Deep Active Inference

Kai Ueltzhöffer

Kai Ueltzhöffer
Kosselstraße 4
69115 Heidelberg, Germany
Tel.: +49-174-9446360
E-mail: kueltzho@gmail.com

Abstract This work combines the free energy principle and the ensuing active inference dynamics with recent advances in variational inference in deep generative models, and evolution strategies to introduce the "deep active inference" agent. This agent minimises a variational free energy bound on the average surprise of its sensations, which is motivated by a homeostatic argument. It does so by optimising the parameters of a generative latent variable model of its sensory inputs, together with a variational density approximating the posterior distribution over the latent variables, given its observations, and by acting on its environment to actively sample input that is likely under this generative model. The internal dynamics of the agent are implemented using deep and recurrent neural networks, as used in machine learning, making the deep active inference agent a scalable and very flexible class of active inference agent. Using the mountain car problem, we show how goal directed behaviour can be implemented by defining appropriate priors on the latent states in the agent's model. Furthermore, we show that the deep active inference agent can learn a generative model of the environment, which can be sampled from to understand the agent's beliefs about the environment and its interaction therewith.

1 Introduction

Active Inference (Friston et al, 2006, 2010; Friston, 2012) is a normative theory of brain function derived from the properties required by active agents to survive in dynamic, changing environments. This theory is able to account for many aspects of action and perception as well as anatomical and physiological features of the brain on one hand (Brown and Friston, 2012; Friston, 2005; Friston et al, 2011; Friston and Kiebel, 2009; Schwartenbeck et al, 2015; Adams et al, 2013), and encompasses many formal theories about brain function on the other (Friston, 2010). In terms of its functional form it rests on the minimisation of a variational upper bound on the agent's average surprise. In this way it is formally very similar to state of the art algorithms for variational inference in deep generative models (Rezende et al, 2014; Kingma and Welling, 2014; Chung et al, 2015). However, optimising this bound for an active agent introduces a dependency on the true dynamics of the world, which an agent usually does not have direct access to, and whose true functional form does not have to coincide with the functional form of the agent's generative model of the world. Here we solve these problems using deep neural networks (LeCun et al, 2015) and recurrent neural networks (Karpathy et al, 2015) as flexible function approximators, which allow the agent's generative model to learn a good approximation of the true world dynamics. Futhermore we apply evolution strategies (Salimans et al, 2017) to estimate gradients on the variational bound, averaged over a population of agents. This formalism allows to obtain gradient estimates even for non differentiable objective functions, which is the case here; since the agent does neither know the equations of motions of the world, nor its partial derivatives.

In this way, our approach pairs active inference with state of the art machine learning techniques to create agents that can successfully reach goals in complex environments while simultaneously building a generative model of themselves and their surroundings. In this work we want to lay out the basic form of these so called "Deep Active Inference" agents and illustrate their dynamics using a simple, well known problem from reinforcement learning, namely the mountain car problem (Moore, 1991). By utilising models and optimisation techniques that have been applied successfully to real world data and large scale problems (Rezende et al, 2014; Kingma and Welling, 2014; Chung et al, 2015; Salimans et al, 2017), our agent can be scaled to complex and rich environments. With this paper we publish the full implementation of the resulting Deep Active Inference agent, together with all scripts used to create the figures in this publication at `https://www.github.com/kaiu85/deepAI_paper`.

In summary, this work casts the free energy principle and active inference in terms of deep learning and evolution strategies. Conceptually, this could be regarded as 'active evolution', where the fitness function is replaced with the path integral of free energy over the dynamics of an agent. One might ask how this relates to active inference, which is generally construed in terms of online data assimilation (i.e., evidence accumulation) and planning as inference. The key move in deep active inference is to convert the inference problem (about latent or hidden states) into a learning problem (akin to amortisation). In other words, deep active inference assumes the existence of a parameterised mapping between observed variables and posterior beliefs (probability distributions) over latent states and optimal action. Crucially, this means one can absorb both the (variational) parameters of this mapping and the parameters of a generative model into the optimisation with respect to the free energy functional (via an evolution strategy). In principle, this enables one to involve or train autonomous agents who learn to infer latent states while, at the same time, learn the generative model and learn to infer (via optimisation of the model and variational parameters respectively).

In section 2 we will briefly recapitulate the active inference principle. In section 3 we describe the general architecture of the deep active inference agent. In section 4 we will discuss experiments in two versions of the mountain car problem: An own implementation and the MountainCar-v0 environment, which is part of OpenAI Gym (Brockman et al, 2016). We will show that the agent can solve both versions of the mountain car problem while simultaneously learning a generative model of its environment. In section 5 we will discuss our results, limitations of our approach, and possible further directions of this approach and its relation to other approaches.

2 Active Inference

Active Inference (Friston et al, 2006, 2010; Friston, 2012) rests on the basic assumption that any agent in exchange with a dynamic, fluctuating environment has to keep certain inner parameters within a well defined range. Otherwise,

it would sooner or later encounter a phase transition due to which it would loose its defining characteristics and therefore disappear. Thus, the agent must restrict itself to a small volume in its state space. This can be formalised using the entropy of the probability distribution $p(s^*)$ of finding the agent in a given state s^* of its state space S^*:

$$H(S^*) = \int_{s^* \in S^*} (-\ln p(s^*)) \, p(s^*) \, \mathrm{d}s^*$$

By minimising this entropy, an agent can counter dispersive effects from its environment and maintain a stable identity in a fluctuating environment.

However, an agent does not have direct access to an objective measurement of its current state. Instead it only perceives itself and the world around it via its sensory epithelia. This can be described by a potentially noisy and non bijective sensory mapping $o = g(s^*)$ from states s^* to sensations o. Defining the sensory entropy

$$H(O) = \int_{o \in O} (-\ln p(o)) \, p(o) \, \mathrm{d}o$$

over the space O of all possible observations of an agent, one can derive the following inequality in the absence of sensory noise (Friston et al, 2010)

$$H(O) \geq H(S^*) + \int_{s^* \in S^*} p(s^*) \ln |g_{s^*}| \, \mathrm{d}s$$

Agents, whose sensory organs do not have a good mapping of relevant physical states to appropriate sensory inputs, do not survive very long. So the mapping between the agent's true states and its sensations is assumed to have an almost constant sensitivity, in terms of the determinant of the Jacobian $|g_{s^*}|$, over the encountered range of states s^*. This makes the last term approximately constant and allows upper bounding the entropy of the agent's distribution on state space by the entropy of its sensory states plus this constant term (Friston et al, 2010). Thus, to maintain its physiological variables within well defined bounds, an agent has to minimize its sensory entropy $H(O)$[1].

Assuming ergodicity, i.e. the equivalence of time and ensemble averages, one can write the sensory entropy as

$$H(O) = \lim_{T \to \infty} -\frac{1}{T} \int_0^T \ln p(o(t)) \, \mathrm{d}t$$

Using this assumption, we can minimise the sensory entropy directly by minimising this time integral, or — using the calculus of variations — by minimising its sensory surprise $-\ln p(o(t))$ at all times, in terms of the following Euler-Lagrange equation:

$$\nabla_o \left(-\ln p(o(t)) \right) = 0$$

---

[1] In more general formulations of active inference, the assumption that the mapping between hidden states and outcomes is constant can be relaxed (Friston et al, 2015).

To be able to efficiently do this, our agent needs a statistical model of its sensory inputs, to evaluate $p(o)$. Since the world in which we live is hierarchically organised, dynamic, and features a lot of complex noise, we assume that the agent's model is a deep, recurrent, latent variable model (Conant and Ashby, 1970). Furthermore we assume that this model is generative, using the observation that humans are able to imagine certain situations and perceptions (like the image of a blue sky over a desert landscape) without actually experiencing or having experienced them. Thus, we work with a generative model $p_\theta(o, s)$ of sensory observations o and latent variables s, that represent the hidden states of the world that cause outcomes, which we can factorise into a likelihood function $p_\theta(o|s)$ and a prior on the states $p_\theta(s)$:

$$p_\theta(o, s) = p_\theta(o|s)p_\theta(s)$$

The set θ comprises the slowly changing parameters that the agent can change to improve its model of the world. In the brain this might be the pattern and strength of synapses. Given this factorisation, to minimize surprise, the agent has to solve the hard task of calculating

$$p_\theta(o) = \int p_\theta(o|s)p_\theta(s) \, \mathrm{d}s$$

by marginalising over all possible states that could lead to a given observation. As the dimensionality of the latent state space S can be very high, this integral is extremely hard to solve. Therefore a further assumption of the free energy principle is, that agents do not try to minimize the surprise $-\ln p_\theta(o)$ directly, but rather minimize an upper bound, which is a lot simpler to calculate.

Using the fact that the Kullback-Leibler (KL) Divergence

$$D_{\mathrm{KL}}(p_a(x)||p_b(x)) = \int_{x \in X} \ln\left(\frac{p_a(x)}{p_b(x)}\right) p_a(x) \, \mathrm{d}x$$

between two arbitrary distributions $p_a(x)$ and $p_b(x)$ with a shared support on a common space X is always greater or equal to zero, and equal to zero if and only if the two distributions are equal, we can define the variational free energy as:

$$F(o, \theta, u) = -\ln p_\theta(o) + D_{\mathrm{KL}}(q_u(s)||p_\theta(s|o)) \geq -\ln p_\theta(o)$$

Here $q_u(s)$ is an arbitrary, so called variational density over the space of hidden states s, which belongs to a family of distributions parameterised by a time dependent, i.e. fast changing, parameter set u. If $q_u(s)$ was a diagonal Gaussian, $u = \{\mu, \sigma\}$ would be the corresponding means and standard deviations. This parameter set can be encoded by the internal states of our agent, e.g. by the neural activity (i.e. the firing pattern of neurons) in its brain. Thus, the upper bound $F(o, \theta, u)$ now only depends on quantities to which our agent

has direct access. Namely the states of its sensory organs o, the synapses encoding its generative model of the world θ and the neural activity representing the sufficient statistics u of the variational density.

Using the definition of the Kullback-Leibler divergence, the linearity of the integral, Bayes' rule, and manipulation of logarithms, one can derive the following equivalent forms of the free energy functional:

$$
\begin{aligned}
F(o, \theta, u) &= -\ln p_\theta(o) + D_{\mathrm{KL}}(q_u(s)||p_\theta(s|o)) \\
&= \langle -\ln p_\theta(o, s) \rangle_{q_u(s)} - \langle -\ln q_u(s) \rangle_{q_u(s)} \\
&= \langle -\ln p_\theta(o|s) \rangle_{q_u(s)} + D_{\mathrm{KL}}(q_u(s)||p_\theta(s))
\end{aligned}
$$

Here $\langle f(s) \rangle_{q_u(s)}$ means calculating the expectation value of $f(s)$ with respect to the variational density $q_u(s)$.

Note that while the concrete choice of the KL-divergence might seem arbitrary at first, the KL-divergence has some desirable properties beyond the fact that it allows for the above reformulations of the variational free energy: It is a measure of relative information, i.e. $D_{\mathrm{KL}}(p_a(x)||p_b(x))$ quantifies how many bits in average have to be transmitted to communicate the density p_a, given that the density p_b is already known. Furthermore it is the only such measure that is local (i.e. local changes in a density have local consequences), coordinate invariant, and extensive (i.e. additive over independent subsystems) (Caticha, 2004; Tran et al, 2017). The variational free energy above is exactly the same objective function used in the variational autoencoder (VAE, (Kingma and Welling, 2014; Rezende et al, 2014)). The negative free energy is also known as an evidence lower bound (ELBO), which underlies much of machine learning and all of variational inference.

If the agent was unable to change its environment, i.e. just a passive observer of the world around it, the only thing it could do to minimize F would be to change the parameters θ of its generative model and the sufficient statistics u of its inner representation. Looking at the first form of the free energy, optimising u would correspond to minimising the Kullback-Leibler divergence between the variational distribution $q_u(s)$ and the true posterior distribution $p(s|o)$, i.e. the probability over hidden states s given the observations o. Thus, the variational density can be seen as an approximation of the true posterior density. Therefore, by minimising F, the agent automatically acquires a probabilistic representation of an approximate posterior on the hidden states of the world, given its sensory input. The optimisation of the sufficient statistics u of the variational density q is therefore what we call "perception". As q has to be optimised on a fast timescale, quickly changing with the sensory input o, it is likely represented in terms of neural activity. This might explain hallmarks of Bayesian computations and probabilistic representations of sensory stimuli in the brain (Ernst and Banks, 2002; Alais and Burr, 2004; Knill and Pouget, 2004; Moreno-Bote et al, 2011; Berkes et al, 2011; Haefner et al, 2016).

As the variational free energy upper bounds the surprise $-\ln p_\theta(o)$, minimising free energy with respect to the parameters θ of the generative model will simultaneously maximise the evidence $p_\theta(o)$ for the agent's generative

model of the world. The resulting optimisation of the parameters θ of the generative model is what we call perceptual learning and what might be implemented by changing the synaptic connections between neurons in the brain. The second form is given only to demonstrate, where the name "free energy" originated from, since its form is isomorphic with the Helmholtz free energy of the canonical ensemble in statistical physics.

The core idea of active inference is now to equip the agent with actuators, that allow it to actively change the state of its environment. Thereby the sensory observations o become a function of the current and past states a of the agent's actuators ("muscles"), via their influence on the state of the world generating the sensory inputs. Now the agent can not only minimize the free energy bound by learning (i.e. optimising the parameters of its generative model) and perception (i.e. optimising the sufficient statistics of the variational density), but also by changing the observations it makes.

Considering the third form of the variational free energy

$$F(o, \theta, u) = \langle -\ln p_\theta(o|s) \rangle_{q_u(s)} + D_{\mathrm{KL}}(q_u(s) || p_\theta(s))$$

we see that our agent can minimize it by seeking out observations o that have a high likelihood $p_\theta(o|s)$ under its own generative model of the world, averaged over its approximate posterior of the state of the world. Thus, the agent will seek out states that conform to its expectations, i.e. that have a high likelihood under its generative model of the world. So one can encode specific goals in the priors of the generative model of the agent: assigning a higher a priori probability to a specific state, the agent will try to seek out this state more often. The first term, which our agent will maximise using its actions, is also called accuracy in Bayesian statistics, while the second term is known as complexity. Complexity measures the degree to which posteriors have to be adjusted in relation to priors to provide an accurate account of sensory data or outcomes.

The ensuing triple optimisation problem can be formalised using the following dynamics for the parameters θ of an agent's generative model, as well as the time courses of its internal states $u(t)$, encoding the sufficient statistics of the variational density q, and of the states of its actuators $a(t)$:

$$(\theta, u, a) = \underset{(\tilde{\theta}, \tilde{u}, \tilde{a})}{\arg\min} \frac{1}{T} \int_t F(o(\tilde{a}(t)), \tilde{\theta}, \tilde{u}(t)) \, \mathrm{d}t$$

The quantity on the right is the time integral of the *variational free energy* and therefore - in analogy to physics - called the *variational free action*. The *variational free action* is a direct upper bound of the sensory entropy, given ergodicity and a sufficiently long integration time T.

In this paper we will train agents by directly minimising their *variational free action* over the entire length of the simulated trials. Furthermore we will use an amortisation approach which replaces the direct optimisation of the trajectories of the sufficient statistics of the variational density and the active states, by highly flexible parameterised functions, namely neural networks,

whose parameters are optimised simultaneously with the parameters of the generative model. This is in contrast to most work in the theoretical neuroscience literature, where the aim is to explain how this minimisation can be implemented (approximately) by agents, using only local (in space and time) interactions, knowledge and dynamics. There, two approaches dominate: First, given dynamics in continuous time and state spaces, one can use the Euler-Lagrange formalism to directly derive differential equations governing the local dynamics of the agents from the global optimisation objective (Friston et al, 2006). Second, given discrete time and state spaces, which means that the dynamics of the agent can be described in terms of (partially observable) Markov decision processes, one can optimise the agents by minimising their expected free energy, a quantity representing the counterfactual, expected free energy over possible trajectories, given sequences of actions (Friston et al, 2015). By optimising the parameters of an amortised agent using its variational free action directly as optimisation objective, the agent's amortised dynamics should approximate such local dynamics automatically. This is indeed what we find and discuss in section 5.2.

Furthermore, in this paper we will consider the variational density or beliefs over hidden states of the world. In more general formulations, this density would cover both the hidden states and the parameters of the generative model. Usually, these beliefs are factorised so that there is a mean field approximation to the true posterior estimates of (time varying) states and (time invariant) parameters (Friston, 2008). We will return to this issue in the discussion. For the present, we will treat the parameters as unknown quantities, whose probability density will be optimised with respect to (expected) free energy using an evolution strategy (as opposed to a variational optimisation).

3 Methods

In this section, we introduce an agent which uses recent advances in stochastic optimisation of deep generative models (Kingma and Welling, 2014; Rezende et al, 2014; Chung et al, 2015) and evolution strategies as efficient, scalable optimisation algorithm for non differentiable objective functions (Salimans et al, 2017) to implement active inference. We show its ability to reach a specific goal while concurrently building a generative model of its environment, using the well known mountain car problem (Moore, 1991). But we first start with some basics that are well known in the deep learning community, but which — due to similar but differently used nomenclature — might lead to confusion among computational neuroscientists.

3.1 Deep Neural Networks

When we talk about deep neural networks, we use this term in the sense of the deep learning literature (LeCun et al, 2015). In this sense, neural networks

are nothing but very flexible function approximators. An excellent and comprehensive introduction to the state of the art in deep learning is given in Goodfellow et al (2016).

3.1.1 Fully Connected Layer

Deep neural networks are composed of individual layers, which are functions $f_\theta : \mathbb{R}^{d_1} \to \mathbb{R}^{d_2}, x \mapsto f_\theta(x)$. Here d_1 represents the number of input neurons, i.e. the dimensionality of the input space, and d_2 represents the number of output neurons, i.e. the dimensionality of the output space. The subscript θ represents the parameters of the function, which can be tuned to approximate or optimise a given objective function.

The canonical functional form of a so called *fully connected* layer is:

$$f_\theta(x) = \mathrm{h}\left(\theta_w x + \theta_b\right)$$

Here $\mathrm{h} : \mathbb{R} \to \mathbb{R}$ is an *elementwise* nonlinear transfer function. This could for example be a tanh, sigmoid or a so called rectified linear function $\mathrm{relu(x)} = \max(0, x)$. The set of parameters $\theta = \{\theta_w, \theta_b\}$ consists of the $d_2 \times d_1$ matrix θ_w, called the weight matrix or just weights, and the bias vector $\theta_b \in \mathbb{R}^{d_2}$. This functional form is loosely inspired by the response function of very simple models of neural ensembles (e.g. the mean field equation for the average firing rate of a large population of leaky integrate and fire neurons as function of the mean inputs to the population, c.f. equations 12 - 15 in Wong and Wang (2006)). These response functions convert the weighted sum of inputs currents (approximated by the firing rates x of the presynaptic neurons, weighted using the parameters θ_w) to the firing rate of the modelled population by a nonlinear, thresholded activation function, which is represented here using the bias parameters θ_b and the nonlinear transfer function h.

3.1.2 Deep Feedforward Networks

A deep feedforward network consists of a stack of individual layers $f_1, f_2, f_3, ..., f_{n-1}, f_n$, that are applied to the input x sequentially to generate an output y:

$$y = f_n(f_{n-1}(...f_3(f_2(f_1(x)))...)) = f_n \circ f_{n-1} \circ ... \circ f_3 \circ f_2 \circ f_1(x)$$

The output of the function f_n is called the output layer, the input x is called the input layer, the intermediate values $f_1(x), f_2(f_1(x)), ..., f_{n-1}(...x)$ are called the hidden layers, since they are not explicitly constrained by the target or objective function. The finite dimensions of the intermediate outputs $f_1(x), f_2(f_1(x)), ..., f_{n-1}(...x)$ can be regarded as the numbers of hidden neurons in each layer.

The flexibility of deep feedforward networks allow us to specify agents, whose generative models of their environment do not need to be from the

same model family as the true generative process. It was shown that a feed-forward network with a linear output layer and at least one hidden layer with a nonlinear activation function, such as a tanh or sigmoid activation function, can approximate any Borel measurable function from one finite dimensional space to another with arbitrary accuracy, given a sufficient dimensionality (i.e. a sufficient number of neurons) of the hidden layer (Hornik et al, 1989).

3.1.3 Recurrent Neural Networks

To enable a learning system to build a memory of its previous actions and experiences, a simple and elegant way is to enhance a neural network

$$y_t = f_\theta(x_t)$$

by feeding its output at the previous time step back as an additional input, i.e.

$$y_t = f_\theta(x_t, y_{t-1})$$

Although there is a wide variety of more complex and powerful architectures (c.f. Karpathy et al (2015); Goodfellow et al (2016)), this simple, prototypical recurrent neural network was recently shown to be able to learn long range temporal dependencies, given a sensible initialisation (Le et al, 2015). We are not using this deterministic type of recurrent neural network directly, but we use the basic idea by feeding back samples from a distribution on the current network state as inputs to calculate the next network state. Similar to the approximation theorem for deep feedforward networks, it was shown that recurrent neural networks are able to learn arbitrary algorithms, as they are Turing complete (Siegelmann, 1995).

3.2 The Deep Active Inference Agent

We will now describe an agent that follows the free energy principle laid out in section 2 (Friston et al, 2006, 2010). It it will interact with a discrete-time environment and motor outputs and its inner workings are implemented and optimised using recent advances in deep learning (Kingma and Welling, 2014; Rezende et al, 2014; Chung et al, 2015; Salimans et al, 2017). Its objective will be to minimise its variational free action, i.e. the sum of its variational free energy over all timesteps of a trial.

3.2.1 The Environments

Our agent will be able to interact with discrete time environments with both continuous and discrete state variables. In its most general form, the environment dynamics describe the evolution of the true state $\mathbf{s}^*(t)$ of the world, depending on the agent's action $\mathbf{a}_t$:

$$\mathbf{s}_{t+1}^* = \mathbf{R}(\mathbf{s}_t^*, \mathbf{a}_t)$$

Note that we use bold face to denote vectors.

The true states of the world cause observations $\mathbf{o_t}$ via the agent's sensory mapping $\mathbf{o}_t = g(\mathbf{s}_t^*)$. It describes, which variables of the true state of the world the agent has access to, and the accuracy and noise of these "measurements".

We will discuss a concrete environment, in terms of an own implementation of the mountain car problem with continuous control variable in detail in section 4.1.1, and we will show that our agent can also perform well in the MountainCar-v0 environment of OpenAI Gym (Brockman et al, 2016) in section 4.2.

3.2.2 The Agent's Generative Model of the World

Prior on State Transitions The agent's internal model of the world describes the dynamics of the world in terms of transitions of its latent variables $\mathbf{s}_t$:

$$p_\theta(\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_T) = \prod_{t=1}^{T} p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})$$

with $\mathbf{s}_0 = (0, ..., 0)^T$.

This factorisation means that the agent's prior on the current hidden state $\mathbf{s}_t$ depends only on the previous state $\mathbf{s}_{t-1}$. The transition function $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})$ is implemented by neural networks and parameterised by a subset of the time independent parameters θ of our agent. Thus, the transition distribution $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})$ encodes the agent's model of the dynamics of the world, i.e. what it has learned about the laws of motion of its environment and its influence on it. This gives our agent's generative model of the world the structure of a stochastic recurrent neural network, which was shown to be a very flexible time series model, given a sufficient dimensionality of the state space (Siegelmann, 1995).

Likelihood function The agent's generative model assumes furthermore that the observations $\mathbf{o}_t$, which it makes, depend only on the current state $\mathbf{s}_t$ of the hidden variables:

$$p_\theta(\mathbf{o}_1, \mathbf{o}_2, ..., \mathbf{o}_T) = \prod_{t=1}^{T} p_\theta(\mathbf{o}_t|\mathbf{s}_t)$$

Again, the likelihood function $p_\theta(\mathbf{o}_t|\mathbf{s}_t)$ of an observation, given a state of the world, is implemented using neural networks whose parameters also are part of the set of time independent parameters θ.

3.2.3 Variational Density

Following Kingma and Welling (2014); Rezende et al (2014) we do not explicitly represent the sufficient statistics of the variational posterior at every time step. This would require an additional, costly optimisation of these variational parameters at each individual time step. Instead we use an inference network, which approximates the dependency of the variational posterior on the previous state and the current sensory input and which is parameterised by time invariant parameters. This allows us to learn these parameters together with the parameters of the generative model and the action function (c.f. section 4.2.5), and also allows for very fast inference later on. In other words, all the time invariant aspects of the agent will jointly optimise a free energy functional; including the parameters of the generative model, the (variational) parameters which underwrite learning to infer and a state action policy (c.f. below). We use the following factorisation for this approximation of the variational density on the states $\mathbf{s}$, given the agent's observations $\mathbf{o}$:

$$q_\theta(\mathbf{s}_1, ..., \mathbf{s}_T | \mathbf{o}_1, ..., \mathbf{o}_T) = \prod_{t=1}^{T} q_\theta(\mathbf{s}_t | \mathbf{s}_{t-1}, \mathbf{o}_t)$$

with initial state $\mathbf{s}_0 = (0, ..., 0)$, before the agent has interacted with the environment.

This means that the agent only uses its previous estimate of the hidden state $\mathbf{s}_{t-1}$ and its current observation $\mathbf{o}_t$ to infer the current state $\mathbf{s}_t$. We again use neural networks to implement the density $q_\theta(\mathbf{s}_t | \mathbf{s}_{t-1}, \mathbf{o}_t)$ and add the parameters of the network to the set of time independent parameters θ. Note that this particular factorisation is distinct from the approximations used in Bayesian filtering; in the sense that beliefs about the current state are conditioned on the previous state. This is also known as the Bethe approximation.

This device – of learning the mapping between inputs and the sufficient statistics of (approximate) posterior over hidden states – is known as amortisation. It has the great advantage of being extremely efficient and fast. On the other hand, it assumes that the mapping can be approximated with a static nonlinear function that can be learned with a neural network. This somewhat limits the context sensitivity of active inference schemes, depending upon the parameterisation of the mapping. In short, amortisation enables one to convert a deep inference or deep deconvolution problem into a deep learning problem – by finding a static nonlinear mapping between (time varying) inputs and (approximate) posterior beliefs about the states generating those inputs.

3.2.4 Action States

In general, the action state $\mathbf{a}_t$ of the agent could be minimised directly at each time step, thereby minimising the free energy by driving the sensory input (which depends on the action state via the dynamics of the world) towards the agent's expectations, in terms of its likelihood function. However, this would

require a costly optimisation for each time step (and every single simulated process). Thus, we use the same rationale as Kingma and Welling (2014); Rezende et al (2014) used for the variational density q, and approximate the complex dependence of the action state on the agent's current estimate of the world (and via this on the true state of the world) by fixed, but very flexible functions, i.e. deep neural networks. This yields an explicit functional dependency

$$p_\theta(\mathbf{a}_t|\mathbf{s}_t)$$

whose parameters we include, together with the parameters of the generative model and the parameters underwriting the variational density, to the set of static parameters θ that we will optimise.

We now can just optimise the time invariant parameters θ of these neural networks together with the parameters of the generative model and the variational density. This approximation makes learning and propagating the agent very fast, allowing for efficient simultaneous learning of the generative model, the variational density and the action function.

3.2.5 The Perception-Action Loop

We can describe the agent's perception-action cycle, which we use to evaluate and optimise the agent's variational free action, as follows: When the agent makes an observation $\mathbf{o}_t$ its internal estimate of the state of the world is updated, using the variational posterior $q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{o}_t)$. The resulting density is approximated by drawing a single sample $\hat{\mathbf{s}}$. This state estimate now can be used to calculate the density on the agents active states $p_\theta(\mathbf{a}_t|\hat{\mathbf{s}}_t)$. Again, the resulting distribution is approximated by drawing a single sample $\hat{\mathbf{a}}_t$. This action now can be used to evaluate the environment $s_{t+1}^* = \mathbf{R}(s_t^*, \hat{\mathbf{a}}_t)$ and generate the next observation $\mathbf{o}_{t+1}$ via the agent's sensory mapping $\mathbf{o}_{t+1} = g(\mathbf{s}_{t+1}^*)$. Please note that in this notation the action $\hat{\mathbf{a}}_t$ taken by the agent at time t is defined to be part of the true state of the world $\mathbf{s}_t^*$ at this time step, from which the next state of the world is generated. Simultaneously, the agent can evaluate its prior expectation $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})$ of the hidden state $\mathbf{s}_t$ based only on its previous estimate $\mathbf{s}_{t-1}$, which it will need to calculate its free energy at this time step. Using this recipe, we can propagate an agent through its environment and estimate its free energy at each time step. Summation over the individual timesteps yields an estimate of the free action of a given agent for a given sampled trajectory, which we can use to update the agent's parameters using evolution strategies, as described in section 3.2.8

The agent's perception-action loop is shown in figure 1.

The implementation of of all of these quantities by (partly deep, partly recurrent) neural networks is the reason why we call this class of agents *Deep Active Inference* agents.
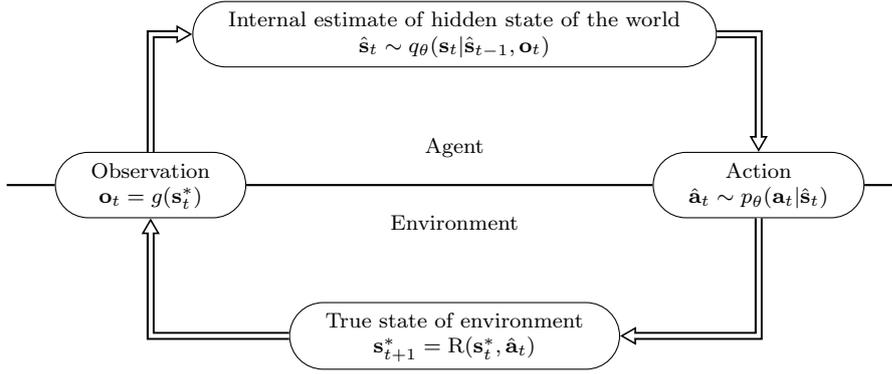
Fig. 1 The agent's perception-action loop. Using the agent's previous estimate of the state of the world $\hat{\mathbf{s}}_{t-1}$ and its current observation $\mathbf{o}_t$, it updates its internal estimate of the hidden state of the world by drawing from the variational density $q_\theta(\mathbf{s}_t|\mathbf{o}_t, \hat{\mathbf{s}}_{t-1})$. Based on its current estimate of the state of the world, it draws an action from the action function $p_\theta(\mathbf{a}_t|\hat{\mathbf{s}}_t)$. This action, together with the previous true state of the world now generates, hidden from the agent, the next state of the world $\mathbf{s}_{t+1}^* = \mathbf{R}(\mathbf{s}_t^*, \hat{\mathbf{a}}_t)$, from which the generative process generates the next observation via $\mathbf{o}_t = g(\mathbf{s}_t^*)$.

3.2.6 The Free Action Objective

Using this loop, we can evaluate all the quantities to calculate the variational free action:

$$FA(o, \theta) = \langle -\ln p_\theta(\mathbf{o}_1, ..., \mathbf{o}_T|\mathbf{s}_1, ..., \mathbf{s}_T)\rangle_q$$
$$+ D_{\mathrm{KL}}(q_\theta(\mathbf{s}_1, ..., \mathbf{s}_T|\mathbf{o}_1, ..., \mathbf{o}_T)||p_\theta(\mathbf{s}_1, ..., \mathbf{s}_T))$$

where $\langle ...\rangle_q$ means the average with respect to the variational density

$$q_\theta(\mathbf{s}_1, ..., \mathbf{s}_T|\mathbf{o}_1, ..., \mathbf{o}_T)$$

Using the factorisations introduced above, the variational free action becomes:

$$FA(o, \theta) = \sum_{t=1}^{T}\left[\langle -\ln p_\theta(\mathbf{o}_t|\mathbf{s}_t)\rangle_{q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{o}_t)} \right.$$
$$\left. + D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}))\right]$$

using $\mathbf{s}_0 = (0, ..., 0)^T$, i.e. just the sum over the variational free energy at each individual time step.

The causal structure of the complete model is shown in figure 2.

To evaluate this expression for an agent with a given parameter set θ, we simulate n_F processes in parallel, which allows us to approximate the variational density q just by a single sample $\hat{\mathbf{s}}_t$ per process and per time step, analogous to Stochastic Gradient Descent or the Variational Autoencoder,
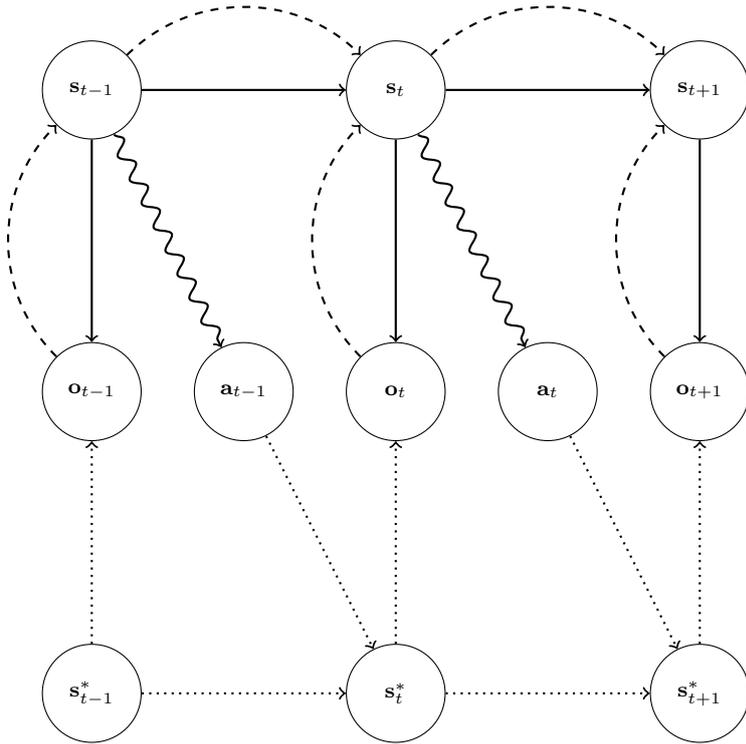
Fig. 2 Graphical representation of causal dependencies. The solid lines correspond to the factors of the agent's generative model, $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t)$ and $p_\theta(\mathbf{o}_t|\mathbf{s}_t)$. The dashed lines correspond to the conditional dependencies modelled by the variational density $q_\theta(\mathbf{s}_t|\mathbf{o}_t, \mathbf{s}_{t-1})$. The dotted lines correspond to the environmental dynamics $\mathbf{s}_{t+1}^* = \mathbf{R}(\mathbf{s}_t^*, \mathbf{a}_t)$ and the true generative process $\mathbf{o}_t = g(\mathbf{s}_t^*)$. The wiggly line describes the dependency of the action $p_\theta(\mathbf{a}_t|\mathbf{s}_t)$ on the hidden states $\mathbf{s}_t$.

where only one sample per data point is enough, since the gradients depend on entire (mini-)batches of datapoints (Kingma and Welling, 2014; Rezende et al, 2014; Chung et al, 2015). The sampling proceeds according to algorithm 1 (please see table).

The evaluation of the KL-divergence $D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}))$ in algorithm 1 can be approximated by sampling. Here we use the fact that during propagation our agent permanently generates samples $\hat{\mathbf{s}}_t$ from its variational posterior $q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)$, which it uses to evaluate its action function $p_\theta(\mathbf{a_t}|\hat{\mathbf{s}}_t)$. Thus, the KL-term can be simply approximated by

$$D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1})) \approx \ln q_\theta(\hat{\mathbf{s}}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t) - \ln p_\theta(\hat{\mathbf{s}}_t|\hat{\mathbf{s}}_{t-1}))$$

Although this estimate is very easy to evaluate, based on a single sample it suffers from a relatively high variance. Thus, another option is to choose the variational density and the prior density from families for which a closed

Algorithm 1 Sampling based approximation of the free action cost. Concrete samples and values of a variable are marked using a hat, e.g. $\hat{\mathbf{s}}_t$. l is the sampling based approximation of the expected likelihood of the observations under the variational density q, i.e. the accuracy term. d is the KL-divergence between the variational density and the prior, i.e. the complexity term.

Initialize the free action estimate with $FA = 0$
Initialize n_F agents with $\hat{\mathbf{s}}_0 = (0, ..., 0)$ and initialize environment with $\mathbf{s}_0^*$.
for each Agent **do**
    for $t = 1, ..., T$ **do**
        Sample an action $\hat{a}_{t-1}$ from $p_\theta(a_{t-1}|\hat{\mathbf{s}}_{t-1})$
        Propagate the environment using $\mathbf{s}_t^* = \mathbf{R}(\mathbf{s}_{t-1}^*, \hat{\mathbf{a}}_{t-1})$
        Generate observations $\mathbf{o}_t$ using $\mathbf{o}_t = g(\mathbf{s}_t^*)$
        Draw a single sample $\hat{\mathbf{s}}_t$ from $q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)$
        Calculate $l = -\ln p_\theta(\mathbf{o}_t|\hat{\mathbf{s}}_t)$
        Calculate $d = D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}))$
        Increment free energy $FA = FA + (d + l)$
        Carry $\hat{\mathbf{s}}_t$ and $\mathbf{s}_t^*$ over to the next time step.
    end for
end for
return FA

form expression of $D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}))$ is available (Kingma and Welling, 2014; Rezende et al, 2014). This is the case for diagonal Gaussian distributions, which we will use in our experiments.

The minimisation of the free action with respect to the parameters θ, by approximating the gradient of the free action with respect to these parameters using the sampling scheme outlined in section 3.2.8, will improve the agent's generative model $p_\theta(o, \mathbf{s})$, by lower bounding the evidence $p_\theta(o)$ of the observations o, given the generative model. Simultaneously it will make the variational density $q_\theta(\mathbf{s}|o)$ a better approximation of the true posterior $p_\theta(\mathbf{s}|o)$, as can be seen from the following, equivalent form of the free energy (c.f. section 2):

$$F(o, \theta) = <-\ln p_\theta(o|\mathbf{s})>_{q_\theta(\mathbf{s}|o)} + D_{\mathrm{KL}}(q_\theta(\mathbf{s}|o)||p_\theta(\mathbf{s}))$$

$$= -\ln p_\theta(o) + D_{\mathrm{KL}}(q_\theta(\mathbf{s}|o)||p_\theta(\mathbf{s}|o))$$

Additionally, the parameters of the action function will be optimised, so that the agent seeks out expected states under its own model of the world, minimizing $<-\ln p_\theta(o|\mathbf{s})>_{q_\theta(\mathbf{s}|o)}$.

3.2.7 Goal Directed Behaviour

If we just propagate and optimise our agent as it is now, it will look for a stable equilibrium with its environment and settle there. However, to be practically applicable to real-life problems, we have to instil some concrete goals in our agent. We can achieve this by defining states that it will expect to be in or alternatively by directly specifying sensations that it will expect to solicit. These are generally called 'prior preferences' and are crucial in defining the free energy minimising, attracting states that are, effectively, the 'goals' of the

agent; in other words, the sorts of states an agent expects itself to seek out. The action states will try to fulfil these expectations, accordingly.

This can be seen as defining homeostatic, i.e. vitally important, state parameters. E.g. the CO_2 concentration in human blood is extremely important and tightly controlled, as opposed to the possible brightness perceived at the individual receptors of the retina, which can vary by orders of magnitude. Though humans might not directly change their behaviour depending on visual stimuli, a slight increase in the CO_2 concentration of their blood and the concurring decrease in pH will trigger chemoreceptors in the carotid and aortic bodies, which in turn will increase the activity of the respiratory centers in the medulla oblongata and the pons, leading to a fast and strong increase in ventilation, which might be accompanied by a subjective feeling of dyspnoea or respiratory distress. These hard-wired connections between vitally important body parameters and direct changes in perception and action might be very similar to our approach to encode the goal-relevant states or sensations explicitly.

3.2.8 Optimization using Evolution Strategies

Without action, the model and the objective would be very similar to the objective functions of Kingma and Welling (2014); Rezende et al (2014); Chung et al (2015). So one could just do a gradient descent on the estimated free action with respect to the parameters θ, using a gradient-based optimisation algorithm such as ADAM (Kingma and Ba, 2014). However, to do this here, we would have to backpropagate the partial derivatives of the free action with respect to the parameters through the dynamics of the world. I.e. our agent would have to know the equations of motions of its environment or at least their partial derivatives. As this is obviously not the case, and as many environments are not even differentiable, we have to resort to another approach.

It was recently shown that evolution strategies allow for efficient, large scale optimisation of complex, non-differentiable objective functions $FA(\theta)$ (Salimans et al, 2017). We apply this algorithm as follows: Instead of searching for a single optimal parameter set θ^*, we introduce a distribution on the space of parameters, which is called the "population density". We optimise its sufficient statistics to minimize the expected free action under this distribution. The population density can be seen as a population of individual agents, whose average fitness is optimised, hence the name. The expected free action over the population as function of the sufficient statistics ψ of the population density $p_\psi(\theta)$ is:

$$\eta(\psi) = \langle FA(\theta) \rangle_{p_\psi(\theta)}$$

Now we can calculate the gradient of η with respect to the sufficient statistics ψ:

$$\nabla_\psi \eta(\psi) = \langle FA(\theta) \nabla_\psi \ln p_\psi(\theta) \rangle_{p_\psi(\theta)}$$

Using a diagonal Gaussian for $p_\psi(\theta) = \mathrm{N}(\theta; \mu^\psi, \sigma^\psi)$ the gradient with respect to the means μ^ψ is:

$$\nabla_{\mu^\psi} \eta(\psi) = \left\langle FA(\theta) \frac{1}{(\sigma^\psi)^2} \left(\theta - \mu^\psi\right) \right\rangle_{p_\psi(\theta)}$$

We will also optimise the standard deviations of our population density, using the corresponding gradient:

$$\nabla_{\sigma^\psi} \eta(\psi) = \left\langle FA(\theta) \frac{\left(\theta - \mu^\psi\right)^2 - \left(\sigma^\psi\right)^2}{(\sigma^\psi)^3} \right\rangle_{p_\psi(\theta)}$$

Drawing samples ϵ_i from a standard normal distribution $\mathrm{N}(\epsilon_i; 0, 1)$, we can approximate samples from $p_\psi(\theta)$ by $\theta_i = \mu^\psi + \sigma^\psi \epsilon_i$. Thus, we can approximate the gradients via sampling by:

$$\nabla_{\mu^\psi} \eta(\psi) \approx \frac{1}{n_{\mathrm{pop}}} \sum_{i=1}^{n_{\mathrm{pop}}} FA(\theta_i) \frac{\epsilon_i}{\sigma^\psi} \tag{1}$$

and

$$\nabla_{\sigma^\psi} \eta(\psi) \approx \frac{1}{n_{\mathrm{pop}}} \sum_{i=1}^{n_{\mathrm{pop}}} FA(\theta_i) \frac{\epsilon_i^2 - 1}{\sigma^\psi}$$

For reasons of stability we are not optimising σ^ψ directly, but calculate the standard deviations using:

$$\sigma^\psi = \mathrm{softplus}(\tilde{\sigma}^\psi) + \sigma^\psi_{\mathrm{min}}$$

with $\mathrm{softplus}(x) = \ln(1 + e^x)$. By choosing $\sigma^\psi_{\mathrm{min}} = 10^{-6}$ constant and optimising $\tilde{\sigma}^\psi$ we prevent divisions-by-zero and make sure that there is no sign-switch during the optimisation. The chain rule gives:

$$\nabla_{\tilde{\sigma}^\psi} \eta(\psi) \approx \frac{1}{n_{\mathrm{pop}}} \sum_{i=1}^{n_{\mathrm{pop}}} FA(\theta_i) \frac{\epsilon_i^2 - 1}{\sigma^\psi} \frac{\partial \sigma^\psi}{\partial \tilde{\sigma}^\psi} = \frac{1}{n_{\mathrm{pop}}} \sum_{i=1}^{n_{\mathrm{pop}}} FA(\theta_i) \frac{\epsilon_i^2 - 1}{\sigma^\psi} \frac{\exp\left(\tilde{\sigma}^\psi\right)}{1 + \exp\left(\tilde{\sigma}^\psi\right)} \tag{2}$$

Using these gradient estimates, we now optimise the expected free action bound under the population density using ADAM as gradient based optimiser (Kingma and Ba, 2014). The corresponding pseudocode is shown in algorithm 2.

Algorithm 2 Optimisation of the free action bound.

Initialize the population density on the parameters θ using randomised sufficient statistics $\psi = \{\mu^\psi, \tilde{\sigma}^\psi\}$
while Expected free action $\eta(\psi)$ has not converged **do**
    Draw n_{pop} samples $\theta_i = \mu^\psi + \sigma^\psi \epsilon_i$, $\epsilon_i \sim \text{N}(0,1)$
    for each Sample **do**
        Approximate the free action bound $FA(\theta_i)$ using algorithm 1 with n_F processes
    end for
    Approximate the gradients $\nabla_\psi \eta$ using equations 1 and 2.
    Perform a parameter update on ψ using ADAM with these gradient estimates.
end while

Algorithm 3 Sampling from the agent's generative model. Concrete samples and values of a variable are marked using a hat, e.g. $\hat{\mathbf{s}}_t$.

Initialize n_p processes with $\hat{\mathbf{s}}_0 = (0, ..., 0)$.
for each Process **do**
    for $t = 1, ..., T$ **do**
        Draw a single sample $\hat{\mathbf{s}}_t$ from $p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1})$
        Sample single observations $\mathbf{o}_t$ from each likelihood $p_\theta(\mathbf{o}_t|\hat{\mathbf{s}}_t)$
        Carry $\hat{\mathbf{s}}_t$ over to the next time step.
    end for
end for

3.2.9 Sampling from the Learned Generative Model

Once the agent has been optimised, we can not only propagate it through the environment, but we can also draw samples from its generative model of the world. This enables us to examine the agent's beliefs about the dynamics of the world. For unconstrained sampling, one can just use the agent's prior on state transitions $p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1})$ to propagate processes through it's latent space and then use the learned likelihood function to generate samples. This is effectively a posterior predictive density over outcomes. The procedure is described in detail in algorithm 3. Sampling many processes ($n_p \approx 10^2 - 10^4$) in parallel yields a good approximation, although only a single sample is drawn from the prior density and from each factor of the likelihood function per individual process and per time step.

    Furthermore, we can also use a Markov-Chain Monte Carlo algorithm to draw constrained samples from its generative model (Rezende et al, 2014). This can on one hand be used to impute missing inputs. E.g. by sampling from

$$p_\theta(o_{1,1}, o_{1,2}, ..., o_{1,T} | \mathbf{o}_{2:d_o,1}, \mathbf{o}_{2:d_o,2}, ..., \mathbf{o}_{2:d_o,T})$$

we can get the agent's estimate on an unobserved sensory channel o_1, given a series of observations of the other sensory channels $o_2, ..., o_{d_o}$, where d_o is the dimensionality of the observations, i.e. the number of sensory channels.

    If one or more of the agent's sensory channels are *proprioceptive* sensory channels, i.e. if they give the agent input about the state of its own effector organs $\mathbf{a}$ (e.g. in terms of corollary discharges, (Crapse and Sommer, 2008)) ,

we can use constrained sampling to examine its beliefs about the trajectory of latent states in the world, given its actions. If we summarise the proprioceptive channels as $\mathbf{o}_a$ and the exteroceptive channels as $\mathbf{o}_e$, we can get an explicit probe of the agent's beliefs about its influence on the world.

$$p_\theta(\mathbf{o}_{e,1}, \mathbf{o}_{e,2}, ..., \mathbf{o}_{x,T} | \mathbf{o}_{a,1}, \mathbf{o}_{a,2}, ..., \mathbf{o}_{a,T})$$

The required algorithm is developed in appendix F of Rezende et al (2014) and described here as algorithm 4. The basic idea is to use the de-noising properties of autoencoders due to the learned, abstract and robust representation and their ability to generate low-dimensional representations capturing the regularity and systematic dependencies within the observational data. Thus, the workings of this algorithm can be understood as follows: First, all but the given sensory channels are randomly initialised. These partly random sensory observations are now encoded using the variational distribution q. The resulting state tries to represent the observation within the low-dimensional, robust representation learned by the agent and should thereby be able to remove some of the "noise" from the randomly initialised channels, just in line with the classic idea of an autoencoder (Hinton and Salakhutdinov, 2006). From this variational distribution a sample is drawn, which can be used to generate new, sensory samples, that are already less noisy. Now the known observations can be reset to their respective values and the denoised observations can again be encoded, using the variational density q. By iteratively encoding the denoised samples together with the given sensory inputs, the iterative samples from the abstract, robust representation will converge to the most probable cause (in terms of the hidden states) for the actually observed sensations under the generative model. As the variational density and the generative model capture the regularities and dependencies within the observations, the observations generated from this representation will converge to the distribution of the unknown observations, given the observed channels. Rezende et al (2014) provided a proof that this is true, given that the unobserved channels are not initialised too far away from the actual values. In summary, this means that we can characterise an agent's beliefs about unobserved outcomes in terms of a posterior predictive density over a subset of outcomes (e.g., exteroceptive), given another subset (e.g., proprioceptive).

4 Experiments

Here we describe an own implementation of the mountaincar environment and a deep active inference agent that we train to solve it.

Furthermore we show that a deep active inference agent can also solve the MountainCar-v0 environment of OpenAI Gym (Brockman et al, 2016). The full code of these experiments and the scripts to reproduce all figures in this paper can be downloaded here: `https://www.github.com/kaiu85/deepAI_paper`.

Algorithm 4 Markov-Chain Monte Carlo algorithm for constrained sampling from the agent's generative model. Concrete samples and values of a variable are marked using a hat, e.g. $\hat{s}_t$. We call the set of sensory channels that we condition on $\mathbf{o}_a$, and the set that we sample from $\mathbf{o}_e$. The noise parameters $\boldsymbol{\mu}_e$, $\boldsymbol{\sigma}_e$ should be set according to the mean statistics of the corresponding channels, e.g. known from previous interactions of the agent with its environment.

> **Given:** Time course $\mathbf{o}_{a,1}, ..., \mathbf{o}_{a,T}$ of known sensory channels $\mathbf{o}_a$
> Initialize n_s processes with $\hat{\mathbf{s}}_0 = (0, ..., 0)$
> **for each** Process **do**
>   **for** $t = 1, ..., T$ **do**
>   Initialize $\hat{\mathbf{o}}_{e,t}^0$ with sample from $\mathrm{N}(\mathbf{o}_{e,t}^0; \boldsymbol{\mu}_e, \boldsymbol{\sigma}_e)$
>   **for** $i = 1, ..., n_i$ **do**
>   Sample $\hat{\mathbf{s}}_t^i$ from $q_\theta(\mathbf{s}_t^i | \hat{\mathbf{s}}_{t-1}, \hat{\mathbf{o}}_{e,t}^{i-1}, \mathbf{o}_{a,t})$
>   Sample new estimates $\hat{\mathbf{o}}_{e,t}^i$ from likelihood $p_\theta(\mathbf{o}_{e,t}^i | \hat{\mathbf{s}}_t^i)$
>   **end for**
>   Set $\hat{\mathbf{s}}_t = \hat{\mathbf{s}}_t^{n_i}$
>   Set $\hat{\mathbf{o}}_{e,t} = \hat{\mathbf{o}}_{e,t}^{n_i}$
>   Return $\hat{\mathbf{o}}_{e,t}$
>   Carry $\hat{\mathbf{s}}_t$ over to the next time step.
>   **end for**
> **end for**

4.1 Own Implementation of Mountaincar Environment

4.1.1 Environment

The agent will act in a discrete time version of the mountain car world (Moore, 1991). It will start at the bottom $x = -0.5$ of a potential landscape $G(x)$, which is shown in figure 3.

The agent itself is a small car on this landscape, whose true, physical state is given by its current position x_t, its velocity v_t, and the current state a_t of its effector organs. In the case of humans this would be the state of the muscles, here the state a_t describes the car's steering direction and the throttle of the car's engine. The dynamics of the environment are given by a set of update equations, which can be summarised symbolically by

$$(x_{t+1}, v_{t+1}) = \mathbf{R}(x_t, v_t, a_t)$$

This equation can be decomposed into a set of equations, as multiple forces will act on the agent. The downhill force F_g due to the shape of the landscape depends on its position x. It is given by

$$F_g(x) = -\frac{\partial}{\partial x} G(x) = \begin{cases} 0.05 \left(-2x - 1\right), & x < 0 \\ 0.05 \left(-(1 + 5x^2)^{-1/2} - x^2(1 + 5x^2)^{-3/2} - x^4/16\right), & x \geq 0 \end{cases}$$

and shown in figure 4.

The agent's motor can generate a force F_a, depending on its current action state a
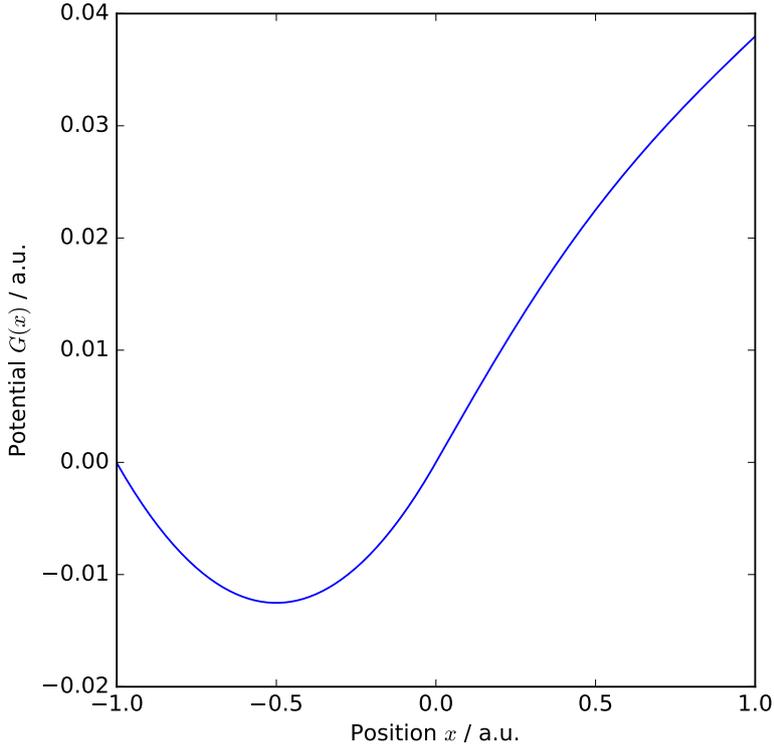
$$F_a(a) = 0.03 \tanh(a)$$

Fig. 3 Potential of the mountain car problem. The agent is a small cart starting at the bottom of the valley at $x = -0.5$

As mentioned above, the action state controls the steering direction (positive or negative) and the throttle of the engine. However, the absolute force that the engine can generate is limited to the interval $(-0.03, 0.03)$ due to the tanh function.

The laminar friction force F_f depends linearly on the agent's current velocity v

$$F_f(v) = -0.25v$$

Thus, the total force action on the agent is

$$F_{\text{total}}(x_t, a_t, v_t) = F_g(x_t) + F_a(a_t) + F_f(v_t)$$

This leads to the following update equations for velocity

$$v_{t+1} = v_t + F_{\text{total}}(x_t, a_t, v_t)$$

and position

$$x_{t+1} = x_t + v_{t+1}$$
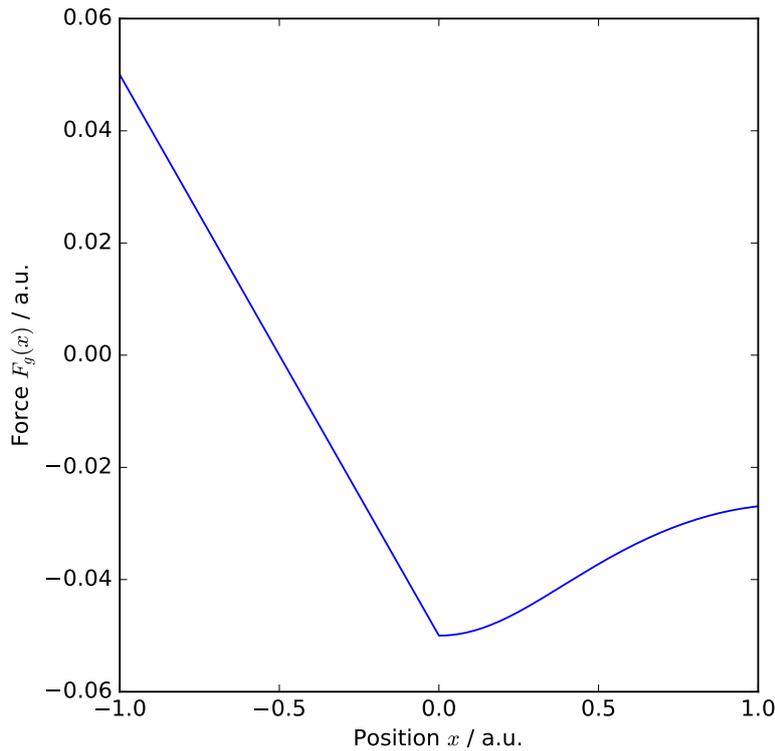
Fig. 4 Downhill force F_g due to the slope of the potential landscape.

Initially, the agent is resting at the bottom of the landscape, i.e. $\mathbf{s}_0^* = (x_0, v_0) = (-0.5, 0.0)$.

We will later set the agent the goal of reaching and staying at $x = 1.0$. However, due to the shape of the potential landscape and the resulting force $F_g(x)$, as shown in figure 4, we notice that the landscape gets very steep at $x = 0.0$. As the force generated by the motor is limited to the interval $(-0.03, 0.03)$, the agent is not strong enough to climb the slope at $x = 0$, which results in a downhill force of $F_g(0) = -0.05$, without some additional momentum. Thus, to overcome this slope the agent has to move uphill to the left, *away* from its target position, at first. In this way, it can acquire the required additional momentum, which allows it to overcome the opposite slope. In this way, the mountain car environment — although very simple — is not completely trivial, as the agent has to learn that a direct approach of the target position will not succeed and it needs to acquire a strategy that initially leads away from its target.

The simulated trials in this environment are always 30 time steps long. The resulting trajectory of outcomes (and posterior beliefs) furnish the quantities necessary to evaluate free energy at each time point. The ensuing path

integral or free action can then be used to update the ensemble; after which the environment is reset.

4.1.2 Sensory Inputs

The agent has a noisy sense $o_{x,t}$ of its real position x_t

$$p(o_{x,t}|x_t) = \mathrm{N}(o_{x,t}; x_t, 0.01)$$

Here $\mathrm{N}(x; \mu, \sigma)$ denotes a Gaussian probability density over x with mean μ and standard deviation σ. To show that the agent can indeed learn a complex, nonlinear, generative model of its sensory input, we add another sensory channel with a nonlinear, non bijective transformation of x_t:

$$o_{h,t} = \exp\left(-\frac{(x_t - 1.0)^2}{2 \cdot 0.3^2}\right)$$

Note that the agent has no direct measurement of its current velocity v_t, i.e. it has to infer it from the sequence of its sensory inputs, which all just depend on x_t.

To see if the agent understands its own actions, we also equip it with a proprioceptive sensory channel, allowing it to observe its action state a_t:

$$o_{a,t} = a_{t-1}$$

Note that having a direct sense of its action state a_t is not necessary for an active inference agent to successfully control its environment (c.f. supplementary figures 1-3). E.g. the reflex arcs that humans use to increase the likelihood of their sensory inputs, given their generative model of the world, feature their own, closed loop neural dynamics at the level of the spinal cord. So humans do not (and do not have to) have direct access to the action states of their muscles, when they just lift their arm by expecting it to lift. However, adding this channel will allow us later to directly sample the agent's proprioceptive sensations from its generative model, to check if it understands its own action on the environment.

4.1.3 The Agent's Generative Model of the Environment

The agent possesses a 10 dimensional latent space, i.e. $\mathbf{s}_t \in \mathbb{R}^{10}$. We model the transition distribution $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t)$ as diagonal Gaussian, where the means and standard deviations are calculated from the current state $\mathbf{s}_t$ using neural networks. I.e.

$$p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t) = \mathrm{N}(\mathbf{s}_{t+1}; \boldsymbol{\mu}_\theta^t(s_t), \boldsymbol{\sigma}_\theta^t(s_t))$$

We use a fully connected single layer network with a tanh nonlinearity to calculate the means $\boldsymbol{\mu}_\theta^t(\mathbf{s}_t)$ and another fully connected single layer network with $\mathrm{softplus(x)} = \ln(1 + e^x)$ as nonlinear transfer function to calculate the

standard deviations $\boldsymbol{\sigma}_\theta^t(\mathbf{s}_t)$. We use θ to encompass all parameters of the generative model, that we are going to optimise. In practice, this means all the weights and bias parameters of the neural networks to calculate the means and standard deviations. The use of diagonal Gaussians in the prior leads to hidden, abstract representations within which independent causes of observations are well separated (i.e., are a priori orthogonal).

Similarly, the likelihood functions for each of the three observables also factorise

$$p_\theta(o_{x,1}, ..., o_{x,T}, o_{h,1}, ..., o_{h,T}, o_{a,1}, ..., o_{a,t}|\mathbf{s}_1, ..., \mathbf{s}_T)$$

$$= \prod_{t=1}^{T} p_\theta(o_{x,t}|\mathbf{s}_t) \prod_{t=1}^{T} p_\theta(o_{h,t}|\mathbf{s}_t) \prod_{t=1}^{T} p_\theta(o_{a,t}|\mathbf{s}_t)$$

So the likelihood of each observable $o_{x,t}, o_{a,t}, o_{h,t}$ for a given time t only depends on the current state $\mathbf{s}_t$. We again use Gaussian distributions, to obtain

$$p_\theta(o_{x,t}|\mathbf{s}_t) = \qquad \qquad \mathrm{N}(o_t; \boldsymbol{\mu}_\theta^x(\mathbf{s}_t), \boldsymbol{\sigma}_\theta^x(\mathbf{s}_t))$$
$$p_\theta(o_{a,t}|\mathbf{s}_t) = \qquad \qquad \mathrm{N}(o_t; \boldsymbol{\mu}_\theta^a(\mathbf{s}_t), \boldsymbol{\sigma}_\theta^a(\mathbf{s}_t))$$
$$p_\theta(o_{h,t}|\mathbf{s}_t) = \qquad \qquad \mathrm{N}(o_t; \boldsymbol{\mu}_\theta^h(\mathbf{s}_t), \boldsymbol{\sigma}_\theta^h(\mathbf{s}_t))$$

We calculate the sufficient statistics of these Gaussian distributions from the state $\mathbf{s}_t$ using a deep feedforward network with three hidden layers of $d_h = 10$ neurons each, using $\mathrm{relu}(x) = \max(0, x)$ as nonlinear transfer function. We use a linear output layer to calculate the means of the Gaussian variables and a second output layer with $\mathrm{softplus}(x) = \ln(1 + e^x)$ as nonlinear transfer function to calculate the standard deviations. Although this structure is on the first glance different from the hierarchical dynamical models developed by Friston (2008), the deep and nonlinear structure of the feedforward network also allows for structured noise to enter at different levels of this hierarchy.

4.1.4 Variational Density

We again use diagonal Gaussians to model the variational density

$$q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t}) = \mathrm{N}(\mathbf{s}_t; \boldsymbol{\mu}_\theta^q(\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t}), \boldsymbol{\sigma}_\theta^q(\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t}))$$

The sufficient statistics are calculated using a deep feedforward network with two hidden layers of $d_h = 10$ neurons each, using $\mathrm{relu}(x) = \max(0, x)$ nonlinearities. The means are again calculated using a linear, and the standard deviations using a softplus output layer. While the use of diagonal Gaussians in the prior leads to hidden, abstract representations within which independent causes of observations are optimally separated, i.e. which has favourable properties, here this choice is just due to practical considerations. The fact that we choose both the variational density and the prior density as diagonal Gaussians, will later allow us to use a closed formula to calculate the Kullback-Leibler Divergence between the densities. However, if a more flexible posterior

is required, normalising flows allow a series of nonlinear transformations of the diagonal Gaussian used for the variational density, by which it can approximate very complex and multimodal posterior distributions (Rezende and Mohamed, 2015; Kingma et al, 2016; Tomczak and Welling, 2016).

4.1.5 Action States

We use a one-dimensional Gaussian form for the action function

$$p_\theta(a_t|\mathbf{s}_t) = \mathrm{N}(a_t; \mu_\theta^a(\mathbf{s}_t), \sigma_\theta^a(\mathbf{s}_t))$$

whose sufficient statistics are calculated using a deep feedforward network with one fully connected hidden layers of $d_h = 10$ neurons, a linear output layer for the mean, and a softplus output layer for the standard deviation.

4.1.6 Goal Directed Behaviour

In this concrete case, we want to propagate the agent for 30 time steps and want it to be at $x = 1.0$ for at least the last ten time steps. As the agent's priors are over the hidden states, we introduce a hard-wired state which just represents the agent's current position. We do this by explicitly encoding the agent's sense of position $o_{x,t}$ to the first dimension of the state vector $s_{1,t}$:

$$q_\theta(s_{1,t}|\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t}) = \mathrm{N}(s_{1,t}; 0.1 o_{x,t}, 0.01)$$

We specify the agent's prior expectations by explicitly setting the prior over the first dimension of the state vector for $t > 20$:

$$p_\theta^t(s_{1,t}|\mathbf{s}_{t-1}) = \mathrm{N}(s_{1,t}; 0.1, 0.01), \text{if } t > 20$$

4.1.7 The Free Action Objective

Now we have everything that we need to put our objective function together. We use the following form of the variational free action bound (c.f section 2):

$$FA(o,\theta) = \sum_{t=1}^{T} \Bigg[\big\langle -\ln p_\theta(o_{x,t}|\mathbf{s}_t) - \ln p_\theta(o_{h,t}|\mathbf{s}_t) - \ln p_\theta(o_{a,t}|\mathbf{s}_t) \big\rangle_{q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t})}$$
$$+ D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, o_{x,t}, o_{h,t}, o_{a,t}) || p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1})) \Bigg]$$

using $\mathbf{s}_0 = (0, ..., 0)^T$.

The sampling proceeds according to algorithm 1, where we use the closed form of the KL-Divergence for diagonal Gaussians:

$$D_{\mathrm{KL}}(q_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1}, \hat{o}_{x,t}, \hat{o}_{h,t}, \hat{o}_{a,t}) || p_\theta(\mathbf{s}_t|\hat{\mathbf{s}}_{t-1})) =$$
$$D_{\mathrm{KL}}(\mathrm{N}(\mathbf{s}_t; \boldsymbol{\mu}^q, \boldsymbol{\sigma}^q) || \mathrm{N}(\mathbf{s}_t; \boldsymbol{\mu}^t, \boldsymbol{\sigma}^t)) = \sum_{i=1}^{n} \frac{1}{2} \left(2 \ln \frac{\sigma_i^t}{\sigma_i^q} + \frac{(\sigma_i^q)^2 + (\mu_i^q - \mu_i^t)^2}{(\sigma_i^t)^2} - 1 \right)$$

Here N($\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}$) denotes a diagonal Gaussian distribution over $\mathbf{x}$ with mean vector $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$

4.1.8 Experimental Set Up and Parameters

The experiments were performed on a desktop PC equipped with a 2013 NVIDIA GTX Titan GPU. The parameter values used in the optimisation algorithm 2 and the required estimation of the free action bound using algorithm 1 are shown in table 1. Note that we approximate the free action by a single process ($n_F = 1$) for each sample from the population density. This is possible, as we draw many ($n_{\mathrm{pop}} = 10^4$) samples from the population density. Using more processes for each sample, while keeping the total number $n_{\mathrm{tot}} = n_{\mathrm{pop}} n_F$ of simulated processes constant, results in slower convergence, as the coverage of the parameter space has to be reduced, while the total variability due to the stochastic approximation of the total bound stays approximately constant. The full code of this implementation and the scripts to reproduce all figures in this paper can be downloaded here: `https://www.github.com/kaiu85/deepAI_paper`. The code is written in Python 2.7, using Theano (Theano Development Team, 2016) for GPU optimised tensor operations.

4.1.9 Results

The evolution strategies based optimisation procedure used less than 300 MB of GPU memory and took less than 0.4 s per iteration. Figure 5 shows the convergence of the free action bound within 30,000 training steps (less than 3.5 hours). It quickly converges from its random starting parameters to a plateau. Here the agent tries to directly climb the hill and gets stuck at the steep slope. However, after only about 250 updates of the population density the agent discovers, that it can get higher by first moving in the opposite direction, thereby gaining some momentum, which it can use to overcome the steep parts of the slope. This insight leads to a sudden, rapid decline in free action (Friston et al, 2017a). The rapid development of the agent's strategy and its quick adoption of the insight, that an initial movement away from its target position is beneficial, is illustrated in figure 6.

The agent's trajectory after about 30,000 training steps is shown in figure 7: It takes a short left swing to gain the required momentum to overcome the steep slope at $x = 0$, then directly swings up to its target position $x = 1.0$, and stays there by applying just the right force to counteract the non-zero downhill force at the target position $x = 1.0$.

After 30,000 optimisation steps, the agent has also developed quite some understanding of its environment. We can compare figure 7, which was generated by actually propagating the agent in its environment, with figure 8, which was generated by sampling from the agent's generative model of its environment, *without any interaction* with the environment. We see that the agent not only learned the time course of its proprioceptive sensory channel
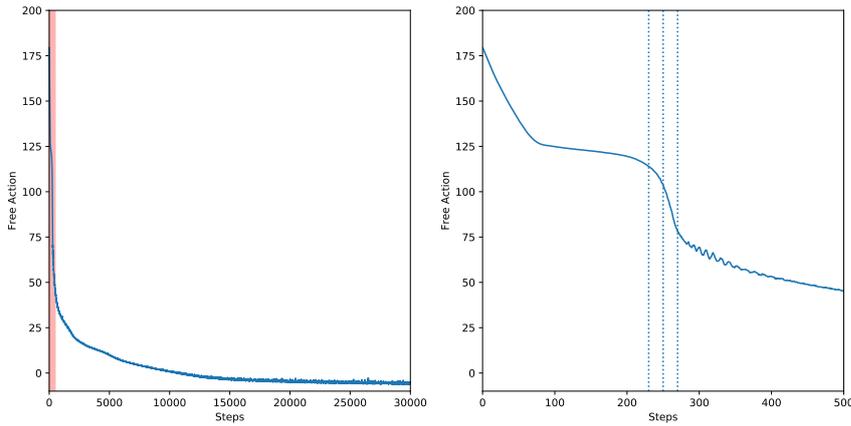
Fig. 5 Convergence of free action, averaged over the current population of agents, using parameters in table 1. The area shaded in red in the left plot was enlarged in the right plot. The iterations for which the agent's performance is shown in figure 6 are indicated by dotted vertical bars.

parameter	description	value
α	learning rate of ADAM optimiser	0.001
(β_1, β_2)	exponential decay rates for moment estimation of ADAM optimiser	(0.9, 0.999)
ϵ	noise parameter of ADAM optimiser	10^{-8}
n_F	number of processes to approximate free action for each sample from the population density	1
n_{pop}	number of samples from the population density	10^4

Table 1 Parameter values used in algorithms 1 and 2.

o_a and its sense of position o_x, but also the — in this setting irrelevant — channel o_h, which is just a very nonlinear transformation of its position. Note that each panel of figure 8 shows 10 processes sampled from the generative model as described in algorithm 3. Note that we are approximating each density just by a single sample per time step and per process. Thus, although our estimates seem a bit noisy, they are very consistent with the actual behaviour of the agent and the variability can easily be reduced by averaging over several processes.

Having learned a generative model of the environment, we can not only propagate it freely, but we can also use it to test beliefs of the agent, given some a priori assumptions on the time course of certain states or sensory channels, using algorithm 4, described in section 3.2.9. We sampled the agent's prior beliefs about his trajectory $o_{x,t}, o_{h,t}$, given its proprioceptive inputs $o_{a,t}$, i.e. $p(o_{x,1}, ..., o_{x,T}, o_{h,1}, ..., o_{h,T} | o_{a,1}, ..., o_{a,T})$. Using the above example we took
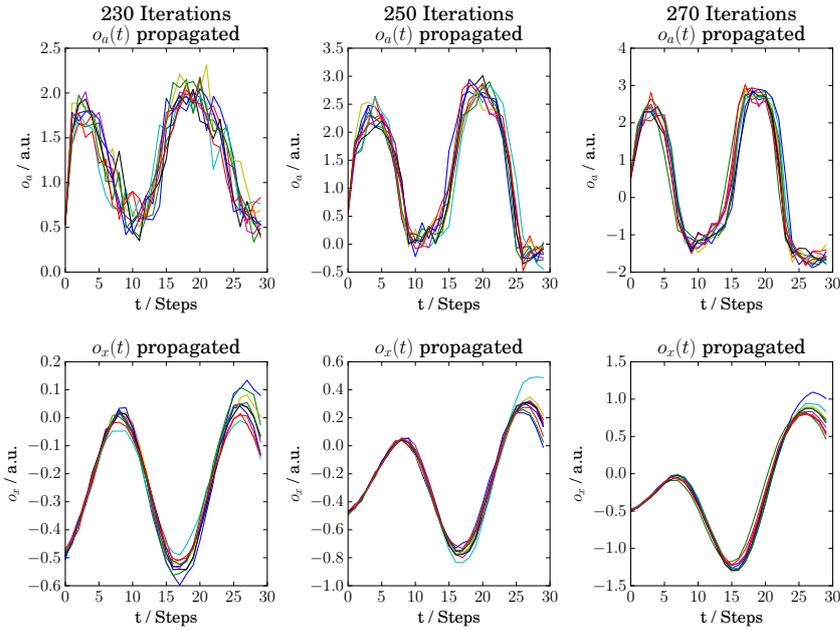
Fig. 6 The agent's performance after 230, 250, and 270 training steps, using the mean parameters of the population density. It has just realised that by moving uphill a bit to the left (from $t = 15$ to $t = 20$), it can reach a higher position (around $t = 27$) than by directly going upwards (c.f. $t = 9$). Shown are the agent's proprioception o_a (upper row), and its sense of position o_x (lower row).

the average time course of the proprioceptive channel for the true interaction with the environment and shifted it back 10 time steps. The results are shown in figure 9. First, we see that not all of the 10 sampled processes did converge. This might be due to the Markov-Chain-Monte-Carlo-Sampling approach, in which the chain has to be initialised close enough to the solution to guarantee convergence. However, for 9 out of 10 processes, the results look very similar to the true propagation (figure 7) and the unconstrained samples from the generative model (figure 8), only shifted back 10 time steps.

4.2 MountainCar-v0 Environment of OpenAI Gym

Here we show that our deep active inference architecture is able to solve a more standardised version of the mountain car task. This implementation is part of OpenAI Gym, a library of reinforcement learning problems widely used throughout the reinforcement learning community (Brockman et al, 2016).
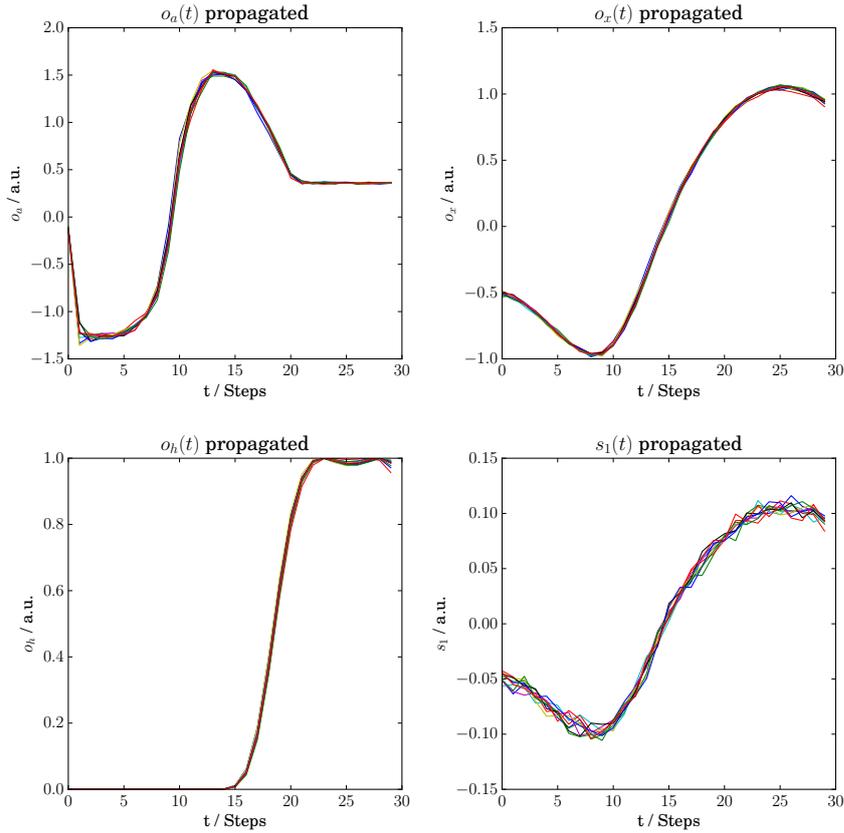
Fig. 7 The agent's performance after 30,000 training steps, using the mean parameters of the population density. It tightly sticks to a strategy, shown here in terms of the proprioceptive sensory channel o_a (upper left). The resulting trajectory (shown in terms of o_x, upper right) first leads uphill to the left, away from the target position, to gain momentum and overcome the steep slope at $x = 0$. The nonlinearly modified sensory channel o_h is shown on the lower left and the "homeostatic" hidden state s_1 on the lower right.

4.2.1 Environment

The MountainCar-v0 environment differs in some aspects from our own implementation:

- The starting position of the agent at the bottom of the valley is randomised.
- Depending on the starting position, the shape of the landscape requires the agent to make two swings to build up enough momentum to reach its target positions.
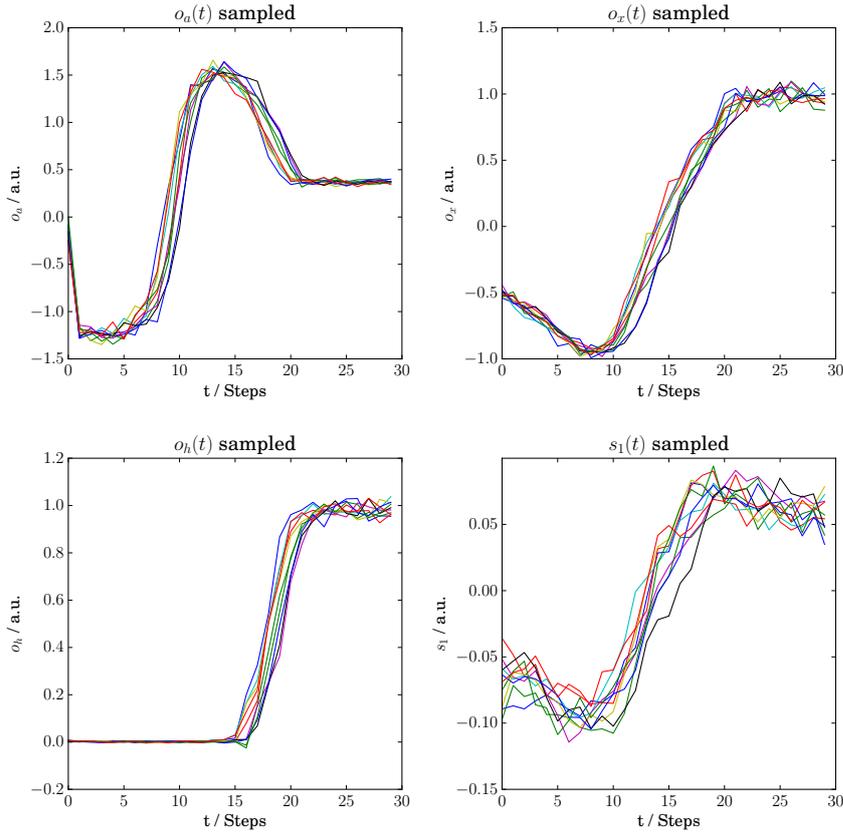- The agent has direct access to its position **and** velocity.

Fig. 8 Ten processes sampled from the agent's generative model of the world after 30,000 training steps, using the mean parameters of the population density. Shown are the prior expectations on the proprioceptive channel o_a (upper left), the agent's sense of position o_x (upper right), a nonlinear transformation of the position o_h, and the agent's prior expectation on its "homeostatic" state variable s_1. Note that each distribution is approximated by a single sample per time step per process.

– The agent has to be trained to maximise the sum of a separate reward channel over each trial.
– The reward channel is always -1, except when the car has reached its final position on top of the hill.
– As soon as the car reaches its target position, the environment terminates. I.e. the lengths of the time series get shorter, as the agent learns to reach its goal earlier. This also makes the reward very sparse, since maximally one time step gets rewarded at each iteration, and only if the agent reaches its goal position.
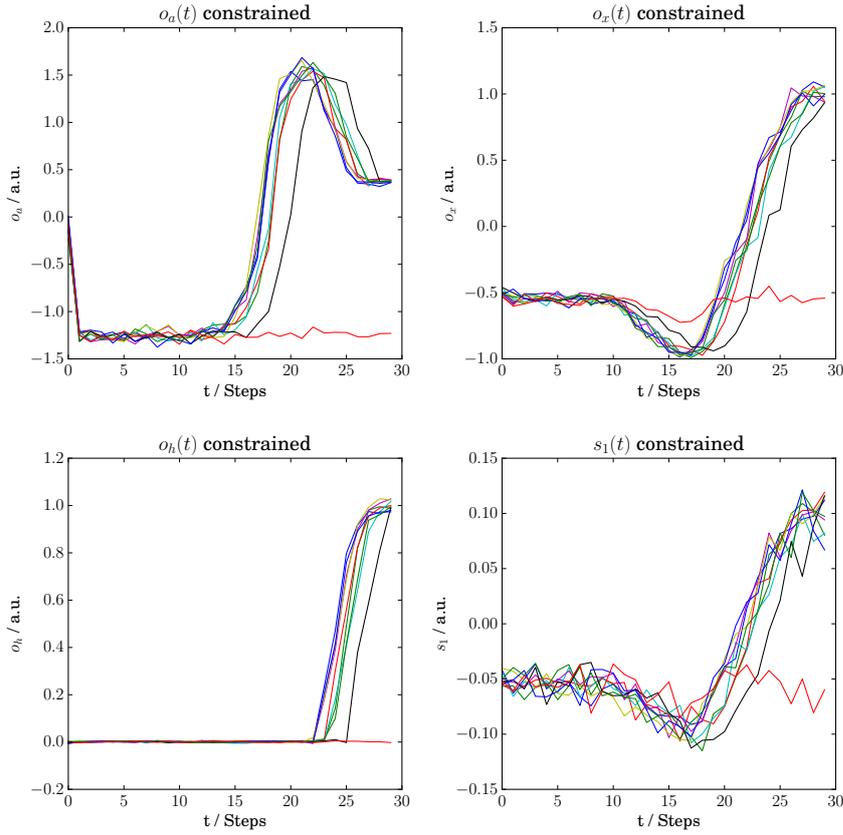
Fig. 9 Ten processes sampled from the agent's generative model of the world after 30,000 training steps, constrained on a given trajectory of the proprioceptive input o_a (upper left), using the mean parameters of the population density. Shown are the constrained expectations on the proprioceptive channel o_a (upper left), the agent's sense of position o_x (upper right), a nonlinear transformation of the position o_h, and the agent's constrained expectation on its "homeostatic" state variable s_1. Note that each distribution is approximated by a single sample per time step per process.

- The action variable $\mathbf{a}_t$ is a categorical variable with 3 possible states: push to the left, do nothing, push to the right.
- When the agent is unable to reach the goal position, the environment resets after 200 steps.

A detailed description of the environment and the code of the environment are given at `https://github.com/openai/gym/wiki/MountainCar-v0` and `https://github.com/openai/gym/blob/master/gym/envs/classic_control/mountain_car.py`.

4.2.2 Sensory Inputs

The agent has direct access to its position and velocity: $\mathbf{o_t} = (x_t, v_t) = \mathbf{s}_t^*$ Furthermore it possesses a reward channel $o_{\text{rew},t}$, that indicates if it is at is goal position ($o_{\text{rew},t} = 1$), or not ($o_{\text{rew},t} = -1$).

4.2.3 The Agent's Generative Model of the Environment

Again, the dimensionality of the agent's latent space is 10, i.e. $\mathbf{s}_t \in \mathbb{R}^{10}$. We use the same architecture as in the previous agent, to model the prior distribution on state transitions by a diagonal Gaussian distribution:

$$p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t) = \mathrm{N}(\mathbf{s}_{t+1}; \boldsymbol{\mu}_\theta^t(s_t), \boldsymbol{\sigma}_\theta^t(s_t))$$

We use a fully connected single layer network with a tanh nonlinearity to calculate the means $\boldsymbol{\mu}_\theta^t(s_t)$ and another fully connected single layer network with softplus(x) = $\ln(1 + e^x)$ as nonlinear transfer function to calculate the standard deviations $\boldsymbol{\sigma}_\theta^t(s_t)$.

We again model the likelihood function using a diagonal Gaussian:

$$p_\theta(\mathbf{o}_t|\mathbf{s}_t) = \mathrm{N}(\mathbf{o}_t; \boldsymbol{\mu}_\theta^o(\mathbf{s}_t), \boldsymbol{\sigma}_\theta^o(\mathbf{s}_t))$$

We calculate the sufficient statistics of these Gaussian distributions from the state $\mathbf{s}_t$ using a deep feedforward network with one hidden layers of $d_h = 10$ neurons each, using tanh as nonlinear transfer function. We use a linear output layer to calculate the means of the Gaussian variables and a second output layer with $0.05 \cdot \mathrm{sigmoid}(x) = 0.05 \cdot (1 + e^{-x})^{-1}$ as nonlinear transfer function to calculate the standard deviations, thereby constraining the maximum standard deviation to 0.05.

4.2.4 Variational Density

We again use diagonal Gaussians to model the variational density

$$q_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{o}_t) = \mathrm{N}(\mathbf{s}_t; \boldsymbol{\mu}_\theta^q(\mathbf{s}_{t-1}, \mathbf{o}_t), \boldsymbol{\sigma}_\theta^q(\mathbf{s}_{t-1}, \mathbf{o}_t))$$

The sufficient statistics are calculated using a deep feedforward network with one hidden layer of $d_h = 10$ neurons each, using tanh nonlinearities. The means are again calculated using a linear, and the standard deviations using a softplus output layer.

4.2.5 Action States

We use a categorical distribution with three classes for the action function

$$p_\theta(\mathrm{a}_t|\mathbf{s}_t) = \mathrm{Cat}(a_t; \pi_\theta(\mathbf{s}_t))$$

The class probabilities $\pi_\theta(\mathbf{s}_t)$ are calculated from the hidden states using a feedforward network with three fully connected hidden layers of $d_h = 10$ neurons each, using tanh nonlinearities, and a $(\mathrm{softmax}(\mathbf{x}))_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$ output layer with three units, representing the three possible action choices.

4.2.6 Goal Directed Behaviour

Instead of creating an explicit latent representation of the reward channel, its simple structure allows us to directly put a prior on the corresponding sensory channel:

$$p(o_{\text{rew},t}) = \text{N}(o_{\text{rew},t}; 1.0, \sigma_{\text{rew}})$$

The standard deviation σ_{rew} allows us to weight the prior on the reward channel relative to the other terms of the variational free action.

4.2.7 The Free Action Objective

Now we have everything that we need to put our objective function together. We use the following form of the variational free action bound (c.f section 2):

$$FA(o, \theta) = \sum_{t=1}^{T} \left[\langle -\ln p_\theta(\mathbf{o}_t | \mathbf{s}_t) - \ln p_\theta(o_{\text{rew},t}) \rangle_{q_\theta(\mathbf{s}_t | \mathbf{s}_{t-1}, \mathbf{o}_t)} \right.$$

$$\left. + D_{\text{KL}}(q_\theta(\mathbf{s}_t | \mathbf{s}_{t-1}, \mathbf{o}_t) || p_\theta(\mathbf{s}_t | \mathbf{s}_{t-1})) \right]$$

using $\mathbf{s}_0 = (0, ..., 0)^T$.

The sampling again follows algorithm 1, using the closed form of the KL-Divergence for diagonal Gaussians.

4.2.8 Experimental Set Up and Parameters

The experiments were performed on the same desktop PC equipped with a 2013 NVIDIA GTX Titan GPU, using the same parameter values in the optimisation algorithm 2 and the required estimation of the free action bound using algorithm 1, as shown in table 1.

As the reward in this version of the mountaincar environment is very sparse, we had to start with a very low standard deviation $\sigma_{\text{rew,start}} = 0.001$ of the prior on the reward channel, to prevent the agent from settling into the local minimum associated to the stable and therefore very predictable state of the agent resting at the bottom of the valley. We annealed the standard deviation from its initial value to $\sigma_{\text{rew,end}} = 0.5$ with a time constant of $\tau_{\sigma_{\text{rew}}} = 0.003$ via

$$\sigma_{\text{rew}} = (\sigma_{\text{rew,start}} - \sigma_{\text{rew,end}}) \exp(-\tau_{\sigma_{\text{rew}}} n_{\text{update}}) + \sigma_{\text{rew,end}}$$

where n_{update} is the number of the current update step. The full code of this implementation and the scripts to reproduce all figures in this paper can be downloaded here: `https://www.github.com/kaiu85/deepAI_paper`.

4.2.9 Results

The evolution strategies based optimisation procedure used less than 300 MB of GPU memory. As the MountainCar-v0 Environment of OpenAI Gym runs on the CPU, we had to evaluate it sequentially for each process. We experimented with CPU parallelisation, but considering the very simple structure of the environment (c.f. `https://github.com/openai/gym/blob/master/gym/envs/classic_control/mountain_car.py`), the overhead to recollect the individual threads outweighed the benefit. Thus, an update step took about 30 s. The agent was able to successfully solve the environment after 1400 steps. Here we define solving according to the OpenAI rules for this environment, i.e. as obtaining a mean reward of more than -110, corresponding to the agent reaching its goal position in less than 110 time steps. To allow for a comparison with state of the art algorithms, we show the convergence of the mean reward (averaged over the population density) over the number of update steps in figure 10. Recall that for each agent, i.e. for each sample from the population density, a full trial is propagated and the resulting free action (and mean reward) are evaluated. I.e. a single update step in figure 10 corresponds to 10,000 simulated trials. Similar to the other agent, we optimised the agent for further 30,000 steps. The resulting dynamics when propagated through the world, and unconstrained samples from its generative model are shown in figure 11.

5 Discussion and Outlook

5.1 A Scalable and Flexible Implementation of Active Inference

In this paper we have shown that the free energy objective proposed by the active inference framework (Friston et al, 2010) enables an agent to find a solution to the mountain car environment, while concurrently building a generative model of itself and its environment. By implementing the internal dynamics of our agent using deep neural networks (LeCun et al, 2015; Goodfellow et al, 2016) and recurrent dynamics (Karpathy et al, 2015), it is able to approximate the true generative process of its environment by its own generative model. Furthermore, as we are using an efficient, black-box optimiser for non differentiable objective functions, the agent does not require direct access to any information regarding the true generative process or its derivatives for any given environment. As the implementation and optimisation of this agent uses methods that are applied to complex, large scale problems in machine learning and artificial intelligence (Chung et al, 2015; Rezende et al, 2016; Kingma et al, 2016), we hope that this class of agent can be of further use to demonstrate that active inference is able to solve more complex and realistic problems, such as the Atari and 3D robotic environments from OpenAI Gym (Brockman et al, 2016). The Atari environments require an agent to learn to play Atari games from raw pixel or RAM input, while the 3D robotic envi-
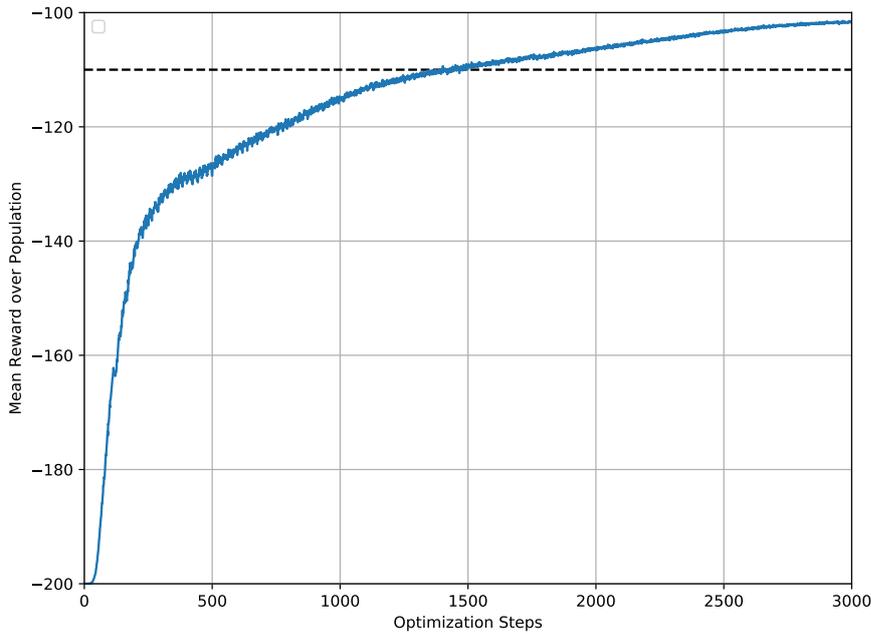
Fig. 10 Average reward over the current population density plotted over the number of update steps. The dashed horizontal line indicates a mean reward of -110, which is the threshold at which the environment is considered solved. Note that for each update, 10,000 samples from the population density are drawn and for each parameter set the free action (and the mean reward) are evaluated.

ronments use the MuJoCo physics engine (Todorov et al, 2012), to accurately simulate 3D robotic control problems.

5.2 Comparison to Original Implementation

In contrast to the original implementation of Friston et al (2010), our implementation is formulated in discrete time steps without an explicit representation of generalised coordinates. I.e. our agent has to learn how its observations of the position o_x are related between successive time steps and to form its own representation of its velocity. Furthermore, the agent's generative model of the world has — in contrast to the former work — not the same functional form as the true generative process. I.e. our agent also has to learn an approximation of the true dynamics in terms of its own generative model. This is possible due to the implementation of the agent's generative model in terms of a high-dimensional, recurrent neural network. These structures were shown to be able to efficiently learn and represent the underlying structure in complex time series data from real life applications (Karpathy et al, 2015; Le
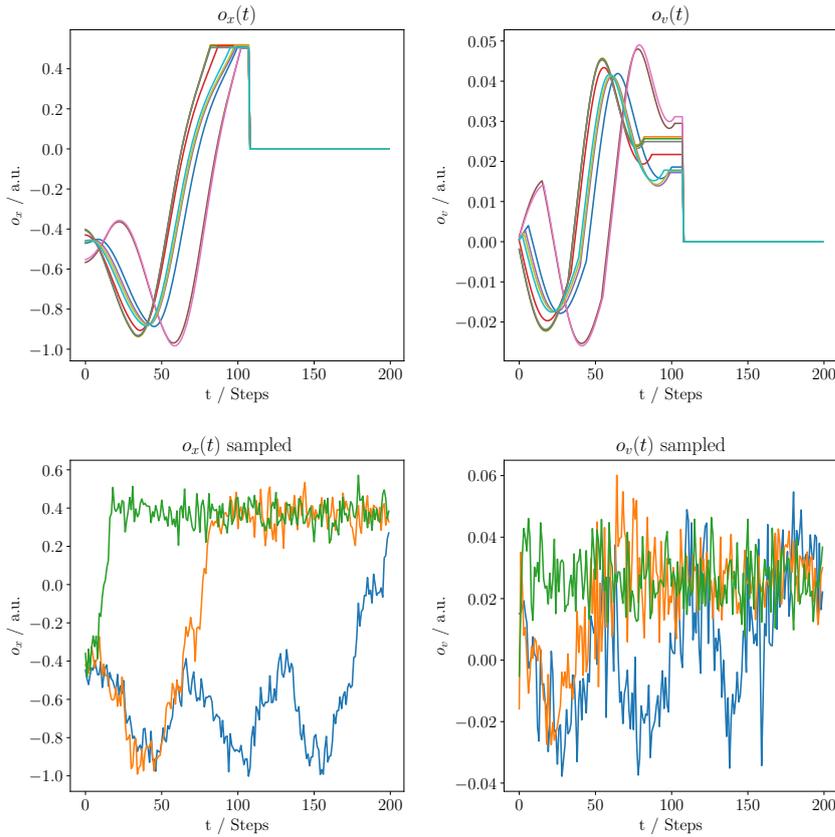
Fig. 11 Top: The agent's performance after 30,000 training steps, using the mean parameters of the population density. Shown are the agent's position (left) and velocity (right). Depending on its initial position, the agent has learned to take one or two swings, to gain enough momentum to reach its goal position. Bottom: Three sample time courses from the agent's generative model after 30,000 training steps. Although the agent always correctly imagines its initial and final positions, the dynamics in between vary. While it sometimes believes it can just go directly up the hill (green), it often converges to the actual dynamics, where it has to first make one or more swings to gain the required momentum (yellow, blue). Note the nice correspondence between the sampled position (left) and velocity (right) trajectories. Note that each distribution is approximated by a single sample per time step per process.

et al, 2015; Chung et al, 2015). Furthermore, the agent has no access to the partial derivatives of its sensory inputs with respect to its actions, as these also depend on a knowledge of the true generative process, which real agents usually do not possess. By using evolution strategies (Salimans et al, 2017), we are able to derive stochastic estimates of these gradients that enable us to train a full active inference agent despite these constraints.

We use a flexible, but static approximations of the complex dependence of the variational density on the current observations and the previous distribution over the latent variables, and of the action on the latent variables. This allows us to propagate and optimise our agent very fast and efficiently, however, it might reduce the context sensitivity as compared to optimising the sufficient statistics of the variational density and the action function at each single time step, which would be the discrete-time equivalent to the optimisation performed by Friston et al (2010). Our fixed mapping from the approximate posterior distribution of states of the world, given the agent's observations, to a distribution on possible actions is known as a state-action policy in control theory. It presupposes a Bayes optimal action for every state of the world. These are very common forms of policies; however, they are not universal policies in the sense that many actions depend upon previously encountered states. However, as we allow the agent to develop its own, very flexible representation of the current state of the world, it can basically include a compressed representation of the history of previous states, if this information is required to guide its action. Indeed, in the mountain car environment, our agent had to develop at least an implicit representation of its current momentum or velocity, which can be regarded as difference between its current and its previous position. Otherwise it would not have been able to successfully solve the mountain car problem, as the agent had to learn to move left from its initial position, to acquire some additional momentum, while it later had to accelerate to the right at the exact same position, but after it had acquired the required additional momentum, which allowed it to climb the steep slope at $x = 0.0$. An important aspect of the context sensitivity - that eludes state action policies - is the fact that they are not belief-sensitive. In other words, unless one uses a belief state space, the optimal action does not depend upon what one knows about the consequences of action. Having said this, the fact that we used optimised the free action, i.e. the path integral of the free energy, over complete trials means that we have optimised our agent to be epistemic; i.e., ambiguity resolving. This brings the beliefs states of the agent into play during the optimisation process.

5.3 Comparison to a recent "Action Oriented" Implementation of Active Inference

Another implementation of Active Inference was recently introduced by Baltieri and Buckley (2017). Similar to our approach, the authors do not give their agent access to the generative process, neither in terms of the functional form of the agent's generative model, nor by providing it with the partial derivatives of its full sensory inputs with respect to its actions. However, their work differs in several crucial aspects from ours: Their agent is formulated in continuous time, using partial differential equations. Furthermore, the agent's generative model is very simple, bearing little resemblance to the actual equations of motions of its environment. They call this class of model "action oriented"

to differentiate it from the so called "perception oriented" implementations, in which the agent's generative model is very similar to the actual equations of motions (e.g. (Friston et al, 2010)).The resulting differential equations are simple enough to explicitly discuss their dynamics. In contrast, we use a discrete time model with high dimensional latent variables and highly nonlinear transfer functions, precluding us from a classical analytical treatment of the resulting dynamics. However, the very flexible form of the generative model in our approach allows our agent to learn an precise approximation of the true dynamics, as shown in figure 8. Furthermore, Baltieri and Buckley (2017) circumvent the lack of explicit partial derivatives of the sensory inputs with respect to the agent's actions, by subdividing its sensory inputs into exteroceptive and proprioceptive channels, where they use action only to suppress prediction errors in the proprioceptive channels.

This leads to a subdivision of sensory channels into interoceptive and exteroceptive channels and requires an explicit proprioceptive channel o_a, which in our approach is actually not necessary to successfully build a generative model of the environment and reach the goals defined by the agent's prior expectations (c.f. supplementary figures 1-3). Furthermore, the classic active inference framework (Friston et al, 2006, 2010) underlines the crucial role of the partial derivatives of exteroceptive sensations (e.g. visual input) with respect to action (e.g. oculomotor activity) and hints at possible approximations implemented in the brain. E.g. the retinotopic maps in the early visual system, such as in the superficial and deep layers of the superior colliculus, allow a quick and simple inversion of the changes in sensory input with respect to small eye movements (Friston et al, 2010).

In terms of our model this would be equivalent to ignoring the partial derivatives of the exteroceptive sensory channels o_x and o_h with respect to the subset of parameters θ parametrising the action function $p_\theta(a|s)$ and updating these parameters only using the gradients of the expected sensory surprise $< -\ln p_\theta(o_{a,t}|\mathbf{s}_t) >_{q(\mathbf{s}_t|\mathbf{s}_{t-1},o_{x,t},o_{h,t},o_{a,t})}$ in the proprioceptive channel o_a under the population density. At least in the concrete case of our agent, updating the parameters of the action function only using the gradients of the expected sensory surprise in the proprioceptive channel severely impaired the agent's behavioural performance and learning (c.f. supplementary figures 4-6). However, this might be due to the fact that our action function is a flexible, but static approximation $p_\theta(a_{t+1}|s_t)$ of the complex, and time-varying dependency of the action on the current state of the world via the internal state of the agent. By this approximation we confound the optimisation of action on the fast timescale, which is done explicitly by Baltieri and Buckley (2017) and which might not depend on exteroceptive sensations, with the learning of the parameters of the generative model. In other words, the fast dynamics corresponding to the action variable a_t can work to suppress proprioceptive prediction error and thus do not have to depend directly on exteroceptive sensations, given the agent has acquired a sufficiently certain generative model on how changes in proprioception and exteroception are related. However, learning this joint generative model of proprioceptive and exteroceptive sensations, i.e. the struc-

ture of the proprioceptive motor loops and their relation to the inferred hidden states of the world, is clearly accountable to all sensory channels.

5.4 Acquisition of a Detailed Generative Model of the World, Emergent Curiosity, and Generalisation

After its homeostatic needs are fulfilled, i.e. after the agent has quickly discovered a way to fulfil its prior expectations in terms of its final position at $x = 1.0$, it takes the agent some time, about 30,000 iterations, until it has learned a very detailed model of its environment. Sampling from this generative model is quick and easy and the sampled time courses closely match the true time course of the agent's interaction with its environment. Optimising this generative model of the world gives the agent a sense of epistemic value (Friston et al, 2015). I.e. it will not only seek out the states, which it expects, but it will also try to improve its general understanding of the world. This is similar to recent work on artificial curiosity (Pathak et al, 2017). However, in contrast to this work, our agent is not only interested in those environmental dynamics which it can directly influence by its own action, but also tries to learn the statistics of its environment which are beyond its control. Recently it was shown that the acquisition of a generative model of several - initially task-irrelevant - environmental variables from raw 2D inputs generated by an immersive 3D environment facilitates the learning of goal-directed policies later on and helps an agent to generalise across tasks and environments (Dosovitskiy and Koltun, 2017). Furthermore, the previous acquisition of a full generative model of a physics based 2D racing game from raw pixel input allowed an agent to solve a complex, stochastic (in terms of randomly generated race tracks) OpenAI Gym environment for the first time (Ha and Schmidhuber, 2018). Even beyond reinforcement learning tasks, generative pre-training, i.e. the previous acquisition of a generative model, was shown to improve the performance on several language understanding tasks in natural language processing (Radford et al, 2018) and other deep-learning tasks (Erhan et al, 2010)

5.5 Homeostatic Priors

The fact that in our example the action based component, i.e. the solution of the prespecified problem (in terms of the agent's prior expectations) precedes the acquisition of the detailed generative model of its environment might be due to the strong weighting of the prior expectations in terms of the fixed and small standard deviations of the homeostatic state s_1 and the corresponding prior ($\sigma = 0.01$). This leads to a large contribution of the corresponding term to the free energy functional and consequently to a strong impact on the gradient estimates. Thus, by using more lenient prior expectations one might give more weight to the epistemic exploration of the environment with the aim of building a detailed understanding of the world, in terms of a good generative model.

Thus, the strong, high precision (i.e. low standard deviation) priors in our case can really be seen as homeostatic expectations, that are crucially for an agent's survival. Thinking about the human body, these might be very precise expectations on the pH and on the oxygen concentration in human blood, which trigger immediate, strong physiological and behavioural responses in terms of respiratory distress and dyspnoea, as soon as they get violated. While this kind of hard-coded inference dynamics and expectations might be fixed for individual agents of a class (i.e. species) within their lifetimes, these mappings can be optimised on a longer timescale over populations of agents by evolution. In fact, evolution might be seen as a very similar learning process, only on different spatial and temporal scales (Watson and Szathmáry, 2016; Baez and Pollard, 2015; Harper, 2009; Campbell, 2016), where free energy is optimised on the level of species and ecosystems rather than individual agents (Friston, 2012; Ramstead et al, 2017).

5.6 Constrained Sampling for Understanding the Learned Models

We demonstrate constrained sampling from the agent's generative model of the world. Comparing the constrained samples (figure 9) to the actual interaction (figure 7) with the environment and the unconstrained samples (figure 8), they look reasonable, but deviate from the true dynamics that one would expect given the conditioned time course: We constrained the sampling on the mean time course of o_a from the agent's true interaction with the world, shifted back in time by 10 time steps. This lead to samples of the other variables, which were also shifted back in time. However, during the first 10 time steps the agent believes that it would stay at its initial position, despite its strong push to the left. This might be due to the fact that the relative weighting of the agent's prior expectations on its position is quite strong. Accordingly, it tightly sticks to the optimal trajectory, as soon as it has learned it. Thus, the dynamics it infers deviate from the true dynamics, as it only ever experiences a very small subset of its phase space. However, if you put the agent in a noisy environment with larger stochastic fluctuations, where it is forced to explore and encounter a wider variety of dynamic states, it should learn a more complete and realistic model of the environmental dynamics. This hope also rests on the fact that by taking away its effector organs, our agent can be reduced to a generative recurrent latent variable model. This class of models has been able to model and generate very complex time series data, such as spoken language on the basis of its raw audio waveform (Chung et al, 2015). Thinking about autonomous agents in a real environment (i.e. robots or autonomous vehicles), constrained sampling from an agent's generative model might be a good way to understand its beliefs about how the world might react to given actions or events, opening up the "black box" associated with classical deep reinforcement learning algorithms.

5.7 Benefits of the Evolution Strategies Optimiser

Evolution strategies allow for exploration of the environment without requiring
the agent's action functions to be probabilistic: Using standard reinforcement
learning algorithms, such as deep Q-learning (Mnih et al, 2015), the only way
that the agent can discover new states, which do not correspond to its current,
local optimum, is by requiring it to stochastically take actions from time to
time. This might be by occasionally sampling completely random actions or by
lower bounding the standard deviation of its actions. Here, no such artificially
forced exploration is required, as the evolution strategies algorithm explores
new solutions on the parameter space (Salimans et al, 2017). So our agent can
(and actually does) settle to an (almost) fully deterministic policy, if such a
policy exists for the respective environment and the agent's priors. Moreover,
as the gradient estimates only depend on the full free action, after the com-
pletion of each individual simulation, they are also robust against long time
horizons and sparse rewards (Salimans et al, 2017). Interestingly, the whole
premise of recent implementations of active inference (Friston et al, 2015) is
that exploratory behaviour is deterministic, purposeful and uncertainty re-
ducing. In our setup, the stochastic exploration of parameter space is about
exploring state space - this is already subsumed under the free energy func-
tional using active inference. Rather, it ensures a thorough and comprehensive
exploration of parameter space in forming a (possibly multimodal) posterior
density over the right sort of parameters.

5.8 Variational Bayesian Perspective on Evolution Strategies

Although we are using evolution strategies mainly as a black box optimiser
for our non differentiable objective function, namely the sampling based ap-
proximation of the free action bound on the agent's surprise, a subtle but
important reinterpretation of the population density highlights the ubiquitous
role of variational Bayes and the implicit minimisation of variational free en-
ergy (c.f. (Friston, 2012; Baez and Pollard, 2015; Harper, 2009)). In brief, if
we rename the population density $p_\psi(\theta)$ with $q_\psi(\theta)$ and write the likelihood
as $p(o|s,\theta)$ instead of $p_\theta(o|s)$ and the variational density as $q(s|o,\theta)$ instead of
$q_\theta(s|o)$, then the expected free energy becomes:

$$\eta(\psi, o) = \left\langle \left\langle -\ln p(o|s,\theta) \right\rangle_{q(s|o,\theta)} + D_{\mathrm{KL}}(q(s|o,\theta)||p(s|\theta)) \right\rangle_{q_\psi(\theta)}$$

If we now add a complexity penalty $D_{\mathrm{KL}}(q_\psi(\theta)||p(\theta))$ on the population den-
sity, based on a prior $p(\theta)$ on the parameters θ, the sum of the expected
variational free energy under the population density plus this KL divergence

becomes

$$
\eta(\psi, o) + D_{\mathrm{KL}}(q_\psi(\theta)||p(\theta))
$$
$$
= \left\langle \left\langle -\ln p(o|s,\theta) \right\rangle_{q(s|o,\theta)} + D_{\mathrm{KL}}(q(s|o,\theta)||p(s|\theta)) \right\rangle_{q_\psi(\theta)} + \left\langle \ln q_\psi(\theta) - \ln p(\theta) \right\rangle_{q_\psi(\theta)}
$$
$$
= \left\langle \left\langle -\ln p(o|s,\theta) \right\rangle_{q(s|o,\theta)} + \left\langle \ln q(s|o,\theta) - \ln p(s|\theta) \right\rangle_{q(s|o,\theta)} + \ln q_\psi(\theta) - \ln p(\theta) \right\rangle_{q_\psi(\theta)}
$$
$$
= \left\langle -\ln p(o|s,\theta) + \ln q(s|o,\theta) - \ln p(s|\theta) + \ln q_\psi(\theta) - \ln p(\theta) \right\rangle_{q(s|o,\theta)q_\psi(\theta)}
$$
$$
= \left\langle -\ln p(o|s,\theta) + \ln q(s|o,\theta)q_\psi(\theta) - \ln p(s|\theta)p(\theta) \right\rangle_{q(s|o,\theta)q_\psi(\theta)}
$$

This is just the variational free energy under a proposal density $q_\psi(s,\theta|o) = q(s|o,\theta)q_\psi(\theta)$ and prior $p(s,\theta) = p(s|\theta)p(\theta)$ which cover both states and parameters, i.e. full variational Bayes (c.f. (Kingma and Welling, 2014), appendix F). On this view, the population dynamics afford a general and robust, ensemble based scheme for optimising model parameters with respect to the variational free energy of beliefs over both model parameters and latent states, neglecting prior information (or using a noninformative prior) on the parameters. This is similar to the more general formulation of Friston (2008), who also absorbed deep inference and deep learning problems under the same imperative (i.e., minimisation of variational free energy) to solve a dual estimation problem. One subtlety here is that the free energy of the beliefs about parameters (i.e., the population density) minimises the path or time integral of free energy – during which the parameters are assumed not to change.

5.9 Limitations

5.9.1 Sample inefficiency

While the evolution strategies algorithm parallelises extremely well and can reach state of the art results in simulated reinforcement learning tasks (Salimans et al, 2017), its enormous use of evaluations of the environment precludes it, at least in its simple form, from being applied to the training of physical agents, such as robots or autonomous vehicles. However, it was recently shown that also complex tasks (VizDoom (Kempka et al, 2016), CarRacing-v0 (Brockman et al, 2016)) can be learned from pixels using covariance-matrix adaptation evolution strategies(c.f. (Hansen, 2016)) on a single desktop-class computer (Ha and Schmidhuber, 2018). Furthermore, recent work proposes using approximate gradient information to guide the evolution of the population density, thereby ameliorating the course of dimensionality and dramatically reducing the required amount of samples from the population density per training step (Maheswaranathan et al, 2018).

5.9.2 Amortisation versus direct optimisation of sufficient statistics of the variational density and action function

By using deep neural networks, we amortise the optimisation of the sufficient statistics of the variational density, approximating the posterior density of hidden states, given observations, and the optimisation of the sufficient statistics of the action function. This approach utilises the fact that deep neural networks were shown to be able to approximate arbitrary functions, given enough neurons in the hidden layers (Hornik et al, 1989). However, given a new environment, which might be characterised by similar, but in detail different statistics, our agent's *static* variational density and its action function will have to rely on interpolation or extrapolation beyond the range of what the agent has experienced before. This might preclude finding new, more suitable minima of the variational free action in new environments, i.e. amortisation impairs the agents ability to have new insights required to understand new environments (Friston et al, 2017a). This said, in practice amortised approaches are able to learn impressively crisp and creative generative models. While the variational autoencoder often suffers from blurred samples, lacking high (spatial) frequency features of the original data, another class of amortised generative models, namely generative adversarial networks (Goodfellow et al, 2014) are able to produce incredibly sharp and convincing samples (Karras et al, 2018) and make impressive use of the latent representation in tasks such superresolution (Ledig et al, 2016) or style transfer or unsupervised image-to-image translation (Zhu et al, 2017; Liu et al, 2017). Thus, we think while amortisation is a crucial limitation, the benefits often supervene, if the statistics of the environment and the goals do not change too much between training and test time. Rather, the concrete structure of the generative model and the approximate posterior might limit the agent's ability to encode all required information in its latent representation, thereby leading to smooth and interpolated samples. However, there are many current streams of research trying to ameliorate this limitation for variational autoencoders, namely in the form of normalizing flows (Rezende and Mohamed, 2015; Kingma et al, 2016; Tomczak and Welling, 2016), auxiliary variables (Maaløe et al, 2016), or implicit representations (Mescheder et al, 2017; Huszár, 2017).

5.10 Possible Extensions

5.10.1 More complex a priori expectations

Right now, our very direct way of hard-wiring the expectations of our agent, in terms of explicit prior distributions on the first dimension of the agent's latent space, seems very ad hoc and restricted. However, we could show that due to the flexibility of our agent's internal dynamics and the robust optimisation strategies, our agent quickly learns to reach these very narrowly defined goals and is able to build a realistic model of the environmental dynamics. In future

work, we plan to look into more complex a priori beliefs. This could be done by sampling from the agent's generative model as part of the optimisation process. E.g. one could sample some processes from the generative model, calculate the quantity on which a constraint in terms of prior expectations of the agent should be placed, and calculate the difference between the sampled and the target distribution, e.g. using the KL divergence. Then one could add this difference as penalty to the free energy. To enforce this constraint, one could use for example the Basic Differential Multiplier Method (Platt and Barr, 1988), which is similar to the use of Lagrange multipliers, but which can be used with gradient descent optimisation schemes. This prospective sampling to optimise the goals of the agent might be actually what parts of prefrontal cortex do when thinking about the future and how to reach certain goals.

As this procedure introduces another constraint, which increases the complexity of the objective function and might slow down or prevent the proper convergence of the optimisation process, another, more natural way to include more complex prior expectations is to include one or more additional levels of latent variables and use simple priors on the more abstract levels to generate more complex, so called empirical priors on the lower levels (c.f. (Friston et al, 2017b) for a review).

5.10.2 Episodic memory

Furthermore, in more complex environments, where rewards and success are sparse and where specific facts have to be remembered, the addition of an episodic memory mechanism was shown to significantly improve performance of simulated agents (Graves et al, 2016).

5.11 A first step from Artificial Life to Artificial General Intelligence

We present a flexible and scalable implementation of a general active inference agent, that is able to learn a generative model of its environment while simultaneously achieving prespecified goals, in terms of prior expectations on its perceived states of the world. We hope that this implementation will prove useful to solve a wide variety of more complex and realistic problems. By this, one could show how general, intelligent behaviour follows naturally from the free energy principle. This principle, in turn, is derived from necessary properties of dynamic systems in exchange with changing environments, which allow them to sustain their identity by keeping their inner parameters within viable bounds (Friston, 2012, 2013). Thus, we hope that our work contributes to more concrete, experimental examples that intelligent behaviour can follow and is hard to separate from the basic imperative to survive in and adapt to changing environments.

6 Acknowledgements

References

Adams RA, Stephan KE, Brown H, Frith CD, Friston KJ (2013) The computational anatomy of psychosis. Frontiers in Psychiatry 4(47)

Alais D, Burr D (2004) The ventriloquist effect results from near-optimal bimodal integration. Current Biology 14(3):257–262

Baez JC, Pollard BS (2015) Relative entropy in biological systems. https://arxivorg/abs/151202742

Baltieri M, Buckley CL (2017) An active inference implementation of phototaxis. https://arxivorg/abs/170701806

Berkes P, Orbán G, Lengyel M, Fiser J (2011) Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. Science 331:83–87

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) Openai gym. https://arxivorg/abs/160601540

Brown H, Friston KJ (2012) Free-energy and illusions: the cornsweet effect. Frontiers in Psychology 3(43)

Campbell JO (2016) Universal darwinism as a process of bayesian inference. https://arxivorg/abs/160607937

Caticha A (2004) Relative entropy and inductive inference. AIP Conference Proceedings 707

Chung J, Kastner K, Dinh L, Goel K, Courville A, Bengio Y (2015) A recurrent latent variable model for sequential data. https://arxivorg/abs/150602216

Conant R, Ashby W (1970) Every good regulator of a system must be a model of that system. International Journal of Systems Science 1(2):89–97

Crapse TB, Sommer MA (2008) Corollary discharge across the animal kingdom. Nature Reviews Neuroscience 9:587–600

Dosovitskiy A, Koltun V (2017) Learning to act by predicting the future. ICLR

Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P (2010) Why does unsupervised pre-training help deep learning? JMLR

Ernst M, Banks M (2002) Humans integrate visual and haptic information in a statistically optimal fashion. Nature 415:429–433

Friston KJ (2005) A theory of cortical responses. Phil Trans R Soc B 360:815–836

Friston KJ (2008) Hierarchical models in the brain. PLoS Computational Biology 4(11)

Friston KJ (2010) The free-energy principle: a unified brain theory? Nature Reviews Neuroscience 11(2):127–38

Friston KJ (2012) A free energy principle for biological systems. Entropy 14:2100–2121

Friston KJ (2013) Life as we know it. Journal of The Royal Society Interface 10

Friston KJ, Kiebel SJ (2009) Predictive coding under the free-energy principle. Phil Trans R Soc B 364:1211–1221

Friston KJ, Kilner J, Harrison L (2006) A free energy principle for the brain. Journal of Physiology Paris 100:70–87

Friston KJ, Daunizeau J, Kilner J, Kiebel SJ (2010) Action and behavior: a free-energy formulation. Biological Cybernetics 192(3):227–260

Friston KJ, Mattout J, Kilner J (2011) Action understanding and active inference. Biological Cybernetics 104:137–160

Friston KJ, Rigoli F, Ognibene D, Mathys C, Fitzgerald T, Pezzulo G (2015) Active inference and epistemic value. Cognitive Neuroscience 6(4):187–214

Friston KJ, Frith CD, Pezzulo G, Hobson JA, Ondobaka S (2017a) Active inference, curiosity and insight. Neural Computation 29:1–51

Friston KJ, Rosch R, Parr T, Price C, Bowman H (2017b) Deep temporal models and active inference. Neuroscience & Biobehavioral Reviews 77:388–402

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. https://arxivorg/abs/14062661

Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, `http://www.deeplearningbook.org`

Graves A, Wayne G, Reynolds M, Harley T, Danihelka I, Grabska-Barwinska A, Gómez Caolmenarejo S, Grefenstette E, Ramalho T, Agapiou J, Puig-domenèch Badia A, Hermann KM, Zwols Y, Ostrovski G, Cain A, King H, Summerfield C, Blunsum P, Kavukcuoglu K, Hassabis D (2016) Hybrid computing using a neural network with dynamic external memory. Nature 538:471–476

Ha D, Schmidhuber J (2018) World models. https://arxivorg/abs/180310122

Haefner R, Berkes P, Fiser J (2016) Perceptual decision-making as probabilistic inference by neural sampling. Neuron 90(3):649–660

Hansen N (2016) The cma evolution strategy: A tutorial. https://arxivorg/abs/160400772

Harper M (2009) The replicator equation as an inference dynamic. https://arxivorg/abs/09111763

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313

Hornik K, Stinchcombe M, White H (1989) Multilayer feedforward networks are universal approximators. Neural Networks 2:359–366

Huszár F (2017) Variational inference using implicit distributions. https://arxivorg/abs/170208235

Karpathy A, Johnson J, Fei-Fei L (2015) Visualizing and understanding recurrent networks. https://arxivorg/abs/150602078

Karras T, Aila T, Laine S, Lehtinen J (2018) Progressive growing of gans for improved quality, stability, and variation. ICLR

Kempka M, Wydmuch M, Runc G, Toczek J, Jaśkowski W (2016) Vizdoom: A doom-based ai research platform for visual reinforcement learning. https://arxivorg/abs/160502097

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. https://arxivorg/abs/14126980

Kingma DP, Welling M (2014) Auto-encoding variational bayes. ICLR

Kingma DP, Salimans T, Jozefowicz R, Chen X, Sutskever I, Welling M (2016) Improving variational inference with inverse autoregressive flow. https://arxivorg/abs/160604934

Knill D, Pouget A (2004) The bayesian brain: the role of uncertainty in neural coding and computation. Trends in Neurosciences 27(12):712–9

Le QV, Jaitly N, Hinton GE (2015) A simple way to initialize recurrent networks of rectified linear units. https://arxivorg/abs/150400941

LeCun Y, Bengio Y, Hinton GE (2015) Deep learning. Nature 521:436–44

Ledig C, Theis L, Huszár F, Caballero J, Cunningham A, Acosta A, Aitken A, Tejani A, Totz J, Wang Z, Shi W (2016) Photo-realistic single image super-resolution using a generative adversarial network. https://arxivorg/abs/160904802

Liu MY, Breuel T, Kautz J (2017) Unsupervised image-to-image translation networks. NIPS

Maaløe L, Sønderby CK, Sønderby SK, Winther O (2016) Auxiliary deep generative models. https://arxivorg/abs/160205473

Maheswaranathan N, Metz L, Tucker G, Sohl-Dickenstein J (2018) Guided evolutionary strategies: escaping the curse of dimensionality in random search. https://arxivorg/abs/180610230

Mescheder L, Nowozin S, Geiger A (2017) Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. https://arxivorg/abs/170104722

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518:529–33

Moore A (1991) Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. In: Proceedings of the Eight International Conference on Machine Learning, Morgan Kaufmann

Moreno-Bote R, Knill D, Pouget A (2011) Bayesian sampling in visual perception. Proc Natl Acad Sci U S A 108(30):12,491–6

Pathak D, Pulkit A, Efros AA, Darrell T (2017) Curiosity-driven exploration by self-supervised prediction. https://arxivorg/abs/170505363

Platt JC, Barr AH (1988) Constrained differential optimization. In: Neural Information Processing Systems, American Institute of Physics, New York,

pp 612–621

Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training. https://blogopenaicom/language-unsupervised/

Ramstead MJD, Badcock PB, Friston KJ (2017) Answering schrödinger's question: A free-energy formulation. Physics of Life Reviews

Rezende DJ, Mohamed S (2015) Variational inference with normalizing flows. JMLR 37

Rezende DJ, Mohamed S, Wierstra D (2014) Stochastic backpropagation and approximate inference in deep generative models. ICML

Rezende DJ, Ali Eslami SM, Mohamed S, Battaglia P, Jaderberg M, Heess N (2016) Unsupervised learning of 3d structure from images. https://arxivorg/abs/160700662

Salimans T, Ho J, Chen X, Sutskever I (2017) Evolution strategies as a scalable alternative to reinforcement learning. https://arxivorg/abs/170303864

Schwartenbeck P, Fitzgerald T, Mathys C, Dolan R, Kronbichler M, Friston KJ (2015) Evidence for surprise minimization over value maximization in choice behavior. Scientific Reports 5(16575)

Siegelmann HT (1995) Computation beyond the turing limit. Science 268:545–548

Theano Development Team (2016) Theano: A Python framework for fast computation of mathematical expressions. https://arxivorg/abs/160502688

Todorov E, Erez T, Tassa Y (2012) Mujoco: A physics engine for model-based control. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)

Tomczak JM, Welling M (2016) Improving variational auto-encoders using householder flow. https://arxivorg/abs/161109630

Tran D, Ranganath R, Blei D (2017) Hierarchical implicit models and likelihood-free variational inference. https://arxivorg/pdf/170208896pdf

Watson RA, Szathmáry E (2016) How can evolution learn? Trends in Ecology and Evolution 31(2):147–157

Wong KF, Wang XJ (2006) A recurrent network mechanism of time integration in perceptual decisions. The Journal of Neuroscience 26(4):1314–28

Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. https://arxivorg/abs/170310593