Deep Q-learning for Active Recognition of GERMS: Baseline performance on a standardized dataset for active learning

Mohsen Malmir[1]
mmalmir@eng.ucsd.edu
Karan Sikka[1]
ksikka@eng.ucsd.edu
Deborah Forster[1]
dforster@ucsd.edu
Javier Movellan[2]
Javier@emotient.com
Garrison W. Cottrell[3]
gary@eng.ucsd.edu

[1] Machine Perception Lab.
University of California San Diego,
San Diego, CA, USA

[2] Emotient, Inc.
4435 Eastgate Mall, Suite 320,
San Diego, CA, USA

[3] Computer Science and Engineering Dept.
University of California San Diego,
San Diego, CA, USA

Abstract

In this paper, we introduce GERMS, a dataset designed to accelerate progress on active object recognition in the context of human robot interaction. GERMS consists of a collection of videos taken from the point of view of a humanoid robot that receives objects from humans and actively examines them. GERMS provides methods to simulate, evaluate, and compare active object recognition approaches that close the loop between perception and action without the need to operate physical robots. We present a benchmark system for active object recognition based on deep Q-learning (DQL). The system learns to actively examine objects by minimizing overall classification error using standard back-propagation and Q-learning. DQL learns an efficient policy that achieves high levels of accuracy with short observation periods.

1 Introduction

Active object recognition (AOR) refers to problems in which an agent interacts with the world (e.g., via commands to servo motors of a robotic arm) and controls its sensor parameters (e.g., camera orientation, gain, sensitivity) to maximize the speed and accuracy with which it recognizes objects. A wide range of approaches have been developed since Wilkes and Tsotsos' [19] pioneering work. These approaches are designed to re-position sensors or change the environment so that the new inputs to the system become less ambiguous [1, 2] with respect to goals such as 3D reconstruction, localization or recognition of objects (see [16] for a comprehensive literature review).

Active object recognition systems include two modules: A recognition module and a control module. Given a sequence of images, the recognition module produces a belief state

about the objects that generated those images. Given this belief state, the control module produces actions that will affect the images observed in the future. The controller is typically designed to improve the speed and accuracy of the recognition module. One of the earliest active systems for object recognition was developed by Wilkes and Tsotsos [19]. They used a heuristic procedure to bring the object into a "standard" view by a robotic-arm-mounted camera. In a series of experiments on 8 Origami objects, they qualitatively report promising results for achieving the standard view and retrieving the correct object labels. Seibert and Waxman explicitly model the views of an object by clustering the images acquired from the view-sphere of the object [18]. The correlation matrices between these are then used to predict the correct object label. Using three model aircraft objects, they show that the belief over the correct object improves with the number of observed transitions compared to randomly generated paths on the view sphere of these objects.

Since these pioneering efforts, more theoretically-motivated approaches have attempted to optimize an objective function, for example the conditional entropy $H(O|M)$ between the original object O and the observed signal M, the expected entropy loss over actions, the belief uncertainty, or simply the variance among the object representations[3, 4, 5, 17]. Paletta & Pinz's work [15] is probably the most similar to our proposed model, as they treat active object recognition as a reinforcement learning problem, using Q-learning to find the optimal policy. They used an RBF neural network with the reward function depending on the amount of entropy loss between the current and the next state. Finally, the architecture for our work was inspired in part by recent work using a DCNN for representation of images in the context of learning to play Atari games with reinforcement learning[11].

A common thread in many of these approaches is the use of small, sometimes custom-designed sets of objects. One exception by Schiele and Crowley[17] used the COIL-100 dataset for their experiments, which consists of 7200 images of 100 toy objects rotated in depth [14]. This dataset is appealing for active object recognition because it provides systematically defined views of objects. However it is not an adequately challenging dataset for several reasons, including the simplicity of the image background, and the high similarity of different views of the objects due to single-track recording sessions. Indeed, by selecting the two most discriminative views of each object, Schiele and Crowley achieved almost perfect recognition accuracy.

This paper makes two main contributions: First, we present and make publicly available the GERMS dataset [1], that was specifically developed for active object recognition. The goal of this dataset is to accelerate progress on active object recognition by providing a common framework for evaluation of active object recognition algorithms. The dataset can be used to simulate the effect of robot actions without the need to have access to the physical robot. Second, we propose an architecture (DQL) for AOR based on deep Q-learning. Deep Q-learning supports learning the optimal policy for action selection from raw images [11]. The proposed model sets a performance baseline on the GERMS dataset that can be improved upon by other research groups. Although there is existing work that utilizes deep convolutional neural networks (DCNN) for mapping raw image sensory to actuator values [9], to our knowledge, this is the first work employing deep Q-learning for active object recognition. In the following sections, we introduce the GERMS dataset and describe its composition, data collection procedure and proposed benchmarks. Then the DQL system is described in detail, and our baseline performance benchmarks are reported.

[1] Available at http://rubi.ucsd.edu/GERMS/

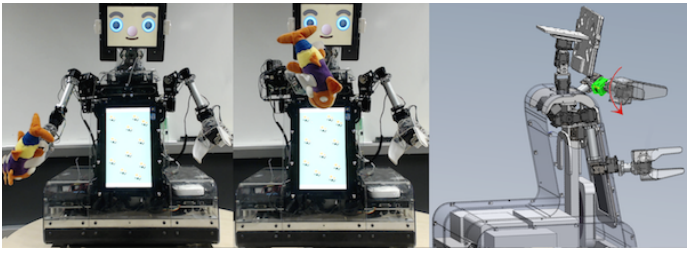


Figure 1: Left: RUBI holding an object. Middle: RUBI brings the object to its center of view and rotates it 180° using its wrist. Right: The red arc shows the direction of rotation of the wrist. The wrist joint is shown in green.

2 The GERMS Dataset

Many of the active object recognition methods are built around a specific hardware system, which makes the replication of their results very difficult. Other systems use off-the-shelf computer vision datasets, which include several views of objects captured by systematically changing object's orientation in the image. However, these datasets do not offer any active object recognition benchmark *per se*. Adapting such datasets for active object recognition ignores the challenges in the active component such as noisy actions. A well-defined benchmark should consist of an established list of training and testing images, along with the baseline results for object recognition. The dataset should also offer methods to simulate active observation with actual robotic sensory systems. In this section, we introduce the GERMS dataset, which aims to accelerate progress on active object recognition by addressing some of the shortcomings of the previous datasets.

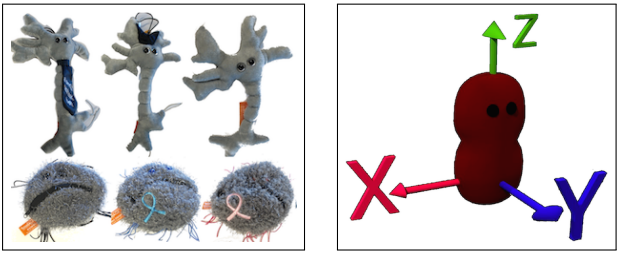
2.1 Data Collection Context

The data collection procedure was motivated by the needs of the RUBI project, whose goal is to develop robots that interact with toddlers in early childhood education environments [7, 8, 10, 12, 13]. As part of this project, we found that one of the activities toddlers liked most was give-and-take games with the robot (RUBI). In these games the toddlers hand objects to RUBI, who then pretends to examine them and gives them back to the toddler. We found that teachers use this type of give-and-take activity as an opportunity to name the objects and teach vocabulary skills. Thus we aim for RUBI to be able to recognize and name the objects given to it. In order to minimize the effect of prior knowledge of the objects that different toddlers have, we choose a large collection of soft toys, whose names the toddlers were unlikely to know.

2.2 Dataset Details

The GERMS dataset consists of 1365 video recordings of give-and-take trials using 136 different objects. The objects are soft toys depicting various human cell types, microbes and disease-related organisms. Figure 2 shows a collage of these toys. Subsets of objects in GERMS exhibit interesting visual similarities that makes it a suitable dataset for active

Figure 2: Object set used in GERMS dataset. The objects represent human cell types, microbes and disease-related organisms.



(a) (b)

Figure 3: (a) Similar objects. Top: 3 brain cells. Bottom: 3 cancer cells. Toys in each row are not distinguishable from the rear view. (b) Coordinate system on *sore throat* (see text).

object recognition. For example, ambiguity between objects in figure 3(a) can be resolved only from certain alternative views.

A trial starts when someone hands an object to RUBI (see Figure 1). RUBI brings the grasped object to its center of view, rotates it by 180° and then returns it. During each trial, RUBI continuously records images from its head-mounted camera at 30 frames per second. For each image, it also reads the positions of its joints. These data are then stored in a *track*, a collection of which constitutes the dataset.

On average, each track contains 265 snapshots of a give-and-take trial, with each snapshot consisting of an image from the head-mounted camera, the capture time, and the joint angles at capture time. These joint angles allow researchers to simulate different active observation policies. Table 1 summarizes the number of images in the dataset. The details of the objects used, and train and test data collection are described in the next two subsections.

2.3 Data Collection

The training data consist of 6 video clips per object, for a total of 816 clips. In each clip an object was handed to RUBI in one of 6 predetermined poses. These poses are determined

Table 1: GERMS dataset statistics (mean±std)

	No. of tracks	**Frames/track**	**Frames with object/track**	**Mean track length (sec)**
Train	816	265 ± 7	157 ± 12	8.94
Test	549	265 ± 7	145 ± 19	8.91

(a) (b)

Figure 4: (a) Six different poses of *flesh eating* object used for training data. The images are captured from RUBI's head-mounted camera, (b) Test poses for *flesh eating* object.

via an object-centered coordinate system (see Figure 3(b)). The Z-axis is aligned with the longest side of the object, with the positive direction determined by the orientation of the object's face. The Y-axis always comes out of the object's face plane. The 6 different poses result from the combination of 3 object orientations with respect to each of 2 grippers (left arm and right arm). Figure 4(a) shows the 6 different configurations for a single object. In each trial, RUBI grabs the object in one of these 6 configurations, brings the object to its center of view, and rotates the object by 180°.

To collect test data, we asked a set of human subjects to hand the GERM objects to RUBI in poses they considered natural. A total of 12 subjects participated in test data collection, each subject handing between 10 and 17 objects to RUBI. They were asked to hand each object to each gripper twice, using a different pose each time. Figure 4(b) shows snapshots of the test data for the same object. The background of the GERMS dataset was provided by a large screen TV displaying video scenes from the classroom in which RUBI operates, including toddlers and adults moving around.

2.4 Annotation

The training and test data were annotated manually with the target object's bounding box in frames where the object was visible. These annotations serve to limit the boundaries of objects in each image since the images are much larger than individual objects. For examples of annotations, see Figure 5. The annotations provide ground truth to test object segmentation algorithms and to allow testing AOR algorithms that assume ideal object segmentation.

2.5 Actuator Data

Accompanying each image in give-and-take trials are recordings of the joints of the robot. These joints include 2-DOF head, and two 7-DOF arms. The servos are MX-28T and MX106T type Dynamixels. Each servo is equipped with a contact-less absolute encoder with 12-bit resolution of 360°. During a give-and-take trial, after each image is captured, its corresponding servo positions are also recorded simultaneously.

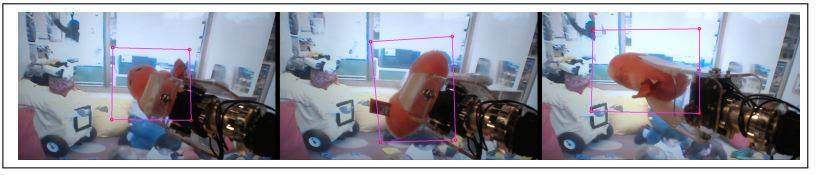


Figure 5: Bounding box annotation examples.

3 Active Object Recognition Using Deep Q-learning

As in previous work by Paletta & Pinz [15], we treat active object recognition as a reinforcement learning problem, using Q-learning to find the optimal policy. We use a DCNN for object representation and policy learning and call it deep Q-learning (DQL). Our model takes a minimalist approach to encoding the state: we extract a belief vector over different object labels and use that as input to the policy learning DCNN. Our method is different from the work by Mnih et al. [11] in an important way. We use objects beliefs as representation of the current states to learn actions, while in [11] actions are learned from the raw image. Here we hypothesize that the next action to disambiguate the current view can be inferred only from the belief over different objects. Another difference between our problem and that of [11] is that object recognition at test-time may not be episodic, that is, there may be no way of knowing when the object inspection is finished. This is a difficult problem to solve automatically, for this work we use a fixed length threshold to finish the object inspection trials.

3.1 Deep Q-learning

The model architecture is shown in figure 6. An image is first transformed into a set of features using a DCNN borrowed from [6] which was trained on ImageNet. We add a softmax layer on top of this model to recognize GERMS objects; the output of this softmax layer is the belief over different GERMS objects given an image. This belief is combined with the accumulated belief from the previous images using Naive Bayes. This accumulated belief represents the *state* of the AOR system in each time step. Let I_i be the input image to the system at the ith time step, the accumulated belief over objects given images from time steps $1,\ldots,n$ is given by,

$$P(O|I_1,\ldots,I_n) \propto \prod_{i=1}^{n} P(O|I_i) \tag{1}$$

where $P(O|I_i)$ is the posterior belief over object label O computed by the first softmax layer in figure 6. The accumulated belief is then transformed by the policy learning network into action values. This network is composed of two Rectified-Linear-Unit (ReLU) layers followed by a Linear-Unit (LU) layer. Each unit in the LU represents the action value for a given accumulated belief and one of the possible actions. We will discuss possible actions in the next subsection. In order to train this module, we employ the Q-learning iterative update:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{R(s,a) + \gamma \max_{a^*} Q(s^*,a^*) - Q(s,a)\}. \tag{2}$$

In the above equation, $Q(s,a)$ is the action value for action a in state s, α is the learning rate and $0 < \gamma < 1$ is the reward discount factor. The Q-learning iterative update is turned into

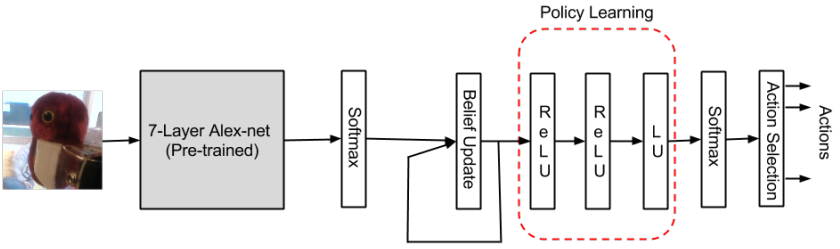


Figure 6: The proposed architecture for DQL. Images are first transformed into beliefs over object labels using a pre-trained Alex-net (gray block). Each block except for 7-layer Alex-net represents one layer in the network.

the following stochastic gradient descent weight update rule for the network:

$$W \leftarrow W - \lambda \left(R_t + \gamma \max_a Q(B_{t+1}, a) - Q(B_t, a_t) \right) \frac{\partial}{\partial W} Q(B_t, a_t). \tag{3}$$

Here, W is the set of weights of the policy learning network, $Q(s,a)$ is the action-value learned by the network for action a in state s, γ is the reward-discount factor and R_t is the reward value at time step t. Also, λ is the learning rate for the neural network and B_t is the vector of beliefs over object labels, which is used here to represent the state of the system.

3.2 Policy Learning

The number of output units in the policy learning network is equal to the number of possible actions. Each output unit calculates the action value $Q(s,a)$ for one action a. We implemented a set of actions which rotate the robot's wrist from its current position by an offset angle. We chose the rotation offsets to be $\pm\pi/64$, $\pm\pi/32$, $\pm\pi/16$, $\pm\pi/8$, $\pm\pi/4$, for a total of ten actions. The allowable range of rotation for both robot wrists is in $[0,\pi]$. We choose these actions because they allow for both fine-grained inspection of the object at hand and large rotations to traverse the entire range of wrist rotation in few actions.

The training procedure for DQL is shown in Algorithm 1. In this algorithm, *AlexNet-Softmax* converts a single image into a belief vector over objects, and *Action-Value* converts the accumulated belief into action-values using the policy network. A *game play* is defined as the sequence of *moves* (wrist rotations) selected by the policy learning network to examine the current object at hand. Each move results in rotation of the object and thus a new belief vector over object labels. The mapping of the image to action values is shown in lines 8-11 of Algorithm 1. For the selected action then we update the parameters of the network according to Equation 3. For each update, we calculate the target value of the selected action by grabbing the next image based on the current pose of the robots wrist and the selected action. The action value for this *look-ahead* image is used as the target value for the current image (lines 12-15).

In Algorithm 1 we show the training for a single image at a time. In practice, the training algorithm uses mini-batches that contain images from different give-and-take trials. In each training iteration, the procedure keeps track of different game plays and updates the neural network parameters using the sum of their gradients. A game play finishes after n moves, after which the training proceeds to the next mini-batch.

Algorithm 1 Training Deep Q-learning network

procedure TRAIN
  $t \leftarrow 1$
  while not converged **do**
    $I_t \leftarrow$ Next-Training-Image$(t-1)$
    $I_c \leftarrow I_t$
    $B_c \leftarrow$ vector of ones of length C
    for move=1 To LengthofGamePlay **do**
      $A \leftarrow$ AlexNet-Softmax(I_c)
      $B_c \leftarrow$ Normalize(elementwise-product(A, B_c))
      $Q(B_c, a) \leftarrow$ Action-Value(B_c)
      $a_c \leftarrow$ Action-Selection$(Q(B_c, a))$
      $I_c^* \leftarrow$ Next-Image(I_c, a_c)
      $A^* \leftarrow$ AlexNet-Softmax(I_c^*)
      $B_c^* \leftarrow$ Normalize(elementwise-product(A^*, B_c))
      $Q(B_c^*, a) \leftarrow$ Action-Value(B_c^*)
      $W \leftarrow W - \lambda \left(R_c + \gamma \max_a Q(B_c^*, a - Q(B_c, a_c))\right) \frac{\partial}{\partial W} Q(B_c, a_c)$
      $I_c \leftarrow$ Next-Image(I_c, a_c)
    end for
    $t \leftarrow t+1$
  end while
end procedure

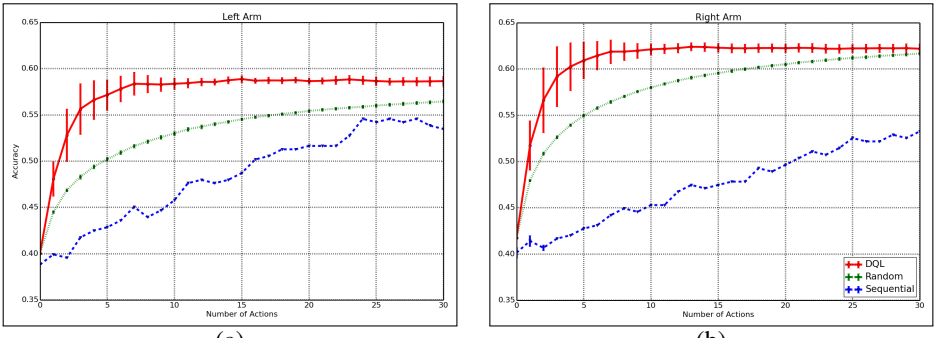
We use the same set of learning parameters to train different models for the left and right arm. The training is done using stochastic gradient descent with mini-batches of size 128. The learning rate starts at 0.01 and is multiplied by 0.1 every 1000 iterations. The training procedure is on-policy, with probabilistic action selection. The training runs for 5 epochs over the training data (3500 iterations on mini-batches), where for each mini-batch a game play of length 5 is followed to update the weights. We found no significant difference in the accuracy of models trained with longer game plays (10 and 20). After each move, a reward R_c of ± 10 was given to the network depending on whether the maximum probability label in the accumulated belief vector is equal to the target label for that give-and-take trial or not.

4 Baseline Results

The GERMS dataset contains two benchmark tasks, one for each arm. We report the accuracy of label prediction on test set trials. On each trial a new test object is selected for recognition. The AOR algorithm chooses a sequence of servo configurations producing a sequence of views of the test object. We report the accuracy of predicting the correct label as a function of the number of actions

We report the accuracy of the DQL active object recognition system as the baseline for GERMS. An instance of the network in Figure 6 is trained using game plays of length 5 using images from the training set. After the model is trained, we measure the performance on the test set as a function of the number of actions. For each action, a new accumulated belief vector is calculated and used to measure the accuracy of the model. The benchmarks are shown in figure 7.

We compared the performance of our model against two alternative policies: sequential



(a) (b)

Figure 7: Performance comparison between DQL, sequential and random policies on test images for RUBI's (a) left and (b) right arm.

Figure 8: From left to right: the sequence of actions chosen by DQL on "Cancer".

and random. The random policy selects a random action with uniform probability, while the sequential strategy always starts from the same position and moves in the same direction to the next immediate position. Figure 7 compares the accuracy of predicting the correct object label as a function of the number of observed images. This performance is averaged over the entire set of images in the test set, that is if the system starts from any image in the test set and selects the next action according to the corresponding policy. We report the average performance of 20 models trained in separate runs of Algorithm 1.

Table 2 shows the required number of step to reach the same level of prediction accuracy over 136 different target classes of GERMS by the sequential, random and DQL strategies. For the samples collected with the left arm, DQL strategy achieves 55% accuracy in only 3 steps which is significantly better than 18 steps by the random and 37 steps by the sequential method. For the same arm, DQL achieves its peak performance (58%) in 7 steps while the other two methods can't achieve this in a maximum of 30 steps. For the right arm, DQL reaches 58% accuracy in 3 steps, compared to 10 steps required by the random strategy. Sequential method can't reach this level in 30 steps. DQL achieves its peak performance of 62% accuracy in 10 steps, while none of the other methods can reach the same accuracy within 30 steps.

5 CONCLUSIONS

Active object recognition has the potential to overcome many of the difficulties encountered in classical vision problems of passive object recognition from static images. While the literature on active object recognition has shown promising results, progress has been slow due to the lack of realistic datasets and benchmarks that can be easily shared by multiple research groups. In this paper, we introduced the GERMS dataset that includes a collection of videos, a set of active object recognition benchmarks and baseline results on those benchmarks. We hope that this dataset will facilitate the comparison of different active object recognition

Table 2: Number of steps required by the sequential, random and DQL policies to reach the same level of prediction accuracy on GERMS dataset.

Method \ Prediction Accuracy(%)	48	53	55	58	62	
Sequential	18	30	-	-	-	**Right Arm**
Random	2	4	6	10	-	
DQL	**1**	**2**	**2**	**3**	**10**	
Sequential	15	24	-	-	-	**Left Arm**
Random	3	10	18	-	-	
DQL	**1**	**3**	**3**	**7**	-	

methods and accelerate progress in the field.

We also proposed an architecture for active object recognition based on deep Q-learning. Instead of the standard approach of encoding the state using a vector of visual features, we used the representation produced by a commodity deep object classification network [11]. This representation was then processed by an additional deep network trained using Q-learning for efficient action selection. The proposed approach outperforms sequential and random action selection policies and serves as baseline for future comparisons. We also observed that different lengths of game plays did not have an impact on the performance of trained model. The model achieves its peak recognition accuracy in 7-10 steps, far fewer than the random and sequential strategies, and always achieves the highest accuracy among them.

The current baseline approach relied on human-annotated bounding boxes. We are currently baselining algorithms that include automatic object segmentation. We are also evaluating potential performance gains achievable by post-training the early perceptual layers of the object recognition network using the policy learning error signals.

References

[1] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *International journal of computer vision*, 1(4):333–356, 1988.

[2] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.

[3] Hermann Borotschnig, Lucas Paletta, Manfred Prantl, and Axel Pinz. Appearance-based active object recognition. *Image and Vision Computing*, 18(9):715–727, 2000.

[4] Bjorn Browatzki, Vadim Tikhanoff, Giorgio Metta, Heinrich H Bulthoff, and Christian Wallraven. Active in-hand object recognition on a humanoid robot. *Robotics, IEEE Transactions on*, 30(5):1260–1269, 2014.

[5] Francesco G Callari and Frank P Ferrie. Active object recognition: Looking for differences. *International Journal of Computer Vision*, 43(3):189–204, 2001.

[6] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[7] Bret Fortenberry, Joel Chenu, and JR Movellan. Rubi: A robotic platform for real-time social interaction. In *Proceedings of the International Conference on Development and Learning (ICDL04), The Salk Institute, San Diego*, 2004.

[8] Daniel Johnson, Mohsen Malmir, Deborah Forster, Morana Alac, and Javier Movellan. Design and early evaluation of the rubi-5 sociable robots. In *Development and Learning and Epigenetic Robotics (ICDL), 2012 IEEE International Conference on*, pages 1–2. IEEE, 2012.

[9] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *arXiv preprint arXiv:1504.00702*, 2015.

[10] Mohsen Malmir, Deborah Forster, Kendall Youngstrom, Lydia Morrison, and Javier R Movellan. Home alone: Social robots for digital ethnography of toddler behavior. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 762–768. IEEE, 2013.

[11] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[12] Javier Movellan, Micah Eckhardt, Marjo Virnes, and Angelica Rodriguez. Sociable robot improves toddler vocabulary skills. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 307–308. ACM, 2009.

[13] Javier R Movellan, Mohsen Malmir, and Deborah Forester. Hri as a tool to monitor socio-emotional development in early childhood education, 2012.

[14] Sameer A Nene, Nayar, and Hiroshi Murase. Columbia object image library (coil-100). Technical report, Technical Report CUCS-006-96, 1996.

[15] Lucas Paletta and Axel Pinz. Active object recognition by view integration and reinforcement learning. *Robotics and Autonomous Systems*, 31(1):71–86, 2000.

[16] Sumantra Dutta Roy, Santanu Chaudhury, and Subhashis Banerjee. Active recognition through next view planning: a survey. *Pattern Recognition*, 37(3):429–446, 2004.

[17] Bernt Schiele and James L Crowley. Transinformation for active object recognition. In *Computer Vision, 1998. Sixth International Conference on*, pages 249–254. IEEE, 1998.

[18] Michael Seibert and Allen M. Waxman. Adaptive 3-d object recognition from multiple views. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):107–124, 1992.

[19] David Wilkes and John K Tsotsos. Active object recognition. In *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*, pages 136–141. IEEE, 1992.