Sleep: Model Reduction in Deep Active Inference

Samuel T. Wauthier, Ozan Çatal, Cedric De Boom,
Tim Verbelen, and Bart Dhoedt

IDLab, Department of Information Technology at Ghent University – imec
Technologiepark-Zwijnaarde 126, B-9052 Ghent, Belgium
{firstname.lastname}@ugent.be

Abstract. Sleep is one of the most important states of the human mind and body. Sleep has various functions, such as restoration, both physically and mentally, and memory processing. The theory of active inference frames sleep as the minimization of complexity and free energy in the absence of sensory information. In this paper, we propose a method for model reduction of neural networks that implement the active inference framework. The proposed method suggests initializing the network with a high latent space dimensionality and pruning dimensions subsequently. We show that reduction of latent space dimensionality decreases complexity without increasing free energy.

Keywords: Active inference · Model reduction · Sleep

1 Introduction

Sleep is a phenomenon that occurs in most animals [10]. It is a topic of intensive research as it has been shown to be important for both the mind [18,12] and the body [14]. In particular, sleep and learning have been connected in many hypotheses [17,3], as well as mental health [4] and memory [20].

Active inference is a theory of behaviour and learning that originated in neuroscience [8]. The basic assumption is that intelligent agents attempt to minimize their variational free energy. Variational free energy — named for its counterpart in statistical physics i.e. Helmholtz free energy — is also known as the evidence lower bound (ELBO) in variational Bayesian methods.

Since its conception, active inference has been explored in multiple subfields of neuroscience and biology [6,5,11] and eventually found its way into the field of computer science [19,15,2]. In particular, Ueltzhöffer [19] and Çatal *et al.* [2] have made developments in *deep active inference*, i.e. the use of deep neural networks to implement active inference.

Recent work [13,9,7] has pointed out the relation between the function of removing redundant connections during sleep and Bayesian model reduction (BMR) in active inference, i.e. complexity minimization through elimination of redundant parameters. In this work, we propose a method for reducing complexity in the deep active inference framework. We evaluate the method through simulation experiments.

2 Deep active inference

Currently, using deep neural networks in active inference to learn state spaces, in addition to policy and posterior, is becoming increasingly popular, which contrasts with active inference on discrete state spaces as described in [9]. In this approach, the dimensionality of the state space is a hyperparameter, i.e. it must be specified before training and cannot change along the way. Here, we briefly introduce the method provided by Çatal *et al.* [2].

Assuming the policy π may be broken up into a sequence of actions $\mathbf{a}_t$ and the current state depends on the previous action instead of the policy, a generative model with observations $\mathbf{o}_t$ and states $\mathbf{s}_t$ is defined as

$$P(\tilde{\mathbf{o}}, \tilde{\mathbf{s}}, \tilde{\mathbf{a}}) = P(\mathbf{s}_0)P(\tilde{\mathbf{a}}) \prod_{t=1}^{T} P(\mathbf{o}_t|\mathbf{s}_t)P(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}), \tag{1}$$

where $\tilde{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$.

Deep neural networks are used to parameterize the prior, likelihood and approximate posterior distributions: $p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})$, $p_\phi(\mathbf{o}_t|\mathbf{s}_t)$ and $q_\xi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)$, respectively. With this, minimization of free energy consists of minimizing the loss function

$$\begin{aligned} &L(\theta, \phi, \xi; \mathbf{o}_t, \mathbf{s}_{t-1}, \mathbf{a}_{t-1}) = \\ &\quad D_{\mathrm{KL}}(q_\xi(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1}, \mathbf{o}_t)||p_\theta(\mathbf{s}_t|\mathbf{s}_{t-1}, \mathbf{a}_{t-1})) - \log p_\phi(\mathbf{o}_t|\mathbf{s}_t). \end{aligned} \tag{2}$$

Prior, likelihood and posterior distributions are chosen to be multivariate normal distributions. As opposed to the standard VAE, optimization is done over

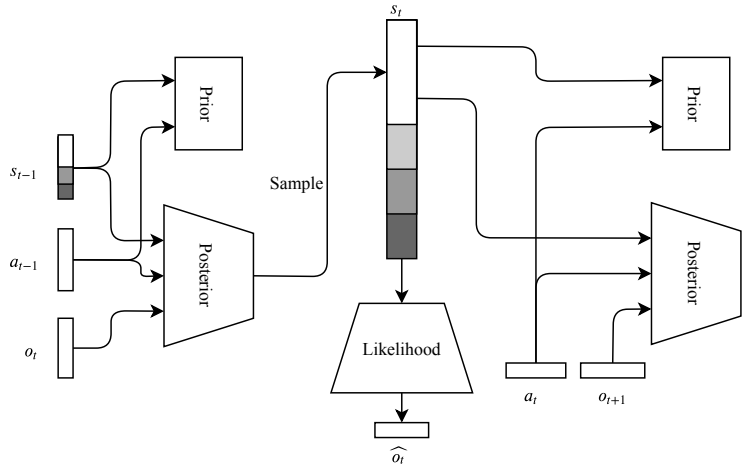

Fig. 1. Information flow of neural networks. The posterior network takes in previous state and action, and current observation. The prior network takes in previous state and action. The likelihood network takes in current state. The state s_t illustrates that dimensions may be pruned, if they are unused.

sequences in time. Additionally, empirical priors are learned, instead of using fixed priors. A chart on the information flow can be found in Figure 1.

3 Latent space dimensionality reduction and sleep

The size of the latent space vector $\mathbf{s}$ is an important hyperparameter. On the one hand, this must be large enough to explain observations in the generative model. On the other hand, it must be kept minimal to reduce complexity as to minimize the required resources, such as memory and power (both computational and electrical). In general, one does not know the optimal size of $\mathbf{s}$. A typical way of finding a well-performing value is a hyperparameter sweep. Parameter sweeps, however, are resource intensive and require many unnecessary training runs. Therefore, we propose a method for dimensionality reduction in the deep active inference framework.

The basic idea is to prune dimensions in the latent space vector $\mathbf{s}$. A popular method for inspecting informative dimensions of a vector space is singular value decomposition (SVD). This technique is used to factorize an $m \times n$ matrix $\boldsymbol{A}$ into three matrices $\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^*$, where a common geometrical interpretation is that the decomposition gives 2 rotation matrices $\boldsymbol{U}$ and $\boldsymbol{V}^*$, and a scaling matrix $\boldsymbol{S}$.

Algorithm 1 lines out the method in the form of pseudo-code. Let n be the dimensionality of the latent space. We sample a latent space vector from m different sequences to construct the column vectors of a matrix $\boldsymbol{A}$. The column space of $\boldsymbol{A}$, denoted $\mathcal{C}(\boldsymbol{A})$, forms a subspace of the latent space. Applying SVD to $\boldsymbol{A}$ gives the scaling matrix $\boldsymbol{S}$. The values on the diagonal of $\boldsymbol{S}$ are the singular values of $\boldsymbol{A}$, and suggest a size for the dimensions of $\mathcal{C}(\boldsymbol{A})$ after rotation with $\boldsymbol{U}$. Dimensions with small singular values are assumed to be unused. To this

Algorithm 1: Sleep

input : A trained model model with dimensionality n
The number of repetitions N and number of sequences m
A threshold α
output : The new latent space dimensionality ν
while $i < N$ **do**
  $\boldsymbol{A} \leftarrow []$ `// make a matrix`
  while $j < m$ **do**
    $a \leftarrow$ `GenerateSequence(model)` `// generate a new sequence`
    $\boldsymbol{v} \leftarrow$ `Sample`(a) `// sample a latent space vector`
    $\boldsymbol{A} \leftarrow [\boldsymbol{A}, \boldsymbol{v}]$ `// insert vector as a new column in matrix`
    $j \leftarrow j + 1$
  $\boldsymbol{S} \leftarrow$ `SVD`$(\boldsymbol{A})$ `// apply SVD to matrix`
  $c_i \leftarrow \#(S_{kk} > \alpha)$ for $0 < k < n$ `// count sv's over threshold`
  $\boldsymbol{c} \leftarrow [\boldsymbol{c}, c_i]$ `// add number to list of outcomes`
  $i \leftarrow i + 1$
$\nu \leftarrow$ `Avg`$(\boldsymbol{c})$ `// average over all outcomes`

end, we define a threshold α for which dimensions corresponding to singular values smaller than α can be pruned. We repeat this procedure N times — by generating m new sequences each time — and average the number of pruned dimensions, in order to obtain a relatively robust outcome.

It is important to stress, here, that SVD does not allow one to find *which* dimensions can be pruned. Instead, it is used to converge to the optimal *number* of dimensions. SVD provides the size of dimensions of the column space of $\boldsymbol{A}$, i.e. $\mathcal{C}(\boldsymbol{A})$, described in a basis of the latent space after a rotation with $\boldsymbol{U}$. The actual basis vectors are a linear combination of the rotated basis vectors. In other words, having a zero dimension in rotated latent space, does not necessarily mean there is one in latent space. However, it does indicate that it is possible to reduce dimensionality by choosing a different rotation, since it shows that there is an orientation of the basis vectors which requires less dimensions to describe the column space. Returning to the model, by retraining with a lower dimensionality n, we essentially force the model to learn the latent space with a different orientation which requires less dimensions.

We have dubbed the method *sleep*, since it replicates synapse pruning, as well as Bayesian model reduction. From an active inference perspective, the proposed method is analogous to BMR in that it considers a generative model with a large number of latent factors and optimizes this number post hoc [16]. In other words, both the goals of the proposed method and BMR are to consider alternative models which may give simpler explanations for the same observations. That said, in both cases, the balance between accuracy and complexity is crucial, i.e. accuracy should not suffer due to simplicity. Indeed, the measure for this trade-off is free energy.

Since latent space in deep active inference is learned using deep neural networks, there is no guarantee that each latent space dimension represents an individual feature. Without knowing what is contained in latent space, it is not possible to target specific parameters to turn off as in BMR. Because of this, algorithm 1 must be succeeded by retraining to obtain a reduced model. In this sense, the earlier analogy is incomplete, since BMR allows one to obtain the reduced model parameters from the full model, i.e. it allows one to find *which* dimensions can be pruned.

In the end, the purpose of the sleep method is to reduce complexity whenever an application (e.g. a robot running a deep active inference implementation) has downtime. The overall sequence of events, then, proceeds as follows. Start with a large value for the latent space dimensionality and train the model. Deploy the model on the application. Each time there is downtime in the application (e.g. the robot is charging), reduce the model by sleeping and retraining. Continue this pattern of sleeping and retraining until the model cannot reduce any further.

4 Experimental setup

Experiments were performed using two environments from the OpenAI Gym [1]. The first experiment employs a modified version of the MountainCar environment,

where noise is added to the observation and only the position can be observed. The goal of this environment is to drive up a steep mountain using an underpowered car that starts in a valley. The car is underpowered in the sense that it cannot produce enough force to go against gravity and drive up the mountain in one go. It must first build up enough momentum by driving up the side(s) of the valley. In this experiment, we know upfront that the model only needs 2 dimensions in latent space: position and velocity. Details about the neural networks used for this experiment can be found in Appendix 2.1.

The second experiment employs the CarRacing environment. The goal of this environment is to stay in the middle of a race track using a race car. The car and track are viewed from a top-down perspective. The car must steer left and right to stay on track. Compared to the MountainCar, the CarRacing environment utilizes more complicated dynamics and produces higher dimensional observations. Examples of the environments can be found in Appendix 1. Details about the neural networks used for this experiment can be found in Appendix 2.2.

5 Results

Fig. 2 shows the evolution of the free energy of MountainCar during training with a fixed number of latent space dimensions (see Appendix 3.1 for a similar figure for CarRacing). It suggests that free energy decreases as more state space dimensions are added. However, it also shows that free energy does not visibly decrease beyond a certain number of dimensions. For the MountainCar, we see that the free energy does not decrease for more than 2 dimensions, while for the CarRacing (Appendix 3.1), we see that the free energy does not decrease for more than 4 dimensions. In essence, there appears to be a critical value of the latent space size. For latent spaces larger than this critical value, the free energy does not reduce. This critical value corresponds to the optimal value for the dimensionality with respect to the accuracy/complexity trade-off.

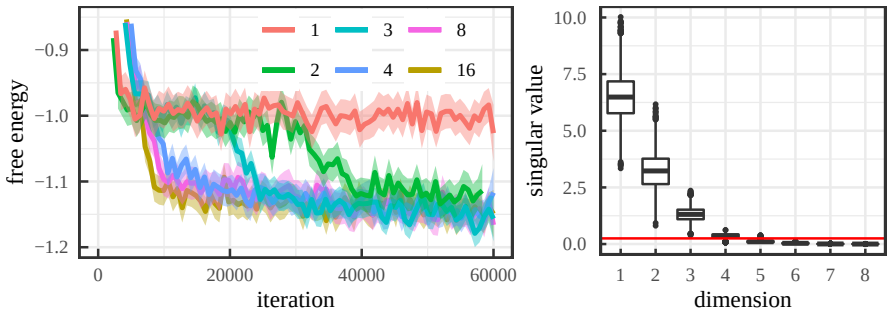

Fig. 2. (Left) Free energy during training of MountainCar for different state space sizes. Curves show smoothed data (LOESS, span 0.02) with 95% standard error bands. (Right) Boxplot of singular values while sleeping at 8 latent space dimensions ($N = 10^4$).

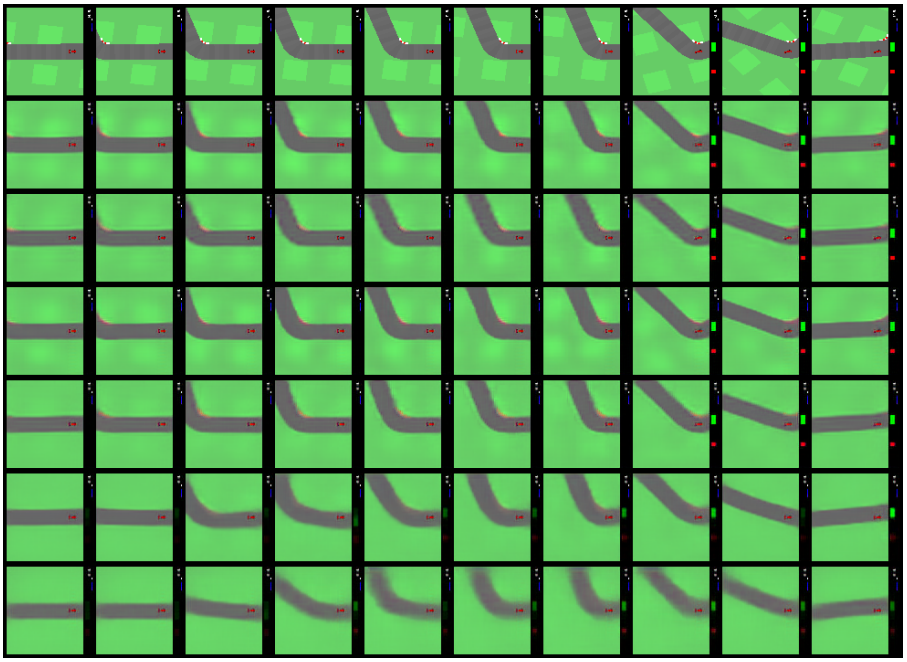

Fig. 3. Reconstructions of CarRacing track over time with different latent space dimensions. From top to bottom: ground truth, 32, 16, 8, 4, 2, 1.

Fig. 3 demonstrates how latent space dimensionality affects reconstruction and how too few dimensions can lead to aspects of the environment not being learned. It shows reconstructions of the CarRacing track for latent space dimensions of 32, 16, 8, 4, 2 and 1 (top to bottom with ground truth in the top sequence). Note how the curvature of the track is not accurately reconstructed through 1 dimension, especially at early time steps. Also, 2 dimensions still seem to lack accuracy (see curvature in second time step). Furthermore, note how the feedback bar is incorrectly encoded by dimensions lower than 4.

Fig. 2 also shows a boxplot for the singular values obtained for the MountainCar with 8 latent space dimensions for $N = 10^4$ iterations (plots for different latent space dimensions can be found in Appendix 3.2). The red line shows a threshold $\alpha = 0.25$. The figure suggests that there is a difference in sizes in the latent space dimensions. Indeed, the first four singular values are on average larger than α, while the remaining values are on average smaller than α. This indicates that certain dimensions are very small, therefore, contain less information, and may be pruned subsequently.

Fig. 4 illustrates the algorithm put into practice with different sleep cycles for the CarRacer with threshold set at $\alpha = 0.25$, where we started with 16 latent space dimensions. In this example, we initiated sleep every 5×10^4 training iterations and checked if dimensionality could be reduced. If so, we pruned and

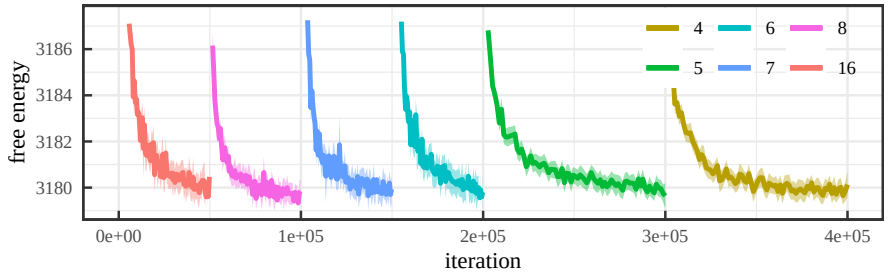

Fig. 4. Free energy over 7 sleep cycles of CarRacer. Setting threshold $\alpha = 0.25$ gives the reduction: $16 \to 8 \to 7 \to 6 \to 5 \to 4$, after which it cannot reduce further. Curves show smoothed data (LOESS, span 0.02) with 95% standard error bands.

restarted training with lower dimensionality, else we continued training for 5×10^4 iterations, until reduction was possible. We stopped the process after 7 sleep cycles. As expected, the sleep sequence manages to reduce the complexity of the model, without impacting the free energy negatively.

When compared to Fig. 8 in Appendix 3.2, the previous result is exactly as expected. Following the steps described there, the state space can effectively be pruned down to 4 dimensions. Observe that if we were to repeat the experiment for the MountainCar, Fig. 7 in Appendix 3.2 shows that setting the threshold at $\alpha = 0.25$ would return a state space dimensionality of 2.

6 Conclusion

Our results show that it is possible to train a deep active inference model by setting a large number of latent space dimensions and subsequently sleeping until minimal complexity is reached. However, the method proposed in this paper is not optimal. A few caveats remain. First of all, the current method requires retraining. After applying SVD, the entire model must be retrained from scratch. Second of all, there exist limitations to SVD. For instance, SVD does not take into account nonlinear transformations. Therefore, relations between different dimensions may remain and the optimal dimensionality may never be reached.

In future work, we will investigate the effects of sleeping at regular intervals during training. For example, we may sleep after every 10^4 time steps to check if we can already reduce the latent space. Another option we will investigate, is to prune both unnecessary dimensions and weights. This way, we may be able to maintain the trained neural network, while reducing complexity. In addition, we want to experiment with different methods for dimensionality reduction, such as nonlinear methods. Another option to be explored is to learn and set unused dimensions to 0 during training.

Acknowledgments

This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme.

References

1. Brockman, G., Cheung, V., Pettersson, L., et al.: Openai gym (2016)
2. Çatal, O., Nauta, J., Verbelen, T., et al.: Bayesian policy selection using active inference pp. 1–9 (apr 2019), http://arxiv.org/abs/1904.08149
3. Fattinger, S., de Beukelaar, T.T., Ruddy, K.L., et al.: Deep sleep maintains learning efficiency of the human brain. Nature Communications **8**(1), 15405 (2017)
4. Freeman, D., Sheaves, B., Goodwin, G.M., et al.: The effects of improving sleep on mental health (oasis): a randomised controlled trial with mediation analysis. The Lancet Psychiatry **4**(10), 749 – 758 (2017)
5. Friston, K., FitzGerald, T., Rigoli, F., et al.: Active inference and learning. Neuroscience and Biobehavioral Reviews **68**, 862–879 (2016)
6. Friston, K., Mattout, J., Kilner, J.: Action understanding and active inference. Biological Cybernetics **104**(1-2), 137–160 (2011)
7. Friston, K., Parr, T., Zeidman, P.: Bayesian model reduction pp. 1–32 (2018), http://arxiv.org/abs/1805.07092
8. Friston, K.J., Daunizeau, J., Kiebel, S.J.: Reinforcement Learning or Active Inference? PLoS ONE **4**(7), e6421 (jul 2009)
9. Friston, K.J., Lin, M., Frith, C.D., et al.: Active Inference, Curiosity and Insight. Neural Computation **29**(10), 2633–2683 (oct 2017)
10. Joiner, W.J.: Unraveling the evolutionary determinants of sleep. Current Biology **26**(20), R1073–R1087 (Oct 2016)
11. Kirchhoff, M., Parr, T., Palacios, E., et al.: The markov blankets of life: Autonomy, active inference and the free energy principle. Journal of the Royal Society Interface **15**(138) (2018)
12. Krause, A.J., Simon, E.B., Mander, B.A., et al.: The sleep-deprived human brain. Nature Reviews Neuroscience **18**(7), 404–418 (2017)
13. Li, W., Ma, L., Yang, G., et al.: Rem sleep selectively prunes and maintains new synapses in development and learning. Nature Neuroscience **20**(3), 427–437 (2017)
14. Newman, A.B., Nieto, F.J., Guidry, U., et al.: Relation of Sleep-disordered Breathing to Cardiovascular Disease Risk Factors : The Sleep Heart Health Study. American Journal of Epidemiology **154**(1), 50–59 (07 2001)
15. Oliver, G., Lanillos, P., Cheng, G.: Active inference body perception and action for humanoid robots (2019), http://arxiv.org/abs/1906.03022
16. Smith, R., Schwartenbeck, P., Parr, T., et al.: An Active Inference Approach to Modeling Structure Learning: Concept Learning as an Example Case. Frontiers in Computational Neuroscience **14**(May), 1–24 (2020). https://doi.org/10.3389/fncom.2020.00041
17. Stickgold, R., Hobson, J.A., Fosse, R., et al.: Sleep, learning, and dreams: Off-line memory reprocessing. Science **294**(5544), 1052–1057 (2001)
18. Tsuno, N., Besset, A., Ritchie, K.: Sleep and depression. The Journal of Clinical Psychiatry **66**(10), 1254–1269 (2005)
19. Ueltzhöffer, K.: Deep active inference. Biological Cybernetics **112**(6), 547–573 (dec 2018)
20. Walker, M.P., Stickgold, R.: Sleep, memory, and plasticity. Annual Review of Psychology **57**(1), 139–166 (2006), pMID: 16318592

Appendix 1 OpenAI Gym examples

Fig. 5 shows snapshots of the MountainCar and CarRacing environments from the OpenAI Gym [1]. Note that observations in the MountainCar environment consist of position and velocity values, while CarRacing provides RGB pixels.

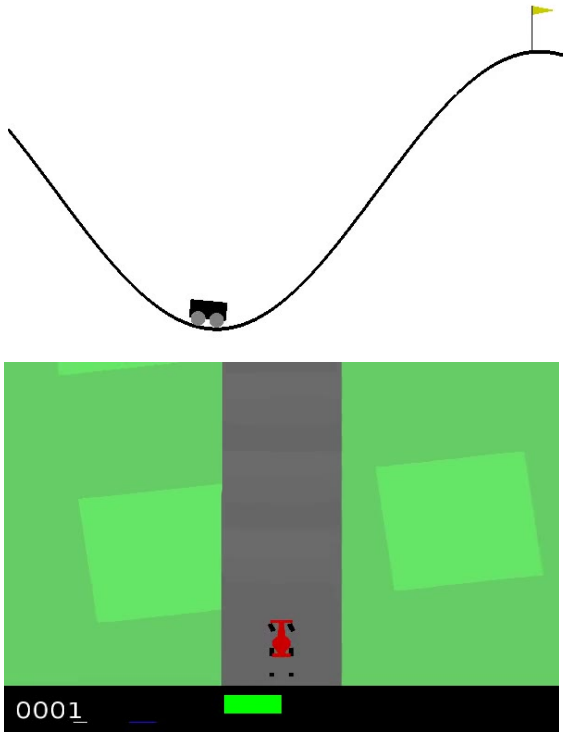

Fig. 5. (Top) Example of MountainCar environment [1]. (Bottom) Example of CarRacing environment [1].

Appendix 2 Neural network definitions

Appendix 2.1 Mountain car

Table 1 shows the neural architecture of the network used in the MountainCar experiments.

Table 1. Specifications of the MountainCar neural network with s latent space dimensions.

	Layer	Neurons/Filters	Activation Function
Posterior	Linear	20	Leaky ReLU
	Linear	$2 \times s$	Leaku ReLU
Likelihood	Linear	20	Leaky ReLU
	Linear	2	Leaky ReLU
Prior	Linear	20	Leaky ReLU
	Linear	$2 \times s$	Leaky ReLU

Appendix 2.2 Car racing

Table 2 shows the neural architecture of the network used in the CarRacing experiments. All filters have 3×3 kernel, as well as stride and padding of 1.

Table 2. Specifications of the CarRacing neural network with s latent space dimensions.

	Layer	Neurons/Filters	Activation Function
Posterior	Convolutional	8	Leaky ReLU
	Convolutional	16	Leaky ReLU
	Convolutional	32	Leaky ReLU
	Convolutional	64	Leaky ReLU
	Convolutional	128	Leaky ReLU
	Convolutional	256	Leaky ReLU
	concat	N/A	N/A
	Linear	$2 \times s$	Leaku ReLU
Likelihood	Linear	$128 \times 2 \times 9$	Leaky ReLU
	Convolutional	128	Leaky ReLU
	Convolutional	64	Leaky ReLU
	Convolutional	32	Leaky ReLU
	Convolutional	16	Leaky ReLU
	Convolutional	8	Leaky ReLU
	Convolutional	3	Leaky ReLU
Prior	LSTM cell	128	Leaky ReLU
	Linear	$2 \times s$	Softplus

Appendix 3 Additional plots

Appendix 3.1 Free energy during training

Fig. 6 shows the evolution of the free energy during training for CarRacing similar to the left plot in Fig. 2. Note how the free energy does not visibly decrease when using more than 4 dimensions.

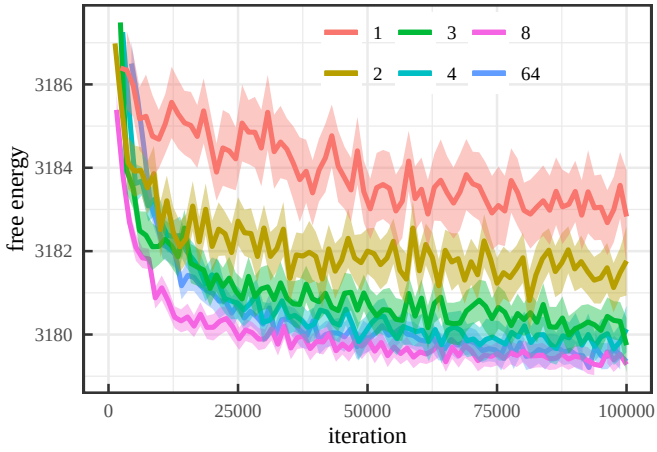

Fig. 6. Free energy during training of CarRacer for different state space sizes. Curves show smoothed data (LOESS, span 0.02) with 95% standard error bands.

Appendix 3.2 Boxplots for different latent space dimensions

Fig. 7 shows boxplots for the singular values obtained for the MountainCar with different latent space dimensions for 10^4 iterations, while Fig. 8 shows the same for the CarRacing. The red line in each plot indicates the threshold $\alpha = 0.25$.

One can do the following mental exercise. Choose a boxplot and count the amount of dimensions that are above threshold on average. This number will be the new dimensionality. Go to the boxplot for that dimensionality and, again, count the amount of dimensions. Repeat this until the dimensionality does not reduce further. Using this process, we can see that the MountainCar will not reduce below 2 and the CarRacing will not reduce below 4.

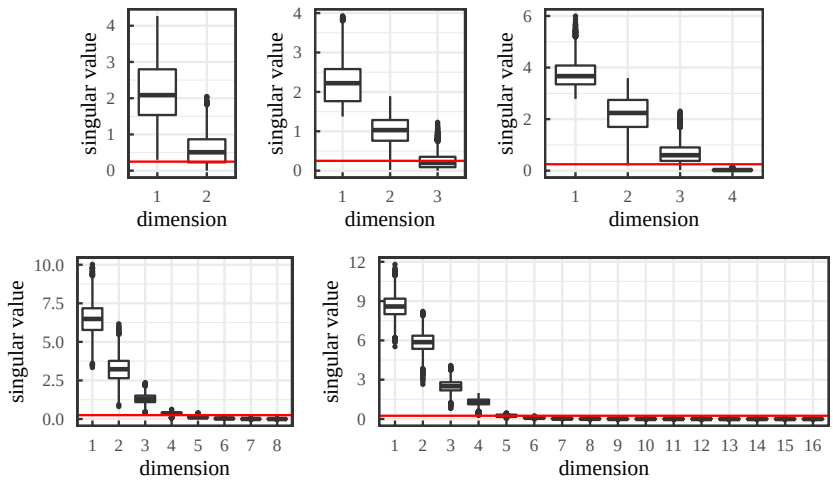

Fig. 7. Boxplots of singular values while sleeping at different latent space dimensions for the MountainCar ($N = 10^4$).

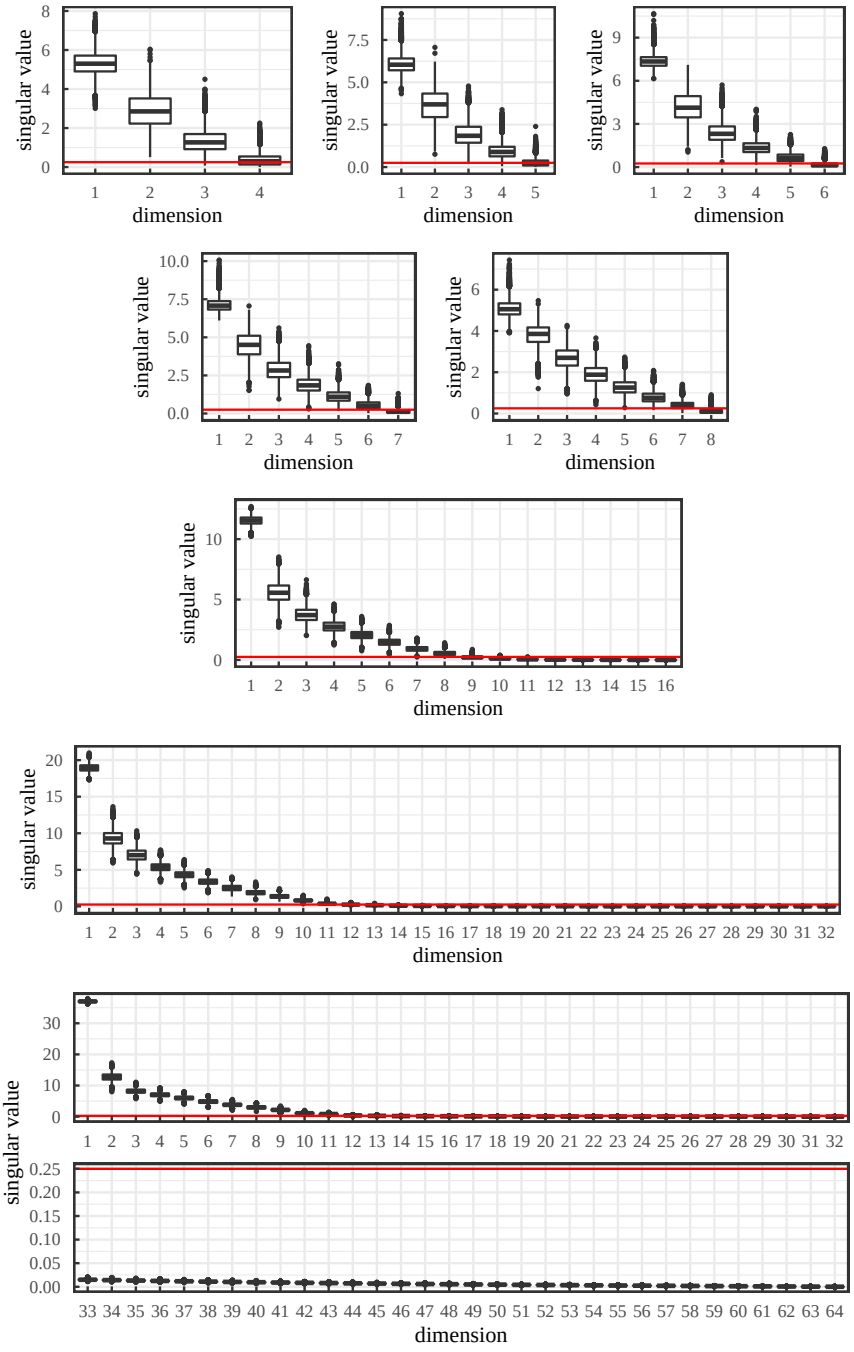

Fig. 8. Boxplots of singular values while sleeping at different latent space dimensions for the CarRacing ($N = 10^4$).