. Furthermore, if perception errors occur when the learned policy is deployed, then the robot may behave incorrectly [6].

(a) (b)

Fig. 1: The two social interaction tasks we consider in this paper. (a) The group interaction task: the robot should reconfigure itself to facilitate conversation in a group. (b) The following task: the robot should follow a group of people. These pictures were taken during a human evaluation session where DBSoC was compared to different baselines.

A second challenge is to cope with the dynamic size of the perceptual input. A robot that interacts with a group of people must be able to cope with groups of different sizes, and even people coming and going within a single conversation. This is a poor match for most LfD methods, which assume observations in the form of fixed-length feature vectors.

Third, in order to be useful for non-experts, the LfD algorithm must be sufficiently generic. In other words, it should be able to learn a wide set of behaviours with a single algorithm, by varying only the data.

In this paper, we propose *deep behavioural social cloning* (DBSoC), a new LfD architecture that overcomes these challenges. *Behavioural cloning* is an approach to LfD in which a control policy is directly inferred using supervised learning, i.e., the demonstrated actions are treated as labels for the corresponding observations, and a mapping from observations to actions is learned. By leveraging the power of deep learning, DBSoC makes behavioural cloning practical for social robotics. In particular, DBSoC 'paints' the output of the robot's person detection and localisation system onto a 2D 'image', which is then fed to a convolutional neural network. In this way, DBSoC can cope elegantly with varying

[1]Informatics Institute, University of Amsterdam, The Netherlands, {k.c.shiarlis,j.messias}@uva.nl

[2]Department of Computer Science, University of Oxford, United Kingdom, shimon.whiteson@cs.ox.ac.uk

numbers of people, as well as automatically discover useful features for selecting actions. In addition, DBSoC employs *long short-term memory* (LSTM) [7] network layers, allowing it to condition action selection on its history of observations and thus filter out transient perceptual errors.

We apply DBSoC to learn two social behaviours for a semi-autonomous holonomic telepresence robot. In the first task, the robot must continually adjust its position and orientation while interacting with a group of people of dynamic size and position. In the second task, the robot must follow two people as their relative positions change.

We evaluated DBSoC on these two tasks by deploying the learned policies on a real telepresence robot and having it interact with actual human subjects. Our quantitative and qualitative results show that DBSoC performs well against a naive controller and outperforms a *gradient boosting regression* baseline. Furthermore, ablation experiments confirm that each element of DBSoC is essential to its success.

II. Problem Statement

In this paper, we consider two social tasks for a telepresence robot. Such a robot acts as an avatar for a remotely present *pilot* controlling the machine. While telepresence robots are typically manually controlled, we aim to automate low-level control so that the pilot need only give high-level commands telling the robot, e.g., to follow a particular person or support a conversation with appropriate body poses. Because telepresence robots are inherently social, and especially since they represent their pilots as avatars, it is essential that any learned behaviour be socially appropriate. In this section, we describe the two particular telepresence tasks considered in this paper, and the demonstration data we collected. For both tasks, we assume the robot is equipped with an imperfect person detection and localisation system. We also assume it is operating in an unstructured environment containing a dynamic number of people, some of whom may not be involved in the pilot's conversation.

A. Group Interaction Task

In the first task, the robot must maintain a socially appropriate position and orientation as the location, position, and size of the group with which the pilot is interacting changes. To do so, the robot must distinguish people who are part of the conversation from those who are merely in the scene. Though this task, shown in Figure 1a, looks deceptively simple, manually coding such behaviour, or even deciding on what features such behaviour should condition, is quite difficult in practice, making it an ideal application for LfD.

Previous work has considered what effect robot repositioning following a change in a group's size and shape has on the people in that group [8], [9]. Here, we consider how to automate such repositioning using LfD. Other work considers detecting groups of people in crowded environments [10] and then interacting with them. In our approach, groups are not explicitly detected, but the learned control policy may implicitly do so when deciding how to act.

B. Following Task

In the second task, the interaction target(s) are moving and the robot should follow them. The robot should be able to deal with variability in the formation of the people it is following. Furthermore, it should be robust to false positive detections and be able to retain a following behaviour despite detecting other people who are in the scene but not relevant to the task. This task is shown in Figure 1b. Following behaviours can be challenging because the robot's sensors often suffer from reliability issues as velocities increase [11].

Previous work used an extended social force model enriched with a goal prediction module to allow a robot to walk side by side with a person [12]. The system tests different social forces parameters and allows the user to give feedback about the experience. Based on this feedback, the robot learns to adapt itself to the specific person. Our work investigates the possibility of bypassing such models and simply learning the required behaviour robustly through data.

Knox et al. [13] train a robot to perform several tasks that are similar to ours, such as maintaining a conversational distance and "magnetic control". However, training is done using human generated reward and reinforcement learning through the TAMER framework [14]. Furthermore, in contrast to our work, their approach can handle only a single interaction target and assumes perfect sensing.

More generally, in social robotics settings, LfD has been used to learn high level behaviours [15], [16] and cost functions for social path planning through the use of *inverse reinforcement learning* (IRL) [17], [18]. In this work, we show how low level reactive social policies can be learned from demonstration. Outside of social settings, LfD is most commonly used for complex manipulation tasks [5]. Deep learning has also been used in an end-to-end fashion to perform behavioural cloning in self-driving cars [19]. However, such an approach is highly intensive in terms of both data and computation. By contrast, we rely for people detection on a sensor fusion pipeline that uses both the RGBD camera and the laser rangefinders of the robot to enable 360° tracking. Consequently, we require only a modest amount of demonstration data (approximately one hour), perform all processing on the robot, and cope successfully with the robot's limited RGBD field of view.

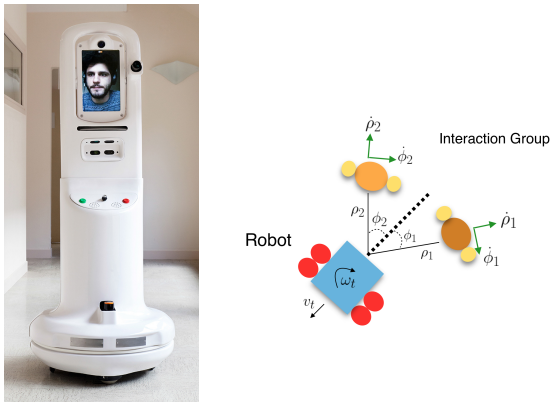
C. Data and Experiments

To collect the data required for learning, we performed one set of experiments for each of the tasks described above. Every experimental session involved two experimenters, one or more volunteers from the University of Amsterdam, and a TERESA[1] intelligent telepresence system, shown in Figure 2a. One experimenter acted as the pilot while the other acted as an interaction target. The interaction target also instructed the volunteers to join or leave the interaction and change the shape of the interaction group (triggering an appropriate reaction from the pilot) in order to generate sufficient variability in the data. For the follow task, the two

[1] `http://www.teresaproject.eu`.

experimenters walked with the robot, making sure that at least one of them was tracked at all times, in order to prevent false data from being recorded.

During these interactions, we recorded the positions and linear velocities of all people detected and tracked by the robot. We also recorded a binary label, indicating the primary interaction target for the robot, i.e., the most important person in the interaction, as indicated by the pilot. The observation o_t of K people tracked at time t, where the first person is the primary interaction target, is therefore defined as, $o_t = \{(\rho_1, \phi_1, \dot{\rho}_1, \dot{\phi}_1, 1), ...(\rho_{,}\phi_{K_t}, \dot{\rho}_{K_t}, \dot{\phi}_{K_t}, 0)\}_t$, where ρ_k and ϕ_k are the distance and angle from the robot respectively and $\dot{\rho}_k$ and $\dot{\phi}_k$ represent the associated velocities (see Figure 2b). We also recorded the linear and angular velocity commands given by the pilot, $a_t = (v_t, \omega_t)$, where a_t denotes the action at time t. Recording took place at 10Hz, which is the control rate the robot uses. Data was gathered in multiple different environments, each containing different numbers of people and different sources of false positives for people detection. We collected in total 43512 and 27311 data points for the following task and group interaction tasks, respectively. This amounts to approximately two hours of data.



(a) (b)

Fig. 2: (a) The TERESA telepresence robot used in our experiments; (b) the data collected for a single frame (t) during our experiments.

III. Method

In this section, we describe deep behavioural social cloning (DBSoC), the learning architecture used to learn a mapping from robot observations to actions for the tasks described above. We first describe our behavioural cloning approach and motivate the use of neural networks in this application. We then build the network architecture piece by piece by looking at different aspects of the data such as input representations, temporal aspects, and output mappings.

A. Behavioural Cloning

Behavioural cloning is a simple approach to LfD that treats the problem as one of supervised learning. The (sequences of) observations are viewed as inputs s_t and actions a_t as labels. Then, any supervised learning paradigm (regression, classification) and algorithm (decision trees, Gaussian processes, neural networks) [20] can be used to learn a mapping from s_t to a_t, which then constitutes a control policy. As mentioned in Section II an alternative to BC is IRL [17], which extracts a cost function from the demonstrations that is subsequently used to plan (or learn [21]) a policy. IRL however, usually requires a model (or simulator) of the social environment, which is hard to obtain in this case. Behavioural cloning is simpler, more practical, and does not require access to a model or the ability to try out candidate policies during learning.

DBSoC uses behavioural cloning with *neural networks* (NNs). Using NNs has two key advantages in this setting. First, given the need for a generic algorithm that can tackle multiple tasks, NNs can learn features, e.g., that can detect if someone is part of a group, that can be reused across related tasks. Second, NNs are modular, allowing us to swap in different modules depending on the high-level application. For example, we describe below how we use recurrent neural networks to deal with noisy perceptual input.

B. Inputs and Convolutional Layers

An important characteristic of our observation vector o_t is that its length depends on the number of people around the robot, which varies over time. This is problematic for many learning algorithms, which expect a fixed-length input vector. One way to alleviate this problem would be to fix K to some reasonable constant. If the number of people in the data is more than K, only the K closest are kept. If the number is smaller than K, then only the first K positions of the vector are filled and the rest set to zeros. However, this implies that one or more people are in the same location as the robot, since $\rho, \theta = (0, 0)$, which could mislead the learning algorithm. Another option would be to add a binary feature to our observation vector to indicate whether the detection is a valid one, yielding, e.g.:

$$o_t = \{(\rho_1, \phi_1, \dot{\rho}_1, \dot{\phi}_1, 1, 1), ...(\rho_{K_t}, \phi_{K_t}, \dot{\rho}_{K_t}, \dot{\phi}_{K_t}, 0, 0)\}_t \quad (1)$$

While this representation accurately describes the observation, it is not convenient for learning. Forcing the algorithm to learn to ignore observations about people whose last bit is zero is an unnecessary additional complication. In addition, the representation ignores crucial symmetries, e.g., rearranging the order of the people in the vector yields a completely different input that describes exactly the same state.

To avoid these difficulties, we instead imprint the information in the observation onto a 2D image. In particular, we fix a maximum distance x_{max}, y_{max} from the robot and 'paint' all the people around the robot within that area on a ternary image, i.e., each pixel can have one of three values: $0 =$ no person is present, $1 =$ a person is present, $2 =$ a primary interaction target is present. We also denote the velocities of people as lines extending from the detections in the direction of the detected velocity and with a length corresponding to its magnitude. Figure 3 shows an example of such an image.

It is straightforward to augment this representation with, e.g., people's orientations or the positions of non-human obstacles, should they be detectable.

A potential downside of this approach, however, is a great increase in the dimensionality of our representation. We tackle this problem by applying *convolutional neural networks* (CNNs) to this input representation. CNNs perform very well on spatially structured images. Although the dimensionality of the input is large, the number of possible images is limited. CNNs have also been proven very effective in order to automatically extract high-level features from the raw pixel data. This in turn means that they would be better suited in building an implicit representation of what a group is. Our results in the next section show that in practice this representation together with CNNs is not only more convenient, but also improves learning and the stability of the resulting behaviour.

Fig. 3: Example images representing the state of the robot. Circles denote people, lines denote velocity, and the white circle represents the primary interaction target.

C. Dynamics and Long Short-Term Memory

Using a single observation as input to a convolutional neural network that outputs a control signal is problematic for three reasons:

1) A single observation instance o_t does not contain higher order information such as acceleration.
2) The people detection module is imperfect and generates some false positives. Conditioning only on the most recent observation makes it impossible to filter such false positives out.
3) Velocity estimates, while helpful, are also inaccurate and noisy, and cannot be improved using only the most recent observation.

These problems can be avoided by using *recurrent neural networks* (RNNs), which implicitly allow the network to condition on its entire history of inputs. *Long short-term memory* [7] is a standard way of implementing recurrence in NNs. Our architecture uses LSTMs to deal with the time-dependent nature of the problem and achieve robustness in the face of uncertain detections.

D. Outputs and Overall Architecture

After processing by the recurrent layers, the architecture then outputs two real continuous values representing the linear and angular velocities $(v_{pred}, \omega_{pred})$ of the robot. An alternative would be to discretise the outputs and treat the prediction as a classification problem. While doing so has some benefits, e.g. allowing multimodal output distributions, we chose regression for two reasons. First, since the right discretisation is problem specific, a regression approach is more general. Second, continuous outputs can generate smoother behaviour.

To train our network, we use backpropagation with the ADAM [22] optimiser to minimise the *mean squared error* (MSE) between the network's prediction and the data.

Figure 4 shows the complete resulting architecture. We employ three convolutional layers consisting of 10 3×3 filters at each layer. We use ReLU nonlinearities followed by max-pooling of size 2. The output from the last convolutional layer is flattened and processed through two LSTM layers of size 200 and 100 respectively. Finally, a densely connected layer is used to output the two required control values. Because the robot selects action s_t at 10Hz, the trained network must be able to forward propagate at that rate. We found that our network could do so comfortably on an Intel i7 processor.

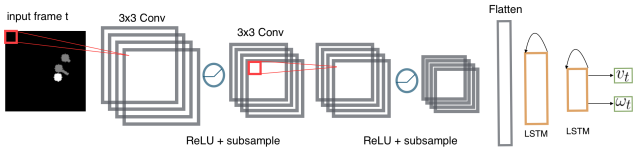


Fig. 4: The complete DBSoC architecture. The image representing the state passes through two convolution-nonlinearity-subsample layers before being fed through two LSTM layers that output the control signals $(v_{pred}, \omega_{pred})$.

E. Preventing Covariate Shift

A well known failure mode of BC is that of covariate shift. This arises because while policy is trained on a dataset that is assumed to be stationary, it is deployed in an inherently non-stationary environment. A policy trained using BC thus does not recover easily from states not seen in the training data and may drift. This phenomenon motivates methods such as DAGGER [23], where the expert policy is queried for the right action at different instances of learning. Although in our setting DAGGER would be hard to apply, we found it was essential to train the robot in a similar manner. Specifically, we performed training sessions iteratively. After a partial data collection, the robot was trained and the policy deployed. During deployment, data of correcting actions was collected whenever the robot behaved inappropriately. The dataset was augmented with this data and the process was repeated.

IV. Evaluation and Results

We evaluate DBSoC by comparing it against three other learning baselines. The first is an off-the-shelf regression method. Due to the mixed types in the inputs, we use *gradient boosting regression* (GBR), which is considered one of the best off-the-shelf algorithms for nonlinear regression. We use this baseline to show that, for traditional regression methods to work well, task-specific feature engineering would need to be performed. Inputs to this baseline are the concatenated observation vectors for K people around the robot (as described in Section III-B) for T timesteps. In this

case, $K = 4$ and $T = 5$ so the input vector representations is $K \times T \times 6 = 120$. We did not use $T = 20$ as DBSoC does because the explosion in input dimensionality caused a degradation in performance. The second baseline is a deep architecture that omits the convolutional layers and only employs the LSTM layers. The input representation is a vector concatenating the features of $K = 4$ people around the robot. The third baseline is an architecture with no memory that uses only the convolutional layers.

When evaluating these algorithms, we are concerned not only with minimising error on the data but also with how they perform when placed on the robot and tested with humans. Therefore, our evaluation relies on a mixture of quantitative and qualitative metrics.

A. Mean Squared Error

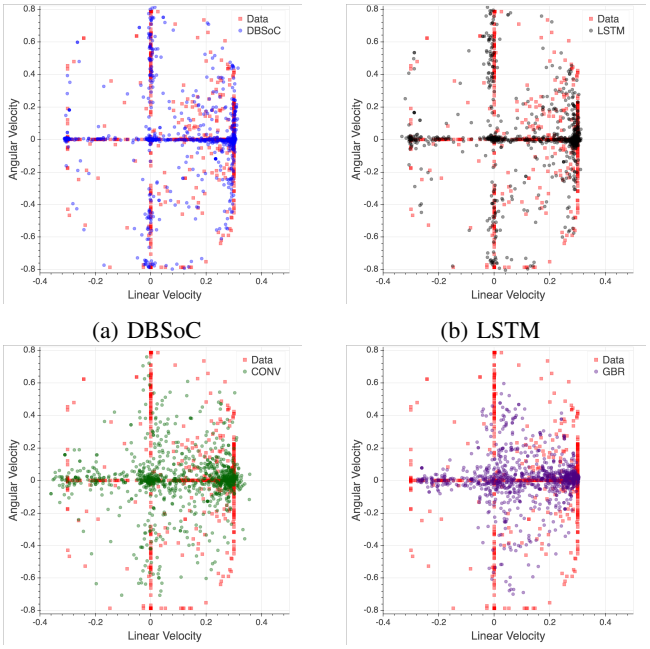
Since the algorithms are trained to minimise MSE, the first performance measure we consider is MSE on both the training set and a test set not used during learning. In our case, the test set contained of 20% of the data for each task, which amounts to 8702 and 5462 data points for the follow and group interaction tasks, respectively. Table I shows the performance of each method on both the training and test sets. We can see that DBSoC outperforms all baselines on the test set, followed by the LSTM method (where no image representation is used). GBR is the worst performer by a significant margin.

		GBR	LSTM	CONV	DBSoC
Group	Train	0.0232	0.0098	0.0087	**0.0018**
	Test	0.0264	0.0120	0.015	**0.0031**
Follow	Train	0.0265	0.00334	0.0079	**0.0030**
	Test	0.0323	0.00450	0.0250	**0.0036**

TABLE I: Train and test MSE for DBSoC and the baselines.

B. Output Distribution

A more qualitative yet insightful means of evaluation can be achieved by examining the distribution of the outputs of each learned policy, conditioned on the inputs. That is, using a portion of the test data, we plot the predicted linear and angular velocities $(v_{pred}, \omega_{pred})$ and compare them with the actual values encountered in the data (v_{dem}, ω_{dem}). This gives an indication of how the output distribution of the predictor matches that of the human demonstrator. This can be more insightful than a single metric value because it sheds some light on the expected behaviour of the robot. Figure 5 overlays (v_{dem}, ω_{dem}) (blue) with $(v_{pred}, \omega_{pred})$ for our method and the baselines for the follow task. From this figure it is clear that GBR captures the demonstrated distribution quite poorly. Its behaviour is expected to not use large angular velocities (y-axis) and will thus be not very responsive. In addition, there is a lot of variance around the origin, possibly leading to oscillatory behaviour between positive and negative velocities. The CONV baseline also seems unable to capture the distribution, as we see many points that are scattered and far away from any data point, so we can expect it to behave poorly too. The LSTM policy on the other hand seems to have similar capabilities to DBSoC, which clearly does a good job at matching the demonstrator's outputs.



(a) DBSoC (b) LSTM

(c) CONV (d) GBR

Fig. 5: Output distributions of the follow data (red) and each of the four learned policies.

C. Human Evaluation

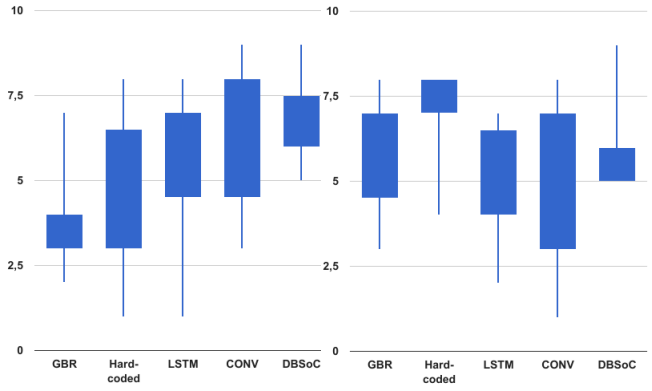
Since the goal of DBSoC is to learn social behaviour, it is essential to evaluate it using human subjects. Furthermore, the policy can only be properly assessed by deploying it on an actual robot. A key reason for this is that behavioural cloning treats learning as a supervised problem, but the learned policy is ultimately deployed in a sequential decision making setting. It is thus crucial to actually deploy the policy and verify that the learned actions do not carry the robot to undesirable states.

Our human evaluation consisted of 5 groups and a total of 15 subjects from the University of Amsterdam interacting with the learned policies as well as a hard-coded controller. This hard-coded behaviour interprets the person closest to the robot as the sole interaction target and attempts to keep a constant distance (ρ) of $1.3m$ from them, in accordance with standard proxemics [24], and an orientation (ϕ) of $0°$. The subjects were first briefed about the purpose of each behaviour they were about to evaluate (i.e., follow or group conversation). Following the briefing was a calibration step where the subjects were shown human-level performance in each of the tasks as well as various policies at random. This allowed the subjects to get a feel for what to expect from the robot. After calibration, the subjects were given approximately two minutes to interact with each policy for each of the tasks. The policies were given anonymously and in a different order each time. The subjects were then given the opportunity to rate the social performance of each policy with a number from 1 to 10, with 10 being human-level

performance. Examples of the tasks and policies in such an evaluation session can be seen in the associated video.[2]

A summary of the scores for each policy is shown as box plots in Figure 6. The results show that different methods perform quite differently depending on the task. For example, the hard-coded controller is perceived to perform well for the follow behaviour—due to the smoothness of its control—whereas in the group interaction task that controller performs poorly. A common comment from subjects was that the hard-coded controller only concentrated on one person and did not adjust to the group. A perhaps more surprising result in that humans perceive GBR to be a decent policy on average for the following tasks. These ratings are quite noisy, however, because the robot's behaviour varied a lot depending on the situations that arose during interaction. DBSoC performs quite well during all evaluations. Remarkably, it never received a score lower than 5, whereas most of the other methods did. Furthermore, when performance is averaged across tasks, it performed the best of all methods, showing greater generality than the baselines. However, it does not perform quite as well as the hard-coded controller on the follow task. This was mainly because control was less smooth for DBSoC, as it was for all the learning baselines.

A more comprehensive evaluation using many more subjects and a more detailed questionnaire would be necessary to reach definitive conclusions. Nonetheless, these results provide some confirmation of the generality and effectiveness of DBSoC. Furthermore, the fact that the learned policies were positively evaluated by subjects who were allowed to interact freely with the robot (rather than having to follow a rigid script), shows that DBSoC has learned useful policies that generalise beyond the data on which they were trained.



(a) Group Interaction (b) Following

Fig. 6: Box plots summarising human evaluation scores for all baselines on the two tasks.

V. Conclusions and Future Work

In this paper, we introduced DBSoC, a generic neural network architecture that allows learning from demonstration for social robotic tasks. We considered two social tasks: group interaction and following. Our architecture was able to learn both tasks well without any programming, demonstrating its generality. Its performance was confirmed quantitatively and also from human evaluation sessions where the learned policies were deployed on a real telepresence robot. In the future, we aim to use reinforcement and inverse reinforcement learning along with behavioural cloning in order to further improve the quality as well as the range of behaviours that can be learned.

[2] http://bit.ly/2vqo9Vv

References

[1] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, Dellaert *et al.*, "Minerva: A second-generation museum tour-guide robot," in *Robotics and automation, 1999.*, vol. 3. IEEE, 1999.

[2] R. Triebel, K. Arras, R. Alami, L. Beyer, S. Breuers, R. Chatila, M. Chetouani, Cremers *et al.*, "Spencer: A socially aware service robot for passenger guidance and help in busy airports," 2015.

[3] Y. Qing-xiao, Y. Can, F. Zhuang, and Z. Yan-zheng, "Research of the localization of restaurant service robot," *International Journal of Advanced Robotic Systems*, vol. 7, no. 3, p. 18, 2010.

[4] K. Shiarlis, J. Messias *et al.*, "TERESA: a socially intelligent semi-autonomous telepresence system," 2015.

[5] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[6] K. Dautenhahn and C. L. Nehaniv, *Imitation in animals and artifacts*. MIT Press Cambridge, MA, 2002.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] H. Kuzuoka, Y. Suzuki, J. Yamashita, and K. Yamazaki, "Reconfiguring spatial formation arrangement by robot body orientation," in *International conference on Human-robot interaction*, 2010.

[9] J. Vroon, M. Joosse, M. Lohse, J. Kolkmeier, J. Kim, K. Truong, G. Englebienne, D. Heylen, and V. Evers, "Dynamics of social positioning patterns in group-robot interactions," in *RO-MAN*, 2015.

[10] B. Lau, K. O. Arras, and W. Burgard, "Multi-model hypothesis group tracking and group size estimation," *International Journal of Social Robotics*, vol. 2, no. 1, pp. 19–30, 2010.

[11] M. Kobilarov, G. Sukhatme, Hyams *et al.*, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *ICRA*, 2006.

[12] G. Ferrer, A. G. Zulueta, F. H. Cotarelo, and A. Sanfeliu, "Robot social-aware navigation framework to accompany people walking side-by-side," *Autonomous Robots*, pp. 1–19, 2016.

[13] W. B. Knox, P. Stone, and C. Breazeal, "Training a robot via human feedback: A case study," in *ICSR*. Springer, 2013, pp. 460–470.

[14] W. B. Knox and P. Stone, "Interactively shaping agents via human reinforcement: The TAMER framework," in *K-CAP*. ACM, 2009.

[15] W.-Y. G. Louie and G. Nejat, "A learning from demonstration system architecture for robots learning social group recreational activities," in *IROS, 2016 IEEE/RSJ International Conference on*. IEEE.

[16] A. Lockerd and C. Breazeal, "Tutelage and socially guided robot learning," in *Intelligent Robots and Systems, 2004.(IROS 2004).*, 2004.

[17] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *ICML*. ACM, 2004, p. 1.

[18] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *ICRA*, 2010.

[19] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv:1604.07316*, 2016.

[20] C. M. Bishop, "Pattern recognition," *Machine Learning*, 2006.

[21] A. Boularias, J. Kober, and J. Peters, "Relative entropy inverse reinforcement learning." in *AISTATS*, 2011, pp. 182–189.

[22] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[24] E. T. Hall, "The hidden dimension," 1966.

This work is funded by the EC-FP7 under grant agreement no. 611153 (TERESA). We would like to thank all the subjects involved in the data collection and evaluation sessions for their time and feedback.