Deep Active Inference for Partially Observable MDPs

Otto van der Himst[(✉)] and Pablo Lanillos

Department of Artificial Intelligence Donders Insitute for Brain,
Cognition and Behaviour Radboud University,
Montessorilaan 3, 6525 Nijmegen, HR, The Netherlands
o.vanderhimst@student.ru.nl, p.lanillos@donders.ru.nl

Abstract. Deep active inference has been proposed as a scalable approach to perception and action that deals with large policy and state spaces. However, current models are limited to fully observable domains. In this paper, we describe a deep active inference model that can learn successful policies directly from high-dimensional sensory inputs. The deep learning architecture optimizes a variant of the expected free energy and encodes the continuous state representation by means of a variational autoencoder. We show, in the OpenAI benchmark, that our approach has comparable or better performance than deep Q-learning, a state-of-the-art deep reinforcement learning algorithm.

Keywords: Deep active inference · Deep learning · POMDP · Control as inference

1 Introduction

Deep active inference (dAIF) [1–6] has been proposed as an alternative to Deep Reinforcement Learning (RL) [7,8] as a general scalable approach to perception, learning and action. The active inference mathematical framework, originally proposed by Friston in [9], relies on the assumption that an agent will perceive and act in an environment such as to minimize its free energy [10]. Under this perspective, action is a consequence of top-down proprioceptive predictions coming from higher cortical levels, i.e., motor reflexes minimize prediction errors [11].

On the one hand, works on dAIF, such as [2,12,13], have focused on scaling the optimization of the Variational Free-Energy bound (VFE), as described in [9,14], to high-dimensional inputs such as images, modelling the generative process with deep learning architectures. This type of approach preserves the optimization framework (i.e., dynamic expectation maximization [15]) under the Laplace approximation by exploiting the forward and backward passes of

the neural network. Alternatively, pure end-to-end solutions to VFE optimization can be achieved by approximating all the probability density functions with neural networks [1,3].

On the other hand, Expected Free Energy (EFE) and Generalized Free Energy (GFE) were proposed to extend the one-step ahead implicit action computation into an explicit policy formulation, where the agent is able to compute the best action taking into account a time horizon [16]. Initial agent implementations of these approaches needed the enumeration over every possible policy projected forward in time up to the time horizon, resulting in significant scaling limitations. As a solution, deep neural networks were also proposed to approximate the densities comprising the agent's generative model [1–6], allowing active inference to be scaled up to larger and more complex tasks.

However, despite the general theoretical formulation, current state-of-the-art dAIF, has only been successfully tested in toy problems with fully observable state spaces (Markov Decision Processes, MDP). Conversely, Deep Q-learning (DQN) approaches [7] can deal with high-dimensional inputs such as images.

Here, we propose a dAIF model[1] that extends the formulation presented in [3] to tackle problems where the state is not observable[2] (i.e. Partially Observable Markov Decision Processes, POMDP), in particular, the environment state has to be inferred directly high-dimensional from visual input. The agent's objective is to minimize its EFE into the future up to some time horizon T similarly as a receding horizon controller. We compared the performance of our proposed dAIF algorithm in the OpenAI CartPole-v1 environment against DQN. We show that the proposed approach has comparable or better performance depending on observability.

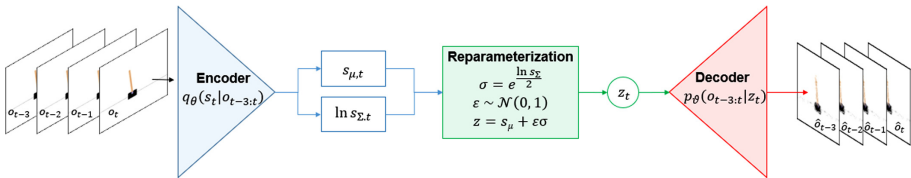## 2   Deep Active Inference Model

Fig. 1. Observations-state neural network architecture. The VAE encodes the visual features that are relevant to reconstruct the input images. The encoder network encodes observations to a state representation of the environment. The decoder reconstructs the input observations from this representation.

---

[1] The code is available on: https://github.com/Grottoh/Deep-Active-Inference-for-Partially-Observable-MDPs.

[2] We formulate image-based estimation and control as a POMDP—see [17] for a discussion.

We define the active inference agent's objective as optimizing its variational free energy (VFE) at time t, which can be expressed as:

$$-F_t = D_{KL}[q(s,a)\|p(o_t, s_{0:t}, a_{0:t})] \tag{1}$$

$$= -E_{q(s_t)}[\ln p(o_t|s_t)] + D_{KL}[q(s_t)\|p(s_t|s_{t-1}, a_{t-1})]$$
$$+ D_{KL}[q(a_t|s_t\|p(a_t|s_t)] \tag{2}$$

where o_t is the observation at time t, s_t is the state of the environment, a_t is the agent's action and $E_{q(s_t)}$ is the expectation over the variational density $q(s_t)$.

We approximate the densities of Eq. 2 with deep neural networks as proposed in [1,3,4]. The first term, containing densities $q(s_t)$ and $p(o_t|s_t)$ concerns the mapping of observations to states, and vice-versa. We capture this objective with a variational autoencoder (VAE). A graphical representation of this part of the neural network architecture is depicted in Fig. 1 – see the appendix for network details.

We can use an encoder network $q_\theta(s_t|o_{t-3:t})$ with parameters θ to model $q(s_t)$, and we can use a decoder network $p_\vartheta(o_{t-3:t}|z_t)$ with parameters ϑ to model $p(o_t|s_t)$. The encoder network encodes high-dimensional input as a distribution over low-dimensional latent states, returning the sufficient statistics of a multivariate Gaussian, i.e. the mean s_μ and variance s_Σ. The decoder network consequently reconstructs the original input from reparametrized sufficient statistics z. The distribution over latent states can be used as a model of the environment in case the true state of an environment is inaccessible to the agent (i.e. in a POMDP).

The second term of Eq. 2 can be interpreted as state prediction error, which is expressed as the Kullback-Leibler (KL) divergence between the state derived at time t and the state that was predicted for time t at the previous time point. In order to compute this term the agent must, in addition to the already addressed $q(s_t)$, have a transition model $p(s_t|s_{t-1}, a_{t-1})$, which is the probability of being in a state given the previous state and action. We compute the MAP estimate with a feedforward network $\hat{s}_t = f_\phi(s_{\mu,t-1}, a_{t-1})$. To compute the state prediction error, instead of using the KL-divergence over the densities, we use the Mean-Squared-Error (MSE) between the encoded mean state s_μ and the predicted state $\hat{s}$ returned by f_ϕ

The third and final term contains the last two unaddressed densities $q(a_t|s_t)$ and $p(a_t|s_t)$. We model variational density $q(a_t|s_t)$ using a feedforward neural network $q_\xi(a_t|s_{\mu,t}, s_\Sigma)$ parameterized by ξ, which returns a distribution over actions given a multivariate Gaussian over states. Finally, we model action conditioned by the state or policy $p(a_t|s_t)$. According to the active inference literature, if an agent that minimizes the free energy does not have the prior belief that it selects policies that minimize its (expected) free energy (EFE), it would infer policies that do not minimize its free energy [16]. Therefore, we can assume that our agent expects to act as to minimize its EFE into the future. The EFE of a policy π after time t onwards can be expressed as:
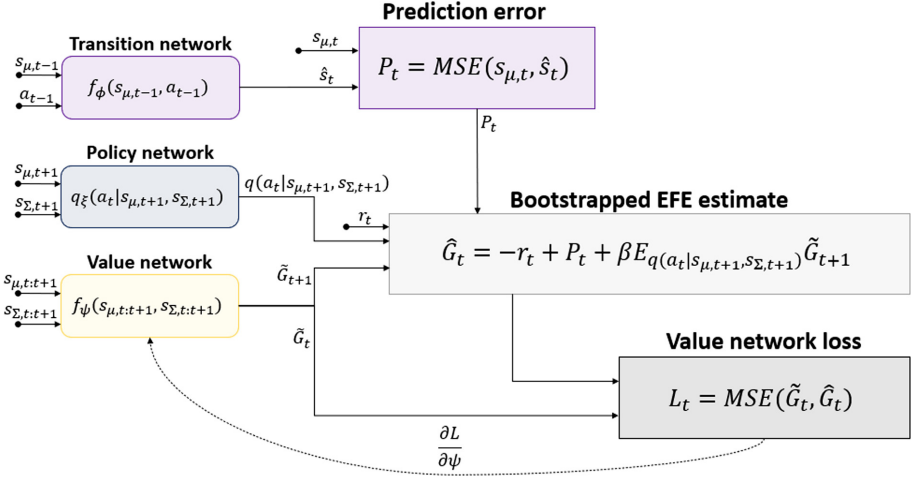
Fig. 2. Computing the gradient of the value network with the aid of a bootstrapped EFE estimate.

$$G_\pi = \sum_{\tau > t} G_{\pi,t}$$
$$G_{\pi,\tau} = \underbrace{-\ln p(o_\tau)}_{-r_\tau} + D_{KL}[q(s_\tau|\pi)\|q(s_\tau|o_\tau)] \tag{3}$$

Note that the EFE has been transformed into a RL instance by substituting the negative log-likelihood of an observation $-\ln p(o_\tau)$ (i.e. surprise) by the reward r_τ [3,18]. Since under this formulation minimizing one's EFE involves computing one's EFE for each possible policy π for potentially infinite time points τ, a tractable way to compute G_π is required. Here we estimate G_π via bootstrapping, as proposed in [3]. To this end the agent is equipped with an EFE-value (feedforward) network $f_\psi(s_{\mu,t}, s_{\Sigma,t})$ with parameters ψ, which returns an estimate $\tilde{G}_t$ that specifies an estimated EFE for each possible action. This network is trained with the aid of a bootstrapped EFE estimate $\hat{G}_t$, which consists of the free energy for the current time step, and a $\beta \in (0, 1]$ discounted value net approximation of the free energy expected under $q(a|s)$ for the next time step:

$$\hat{G}_t = -r_t + D_{KL}[q(s_t)\|q(s_t|o_t)] + \beta E_{q(a_{t+1}|s_{t+1})}\tilde{G}_t \tag{4}$$

In this form the parameters of $f_\psi(s_{\mu,t}, s_{\Sigma,t})$ can be optimized through gradient descent on (see Fig. 2):

$$L_t = MSE(\tilde{G}_t, \hat{G}_t) \tag{5}$$

The distribution over actions can then at last be modelled as a precision-weighted Boltzmann distribution over our EFEs estimate [3,16]:

$$p(a_t|s_t) = \sigma(-\gamma\tilde{G}_t) \tag{6}$$

Finally, Eq. 2 is computed with the neural network density approximations as –
see Fig. 3.

$$-F_t = - E_{q_\theta(s_t|o_{t-3:t})}[\ln p_\vartheta(o_{t-3:t}|z_t)]$$
$$+ MSE(s_{\mu,t}, f_\phi(s_{\mu,t-1}, a_{t-1}))$$
$$+ D_{KL}[q_\xi(a_t|s_{\mu,t}, s_{\Sigma,t})\|\sigma(-\gamma f_\psi(s_{\mu,t}, s_{\Sigma,t}))] \tag{7}$$

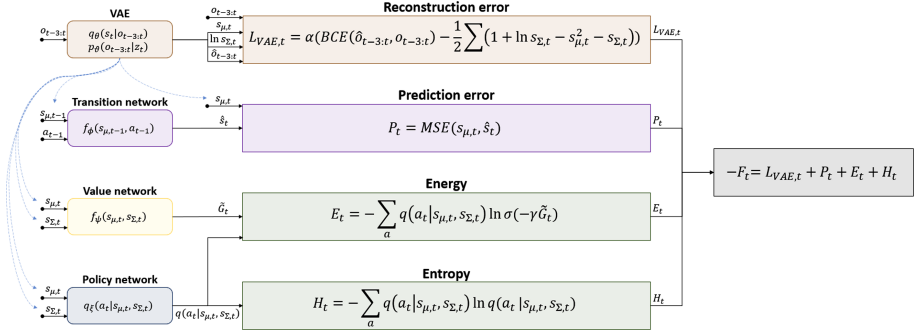where $s_{\mu,t}$ and $s_{\Sigma,t}$ are encoded by $q_\theta(s_t|o_{t-3:t})$.

Fig. 3. Variational Free Energy computation using the approximated densi-
ties. The VAE encodes high-dimensional input as a latent state space, which
is used as input to the other networks. Note that the third term of Eq. 7
($D_{KL}[q_\xi(a_t|s_{\mu,t}, s_{\Sigma,t})\|\sigma(-\gamma f_\psi(s_{\mu,t}, s_{\Sigma,t}))]$) has been split into an energy and an
entropy term (Any KL divergence can be split into an energy term and an entropy).

3 Experimental Setup

To evaluate the proposed algorithm we used the OpenAI Gym's CartPole-v1, as
depicted in Fig. 4. In the CartPole-v1 environment, a pole is attached to a cart
that moves along a track. The pole is initially upright, and the agent's objective
is to keep the pole from tilting too far to one side or the other by increasing or
decreasing the cart's velocity. Additionally, the position of the cart must remain
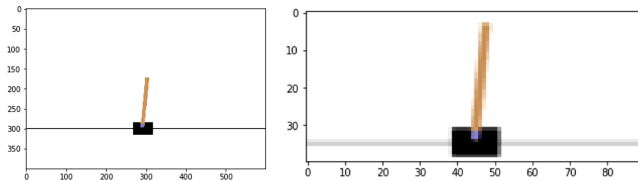within certain bound. An episode of the task terminates when the agent fails

Fig. 4. Cartpole-v1 benchmark (left) and cropped visual input used (right).

either of these objectives, or when it has survived for 500 time steps. Each time step the agent receives a reward of 1.

The CartPole state consists of four continuous values: the cart position, the cart velocity, the pole angle and the velocity of the pole at its tip. Each run the state values are initialized at random within a small margin to ensure variability between runs. The agent can exact influence on the next state through two discrete actions, by pushing the cart to the left, or by pushing it to the right.

Tests were conducted in two scenarios: 1) an MDP scenario in which the agent has direct access to the state of the environment, and 2) a POMDP scenario in which the agent does not have direct access the environment state, and instead receives pixel value from which it must derive meaningful hidden states. By default, rendering the CartPole-v1 environment returns a $3 \times 400 \times 600$ (color, height, width) array of pixel values. In our experiments we provide the POMDP agents with a downscaled and cropped image. There the agents receive a $3 \times 37 \times 85$ pixel value array in which the cart is centered until it comes near the left or right border.

4   Results

The performance of our dAIF agents was compared against DQN agents for the MDP and the POMDP scenarios, and against an agent that selects it actions at random. Each agent was equipped with a memory buffer and a target network [19]. The memory buffer stores transitions from which the agent can sample random batches on which to perform batch gradient descent. The target network is a copy of the value network of which the weights are not updated directly through gradient descent, but are instead updated periodically with the weights of the value network. In between updates this provides the agent with fixed EFE-value or Q-value targets, such that the value network does not have to chase a constantly moving objective.
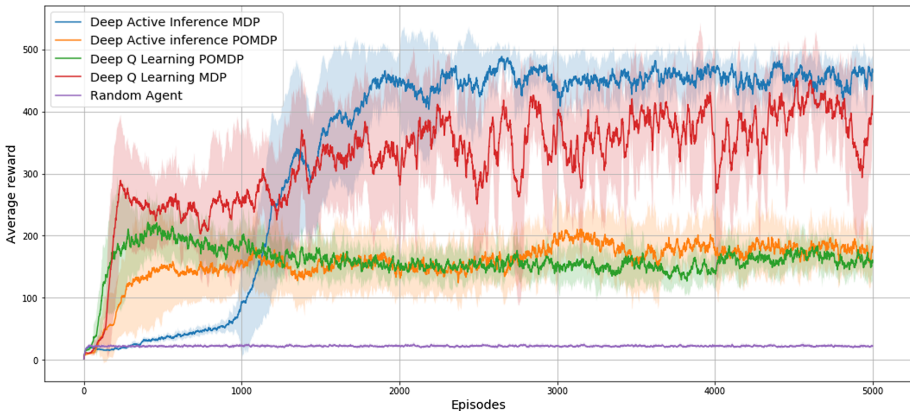

Fig. 5. Average reward comparison for the CartPole-v1 problem.

The VAE of the POMDP dAIF agent is pre-trained to deconstruct input images into a distribution over latent states and to subsequently reconstruct them as accurately as possible.

Figure 5 shows the mean and standard deviation of the moving average reward (MAR) over all runs for the five algorithms at each episode. Each agent performed 10 runs of 5000 episodes. The moving average reward for an episode e is calculated using an smoothing average:

$$MAR_e = 0.1CR_e + 0.9MAR_{e-1} \qquad (8)$$

Where CR_e is the cumulative reward of episode e and MAR_{e-1} is the MAR of the previous episode.

The dAIF MDP agent results closely resemble those presented in [3] and outperforms the DQN MDP agent by a significant margin. Further, the standard deviation shadings show that the dAIF MDP is agent is more consistent between runs than the DQN agent. The POMDP agents are both demonstrated to be capable of learning successful policies, attaining comparable performance.

We have exploited probabilistic model based control through a VAE that encodes the state. On the one hand, this allows the tracking of an internal state which can be used for a range of purposes, such the planning of rewarding policies and the forming of expectations about the future. On the other hand, it makes every part of the algorithm dependent on the proper encoding of the latent space, conversely to the DQN that did not require a state representation to achieve the same performance. However, we expect our approach to improve relative to DQN in more complex environments where the world state encoding can play a more relevant role.

5   Conclusion

We described a dAIF model that tackles partially observable state problems, i.e., it learns the policy from high-dimensional inputs, such as images. Results show that in the MDP case the dAIF agent outperforms the DQN agent, and performs more consistently between runs. Both agents were also shown to be capable of learning (less) successful policies in the POMDP version, where the performance between dAIF and DQN models was found to be comparable. Further work will focus on validating the model on a broader range of more complex problems.

Appendix

Deep Q Agent MDP	
Networks & params.	Description
N_s	Number of states.
N_a	Number of actions.
Q-value network	Fully connected network using an Adam optimizer with a learning rate of 10^{-3}, of the form: $N_s \times 64 \times N_a$.
γ	Discount factor set to 0.98
Memory size	Maximum amount of transition that can be stored in the memory buffer: 65,536
Mini-batch size	32
Target network freeze period	The amount of time steps the target network's parameters are frozen, until they are updated with the parameters of the value network: 25

Deep Q Agent POMDP	
Networks & params.	Description
N_a	Number of actions.
Q-value network	Consists of three 3D convolutional layers (each followed by batch normalization and a rectified linear unit) with $5 \times 5 \times 1$ kernels and $2 \times 2 \times 1$ strides with respectively 3, 16 and 32 input channels, ending with 32 output channels. The output is fed to a 2048×1024 fully connected layer which leads to a $1024 \times N_a$ fully connected layer. Uses an Adam optimizer with the learning rate set to 10^{-5}.
γ	Discount factor set to 0.99
Memory size	Maximum amount of transition that can be stored in the memory buffer: 65,536
Mini-batch size	32
Target network freeze period	The amount of time steps the target network's parameters are frozen, until they are updated with the parameters of the value network: 25

Deep Active Inference Agent MDP	
Networks & params.	Description
N_s	Number of states.
N_a	Number of actions.
Transition network	Fully connected network using an Adam optimizer with a learning rate of 10^{-3}, of the form: $(N_s + 1) \times 64 \times N_s$.
Policy network	Fully connected network using an Adam optimizer with a learning rate of 10^{-3}, of the form: $N_s \times 64 \times N_a$, a softmax function is applied to the output.
EFE-value network	Fully connected network using an Adam optimizer with a learning rate of 10^{-4}, of the form: $N_s \times 64 \times N_a$.
γ	Precision parameter set to 1.0
β	Discount factor set to 0.99
Memory size	Maximum amount of transition that can be stored in the memory buffer: 65,536
Mini-batch size	32
Target network freeze period	The amount of time steps the target network's parameters are frozen, until they are updated with the parameters of the value network: 25

Deep Active Inference Agent POMDP	
Networks & params.	Description
N_l	Size of the VAE latent space, here set to 32.
N_a	Number of actions.
Encoder-network $q_\theta(s_t\|o_{t-3:t})$	Consists of three 3D convolutional layers (each followed by batch normalization and a rectified linear unit) with $5 \times 5 \times 1$ kernels and $2 \times 2 \times 1$ strides with respectively 3, 16 and 32 input channels, ending with 32 output channels. The output is fed to a 2048×1024 fully connected layer which splits to two additional $1024 \times N_l$ fully connected layers. Uses an Adam optimizer with the learning rate set to 10^{-5}.
Decoder-network $p_\vartheta(o_{t-3:t}\|z_t)$	Consists of a $N_l \times 1024$ fully connected layer leading to a 1024×2048 fully connected layer leading to three 3D transposed convolutional layers (each followed by batch normalization and a rectified linear unit) with $5 \times 5 \times 1$ kernels and $2 \times 2 \times 1$ strides with respectively 32, 16 and 3 input channels, ending with 3 output channels. Uses an Adam optimizer with the learning rate set to 10^{-5}.
Transition-network $f_\phi(s_{\mu,t}, a_t)$	Fully connected network using an Adam optimizer with a learning rate of 10^{-3}, of the form: $(2N_l+1) \times 64 \times N_l$.
Policy-network $q_\xi(s_{\mu,t}, s_{\Sigma,t})$	Fully connected network using an Adam optimizer with a learning rate of 10^{-3}, of the form: $2N_l \times 64 \times N_a$, a softmax function is applied to the output.
EFE-value-network $f_\psi(s_{\mu,t}, s_{\Sigma,t})$	Fully connected network using an Adam optimizer with a learning rate of 10^{-4}, of the form: $2N_l \times 64 \times N_a$.
γ	Precision parameter set to 12.0
β	Discount factor set to 0.99
α	A constant that is multiplied with the VAE loss to take it to the same scale as the rest of the VFE terms, set to 4×10^{-5}
Memory size	Maximum amount of transition that can be stored in the memory buffer: 65,536
Mini-batch size	32
Target network freeze period	The amount of time steps the target network's parameters are frozen, until they are updated with the parameters of the value network: 25

References

1. Ueltzhöffer, K.: Deep active inference. Biol. Cybern. **112**(6), 547–573 (2018). https://doi.org/10.1007/s00422-018-0785-7
2. Sancaktar, C., van Gerven, M., Lanillos, P.: End-to-end pixel-based deep active inference for body perception and action. arXiv preprint arXiv:2001.05847 (2019)
3. Millidge, B.: Deep active inference as variational policy gradients. J. Math. Psychol. **96**, 102348 (2020). https://doi.org/10.1016/j.jmp.2020.102348
4. Tschantz, A., Baltieri, M., Seth, A.K., Buckley, C.L.: Scaling active inference. arXiv Prepr. arXiv:1911.10601v1 (2019)
5. Çatal, O., Wauthier, S., Verbelen, T., Boom, C.D., Dhoedt, B.: Deep active inference for autonomous robot navigation. arXiv Prepr. arXiv:2003.03220v1 (2020)
6. Fountas, Z., Sajid, N., Mediano, P.A.M., Friston, K.: Deep active inference agents using monte-carlo methods. arXiv Prepr. arXiv:2006.04176v1 (2020)
7. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv Prepr. arXiv:1312.5602v1 (2013)
8. Arulkumaran, K., Deisenroth, M.P., Brundage, M., Bharath, A.A.: Deep reinforcement learning: a brief survey. IEEE Signal Process. Mag. **34**(6), 26–38 (2017)
9. Friston, K.J., Daunizeau, J., Kilner, J., Kiebel, S.J.: Action and behavior: a free-energy formulation. Biol. Cybern. **102**, 227–260 (2010). https://doi.org/10.1007/s00422-010-0364-z
10. Friston, K.J.: The free-energy principle: a unified brain theory? Nature **11**, 127–138 (2010). https://doi.org/10.1038/nrn2787
11. Adams, R.A., Shipp, S., Friston, K.J.: Predictions not commands: active inference in the motor system. Brain Struct. Funct. **218**(3), 611–643 (2012). https://doi.org/10.1007/s00429-012-0475-5
12. Lanillos, P., Pages, J., Cheng, G.: Robot self/other distinction: active inference meets neural networks learning in a mirror. In: Proceedings of the 24th European Conference on Artificial Intelligence (ECAI), pp. 2410–2416 (2020). https://doi.org/10.3233/FAIA200372
13. Rood, T., van Gerven, M., Lanillos, P.: A deep active inference model of the rubber-hand illusion. arXiv Prepr. arXiv:2008.07408 (2020)
14. Oliver, G., Lanillos, P., Cheng, G.: Active inference body perception and action for humanoid robots. arXiv preprint arXiv:1906.03022 (2019)
15. Friston, K., Trujillo-Barreto, N., Daunizeau, J.: Dem: a variational treatment of dynamic systems. NeuroImage **41**(3), 849–885 (2008). https://doi.org/10.1016/j.neuroimage.2008.02.054
16. Parr, T., Friston, K.J.: Generalised free energy and active inference. Biol. Cybern. **113**, 495–513 (2019). https://doi.org/10.1007/s00422-019-00805-w
17. Hausknecht, M., Stone, P.: Deep recurrent q-learning for partially observable MDPs. In: AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents (AAAI-SDMIA15), November 2015
18. Friston, K.J., Samothrakis, S., Montague, R.: Active inference and agency: optimal control without cost functions. Biol. Cybern. **106**, 523–541 (2012). https://doi.org/10.1007/s00422-012-0512-8
19. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**, 529–533 (2015). https://doi.org/10.1038/nature14236