Maximizing Information Gain in Partially Observable Environments via Prediction Rewards

Yash Satsangi
University of Alberta
ysatsang@ualberta.ca

Sungsu Lim
University of Alberta
sungsu@ualberta.ca

Shimon Whiteson
University of Oxford
shimon.whiteson@cs.ox.ac.uk

Frans A. Oliehoek
Technical University Delft
f.a.oliehoek@tudelft.nl

Martha White
University of Alberta
whitem@ualberta.ca

ABSTRACT

Information gathering in a partially observable environment can be formulated as a reinforcement learning (RL), problem where the reward depends on the agent's uncertainty. For example, the reward can be the negative entropy of the agent's belief over an unknown (or hidden) variable. Typically, the rewards of an RL agent are defined as a function of the state-action pairs and not as a function of the belief of the agent; this hinders the direct application of deep RL methods for such tasks. This paper tackles the challenge of using belief-based rewards for a deep RL agent, by offering a simple insight that maximizing any convex function of the belief of the agent can be approximated by instead maximizing a prediction reward: a reward based on prediction accuracy. In particular, we derive the exact error between negative entropy and the expected prediction reward. This insight provides theoretical motivation for several fields using prediction rewards—namely visual attention, question answering systems, and intrinsic motivation—and highlights their connection to the usually distinct fields of active perception, active sensing, and sensor placement. Based on this insight we present deep anticipatory networks (DANs), which enables an agent to take actions to reduce its uncertainty without performing explicit belief inference. We present two applications of DANs: building a sensor selection system for tracking people in a shopping mall and learning discrete models of attention on fashion MNIST and MNIST digit classification.

KEYWORDS

reinforcement learning; partially observability; information gain

ACM Reference Format:
Yash Satsangi, Sungsu Lim, Shimon Whiteson, Frans A. Oliehoek, and Martha White. 2020. Maximizing Information Gain in Partially Observable Environments via Prediction Rewards. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

To act intelligently, an agent must be able to reason about its uncertainty over certain variables in its environment. *Active perception* [4, 5] is the ability of an agent to reason about its uncertainty and take actions to reduce it. The aim of the agent is to take actions, to collect observations, that help it predict the value of an unknown[1] variable, say y at each time step t. For example, consider the sensor selection task [19, 41], where an agent has access to a set of available sensors to infer the unknown position of a person in a shopping mall (y). At each time step t, due to resource constraints, the agent must select a subset of the sensors from which to collect the observations. Another example is the visual attention task [33], where an agent must sequentially attend to parts of an image to determine if an object is present ($y = 1$ or 0).

The problem of taking informative actions—or selecting informative observations—to minimize (future) uncertainty can be formulated as a reinforcement learning problem. The agent takes actions and receives rewards for reducing uncertainty. The key question is how to compute such rewards. The most straightforward approach is as follows. At each time step, the agent maintains a probability distribution over the unknown variable y. The agent takes actions a^t to collect observations (denoted by z) about this unknown variable. The agent can then update its probability distribution over the unknown variable $p^{t+1}(y) = \Pr(y|z^1, z^2, \ldots, z^{t+1}, a^0, a^1, \ldots, a^t)$. The reward corresponds to expected reduction in uncertainty, after taking an action. A common definition for reduction in uncertainty is the expected *information gain* [29]: $\mathbb{E}_{\Pr(z^{t+1}|p^t,a)}[H(p^t) - H(p^{t+1})]$, where $H(p^t) = -\sum_{y \in Y}(p^t(y)\log(p^t(y))$ is the entropy of the the probability distribution p^t. The expectation is over the possible observations z^{t+1} if the agent takes action a.

Unfortunately, computing these rewards can be prohibitively expensive. Given a model of the world—the conditional probability distributions $\Pr(z^{t+1}|y^{0:t+1}, a^{0:t})$ and $\Pr(y^{t+1}|y^{0:t}, a^{0:t})$—the agent can perform explicit belief inference to exactly compute the information gain of taking an action and so compute the action that maximizes it [29, 41]. Such models must be either manually specified, or learned if a dataset is available, which requires substantial expert knowledge and significant human effort. Even when a model of the world is available, performing explicit belief inference can be expensive or even intractable. In such cases, approximate belief inference methods such as particle filters [13] or variational approximation [20] must be used to compute the information gain.

In this paper we present a simple model-free reinforcement learning approach that allows an agent to take actions that maximize its information gain *without* performing explicit belief inference. We start by presenting a simple insight that shows that any convex

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

[1] We use the term unknown variable instead of hidden variable, because we assume that we have access to this unknown variable during training, as is standard in supervised learning. A hidden variable, on the other hand, is never available.

function of the belief of an agent (about an unknown variable) can be approximated simply by using prediction rewards, for example, +1 for a correct prediction and 0 for an incorrect prediction. Given an arbitrary prediction reward, we establish the exact error bounds the agent would incur for acting greedily with respect to the given prediction reward in comparison to actions that maximize the information gain of the agent. We show that in principle the prediction rewards can be designed to optimize this error.

The practice of providing an agent with prediction rewards is common in sub-fields such as visual attention [33], question answering systems [37] and intrinsic motivation [40]; this work provides theoretical motivation for these strategies and further generalizes the types of rewards and prediction problems that can be considered. Furthermore, the framework put forth unifies disparate areas that are in fact working on similar approaches, namely the fields already using prediction rewards and fields where it is common to maximize information gain, including active perception [41], active sensing [30] and sensor placement [29].

We use the our theoretical result to develop *deep anticipatory networks* (DANs) as a principled framework to leverage the power of deep RL to minimize uncertainty without performing explicit belief inference. A DAN consists of two neural networks: a Q network that selects sensory actions and a model, M network that predicts the state of the world based on the observations generated by those sensory actions. The main idea behind DAN is to train the Q network and M network simultaneously: the Q network learns a Q-function that estimates how much each sensory action would help the M network to predict the current state. Given some ground truth data, the M network learns to predict the current state in a supervised way, given the observations generated by the sensory actions that were selected according to the Q-network.

Finally, we empirically test our algorithm in two settings: sensor selection and attention. We build a sensor selection system for tracking people that scales to a large number of people. Using DAN we learn a policy for sensor selection and we show its performance on test data (when deployed) in comparison to other baselines that reward the agent using a heuristic that is based on the coverage of the sensor. We also apply DAN to a visual attention task where an agent must predict an MNIST class given only a partial observation of it. Our experiments on the MNIST [32] and fashion MNIST [51] datasets show that formulating the visual attention tasks as a continual problem where the agent is rewarded throughout the episode is superior to the terminal reward formulation common in the literature.

2 PROBLEM SETTING

We model the world as a partially observable Markov decision process (POMDP) [24] with finite state, action and observation space. At each time step t, the environment is in hidden state $s \in \mathcal{S}$, the agent takes an action $a \in \mathcal{A}$ and the environment transitions to a new state $s' \in \mathcal{S}$. Additionally, the agent receives an observation $z \in \Omega$ that is correlated with a target variable $y \in Y = \{1, 2, \ldots, n_y\}$ that is a function of s, $y = I(s)$.

The aim of the agent is to predict the target correctly on each step. At each time step, the agent maintains a probability distribution over y given the previous actions and observations, $\Pr(y|z^t, z^{t-1} \ldots z^1, a^{t-1}, a^{t-2} \ldots a^0)$. After taking action a^t and receiving observation z^{t+1}, the agent can update the probability distribution $\Pr(y|z^{t+1}, z^t, \ldots, z^1, a^t, a^{t-1}, \ldots a^0)$ using the Bayes rule. This has been formalized as a ρPOMDP [2] where the reward is defined as the negative entropy of the probability distribution over y. This formulation, however, requires access to the true probability distributions of the POMDP. Instead, we only assume access to a labelled dataset for training, where for a sequence of observations we are given the corresponding targets. For a sensor selection task, such a dataset can be obtained by investing a one-time effort to collect and label sets of observations, without inferring or knowing anything about hidden states or the underlying probabilities.

3 A CONNECTION BETWEEN INFORMATION GAIN AND PREDICTION REWARDS

In this section we provide a bound between the negative entropy and prediction rewards, which correspond to rewarding the agent for correct predictions of the target variable. In particular, we show that prediction rewards provide a set of tangents that form a lower-bound to the negative entropy. We discuss at the end of the section how this implies that maximizing expected prediction rewards—as is done by a reinforcement learning agent—provides an effective proxy to maximizing expected information gain. We first provide an informal theorem statement, and then introduce the required notation to prove the main results.

Let $\mathbf{b} = (b_1, b_2, \ldots b_{n_y})$ denote a probability vector in an n_y dimensional vector space such that $\sum_{i \in \{1,2,\ldots n_y\}} b_i = 1$ ($Y = \{1, 2, \ldots, n_y\}$), and let $H(\mathbf{b})$ be the Shannon entropy defined by $H(\mathbf{b}) = -\sum_{i \in Y} b_i \log b_i$. The vector $\mathbf{b}$ corresponds to the agents prediction about the what target variable is most probable, given the history of observations. The goal of the agent is to select actions to maximize information gain, and so decrease the entropy of the probabilities $\mathbf{b}$: maximize the negative entropy. We can instead consider maximizing an expected 0-1 prediction reward for the most probable class, $\max_i b_i$.

Informal Theorem Statement: The difference between the negative entropy $-H(\mathbf{b})$ and the expected 0-1 prediction reward $\max_i b_i$ (shifted by the a constant that is the same on every step) is upper bounded by $-1 + \log(e + n_y - 1)$.

3.1 Main Theoretical Result

Let $\rho(\mathbf{b})$ be any convex function of the probabilities $\mathbf{b}$, such as $\rho(\mathbf{b}) = -H(\mathbf{b})$. The equation of a tangent plane to ρ is given by: $\langle \mathbf{b}, \nabla\rho(\mathbf{b_0}) \rangle + c_{\mathbf{b}_0}$, where $c_{\mathbf{b_0}}$ is a constant and $\nabla\rho(\mathbf{b}_0)$ is the gradient of ρ. Though generically complex to compute, $c_{\mathbf{b}_0}$ can be computed analytically for certain functions, using Fenchel conjugates (see Boyd and Vandenberghe [9] for a comprehensive introduction). Here, we describe the two most relevant properties for this paper:

Property 1: If $\rho(\mathbf{b})$ is convex, closed and differentiable, then $c_{\mathbf{b}_0}$ is the negative of the *Fenchel conjugate* of $\rho(\mathbf{b})$ at $\nabla\rho(\mathbf{b}_0)$, that is, $c_{\mathbf{b}_0} = -\rho^*(\nabla\rho(\mathbf{b_0}))$, where ρ^* denotes Fenchel conjugate of ρ [6, 9].

Property 2: The Fenchel conjugate of the negative entropy is the log-sum-exp function, $\log(\sum_i e^{x_i})$ [9, Page 93].

Property 1 and 2 give that for $\rho(\mathbf{b}) = -H(\mathbf{b})$, the constant term is $c_{\mathbf{b}_0} = -\log(\sum_{i=1}^{n} e^{\nabla\rho(\mathbf{b}_0)_i})$, where $\nabla\rho(\mathbf{b}_0)_i$ denotes the i^{th} entry in the vector $\nabla\rho(\mathbf{b}_0)$. Now, let $\hat{y} \in Y = \{1, 2, 3 \ldots, n_y\}$ denote a

Table 1: Summary of notation

$\hat{y}$	a random variable that denotes a prediction
h^t	the action (a)-observation(z) history $h^t = \langle a^0, z^1, a^1, \ldots, a^{t-1}, z^t \rangle$.
$\mathbf{b}$	denotes a probability vector
$\rho(\mathbf{b})$	a convex and differentiable function of $\mathbf{b}$
$\rho^*(\mathbf{b})$	the Fenchel conjugate of $\rho(\mathbf{b})$
$\nabla\rho(\mathbf{b})$	the gradient of $\rho(\mathbf{b})$
$\nabla\rho(\mathbf{b})_i$	the i^{th} entry in the vector $\nabla\rho(\mathbf{b})$
$R(y, \hat{y})$	the prediction reward function
$\mathbf{r}_j$	a reward vector, each entry r_i of $\mathbf{r}_j$ is the scalar reward agent gets for $\hat{y} = j$ when true $y = i$.
log	natural logarithm

prediction that is input to a reward function $R(y, \hat{y})$, which gives a scalar value $r_{i,j}$ for each combination of $i, j \in Y$. Let $R(y, \hat{y} = j)$, the reward vector associated with predicting y as j using $\hat{y}$ be denoted by the vector $\mathbf{r}_j$. That is, each entry r_i in $\mathbf{r}_j$ is the reward for predicting $\hat{y}$ as j when the true value of y is i. Given a probability vector $\mathbf{b}$, the expected reward for assigning $\hat{y} = j$ is

$$\rho'(\mathbf{b}, \hat{y} = j) = \langle \mathbf{b}, \mathbf{r}_j \rangle = \sum_{i \in Y} b_i r_{i,j}, \tag{1}$$

which leads to the following lemma.

Lemma 3.1. *If ρ is a closed, convex and differentiable function of $\mathbf{b}$ and $\mathbf{r}_j$ is in the set of all possible values of the gradients of ρ then $\rho'(\mathbf{b}, j) - \rho^*(\mathbf{r}_j) = \langle \mathbf{b}, \mathbf{r}_j \rangle - \rho^*(\mathbf{r}_j)$ is a tangent to the curve $\rho(\mathbf{b})$ at $\mathbf{b_0}$ that satisfies $\nabla\rho(\mathbf{b_0}) = \mathbf{r}_j$ for any fixed $j \in Y$.*

Proof. Property 1 imply that the equation of a tangent to the curve $\rho(\mathbf{b})$ is $\langle \mathbf{b}, \nabla\rho(\mathbf{b_0}) \rangle - \rho^*(\nabla\rho(\mathbf{b_0}))$. If $\mathbf{r}_j = \nabla\rho(\mathbf{b_0})$ then $\langle \mathbf{b}, \mathbf{r}_j \rangle - \rho^*(\mathbf{r}_j)$ is a tangent to the curve $\rho(\mathbf{b})$. The condition that $\mathbf{r}_j$ is in the set of all possible values of gradients of ρ is required for ρ^* to be defined (and for $\nabla\rho(\mathbf{b_0}) = \mathbf{r}_j$ to have a solution). □

We can use this lemma to show that the maximum over these tangent planes forms a lower bound on $\rho(\mathbf{b})$. When ρ is the negative entropy, this maximum over tangent planes precisely corresponds to the expected prediction reward, shifted by a constant as shown in Theorem 3.3.

Proposition 3.2. *If ρ is a closed, convex, and differentiable function of $\mathbf{b}$ and $\mathbf{r}_j$ is in the set of all possible values of the gradients of ρ then the maximum error between $\rho(\mathbf{b})$ and $\rho'(\mathbf{b}) \triangleq \max_{\hat{y} \in Y}(\langle \mathbf{b}, \mathbf{r}_{\hat{y}} \rangle - \rho^*(\mathbf{r}_{\hat{y}}))$ is bounded and positive for $\mathbf{b} \in$ dom ρ.*

Proof. Since $\rho'(\mathbf{b})$ is the maximum over a family of tangents to a convex function $\rho(\mathbf{b})$ it is guaranteed to be a lower bound to $\rho(\mathbf{b})$. Furthermore, if $\rho'(\mathbf{b})$ is defined for $\mathbf{b} \in$ dom ρ then this error is maximal either at one of the intersection points of the tangents or at the extreme points of the domain of $\mathbf{b}$. In both cases it is finite and positive and can be calculated exactly for given values of $\mathbf{r}_j$ and definition of $\rho(\mathbf{b})$. □

The above proposition bounds the error between a convex function and prediction rewards using its Fenchel conjugate. The Fenchel conjugate is known for several convex functions such as negative entropy (see Property 2), KL-divergence, and χ^2-divergence. Given an arbitrary prediction reward, we can derive exactly how well it approximates a given convex function, such as, negative entropy.

In the rest of this section we perform this analysis for the case where $\rho(\mathbf{b})$ is the negative belief entropy. We restrict ourselves to the common reward functions where the agent is rewarded with r' for correctly predicting y and penalized with r'' (or not rewarded $r'' = 0$) otherwise, with $r' \geq r''$

$$R(y, \hat{y}) = \begin{cases} r' & \text{if } y = \hat{y}, \forall y, \hat{y} \in Y; \\ r'' & \text{otherwise.} \end{cases} \tag{2}$$

Using Proposition 3.2 the difference between $\rho(\mathbf{b})$ and $\rho'(\mathbf{b})$ can be quantified as:

$$\rho(\mathbf{b}) - \rho'(\mathbf{b}) = -H(\mathbf{b}) - \max_{j \in Y}(\langle \mathbf{b}\mathbf{r}_j \rangle - \rho^*(\mathbf{r}_j)) \tag{3}$$

For the reward defined in (2), $\mathbf{r}_1$ is the vector $(r', r'', r'', \ldots, r'')$, $\mathbf{r}_2$ is the vector $(r'', r', r'', \ldots, r'')$ and so on. We start by observing that $\rho^*(\mathbf{r}_j)$ is a constant term independent of j and it evaluates to: $\rho^*(\mathbf{r}_1) = \rho^*(\mathbf{r}_2) = \cdots = \rho^*(\mathbf{r}_{n_y}) = \log(e^{r'} + (n_y - 1)e^{r''})$. The term $\max_{j \in Y} \langle \mathbf{b}\mathbf{r}_j \rangle$ can be simplified as max over the following terms $\{(b_1 r' + b_2 r'' + \ldots b_{n_y} r''), (b_1 r'' + b_2 r' + \ldots b_{n_y} r''), \ldots, (b_1 r'' + b_2 r'' + \ldots b_{n_y} r')\}$. Since $b_1 + b_2 + \ldots b_{n_y} = 1$ and since $r' > r''$, the maximum over these aforementioned terms is simply equal to: $\max_{j \in Y} \langle \mathbf{b}, \mathbf{r}_j \rangle = r' \max_{i \in Y} b_i + r''(1 - \max_{i \in Y} b_i)$.

Using above simplifications ρ' can be written as:

$$\rho'(\mathbf{b}) = (r' - r'') \max_{i \in Y} b_i + r'' - \log(e^{r'} + (n_y - 1)e^{r''}), \tag{4}$$

and the difference between $\rho(\mathbf{b}) - \rho'(\mathbf{b})$ can be characterized as:

$$\rho(\mathbf{b}) - \rho'(\mathbf{b}) = -H(\mathbf{b}) - (r' - r'') \max_{i \in Y} b_i - r'' + \log(e^{r'} + (n_y - 1)e^{r''}). \tag{5}$$

This equation provides the exact error from using the tangents, rather than the negative entropy, and can be queried for a specific $\mathbf{b}$ to provide insights into the level of approximation. We can, however, also bound this difference for all $\mathbf{b}$, as given in the next theorem.

Theorem 3.3. *Let $m = r' - r''$ and let[2] $1 \leq m \leq n_y$. For every $\mathbf{b} \in [0, 1]^{n_y}$ s.t. $\sum_{i \in Y} b_i = 1$,*

$$\rho(\mathbf{b}) - \rho'(\mathbf{b}) \leq \max\{\epsilon_1, \epsilon_2\} + -r'' + \log(e^{r'} + (n_y - 1)e^{r''})$$

$$\text{where} \quad \epsilon_1 = \log\left(\tfrac{1}{r' - r''}\right) - 1, \text{ and } \quad \epsilon_2 = \log\left(\tfrac{1}{n_y}\right) - \frac{(r' - r'')}{n_y}.$$

Proof. Starting from (5),
$\rho(\mathbf{b}) - \rho'(\mathbf{b}) = -H(\mathbf{b}) - (r' - r'') \max_{i \in Y} b_i - r'' + \log(e^{r'} + (n_y - 1)e^{r''})$.
Wlog, let $b_1 = \max_{i \in Y} b_i$, then

$$\rho(\mathbf{b}) - \rho'(\mathbf{b}) = -H(\mathbf{b}) - (r' - r'')b_1 - r'' + \log(e^{r'} + (n_y - 1)e^{r''}). \tag{6}$$

For a fixed maximal element b_1, the optimal choice to maximize $-H(\mathbf{b})$ is to concentrate the remaining probability mass on as few elements as possible subject to constraints that $b_i \leq b_1$ for $i \neq 1$ and $i \in Y$. This means setting $b_2 = 1 - b_1$ if $b_1 > 0.5$. Of course, b_1 might be less than 0.5. In general, for some $k \geq 1$, we set $b_{1:k} = b_1$ and

[2] We can get bounds for $m < 1$ and $m > n_y$, but this introduces more cases and reduces the clarity of the result. We focus the result for the most common m.

then $b_{k+1} = 1 - kb_1$ for the remaining probability. The resulting $-H(\mathbf{b}) = kb_1 \log(b_1) + (1 - kb_1)\log(1 - kb_1)$ upper bounds the negative entropy for any distribution with max element b_1.

For $m \doteq r' - r'' \geq 0$, define

$$g(b_1) \doteq kb_1 \log(b_1) + (1 - kb_1)\log(1 - kb_1) - mb_1$$

where $n_y \geq k \geq 1$ and $b_1 \in [\frac{1}{n_y}, \frac{1}{k}]$. Finding b_1 that is maximal for g will will be the same b_1 that is maximal for the rhs of (6) and so give an upper bound on $\rho(\mathbf{b}) - \rho(\mathbf{b}')$. Therefore, we only need to find an upper bound on $g(b_1)$ to prove the theorem. First, we know that $g(b_1)$ is a convex function for b_1 where $\frac{1}{n_y} \leq b_1 \leq \frac{1}{k}$ because

$$\begin{aligned} g'(b_1) &= k + k\log(b_1) - k\log(1 - kb_1) - k - m \\ &= k\log(b_1) - k\log(1 - kb_1) - m, \end{aligned}$$

and

$$\begin{aligned} g''(b_1) &= \frac{k}{b_1} - \frac{k}{1 - kb_1}(-k) \\ &= \frac{k}{b_1} + \frac{k^2}{1 - kb_1} > 0 \end{aligned}$$

Therefore $g(b_1)$ is maximal at the endpoints $b_1 = \frac{1}{n_y}$ or at $b_1 = \frac{1}{k}$, where $n_y \geq k \geq 1$.

If $b_1 = \frac{1}{k}$ ($b_1 \to \frac{1}{k}$ to be more precise), then

$$g\left(b_1 = \tfrac{1}{k}\right) = \log\left(\tfrac{1}{k}\right) + 0 - \frac{m}{k}$$

We can again reason about this function, and find the k that makes this maximal and so provides an upper bound on g. Let $f(k) \doteq \log\left(\frac{1}{k}\right) - \frac{m}{k}$. $f'(k) = -\frac{1}{k} + \frac{m}{k^2} = 0$ gives $k = m$. Further, for $1 \leq m \leq n_y$, we know this function is concave for the region $0 \leq k \leq 2m$ because $f''(k) = \frac{1}{k^2} - \frac{2m}{k^3} < 0$ if $k \leq 2m$. Since this stationary point $k = m$ is in this concave region, we know it is a local maxima. Further, for $k > 2m$, the function becomes convex, but only decreases because there is no stationary points other than $k = m$. Therefore, for this case, the maximal g is

$$\epsilon_1 = \log\left(\tfrac{1}{m}\right) - 1.$$

If $b_1 = \frac{1}{n_y}$, then

$$\epsilon_2 = g\left(b_1 = \tfrac{1}{n_y}\right) = \log\left(\tfrac{1}{n_y}\right) - \frac{m}{n_y}.$$

Putting it all together, since we found $\max(\epsilon_1, \epsilon_2)$ as an upper bound on $g(b_1)$ for all b_1, we get that

$$\begin{aligned} \rho(\mathbf{b}) - \rho'(\mathbf{b}) &= g(b_1) - r'' + \log(e^{r'} + (n_y - 1)e^{r''}) \\ &\leq \max(\epsilon_1, \epsilon_2) - r'' + \log(e^{r'} + (n_y - 1)e^{r''}). \end{aligned}$$

□

Corollary 3.4 (0-1 Prediction Rewards). *If $r' = 1$ and $r'' = 0$, then for every $\mathbf{b} \in [0, 1]^{n_y}$ s.t. $\sum_{i \in Y} b_i = 1$,*

$$\rho(\mathbf{b}) - \rho'(\mathbf{b}) \leq -1 + \log(e + n_y - 1).$$

Proof. Direct application of Theorem 3.3. Substituting $m = r' - r'' = 1 - 0 = 1$, we get $\epsilon_1 = -1$ and $\epsilon_2 = \log(\frac{1}{n_y}) - \frac{1}{n_y}$. Since Since $-1 \geq \log(\frac{1}{n_y}) - \frac{1}{n_y}$ for $n_y \geq 1$, and substituting $r' = 1$ and $r'' = 0$, we get $\rho(\mathbf{b}) - \rho'(\mathbf{b}) \leq -1 + \log(e + n_y - 1)$. □

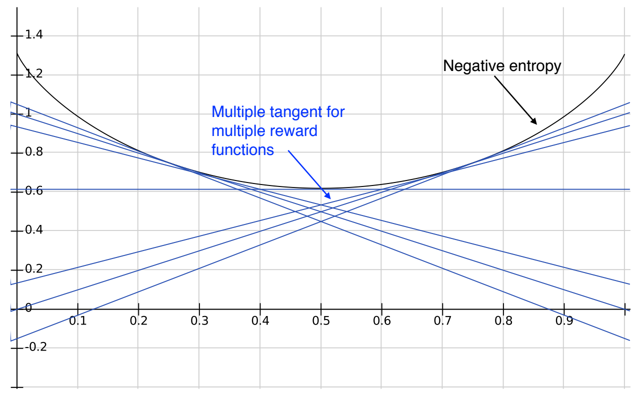

Figure 1: Approximation induced by prediction rewards to a translated negative entropy curve.

3.2 Consequences of the Theory

Computing the optimal action The previous results showed that $\rho'(\mathbf{b}) = \max_{j \in Y}\langle \mathbf{b}, \mathbf{r}_j\rangle - \rho^*(\mathbf{r}_j)$ is an approximation to $\rho(\mathbf{b})$ if ρ is convex. Fortunately, to compute the action $a^{*,t}$ that maximizes the information gain of the agent we do not need to compute $\rho^*(\mathbf{r}_j)$ as it is independent of the actions and is a constant for a fixed $j = \arg\max_{j \in \mathcal{S}}\langle \mathbf{b}, \mathbf{r}_j\rangle - \rho^*(\mathbf{r}_j)$ equal to $\log(e^{r'} + (n_y - 1)e^{r''})$ (for reward defined in (2)). The agent can approximate $a^{*,t} = \arg\max_{a \in \mathcal{A}} \mathbb{E}[H(p^t) - H(p^{t+1})]$ (here p^{t+1} depends on a) by picking actions that maximize $\mathbb{E}_{\Pr(z^{t+1}|p^t, a)}[\max_{\hat{y} \in Y} \sum_y p^{t+1}(y)R(y, \hat{y})]$ or an sample estimate of it. This sample estimate can be computed without maintaining an explicit distribution p^t but instead by training an agent to make correct predictions based on history of action and observations. In the next section we do exactly that.

Reducing the error to zero: The error between prediction reward and information gain can be further reduced by giving the agent the choice of selecting from one of many prediction variables, each of which defines a separate prediction reward as shown in Figure 1. To do so we define multiple prediction reward $R^l(y, \hat{y}^l)$, each of which takes as input a separate prediction variable $\hat{y}$. Furthermore, define $\rho'(\mathbf{b}) = \max_{\{l,j\} \in \{M \times Y\}}(\langle \mathbf{b}, \mathbf{r}^l_j\rangle - \rho^*(\mathbf{r}^l_j))$, where M is the set of all values l can take (4 in this case). Each of these reward functions projects a tangent (or tangent hyperplane) to the original ρ, in this case the entropy, with $\hat{y}^4$ (corresponding to the blue tangent line parallel to x-axis) being unique in that it rewards the agent equally for correct or incorrect predictions. In this way, $\hat{y}^4$ offers the agent an the option to abstain, which is optimal when it is most uncertain (bottommost point of the negative entropy curve). As more and more tangents are defined using new prediction variables, the upper surface of the tangents can approximate the original ρ more and more closely.

3.3 Connection to Existing Literature

An important consequence of this section is that it ties the problem of maximizing information gain [29, 38, 41, 52] to many recent deep RL approaches, that are based on making a correct predictions at the end of an episode [21, 33, 36, 37, 39]. For example, both visual attention approaches [17, 33, 36] and question answering systems [37] train deep RL agents on a 0-1 prediction reward for classifying an image and answering a query correctly respectively. Visual attention, question answering systems, intrinsic motivation, active perception, sensor placement, and active sensing are separate sub-fields of artificial intelligence, that do not necessarily refer to each other very often, however, our results show that they are in fact solving the same problem (or a close approximation of it).

Our theoretical results are related to ρPOMDPs [2] and POMDP-IR [44] and their equivalence as established in [41]. This works shows that given a ρPOMDP—which has a reward function defined by a set of vectors that approximate a convex curve—it is possible to design an equivalent POMDP-IR with a prediction reward. However, they do not give any direction as to how to compute the vectors that closely approximate the convex curve. We circumvent the procedure of computing these vectors by using the theory of Fenchel conjugates that gives us direct and analytical expressions for computing the tangent hyperplanes to a convex curve. Consequently, we are able to derive the exact error bound caused by a prediction reward, for example, a 0-1 prediction reward.

4 DEEP ANTICIPATORY NETWORKS

The insights in the previous section motivate that we no longer need an explicit belief to evaluate the information gain of an action, and can instead employ existing deep RL algorithms such as deep Q-learning to learn a policy that maximizes prediction rewards. In this section we introduce *deep anticipatory networks* (DANs), an algorithm that enables an agent to take actions that help it predict the current and future values of y accurately. DAN consists of two different networks: a Q network and a model M network. The Q network takes as input the action-observation history $h^t = \langle a^0, z^1, a^1, \dots, a^{t-1}, z^t \rangle$ of the agent and outputs the Q-values of all available actions. The agent takes an action a^t (t denoting the current time step) that maximizes the Q-values and receives an observation z^{t+1} that is correlated with the unknown variable y at time step $t+1$. This new action-observation pair is added to the history and fed into the M network.

The M network takes as input the agent's action-observation history and predicts the value of the unknown variable. The M network is trained in a supervised fashion using the agent's dataset of action-observation histories labelled with the corresponding true **Y**. If the M network predicts the state of the world correctly, then the Q network is rewarded +1 and otherwise 0. In other words, the Q network is rewarded for learning a Q-function that takes actions that help the model to predict the state from partial observations. Figure 2 illustrates an abstract DAN.

To train DAN, both the Q and the M networks are trained simultaneously on small mini-batches of data. Since one of the components in DAN is DQN, we additionally borrow the techniques used to train DQNs to train DAN. Specifically, each history-action pair that the agent encounters is stored in an experience buffer to be sampled later to train both the Q and the M networks. We maintain two separate target networks for Q and M networks to get stable target values when updating the Q network.

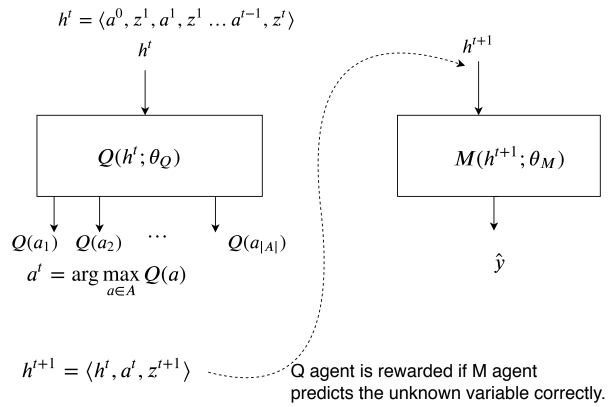

Figure 2: An abstract model of DAN that consists of a Q network and an M network. The Q network controls the input to the M network and the M network controls the reward the Q network gets.

In each iteration, for each episode, the agent follows the policy that is greedy with respect to the Q-values of the Q network. The accumulated experience is added to the experience buffer in the form of the tuple $\langle h^t, a^t, r^{t+1}, h^{t+1}, y^{t+1} \rangle$ that is later used to train the Q network. The observations z^{t+1} and the true y^{t+1} are obtained from the dataset while the reward r^{t+1} is obtained from the target M network. At each time step, the agent samples random experience tuples from the experience buffer and updates θ_{Q} using a Q-learning update, with a target network. Once θ_{Q} is updated, θ_{M} is updated by gradient descent with a cross-entropy loss: $\theta_{\mathrm{M}} = \theta_{\mathrm{M}} + \alpha \nabla_\theta L_{\mathrm{M}}(\theta_{\mathrm{M}})$, where $L_{\mathrm{M}}(\theta_{\mathrm{M}}) = \texttt{cross-entropy}(\mathrm{M}(h^t | \theta_{\mathrm{M}}), y)$.

The idea of learning sensory actions (Q) and a predictive model (M) simultaneously have appeared in earlier literature, with [33] the closest of all architectures. Similar architecture are presented in [3, 17, 36]. The specific architectures in [33], [17], [3] and [36] differ, but they share a common idea: to train the neural network architecture with policy gradient methods on a single unified objective, for example, using REINFORCE [50] or proximal policy optimization [42]. We chose to use DQN, particularly because it facilitates the use of factorization of the state-space and because we use knowledge of the exact action-values for the sensor selection system.

Otherwise, this choice is not critically different: either policy gradient methods or Q-learning methods can be used to solve this problem. A more interesting distinction is in the fact that the DAN architecture makes the it clear how general RL problem definitions can be used. It is common to model the problem of classification as a terminal-reward problem where the agent is rewarded only at the end of the episode (after a fixed number of steps). This is applicable when y is not changing with time. We explicitly formulate this problem as a continual problem where the agent is rewarded at each time step if it correctly prediction the unknown variable y. Such a formulation is critical when y changes with time, for example, in the sensor selection problem. But even in cases when y does not change with time, our experiments suggests that providing feedback on every step leads to faster learning. This has important implications for training visual attention and question answering systems.

5 EXPERIMENTS

In this section we present two different applications of DAN: sensor selection for tracking people in a shopping mall and discrete visual attention for classifying MNIST digits. Code for our experiments is available online.[3]

We apply DAN to build a sensor selection system that we demonstrate can scale to arbitrarily large spaces. We use DAN to learn a sensor selection policy to track people in a shopping mall. The problem was extracted from a real-world dataset collected in a shopping mall [8]. The dataset was gathered over 4 hours using 13 CCTV cameras. Each person's position is represented by x-y coordinates, where both x and y take values in the set $\{1, 2, \dots 50\}$ resulting in a total of 50×50 cells. At each time step, the agent selects one camera out of 10 to get an observation about the location of the person in the image. Each camera covers a subset of 50×50 cells and provides a noisy observation regarding the position of the person. If the person is not present in the image then a null observation is received. This observation along with the selected camera is passed to the M network that predicts which of the $50 \times 50 (= 2500)$ cells the person occupies.

5.1 Sensor Selection

The number of states of the world increases rapidly with the number of people in the scene. To address this, we assume that the movement of a person in the x-direction is independent of his/her movement in the y-direction and vice-versa. We train two separate DAN architectures, DAN-x and DAN-y for separately predicting the x and y coordinates of the position of a person. Furthermore, we assume that the movement of people present in the scene is independent of each other. These approximations let us build a sensor selection system that can scale to larger spaces and numbers of people.

For sensor selection, both the Q and M networks share an identical architecture: three fully connected layers of output size 60, 30, and 128, followed by a recurrent layer of output size 128, and a final fully connected output layer of size 10 (the number of cameras) and 51 (the number of possible cells + null observation). Strictly speaking, here we are using deep recurrent Q network (DRQN)[18] in the DAN architecture instead of DQN. We use ReLU activation for all fully connected layers except the last, and use L2 weight regularization (scale=0.01). We use the discount factor $\gamma = 0.99$ and perform a double DQN [47] update to train Q network with the Adam optimizer [27]. We also train following baselines for comparison. **Coverage baseline** — train only the Q network using the popular state-based reward (i.e., reward the agent for selecting the camera corresponding to the person's current location and getting a positive observation) without the M net. It uses its observations as final predictions, and during evaluation the agent only has to obtain a positive observation to be considered to have made a correct prediction. **Random Policy baseline** — only train the M network with a random policy for camera selection. **DAN + Coverage baseline** — use a combination of DAN reward and coverage reward, in

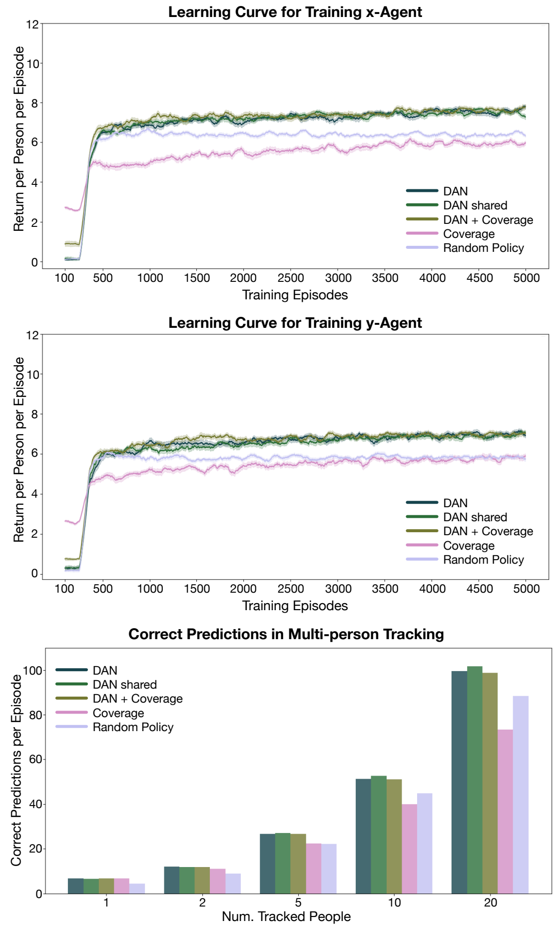

Figure 3: Training curves and multi-person tracking results for sensor selection for DAN agent.

which case the agent is rewarded +1 for correctly predicting the state, +0.2 for not being correct but getting a positive observation, and 0 otherwise (but we still use the M network to predict the x and y coordinates). **DAN-shared** is when the Q and M networks share representations, that is the top layers share the same parameters for both the Q and M network, but the last layer is separated.

We also compare to a model-based particle filter approach and to a DAN model that is trained on a terminal reward (only provided at the end of the episode during training) instead of a continuous reward that is provided at each time step of the training. However, these two baselines performed particularly poorly. The particle filter based approach that had access to the learned transition dynamics (under Gaussian assumption) and the true observations noise results in a performance of 1.9 (less than 1/3 of DAN's) total reward per trajectory for 400 particles and saturates at 3.5 (less than 1/2 of DAN's performance) for 1500 particles and after tuning many parameters of the particle filter. Rewarding an agent only at the termination of the episode does not work either as for tracking the agent needs continuous feedback. We did not experiment further with these baselines.

[3] https://github.com/sungsulim/DeepAnticipatoryNetworks

For training DAN and baseline methods, we swept over the exploration probability $\epsilon : \{0.1, 0.3, 0.5\}$ and Q/M network learning rate: $\{0.01, 0.001, 0.0001\}$. For all methods we found $\epsilon = 0.1$ and Q/M network step-size $= 0.001$ to work the best. We first train x, y-agents for tracking a single person, and the training curves are shown in Figure 3 (a) and (b). We perform 25 runs for each agent. We use track length of 12, sampled from the training track dataset, and train it for 60,000 steps (or 5000 episodes). We also collect experience without training for 3,000 steps (250 episodes). For updating the networks, we use a mini-batch of size 4 to sample episodes from the replay buffer, with trace length of 8 (not updating on the first 4 steps of the episode).

We test the trained DAN agents in single-person and multi-person tracking. For single person tracking, at each time step the agent queries the Q-values from both the DAN-x agent and DAN-y agent and selects the camera (action) that maximizes the average Q-value among all the available actions. For multi-person tracking we transfer the policy learned for single-person tracking to track multiple people. The same Q network is used to compute the Q-values of selecting each camera for each person independently. Finally, the agent selects the camera that maximizes the average Q-value from all the people present in the scene, and the M network predicts the location of all the people based on the observation. During evaluation the agent is rewarded +1 only if both x and y coordinates are predicted correctly. Figure 3 (c) shows the result of multi-person tracking of 500 test tracks. In all cases, variants of DAN outperform the random and coverage baselines. Surprisingly, sharing representation is comparable to DAN with separate representations for Q and M networks, which is good as sharing representations reduces the number of parameters.

5.2 Discrete attention

In this set of experiments, we apply DAN to learn discrete models of attention in which the agent can observe the unknown variable only via a discrete set of available glimpses. As compared to sensor selection here the hidden variable is not changing and selecting one of the available glimpse does not necessarily provides the agent enough information for predicting the digit in the image. So ideally the agent must learn representation that help it predict the digits from as little glimpses as possible. At the start of the episode the agent receives a blank image and as it makes its selections, glimpses of the images are revealed. This task is discussed in earlier papers [33] with different glimpse styles depending on the motivation of the paper. However, many earlier approaches based on deep reinforcement learning model this task with a terminal reward the agent receives the feedback (reward and true label) about its policy only at the end of the episode. Our formulation models this as a continuous feedback task, where the agent makes a prediction at each time step and is rewarded at every time step for making correct predictions. Since during the training the true label is available to the agent, there is no point of making this label available to the agent only at the end of the episode.

For this experiment, the Q and M networks are identical convolutional neural networks (CNN) with two convolutional layers. This is followed by a max pooling layer and two fully connected layers with a dropout [45] probability of 0.5. ReLUs are used as activation

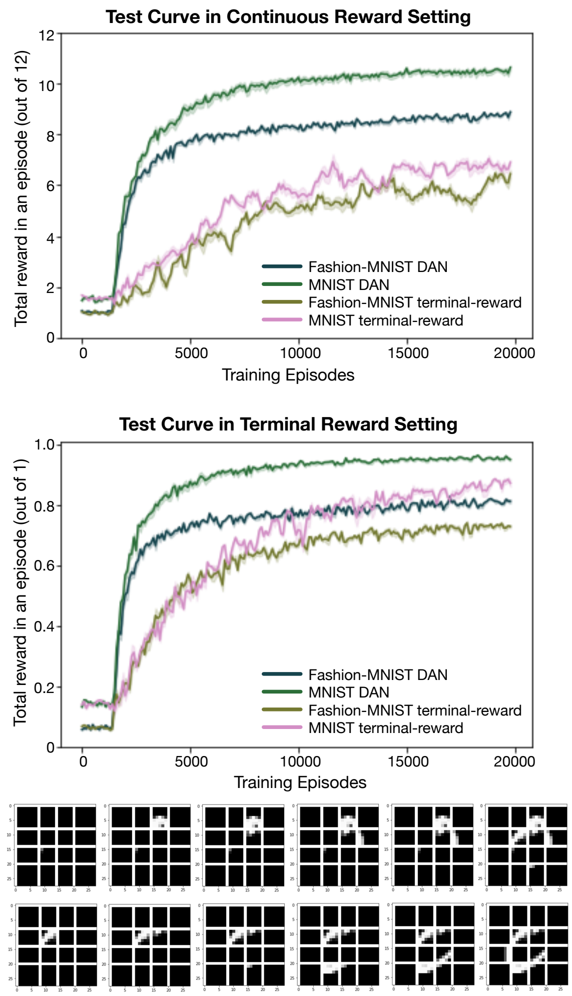

Figure 4: (top, middle) Performance results for discrete attention in continuous/terminal reward setting averaged over 10 runs, (bottom) Sequence of MNIST glimpses selected by the DAN agent for two separate examples.

units for all layers. The length of the episode is kept to 12 and the networks are updated every 4 steps. A learning rate of 0.0005 (after performing a parameter sweep over {0.05, 0.005, 0.0005}) is used with the Adam optimizer [27]. An exploration probability of 0.05 is used throughout training but an exploration probability of 1 is used during the first 1500 episodes.

We compare and evaluate DAN trained with continuous reward and DAN trained with terminal reward in two different setting (a) continuous reward and in (b) terminal reward settings. For the evaluation in the continuous reward setting the agent is rewarded at each time step for an episode of length 12 (so the agent can earn a maximum reward of 12) where as in the terminal reward setting the agent is evaluated on a terminal reward that the agent receives at the end of the episode. Figure 4 shows the average test reward on 500 test images (sampled from a set of 10000 test images at every evaluation) as a function of the training episode for both MNIST and fashion MNIST. The top figure shows the results for when the agent is rewarded at each time step and the middle figure shows results when evaluating on a terminal reward. In both settings the agent trained on continuous reward is significantly faster than the terminal reward setting simply because (a) it is simultaneously trained to select glimpses that can most quickly identify the classes as well as to identify classes from as few glimpses as possible; (b) it better uses the same set of experience to make more updates to its parameters because of the continuous feedback. DAN with terminal rewards performs particularly poorly in the continuous reward setting, as the M network in the terminal reward DAN is not trained to predict the class from smaller number of glimpses. Furthermore, the results also show that, at least for MNIST, it is possible to identify the digits from only one or two glimpses, as the DAN agent gets an average reward of more than 10 out of 12 on test images, whereas for the fashion MNIST, correctly predicting the right class requires a couple of more glimpses.

6 RELATED WORK

Prediction rewards are popular in reinforcement learning, for example, visual attention models [17, 33], question answering systems [11, 37], learning active learning strategies [3], intrinsic motivation [40]. On the other hand, literature such as active perception [41], sensor placement [29], and active sensing [30], formulate the problem of either sensor management/selection/fusion with information gain as the objective function. Our paper ties these fields together by exactly establishing the relationship between prediction rewards and information gain.

Model-based methods as proposed in various active perception [1, 5, 10, 12, 26, 48, 53] and sensor selection [19, 23, 30, 35, 43, 46, 49] literature require a model of the world for their application. The model-free nature of DAN lets us to deploy deep RL machinery for sensor selection in a principled manner. Recently, attempts to perform *online active perception* [15, 36] either focus on fast subset selection or on neural network architecture improvement, e.g., for MNIST, but offer no insight on connecting prediction rewards to information gain.

Neural models of visual attention, such as that of [33] and [17], consider a classification task where the unknown variable is not changing at every time step. Consequently they model the loss function as one conditioned on a terminal reward that the agent receives if it correctly classifies the image after certain time steps. By contrast, sensor selection is a continual learning setting where the position of the person is continuously changing and the agent must predict it at each time step using noisy observations. Moreover, the agent in the classification task is free to adjust the size and shape of the glimpse. By contrast, in sensor selection the agent can only attend to the scene with a fixed (already deployed) set of glimpses that cannot be resized.

Approaches that use intrinsic motivation [40] and auxiliary tasks [21] use the prediction reward as a means to train an agent to solve a specific task. The performance of the policy is evaluated on an *extrinsic* state-based reward; the goal is not prediction accuracy. By contrast, our aim is to maximize the prediction reward and not use it achieve any other target.

DANs are related to learning in POMDPs/MDPs [22, 25] but are designed to learn hidden representations of the world as opposed to the transition or observation function after assuming/designing the representation of the world. Generative adversarial networks (GANs) [16] and DIAYN [14] train two different networks on each other's feedback. However, GANs assume an adversarial relationship between the two networks leading to a min-max formulation of the final objective, while DANs lead to max-max formulation of the final objective. DIAYN [14] consists of two networks, one of which tries to help the other discriminate between objects in order to learn various skills, whereas our aim is to predict the unknown variable and maximize the prediction reward in itself.

Neural estimators based on variational lower bound to KL divergence [7, 34] do not acknowledge the connection between prediction rewards and negative entropy as we do. These approaches also do not categorize the error between the variational lower bound and information gain as we do, which can be further exploited to vanish this error. Thanks to the theory of convex duality, our insights are extendible to any convex functions of the belief and not just KL-divergence. Furthermore, these approaches propose an estimator but do not demonstrate the use of these estimator in a partially observable setting for sensor selection as we do.

Our results are also related to ρPOMDP [2] and POMDP-IR [44] and their equivalence as established in [41]. Apart from the distinction made earlier in Section 3, this paper present a deep reinforcement learning algorithm as compared to a model-based planning method they propose. Approaches [28, 31] that model active perception tasks with surrogate state-based rewards are fundamentally different from our formulation because of the definition of the reward.

7 CONCLUSIONS & FUTURE WORK

This paper established that an agent trying to maximize a prediction reward naturally maximizes a lower bound on the information gain. This insight helps tie together multiple disparate sub-fields of machine learning that use prediction rewards and information gain separately. The DAN algorithm follows as a consequence of these results, which uses a model-free RL agent to gather data, based on prediction rewards, while simultaneously learning the predictions. We show that the approach improves performance in both a sensor selection and two visual attention tasks.

8 ACKNOWLEDGEMENT

We would like to thank anonymous reviewers for their comments. This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). This project had received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 758824 —INFLUENCE).

erc
European Research Council
Established by the European Commission

REFERENCES

[1] P K Allen. 1985. *Object recognition using vision and touch.* Ph.D. Dissertation. U of Penn.

[2] M Araya-lópez, V Thomas, O Buffet, and F Charpillet. 2010. A POMDP extension with belief-dependent rewards. In *NeurIPS*. 64–72.

[3] P Bachman, A Sordoni, and A Trischler. 2017. Learning algorithms for active learning. In *ICML*. JMLR. org, 301–310.

[4] R Bajcsy. 1988. Active perception. *Proc. IEEE* 76, 8 (1988), 966–1005.

[5] R Bajcsy, Y Aloimonos, and J K Tsotsos. 2018. Revisiting active perception. *Autonomous Robots* 42, 2 (2018), 177–196.

[6] H Bauschke and Y Lucet. 2012. What is a fenchel conjugate? *Notices of the AMS* (2012), 44–46.

[7] M I Belghazi, A Baratin, S Rajeswar, S Ozair, Y Bengio, A Courville, and R D Hjelm. 2018. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062* (2018), 2122–2131.

[8] H Bouma, J Baan, S Landsmeer, C Kruszynski, G van Antwerpen, and J Dijk. 2013. Real-time tracking and fast retrieval of persons in multiple surveillance cameras of a shopping mall. In *Multisensor, Multisource Information Fusion*, Vol. 8756. 87560A.

[9] S Boyd and L Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[10] N DB Bruce and J K Tsotsos. 2009. Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision* 9, 3 (2009), 5–5.

[11] C Buck, J Bulian, M Ciaramita, W Gajewski, A Gesmundo, N Houlsby, and W Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. (2018), 1–15.

[12] W Burgard, D Fox, and S Thrun. 1997. Active mobile robot localization by entropy minimization. In *EUROMICRO Workshop*. IEEE, 155–162.

[13] A Doucet and A M Johansen. 2009. A tutorial on particle filtering and smoothing: Fifteen years later. (2009), 656–704.

[14] B Eysenbach, A Gupta, J Ibarz, and S Levine. 2018. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070* (2018), 1–22.

[15] M Ghasemi and U Topcu. 2019. Online active perception for partially observable Markov decision process with limited budget. *arXiv preprint arXiv:1910.02130* (2019), 1–7.

[16] I Goodfellow, J Pouget-Abadie, M Mirza, B Xu, D Warde-Farley, S Ozair, A Courville, and Y Bengio. 2014. Generative adversarial nets. In *NeurIPS*. 2672–2680.

[17] A Haque, A Alahi, and L Fei-Fei. 2016. Recurrent attention models for depth-based person identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1229–1238.

[18] M. Hausknecht and P Stone. 2015. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*. 29–37.

[19] A O Hero and D Cochran. 2011. Sensor management: Past, present, and future. *IEEE Sensors Journal* 11, 12 (2011), 3064–3075.

[20] M Igl, L Zintgraf, T A Le, F Wood, and S Whiteson. 2018. Deep variational reinforcement learning for POMDPs. In *ICML*. 2117–2126.

[21] M Jaderberg, V Mnih, W M Czarnecki, T Schaul, J Z Leibo, D Silver, and K Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. In *ICLR*. 1–17.

[22] M R James and S Singh. 2009. SarsaLandmark: an algorithm for learning in POMDPs with landmarks. In *AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 585–591.

[23] S Joshi and S Boyd. 2009. Sensor selection via convex optimization. *IEEE TSP* (2009), 451–462.

[24] L P Kaelbling, M L. Littman, and A R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* (1998), 99–134.

[25] S Katt, F A Oliehoek, and C Amato. 2017. Learning in POMDPs with Monte Carlo Tree Search. In *ICML (Proceedings of Machine Learning Research)*, Vol. 70. PMLR, 1819–1827.

[26] M D Kelly. 1971. Edge detection in pictures by computer using planning. *Machine Intelligence* (1971), 397–409.

[27] D Kingma and J Ba. 2014. Adam: A method for stochastic optimization. *ICLR*, 1–15.

[28] I Kostrikov, . Erhan, and S Levine. 2016. End to end active perception. In *NIPS 2016 Deep Learning Symposium*. 1–9.

[29] A Krause and C Guestrin. 2005. Near-optimal nonmyopic value of information in graphical models. In *UAI*. 324–331.

[30] C Kreucher, K Kastella, and A O Hero. 2005. Sensor management using an active sensing approach. *Signal Processing* 85, 3 (2005), 607–624.

[31] Q V Le, A Saxena, and A Y Ng. 2008. Active perception: Interactive manipulation for improving object detection. *Stanford University Journal* (2008), 1–9.

[32] Y LeCun, L Bottou, Y Bengio, and P Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[33] V Mnih, N Heess, A Graves, and K Kavukcuoglu. 2014. Recurrent models of visual attention. In *NeurIPS*. 2204–2212.

[34] S Mohamed and D J Rezende. 2015. Variational information maximisation for intrinsically motivated reinforcement learning. In *Neurips*. 2125–2133.

[35] E Monari and K Kroschel. 2010. Dynamic sensor selection for single target tracking in large video surveillance networks. In *IEEE AVSS*. IEEE, 539–546.

[36] H K Mousavi, G Liu, W Yuan, M Takáč, H Muñoz-Avila, and N Motee. 2019. A layered architecture for active perception: Image classification using deep reinforcement learning. *arXiv preprint arXiv:1909.09705* (2019), 1–7.

[37] K Narasimhan, A Yala, and R Barzilay. 2016. Improving information extraction by acquiring external evidence with reinforcement learning. In *EMNLP*. 2355–2365.

[38] S Nowozin. 2012. Improved information gain estimates for decision tree induction. In *ICML*. 571–578.

[39] J Oh, V Chockalingam, S Singh, and H Lee. 2016. Control of memory, active perception, and action in minecraft. In *ICML*. 2790–2799.

[40] D Pathak, P Agrawal, A A Efros, and T Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *ICML*. 2778–2787.

[41] Y Satsangi, S Whiteson, F A. Oliehoek, and M Spaan. 2018. Exploiting submodularity for scaling Up active perception. *Autonomous Robots* 42, 2 (2018), 209–233.

[42] J Schulman, F Wolski, P Dhariwal, A Radford, and O Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017), 1–12.

[43] M T J Spaan and P U Lima. 2009. A decision-theoretic approach to dynamic sensor selection in camera networks. In *ICAPS*. 279–304.

[44] M T J Spaan, T S Veiga, and P U Lima. 2015. Decision-theoretic planning under uncertainty with information rewards for active cooperative perception. *AAMAS* 29, 6 (2015), 1157–1185.

[45] N Srivastava, G Hinton, A Krizhevsky, I Sutskever, and R Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.

[46] L Tessens, M Morbee, H Aghajan, and W Philips. 2014. Camera selection for tracking in distributed smart camera networks. *ACM TOSN* 10, 2 (2014), 23.

[47] H Van Hasselt, A Guez, and D Silver. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*. 2094–2100.

[48] D Wilkes and J K Tsotsos. 1992. Active object recognition. In *CVPR*. IEEE, 136–141.

[49] J L Williams, J W Fisher, and A S Willsky. 2007. Approximate dynamic programming for communication-constrained sensor network management. *IEEE TSP* 55, 8 (2007), 4300–4311.

[50] R J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[51] H Xiao, K Rasul, and R Vollgraf. 2017. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[52] S C H Yang, D M Wolpert, and M Lengyel. 2016. Theoretical perspectives on active sensing. *Current opinion in behavioral sciences* 11 (2016), 100–108.

[53] Y Ye and J K Tsotsos. 1995. Where to look next in 3d object search. In *ISCV*. IEEE, 539–544.