Intrinsically Motivated Self-Supervised Deep Sensorimotor Learning for Grasping

Takeshi Takahashi Michael W. Lanighan Roderic A. Grupen

***Abstract*— Deep learning has been successful in a variety of applications that have high-dimensional state spaces such as object recognition, video games, and machine translation. Deep neural networks can automatically learn important features from high-dimensional state given large training datasets. However, the success of deep learning in robot systems in the real-world is limited due to the cost of obtaining these large datasets. To overcome this problem, we propose an information-theoretic, intrinsically motivated, self-labeling mechanism using closed-loop control states. Taking this approach biases exploration to informative interactions—as such, a robot requires much less training to achieve reliable performance. In this paper, we explore the impact such an approach has on learning how to grasp objects. We evaluate different intrinsic motivators present in the literature applied appropriately in our framework and discuss the benefits and drawbacks of each.**

I. Introduction

Deep learning has been successful in a variety of fields such as object recognition [1], video games [2], Go [3], and translation [4] because of its ability to learn important features and generalize to previously unseen data. Recent advancements in convolutional neural networks have shown great improvements in image recognition tasks over traditional hand-crafted features thanks to large-labeled datasets [1]. However, deep learning has had limited success in robotics because of challenges present in robotics domains. These challenges include:

1) Collecting large labeled datasets is non-trivial. A human typically hand labels training data.
2) Data is expensive in terms of time and energy because the robot executes exploratory actions whose consequences are observed in real-time in order to obtain samples.

To address these challenges, self-supervision techniques have been used to supply labels automatically without human experts [5], [6]. However, these techniques still require a large number of training samples to achieve reliable performance. We propose using active learning techniques to reduce the number of training samples necessary to achieve high levels of performance. We evaluate the effectiveness of using these techniques to decrease training time in learning how to grasp objects with a real robot using self-supervised deep learning. The overall approach is outlined in Figure 1 and is explained in Section III.

This paper provides two contributions in self-supervised deep sensorimotor learning:

The authors are with the Laboratory for Perceptual Robotics, College of Information and Computer Sciences, University of Massachusetts Amherst MA 01003, USA. {ttakahas,lanighan,grupen}@cs.umass.edu

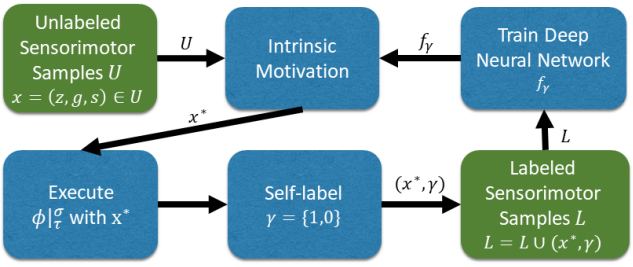


Fig. 1: Intrinsic motivation selects the most informative sample x^* to query. The robot self-labels the action outcome using the status of closed loop controllers. Variables in this figure are defined in Section III.

1) We show that active learning techniques can reduce the training time and the number of samples required to learn deep sensorimotor skills.
2) We evaluate different intrinsic motivators for active learning and discuss the benefits and drawbacks of each.

II. Related Work

Minh *et al.* proposed Deep Q networks (DQNs) to solve simple video games using raw images as an input. By combining reinforcement learning (RL) with convolutional neural networks and a replay-memory they effectively learn policies in domains with discrete actions [2]. This approach is not directly applicable to robotics domains as the action space in robotics is continuous. Lillicrap *et al.* extended DQN for problems with continuous action space [7] by applying a Deterministic Policy Gradient [8] with actor and critic networks to efficiently train policies. Their results in simulation are competitive to those of optimal control policies. However, real robot experiments were not performed. Unfortunately, it is not the case that networks learned in simulation easily transfer to real robot systems despite millions of training simulations over hundreds of hours [9].

We use a method called *self-supervision* that allows a robot to explore and evaluate its own training experiences. Self-supervision is similar to work reported by Pinto and Gupta that uses hundreds of hours of grasp data and heuristic supervision functions for labeling [5]. Recent work by Levine *et al.* incorporate a similar predefined metric to self-supervise a network built to predict the probability of grasp success [6]. Our approach differs in the way that self-supervision is determined using the state of closed-loop controllers. This captures the robot's evaluation of its own actions through

direct closed-loop interactions with the environment. The output of our network corresponds to a predicted status of these primitive closed-loop controllers. We use active learning techniques to intrinsically motivate the robot to select more effective training trials. In comparison, Pinto *et al.* use random grasps and importance sampling to collect data [5] while Levine *et al.* use random and on-policy data collection [6].

We believe that intrinsic motivation (see [10], [11] for surveys) is one of the most important components needed for robots to learn skills autonomously in the real-world. Intrinsic motivations have been extensively investigated in psychology [12], [13]. Barto *et al.* applied intrinsic motivation to reinforcement learning to learn skills by defining a reward function that captures the intrinsic motivation [14]. Intrinsically motivated learning and active learning share many attributes [15] and we will use the terms interchangeably. Active learning aims to select a subset of unlabeled data points to train a system efficiently (see [16] for a survey). Active learning is often used when randomized exploration is inefficient and querying labels is costly in supervised learning tasks. Existing active learning techniques scale poorly to high-dimensional data [17]. Recently, researchers have started investigating applying active learning techniques to convolutional neural networks [18], [19], [20]. Gal *et al.* introduced deep Bayesian active learning in image recognition applications to select unlabeled data efficiently [18]. They use Dropout layers to better approximate the uncertainty present in the network. Batch-query active learning algorithms have also been proposed for convolutional networks in classification tasks that query a large portion of unlabeled images at the same time [19], [20]. However, a robot can query only one sample at a time, so the batch active learning method is not easily applicable. The use of active learning methods for self-supervised deep sensorimotor learning in robotics has not been well studied.

III. Problem Formulation

The overall approach is summarized in Figure 1. Using intrinsic motivation, a sample x^* from an unlabeled sensorimotor sample set $\mathcal{U}$ is selected. x^* is executed by a robot, which observes the outcome in real-time and self-labels the outcome γ. The status of closed-loop controllers alone provides a method for self-supervision in which a robot labels its training experiences autonomously. This new labeled sample (x^*, γ) is used in addition to previously obtained labeled outcomes $\mathcal{L}$ to train a deep neural network which learns the affordance (the possibilities for action) of the current environment. The network predicts controller state given visual observation, proprioceptive state of the robot, and controller goals.

A. Self Supervision via Closed-loop Controller Dynamics

This paper uses a deep convolutional network whose purpose is to predict controller status. The inputs of the network are multi-modal sensory inputs, while the output is predicted control state. By using the control state of closed-loop motion primitives the dimensionality of the output layer of the network is reduced. This representation encodes interactions between the robot and the world and can be used to predict whether given multi-modal sensory inputs afford a particular control action. In other words, this encoding supports expectations for control actions that yield (or afford) reliable tactile interactions. The particular choice of learning the activation of a closed-loop grasp primitive is due to the convergence results demonstrated in [21] for regular convex prismatic objects.

We employ the *Control Basis* formulation to enumerate closed-loop perceptual control policies. These controllers $\phi|_\tau^\sigma$, consist of a combination of potential functions ($\phi \in \Phi$), sensory inputs ($\sigma \in \Sigma$), and motor resources ($\tau \in \mathcal{T}$) [22]. Controllers achieve their objective by following gradients in the potential function $\phi(\sigma)$ with respect to changes in the value of the motor variables Δu_τ. Gradients are described by the error Jacobian $J = \partial\phi(\sigma)/\partial u_\tau$ and references to low-level motor units are computed as $\Delta u_\tau = -kJ^{\#}\phi(\sigma)$, where $J^{\#}$ is the Moore-Penrose pseudoinverse of J [23].

The interaction between the embodied system and the environment is modeled as a dynamical system, allowing the robot to evaluate the status of its actions as the time varying state of a closed-loop system. The time history of the *error* state $(\phi, \dot{\phi})$ is a strong surrogate for the state of the interaction with an unknown environment [21], [24]. The state description γ^t at time t, is derived directly from the state $(\phi, \dot{\phi})$ of the controller using a simple classifier:

$$\gamma^t(\phi|_\tau^\sigma) = \begin{cases} \text{-}: & |\dot{\phi}| > \epsilon_1 \\ 1: & |\dot{\phi}| \leq \epsilon_1 \text{ and } |\phi| \leq \epsilon_2 \\ 0: & |\dot{\phi}| \leq \epsilon_1 \text{ and } |\phi| > \epsilon_2 \end{cases} \quad (1)$$

where ϵ_1 and ϵ_2 are small constants. The neural network predicts this controller state. Thus, the network is a hyperparametric function approximator $f_{\gamma^T(\phi|_\tau^\sigma)} : X \mapsto P(\gamma^T(\phi|_\tau^\sigma))$ where $x \in X$ is (z, g, s), z is the sensory input from the environment, g are controller goals, and s is proprioceptive feedback of the robot. In this study, we assume $lim_{t\to\infty}|\dot{\phi}| \leq \epsilon_1$, and the neural network predicts either 1 or 0. To simplify notation, we use γ instead of $\gamma^T(\phi|_\tau^\sigma)$ in the rest of the paper.

B. Intrinsic Motivators for Active Learning

Let $\gamma \in \Gamma$ be a label ($\Gamma = \{1, 0\}$) the robot assigns given x, $\mathcal{L} = \{(x_i, \gamma_i)\}_{i=1}^{|\mathcal{L}|}$ be a set of labeled sensorimotor samples, and $\mathcal{U} = \{x_i\}_{i=1}^{|\mathcal{U}|}$ be a set of unlabeled sensorimotor samples. Initially $\mathcal{L}$ is small and the robot will collect labeled pairs to train f_γ. In this paper, we want to minimize the training cost to achieve reliable performance of f_γ. The cost can be the number of actions (queries) or training time (computation time + query time). The action the robot selects will query $x_i \in \mathcal{U}$ to obtain γ_i. The query process takes time because the robot needs to execute exploratory actions whose consequences are observed in real-time. The robot selects the most informative sample x_i^* for a query based on its intrinsic motivation. We investigate the behaviors

of the following intrinsic motivators: Maximum Entropy [16], Maximum Entropy with Monte Carlo (MC) Dropout [18], Maximum Entropy with Input Noise, Expected Error Reduction [25], and a Hybrid approach that combines the Maximum Entropy and Expected Error Reduction approaches. To form a baseline, we use a Random approach commonly used in state of the art learning approaches.

1) Maximum Entropy (ME): The Maximum Entropy method selects samples based on which ones are most uncertain [16]. As entropy measures uncertainty, the most uncertain sample x^* will be selected. x^* is computed with

$$\begin{aligned} x^* &= \arg\max_{x \in \mathcal{U}} (H_\theta[\Gamma|x]) \\ &= \arg\max_{x \in \mathcal{U}} (E_{\gamma|\theta,x}(-log(P_\theta(\gamma|x)))) \end{aligned}$$

where θ denotes parameters of a neural network f_γ. The approximate cost of this method is $\mathcal{O}(\frac{|\mathcal{U}|F}{B})$ where F is cost of feed-forward computation of f_γ and entropy computation given one batch, and B is a batch size for the feed-forward computation. The computation will be expensive if we evaluate all samples in $|\mathcal{U}|$. In the case of robotics domain, the goal of the controller g is continuous, and $|\mathcal{U}|$ can be infinite. To approximate $|\mathcal{U}|$, we use $\mathcal{U}_s$, a subset of $\mathcal{U}$, for this evaluation. We create $\mathcal{U}_s$ by randomly sampling from $\mathcal{U}$. The resulting equation is

$$x^* = \arg\max_{x \in \mathcal{U}_s} (H_\theta[\Gamma|x]) \tag{2}$$

The cost of this method is $\mathcal{O}(M_F F)$ where $M_F = \lceil \frac{|\mathcal{U}_s|}{B} \rceil$ is the number of batches for computing entropy.

2) Maximum Entropy with MC Dropout (MEDO): To better estimate the uncertainty of f_γ, we can compute the entropy by making use of *dropout layers* in the neural network [18]. A sample x^* is selected with

$$x^* = \arg\max_{x \in \mathcal{U}_s} (\frac{1}{N} \sum_n (H_{\hat{\theta}_n}[\Gamma|x])) \tag{3}$$

where $\hat{\theta}_n$ follows the dropout distribution and N is the number of iterations to estimate the distribution. This finds the sample that is on average most uncertain over different parameter settings. This is because each feed-forward-pass generates slightly different outputs given the same input as some neurons in the dropout layers are randomly disabled. Gal *et al.* demonstrated how this approach performs better than using the maximum entropy on the MNIST dataset [18] regarding the number of queries. However, this approach incurs a high cost $\mathcal{O}(NM_F F)$ and requires the additional implementation of dropout layers.

3) Maximum Entropy with Input Noise (MEIN) : Instead of using *dropout layers* to estimate the output distributions, they can be estimated by adding noise to the controller goals. With this method a sample x^* is selected using

$$x^* = \arg\max_{x \in \mathcal{U}_s} \left(\frac{1}{N_\epsilon} \sum_n H_\theta[\Gamma|x = (z, g + \epsilon_n, s)] \right) \tag{4}$$

where ϵ_n is noise and N_ϵ is the number of ϵ_n. Each element $\epsilon_{i,n}$ of ϵ_n follows $Uniform(0, r_i)$ where r_i is a small constant. This method estimates uncertainty in the model without relying on the use of dropout layers. This is a good approximation of actual behavior as actions performed by a robot are stochastic. With this approach, the output of the neural network should be robust against small amounts of noise. The cost of this method is $\mathcal{O}(N_\epsilon M_F F)$.

4) Expected Error Reduction (EE): Expected error reduction [25] has not been investigated for use in deep convolutional neural networks. While previous methods estimate the current uncertainty of the model, the expected error reduction estimates the future model uncertainty by retraining the model. This method is computationally expensive because it needs to retrain the neural network considering all possible outcomes. To compute this efficiently, we compute the average of future performance over $\mathcal{U}_e$, a subset of $\mathcal{U}$ instead of using $\mathcal{U}$. We also use only one sample (x, γ) to train the network for computing the future entropy instead of using $\mathcal{L} \cup (x, \gamma)$ where x is a sample we are examining and γ is a possible outcome. The most informative sample x^* is selected with

$$x^* = \arg\max_{x \in \mathcal{U}_s} \left(- \max_{\gamma \in \Gamma} \left(P(\gamma|\theta, x) \hat{H}_{\theta|(x,\gamma)} \right) \right) \tag{5}$$

$$\hat{H}_{\theta|(x,\gamma)} = \frac{1}{|\mathcal{U}_e|} \sum_{x' \in \mathcal{U}_e} H_{\theta|(x,\gamma)}[\Gamma|x'] \tag{6}$$

The cost is $\mathcal{O}(|\mathcal{U}_s||\Gamma|(M_G G + M_e F))$ where $|\Gamma|$ denotes the number of labels, G is cost of the back propagation, M_G is the number of iterations, and $M_e = \lceil \frac{|\mathcal{U}_e|}{B} \rceil$ is the number of batches for computing entropy. Although this is still an expensive method if the number of labels is large, we only consider two outcomes in our manipulation task. Thus, it is still feasible to compute this intrinsic motivator in a reasonable time frame.

5) Hybrid Approach (H): Another possible measure is to combine both the maximum entropy and the expected error reduction approaches. The expected error reduction approach uses the expected future entropy, $\hat{H}_{\theta|(x,\gamma)}$, the estimation of which may not be accurate. To compensate for this, we measure the difference between the current entropy and the expected future entropy. The computational cost is the same as the expected error reduction approach. The most informative sample x^* is selected using

$$x^* = \arg\max_{x \in \mathcal{U}_s} \left(H_\theta[\Gamma|x] - \max_{\gamma \in \Gamma} \left(P(\gamma|\theta, x) \hat{H}_{\theta|(x,\gamma)} \right) \right) \tag{7}$$

6) Baseline: Random (R): The random approach randomly samples x from $\mathcal{U}$.

$$x^* = x_{i^*} \text{ where } i^* \sim Uniform(1, |\mathcal{U}|) \quad (8)$$

The random approach does not consider the current state of the model, and does not bias exploration to informative areas. However, the complexity of this approach is $\mathcal{O}(1)$. As such, the robot can execute more actions within a specific time frame compared to the other approaches.

IV. Experiments

To investigate the benefits of intrinsic motivation and to evaluate the different approaches outlined in Section III-B, we conduct experiments on a manipulation task where a robot learns to grasp novel objects. The objective is to predict grasp success given depth images, controller goals, and proprioceptive states while minimizing the amount of training required.

A. Robot

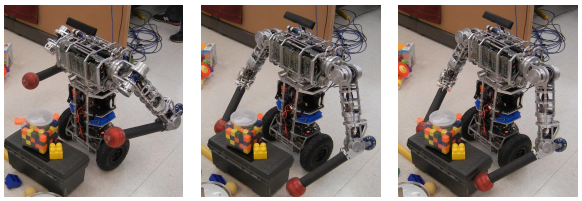
We use the uBot-6 [26] in the experiments. The uBot-6 robot platform (shown grasping an object in Figure 2) is a 13 DOF, toddler-sized, dynamically balancing, humanoid mobile manipulator equipped with an Asus Xtion Pro Live RGB-D camera, designed and built at the Laboratory for Perceptual Robotics. The robot uses Robot Operating System (ROS) middleware [27]. To grasp objects, the robot uses two controllers (REACH and COMPRESS) in sequence (Figure 2). This sequence pre-grasps then grasps an object in an antipodal configuration. REACH is a bimanual endpoint position controller and COMPRESS drives the end effectors together in search of a reaction force.

B. Network Architecture

The network predicts the outcome of grasping events. The network uses input $x = (z, g, s)$ where z is a depth image (size is 120 $\times$ 160), s is a head angle, and $g = (\sigma_{ref_L}, \sigma_{ref_R}, F_{ref})$. $\sigma_{ref_L} = (x_L, y_L, z_L)_{robot}$ is a left hand reference position (pre-grasp position) in the robot frame, and $\sigma_{ref_R} = (x_R, y_R, z_R)_{robot}$ is a right hand reference position in the robot frame, and F_{ref} is the reference force. We use a fixed F_{ref} in this experiment. The goal of REACH controller is $(\sigma_{ref_L}, \sigma_{ref_R})$ and the goal of COMPRESS controller is F_{ref}. The network architecture used in this paper is illustrated in Figure 3. We use a depth image as sensory input. In addition we use controller goals and robot state as additional inputs. This allows us to generalize over different robot-environment configurations. We use Adam [28] to update the weights of the network with a learning rate $\alpha = 10^{-4}$.

C. Dataset

In order to statistically compare the outcomes of each approach, we first collect a training dataset of grasps on a fixed set of objects. We investigate each intrinsic motivator over this dataset. During data collection, the robot was presented



(a) (b) (c)

Fig. 2: Grasping pipeline: The REACH controller and the COMPRESS controller compose the REACH-COMPRESS control program. (a) shows the initial pose of the robot as the robot activates the REACH controller. (b) shows the REACH controller converges to an antipodal pre-grasp. The robot then activates the COMPRESS controller. (c) After COMPRESS controller converges, the object is grasped.

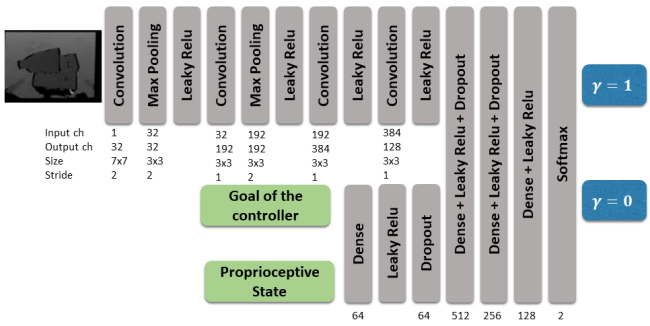
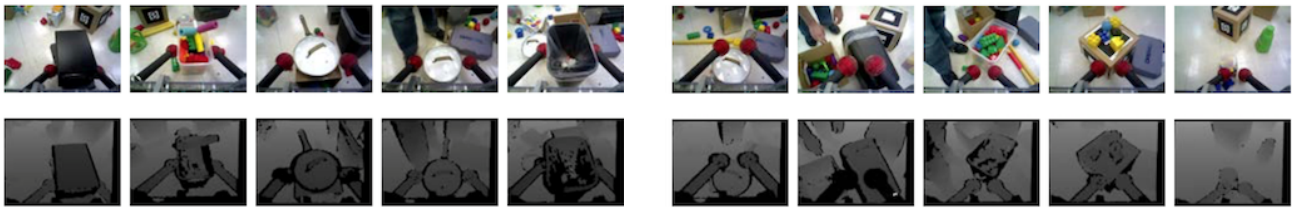


Fig. 3: Network Architecture. Relu is a rectifier linear unit.

with training objects at different positions and orientations in the scene during each trial. Additional objects were added to the scene as distractors that were either placed out of reach, on top of other objects, or occluded areas in the scene. For each trial, the robot performs a grasping action consisting of REACH and COMPRESS actions with a randomly set head angle. Pre-grasp (REACH) goals are selected for the left and right end effectors by sampling $x_L \sim Uniform(0.25, 0.43)$, $y_L \sim Uniform(0.29, 0.31)$, $z_L \sim Uniform(0.1, 0.5)$ with $x_R = x_L, y_R = -y_L, z_R = z_L$. We also used goal positions specified by authors.

After the robot has reached the specified pre-grasp posture the COMPRESS controller $\phi|_\tau^\sigma$ is executed until $|\dot{\phi}| \leq \epsilon_1$ is satisfied, indicating that the controller is no longer making progress towards its target. The asymptotic control state γ becomes the label that is assigned to the whole grasp session—each of the saved images inherits this label. If $|\phi| \leq \epsilon_2$, we consider this as a successful grasp ($\gamma = 1$). We use a quadratic error function ϕ that computes an error between the magnitude of the reference force and the magnitude of the current force. If the end effectors does not move ($|\dot{\phi}| \leq \epsilon_1$) and does not satisfy $|\phi| \leq \epsilon_2$, we consider this as a failure grasp ($\gamma = 0$). This can also happen when the controllers violate safety conditions (for example two hands are too close), and the controllers stop.

Each grasping trial is completed by calling a homing procedure to reposture the robot. The data collection process



(a) (b)

Fig. 4: Selected grasp examples consisting of RGB (top) and depth (bottom) that resulted in $\gamma = 1$ (a) and $\gamma = 0$ (b). The robots hands slipped along the surface of the object in some of the examples in (b).

(a) (b)

Fig. 5: (a) shows training set consists of varying objects with a number of distractors such as blocks, bats, and toys. (b) shows test set.

consisted of 512 grasping trials. Each trial took about 30 seconds. 385 grasp experiences are used as training input to a deep convolutional neural network while 127 grasp experiences are used as testing data. Novel objects were used for testing. We saved depth images and head angles at 10 Hz. Each grasping trial consists of a couple hundred of samples. We select 20 samples from each grasping trial for training and 10 samples for testing to make a large dataset. The overall dataset used 7700 training samples and 1270 testing samples. Each sample contains a depth image, controller goals, a proprioceptive state, and a label the robot assigned. Figure 4 show examples of the outcome states of the closed loop controller. Figure 5 shows objects used in the experiment.

D. Experimental Settings

To investigate the behavior of each intrinsic motivator, we compare the area under the learning curve (ALC) of each intrinsic motivator in three conditions: the number of actions (Q_A), a 30-second-query-time + computation time (Q_{30}), and a 1-second-query-time + computation time (Q_1). The query time is equivalent to the robot action time (executing a grasp action, labeling the outcome, and going back to the initial posture). In each experiment we randomly draw five samples of $\gamma = 1$ and five samples of $\gamma = 0$ as $\mathcal{L}$. We treat the rest of samples as $\mathcal{U}$. After each query, we update the neural network and remove the queried sample from $\mathcal{U}$ and add it to $\mathcal{L}$. Each experiment continues until $|\mathcal{L}| = 1000$. Before experiments, we explored the hyper-parameters of the algorithms. Empirically we found using $|\mathcal{U}_s| = 200$ to yield good performance. Due to the existence of uncertain samples that do not carry much information, performance does not increase even with larger $|\mathcal{U}_s|$. We suspect that these uninformative samples were generated by stochastic robot actions. We use $N = 30$, $N_\epsilon = 30$, $|\mathcal{U}_e| = 64$, $M_G = 1$, $B = 128$, $r_i = 0.03$, $M_F = 2$, and $M_e = 1$.

V. Results and Discussions

For each experiment we ran ten episodes of each intrinsic motivator with a different initial $\mathcal{L}$ sampled from our dataset. If we use the entire training dataset as $\mathcal{L}$, the accuracy of the network asymptotes to 85.6%. The mean learning curves are shown in Figures 6, 7, and 8 for each experimental setting. For each we compute the area under the learning curve (ALC) to evaluate the performance of the intrinsic motivators. The results of the ALC computation are shown in Table II. The higher the ALC is, the more efficient and accurate a method is. To determine statistical significance between each pair of methods, we used a randomization test with $p < 0.05$. The results of the statistical significance tests are shown in Table I. An entry in a table cell indicates that the approach of the row is statistically significantly better performing than the approach in the column. Entries indicate which setting is statistically significant, number of samples (Q_A), query time of 30 seconds (Q_{30}), and query time of 1 second (Q_1). The absence of an entry indicates the performance of the pair is statistically insignificant.

A. Analysis: the number of samples (Q_A)

When performance is evaluated solely using the number of queries, all intrinsic motivators perform statistically significantly better than random, as seen in Table I and Figure 6. However, there is no statistically significant difference in performance between the intrinsically motivated approaches. Although no statistically significant difference in performance among intrinsic motivators exist, the mean of the learning curve of the Expected Error Reduction is slightly lower compared to the other intrinsic motivators. This is likely due to the difficulty this method has of estimating the impact future actions has on entropy.

Figure 6 shows that the Maximum Entropy with MC dropout (MEDO), the Maximum Entropy with Input Noise (MEIN), and the Hybrid approach (H) reached 70% accuracy more than 30 samples earlier than the Maximum Entropy (ME), which indicates that these approaches work better with fewer number of samples. This is likely due to the fact that initially the labeled set does not cover a large

space of the domain. As such the exploration ME undertakes initially will not provide good guidance, as any sample is potentially informative. As MEDO and MEIN better estimate the true uncertainty by applying small perturbations, they will guide the system better initially. Although the expected error reduction itself trails other intrinsic motivators, when combined with ME in the Hybrid approach performance exceeds ME as it biases the system towards areas that will bring more information in the future.

B. Analysis: Computation time and Query time (Q_{30}, Q_1)

When computation time is dominated by execution time (as in the Q_{30} experiment) every intrinsic motivator outperforms Random as seen in Table I and Figure 7. There is no statistical significance in the performance among ME, MEDO, and MEIN in Q_{30}. However, when execution time does not dominate computation time, as in the Q_1 experiment Random outperforms the Expected Error Reduction and Hybrid approaches (see Figure 8). This is because Random is able to query more samples in the equivalent time. Similarly, ME outperforms all other methods in Q_1 due to its short computation time coupled with the guidance it provides.

The computation time shows that MEDO and MEIN took about 30 times more than ME. As we described in Section III-B, the cost of ME is $\mathcal{O}(M_F F)$, MEDO is $\mathcal{O}(N M_F F)$ and MEIN is $\mathcal{O}(N_\epsilon M_F F)$. With $N = 30$ and $N_\epsilon = 30$ the result matches our cost analysis. The cost of expected error reduction and hybrid approaches is $\mathcal{O}(|\mathcal{U}_s||\Gamma|(M_G G + M_e F))$. The time to compute these approaches is more than 200 times that of ME (as expected since $|\mathcal{U}_s| = 200$, $|\Gamma| = 2$ $M_G = 1$, and $M_e = 1$).

When computation and execution times are accounted for there is a clear difference in the performance of the different motivators that is not apparent when only the number of samples is considered. These results indicate that depending on the time a robot takes to execute actions different motivators will provide better performance than others. For instance, as seen by comparing the outcomes of Q_{30} and Q_1 the ME approach may not be the most efficient approach when action execution dominates computation time, but it is the most efficient approach when action execution time does not dominate computation time. These results suggest that we need to consider computation and query time to validate the effectiveness of intrinsic motivators in deep sensorimotor learning tasks even though these times are often omitted in the active learning literature.

VI. Conclusions and Future work

Applying deep-learning techniques to robotics domains remains an open challenge. In this paper, we have evaluated the use of active learning techniques to better improve the learning rates of self-supervised deep sensorimotor learning approaches. The results indicate that intrinsic motivators outperform random exploration and reduces the number of actions and training time required for reliable performance. Although no statistical differences are observed between the intrinsic motivators when only the number of samples used are considered, when we consider the computation time + query time differences can emerge. As the results indicate, if the robot can perform queries quickly (computation time is not dominated by execution time), the maximum entropy approach outperforms all other methods. However, when execution time does not dominate computation time, other approaches that better capture the uncertainty of the network perform better.

Our results also indicate that using Expected Error Reduction as a motivator is not ideal for deep sensorimotor learning because of both the computation time required and the difficulty of estimating future uncertainty. Even when the future entropy is approximated using just one sample, significant computation time is required. With our dataset, if neural networks do not have drop-out layers, the maximum entropy or the maximum entropy with input noise can be used to achieve equivalent performance to maximum entropy with MC dropout. In future work, we would like to examine these intrinsic motivators in other manipulation tasks such as touching and pushing.

TABLE I: Statistically significant results ($p < 0.05$)

	ME	MEDO	MEIN	EE	H	R
ME	-	Q_1	Q_1	Q_1,Q_{30}	Q_1	Q_1,Q_{30}, Q_A
MEDO		-		Q_1,Q_{30}	Q_1,Q_{30}	Q_1,Q_{30}, Q_A
MEIN			-	Q_1,Q_{30}	Q_1	Q_1,Q_{30}, Q_A
EE				-		Q_{30}, Q_A
H					-	Q_{30}, Q_A
R				Q_1	Q_1	-

An entry in a table cell indicates that the approach of the row is statistically significantly better performing than the approach in the column. Entries indicate which setting is statistically significant (Q_A, Q_{30}, Q_1). The absence of an entry indicates the performance of the pair is statistically insignificant. For example, random approach outperforms EE and H in Q_1.

TABLE II: The mean of ALC and Computation Time (CT)

	ALC(Q_A)	ALC(Q_{30})	ALC(Q_1)	CT (sec)
ME	0.811 ± 0.02	0.761 ± 0.02	0.760 ± 0.02	0.02
MEDO	0.815 ± 0.01	0.766 ± 0.01	0.741 ± 0.01	0.61
MEIN	0.806 ± 0.02	0.760 ± 0.01	0.733 ± 0.02	0.62
EE	0.804 ± 0.03	0.741 ± 0.02	0.624 ± 0.02	5.24
H	0.808 ± 0.02	0.748 ± 0.02	0.641 ± 0.03	5.15
R	0.755 ± 0.03	0.708 ± 0.03	0.708 ± 0.03	8.14×10^{-5}

ACKNOWLEDGMENT

The authors would like to thank Jay Ming Wong for his contributions. This material is based upon work supported under Grant NASA-GCT-NNX12AR16A. Any opinions, findings, conclusions, or recommendations expressed in this material are solely those of the authors and do not necessarily reflect the views of the National Aeronautics and Space Administration.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), 2012.

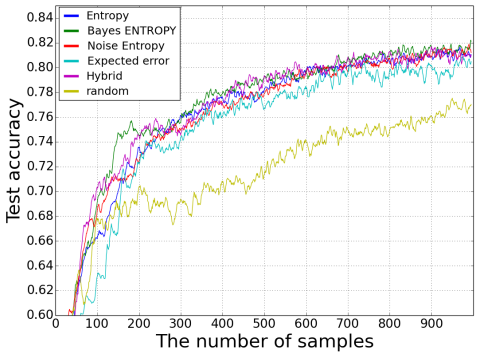


Fig. 6: Mean test accuracy in terms of the number of actions executed. A mean filter over five steps was applied.

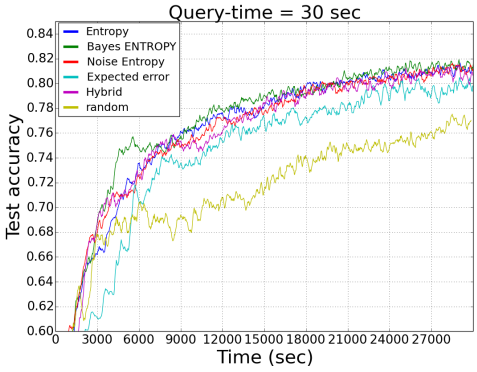


Fig. 7: Mean test accuracy over execution and computation time. A mean filter over five steps was applied. Each query took 30 seconds to execute but took varied times to compute.

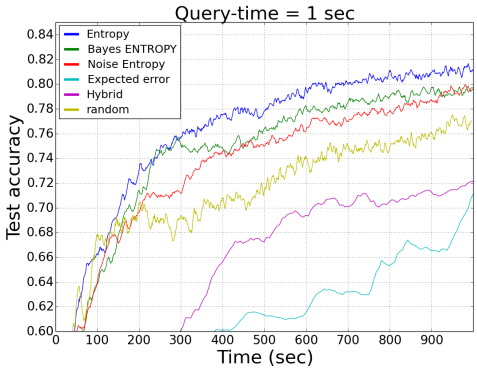


Fig. 8: Mean test accuracy over execution and computation time. A mean filter over five steps was applied. Each query took one second to execute but took varied times to compute.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[4] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016.

[5] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *arXiv preprint arXiv:1509.06825*, 2015.

[6] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *arXiv preprint arXiv:1603.02199*, 2016.

[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[8] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31th International Conference on Machine Learning*, pp. 387–395, 2014.

[9] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[10] G. Baldassarre and M. Mirolli, *Intrinsically Motivated Learning in Natural and Artificial Systems*. Springer Science & Business Media, 2013.

[11] A. Cangelosi, M. Schlesinger, and L. B. Smith, *Developmental robotics: From babies to robots*. MIT Press, 2015.

[12] C. Hull, "Principles of behavior," 1943.

[13] D. E. Berlyne, "Curiosity and exploration," *Science*, vol. 153, no. 3731, pp. 25–33, 1966.

[14] A. G. Barto, S. Singh, and N. Chentanez, "Intrinsically motivated learning of hierarchical collections of skills," in *Proceedings of the 3rd International Conference on Development and Learning*, pp. 112–19, 2004.

[15] M. Lopes and P.-Y. Oudeyer, "Guest editorial active learning and intrinsically motivated exploration in robots: Advances and challenges," *IEEE Transactions on Autonomous Mental Development*, vol. 2, no. 2, pp. 65–69, 2010.

[16] B. Settles, "Active learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 6, no. 1, pp. 1–114, 2012.

[17] S. Tong, *Active learning: theory and applications*. Stanford University, 2001.

[18] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," *CoRR*, vol. abs/1703.02910, 2017.

[19] O. Sener and S. Savarese, "Active Learning for Convolutional Neural Networks: A Core-Set Approach," *ArXiv e-prints*, Aug. 2017.

[20] K. Wang, D. Zhang, Y. Li, R. Zhang, and L. Lin, "Cost-Effective Active Learning for Deep Image Classification," *ArXiv e-prints*, Jan. 2017.

[21] J. A. Coelho and R. A. Grupen, "A control basis for learning multifingered grasps," *Journal of Robotic Systems*, vol. 14, no. 7, pp. 545–557, 1997.

[22] M. Huber, W. S. MacDonald, and R. A. Grupen, "A control basis for multilegged walking," in *IEEE International Conference on Robotics and Automation*, Apr 1996.

[23] Y. Nakamura, *Advanced robotics: redundancy and optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[24] F. Takens *et al.*, "Detecting strange attractors in turbulence," *Lecture notes in mathematics*, vol. 898, no. 1, pp. 366–381, 1981.

[25] N. Roy and A. McCallum, "Toward optimal active learning through monte carlo estimation of error reduction," 2001.

[26] D. Ruiken, M. W. Lanighan, and R. A. Grupen, "Postural modes and control for dexterous mobile manipulation: the umass ubot concept," in *IEEE-RAS International Conference on Humanoid Robots*, Oct 2013.

[27] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, 2009.

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