Deep Knowledge Based Agent: Learning to do tasks by self-thinking about imaginary worlds

Anonymous Authors[1]

Abstract

Lifelong and meta learning aim to learn quickly new skills by transferring knowledge from past experiences. Current algorithms learn a meta-policy from a set of hand-crafted training tasks and adapt their policy to new but similar problems. In contrast, human are able to learn from very sparse data and apply learned skills to very different situations. In this work, we integrate the old-standing idea of knowledge-based agents with deep model-based reinforcement learning. We show that our model can transfer knowledge from completely randomly generated environments to real tasks. We argue that the weights of forward models can be used as a knowledge-base by an inference engine to infer optimal policy in real environments. We demonstrate that this approach automatizes both task-specification and knowledge base construction.

1. Introduction

Combining non-linear function approximations through deep neural networks with long-standing approaches in reinforcement learning (RL) led to unprecedented success in complex task domains such as Atari-2600 (Mnih et al., 2013) and Go (Silver et al., 2017). These ground breaking achievements are very promising for developing real world artificial intelligence. However, in order to outperform human performance, these methods usually require a large amount of training data gathered through interaction with environments, which is computationally challenging. Also, reusing past knowledge and experiences for faster learning of new tasks is a major challenge for current DRL approaches.

In order to overcome these obstacles meta and lifelong reinforcement learning have been proposed and explored extensively (Wang et al., 2016; Finn et al., 2017; Chen et al., 2018). Meta learning is a powerful paradigm to increase the data efficiency of learning models by transferring knowledge from a set of training tasks to new related and similar test tasks. One common assumption made by most meta-reinforcement learning models is that the test tasks are drawn from the same distribution of training samples and performance of meta-learned models strongly depends on the choice of this distribution. This requires a careful manual-design engineering which limits the scope of meta RL applications. On the other hand, lifelong RL tries to reuse any applicable skill gained bt agent during its lifelong to improve its performance on future tasks. Ideally, a lifelong learner should be able to transfer knowledge between completely uncorrelated environments. However, in both cases of meta and lifelong learning, practical implementations don't go beyond strongly connected test and train tasks. Furthermore, providing training tasks is a cumbersome procedure as it requires carefully hand-designing task distributions.

Motivated by this discussion, we propose a novel framework for meeting above mentioned challenges. The key idea is to augmenting the agent with a knowledge base which carry the burden of knowledge-transformation. Indeed having an agent which does reasoning on internal representation of knowledge is an old standing idea in artificial intelligence, usually termed as knowledge-based agent (Russell & Norvig, 2016). Traditional KB agents are based on manually provided KB and using logical programming such as Prolog to infer statement about world from KB and making a plan to reach the goal (Minker, 2012).

Here, we show that the idea of knowledge-based agent can be naturally integrated with model-based RL technique and resulting to a completely automated meta and lifelong approach, which we refer to as deep knowledge-base agent (DKBA). We show that our approach automatizes both task specification and knowledge-base building. As we shall illustrate, under proper circumstances, DKBA is capable of learning a meta-policy from completely unrelated and unstructured tasks, i.e. tasks those are sampled by uniform distribution. We then evaluate our model on some environ-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

ments with specific and different distributions, namely real worlds. Interestingly, DKBA performs closely to optimal policy on these new domains by doing reasoning on KB provided by forward models.

2. Related Work

Meta-learning aims to learn the learning process (Schmidhuber, 1987; Thrun & Pratt, 2012; Bengio et al., 1990; Andrychowicz et al., 2016; Finn et al., 2017). There are also a plenty of works in extending these results to reinforcement learning setting, including (Schweighofer & Doya, 2003; Duan et al., 2016; Xu et al., 2018; Gupta et al., 2018; Nagabandi et al., 2018a).

Lifelong learning, on the other hand, focuses on developing versatile systems that accumulate and refine their knowledge over time (Chen et al., 2018). In the context of reinforcement learning, this means that test tasks can be very different from train ones (Ring, 1997; Tanaka & Yamamura, 1997; Wilson et al., 2007; Ammar et al., 2015).

Deep Model-based reinforcement learning has been explored in (Nagabandi et al., 2018b; Feinberg et al., 2018; Kurutach et al., 2018).

Knowledge-based and logical agents have a long story and we refer interested reader to (Russell & Norvig, 2016; Minker, 2012) and references therein for more details and information.

3. Background

Reinforcement Learning. We consider the standard reinforcement learning paradigm where an agent interacts with an environment in discrete time steps t. We may model the environment as finite episodic Markov decision process (MDP), which is defined by 6-tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \pi, \gamma)$. Here, $\mathcal{S}$ is finite set of states ($\mathrm{S} = |\mathcal{S}|$), $\mathcal{A}$ the finite action space ($\mathrm{A} = |\mathcal{A}|$), $p(s_{t+1}|s_t, a_t)$ is the transition distribution conditioned on state-actions, and $r(s_t, a_t)$ is a reward function. Furthermore $\mathcal{M}$ has a policy $\pi(a_t|s_t)$ for selecting actions at each time step and a discount factor $\gamma \in [0, 1)$. The goal of agent is to maximize long-term expected discounted return $\mathbb{E}_\pi[\sum_{t=0}^{T} \gamma^t \, r(s_t, a_t)]$. Without loss of generality we set $\gamma = 1$ in what follows.

Model-based Reinforcement Learning. In model-based reinforcement learning agent predict how the environment will respond to its actions by learning a parametric transition function $f_{\boldsymbol{\theta_s}}(s_t, a_t)$ for predicting next state s_{t+1} and a reward model $f_{\boldsymbol{\theta_r}}(s_t, a_t)$ which predicts reward r_t. In high dimensional domains, deep neural networks, as universal function approximators, can be used to represents these models of world. The parameters of learned models, $\boldsymbol{\theta_s}$ and $\boldsymbol{\theta_r}$ or $\boldsymbol{\theta}$ for brevity, are correspond to weights of deep neural networks. The networks can be trained in standard supervised fashion by minimizing the prediction errors $\mathcal{E}^{s,r}$ over a set of trajectories $\mathcal{D}$ gathered by agent's own experience:

$$\underset{\boldsymbol{\theta_s}}{\operatorname{argmin}} \sum_{D} \mathcal{E}^s(s_{t+1}, f_{\boldsymbol{\theta_s}}(s_t, a_t)),$$
$$\underset{\boldsymbol{\theta_r}}{\operatorname{argmin}} \sum_{D} \mathcal{E}^r(r_t, f_{\boldsymbol{\theta_r}}(s_t, a_t))$$

Where $\mathcal{E}^s$ and $\mathcal{E}^r$ are appropriate loss objectives for estimating state and reward functions. We use cross entropy for both of them in this work but this choice is task-dependent, essentially.

The learned models can then be used in conjunction with a planing mechanism or control method to simulate the environment and planning actions to maximize rewards. The intention behind using model based methods is their ability in reaching high performance with significantly fewer experiences comparing with model-free approaches and improving sample complexity.

Meta Reinforcement Learning. Meta-RL primarily aims to learn quickly a new optimal policy in MDPs, $\mathcal{M}$, drawn from a distribution $\rho(\mathcal{M})$. In order to train agent a set of RL tasks, $\mathcal{M}_k$, are generated according to distribution $\rho(\mathcal{M})$. Sampled environments might have any new aspect including reward and transition function, however typically it is assumed that state and action space are fixed. The goal of few-shot meta learner is to quickly acquire a policy close to optimal one for test tasks using only a small number of samples. Collecting training set of tasks, the meta-learner's objective is to maximize expected return over these environments by learning a meta policy $\pi_{\rho(\mathcal{M})}$ through solving following optimization problem:

$$\underset{\pi_{\rho(\mathcal{M})}}{\operatorname{argmax}} \; \mathbb{E}_{\substack{\mathcal{M}_k \sim \rho(\mathcal{M}) \\ s_t \sim p_k \\ a_t \sim \pi_{\rho(\mathcal{M})}}} \left[\sum_{t=0}^{T} r(s_t, a_t) \right] \tag{1}$$

At test time, the meta learner is evaluated on unseen tasks generated from the same distribution $\rho(\mathcal{M})$. Although meta-RL generalizes the learning ability of simple RL agent, in practice test and train tasks are strongly correlated. For example the agent is trained on several mazes and at the test time agent should navigate new maps.

Lifelong Reinforcement Learning. One common assumption made by meta-learning techniques is that MDPs in training and test data are from the same distribution $\rho(\mathcal{M})$. Lifelong learning goes one step further by relaxing

this condition and tries to use any knowledge from past experiences that are applicable to new MDPs. Namely, the distribution of MDPs is unknown and fixed in lifelong-RL setting and the agent has to transfer knowledge across the task distribution. Although, in practice lifelong tasks are strongly connected. For instance, the agent is trained on several levels of a game and then evaluated on new levels of the same game.

Knowledge-Based Agent. One early approach in creating artificial intelligent system was building a knowledge base and using an inference mechanism to infer optimal decisions from that. A knowledge base (KB) contains any relevant information about the domain of application, including facts, rules and relations between elements and possibly also methods for solving problems. A knowledge-based agent is composed of a knowledge base and an inference system which knows how to use the knowledge in the base and do reasoning.

The language whose sentences represent facts about the world in KB is called knowledge representation (KR) language . Logical agents are subclass of knowledge-based agents where KR language is propositional or first order logic and inference mechanism is the standard logical deduction. In other words, logical AI involves representing knowledge of world, goals and current state of agent by sentences or axioms in logic, and agent can find a sequence of actions such that these axioms entail the goal by employing inference engine. This approach to AI in its declarative form led to logical programming frameworks including Prolog programming system.

4. Model

One drawback of conventional knowledge-based approaches to AI is that providing knowledge bases is not an automated process and for complex domains requires a tedious handmade programming. Also reusing learned knowledge for new tasks is not a straightforward procedure. In this work we suggest a method to address these obstacles by integrating the concept of knowledge-based agents with model-based reinforcement learning. This approach that we term as deep knowledge-based agent (DKBA) leads to an automated meta and lifelong reinforcement learning. The core idea is using the parameters of learned models, $\boldsymbol{\theta}$, as a knowledge base and utilizing an inference mechanism to infer optimal actions from it. However, instead of using logical rules, we train inference module in supervised manner by providing samples from randomly generated environments, $\mathcal{M}_i$. Each sample-point comprises 4-tuple $(\boldsymbol{\theta}^*_{\boldsymbol{s},i},\ \boldsymbol{\theta}^*_{\boldsymbol{r},i},\ s_{t,i},\ a^*_{t,i})$, where $\boldsymbol{\theta}^*_{\boldsymbol{s},i}$ and $\boldsymbol{\theta}^*_{\boldsymbol{r},i}$ are weights of trained forward models and $a^*_{t,i}$ is the optimal action in state $s_{t,i}$, all corresponding to $\mathcal{M}_i$. The inference engine can be represented by a meta-policy $\Pi_\Theta(a_{t,j}|s_{t,j};\boldsymbol{\theta}^*_{\boldsymbol{s},j},\boldsymbol{\theta}^*_{\boldsymbol{r},j})$ which predicts optimal action in state $s_{t,j}$ using knowledge about environment $\mathcal{M}_j$ provided by $\boldsymbol{\theta}^*_{\boldsymbol{s},j}$ and $\boldsymbol{\theta}^*_{\boldsymbol{r},j}$. The parameters of meta-policy, Θ, are computed by minimizing following cross entropy over the set of training tasks, $\mathcal{D}$:

$$\underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{\mathcal{D}} -\boldsymbol{a^*_{t,i}} \log \Pi_\Theta(a_{t,i}|s_{t,i};\boldsymbol{\theta}^*_{\boldsymbol{s},i},\boldsymbol{\theta}^*_{\boldsymbol{r},i}) \tag{2}$$

We show that if forward models are trained in a deterministic way, the DKBA is able to infer optimal actions based on knowledge encoded in parameters $\boldsymbol{\theta}^*$. This also allows us to transfer knowledge between completely dynamically-unrelated domains.

Furthermore, DKBA automatizes both task-designing and knowledge-base construction. Actually, in practice we adapt extreme situation by training agent on MDPs whose dynamics are drawn from uniform distribution and are completely uncorrelated. Indeed these generated environments may not correspond to any realistic world. At the test time, we evaluate the agent on real environments. Interestingly, DKBA is able to perform close to optimal agent by transferring knowledge from imaginary to real worlds.

Algorithm 1 provides an overview of vanilla DKBA. In train phase, we start by gathering some samples from randomly generated environments according to uniform distribution. We train world models to learn the dynamics of sampled environments using deterministic gradient descent. More specifically, we always begin from a fixed random parameters, $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{r}}$ for each environment and perform gradient descent for a fixed number of epochs and learning rate. In this way we can assign a unique knowledge base to each task. Empirically, we found this way of training very crucial in order to inference mechanism be able recognize similar patterns among different knowledge bases and transfer knowledge between tasks. The algorithm continues by finding optimal policy for the generated MDP. We may employ any standard RL method for this step. After collecting data, we optimize the parameters of meta-policy according to eq 2. At test time the agent receives new tasks generated by any distribution as long as action and state spaces are compatible with training data. We then train $f_{\boldsymbol{\theta_s}}$ and $f_{\boldsymbol{\theta_r}}$ exactly in same way as training phase, i.e. starting from the same initial weights, $\hat{\boldsymbol{\theta}}$, and using the same values for hyper parameters including learning rate and number of epochs. At the end agent returns prediction for optimal policy by using learned parameters in the previous step.

5. Experiments

In order to evaluate our approach that outlined above, we provide a detailed experimental analysis in this section. We

Algorithm 1 DEEP KNOWLEDGE-BASED AGENT (DKBA)

Training Phase

Input: Size of action space, dimentionality of state's representation, random parameters $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{r}}$, number of generated samples N, step size hyperparameters α and number of epochs E for training forward models.
Output: A parametric meta-learned policy $\boldsymbol{\Pi}_{\boldsymbol{\Theta}}$.

• Initialize dataset $\mathcal{D} \leftarrow \emptyset$..
for $n = 1$ **to** N **do**
• Ggenerate a MPD, $\mathcal{M}_n$, according to uniform distribution with given action size and state representation.
• Sample K trajectories from $\mathcal{M}_n$.
• Initilize parameters of $f_{\boldsymbol{\theta}_s}$ and $f_{\boldsymbol{\theta}_r}$ to $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{r}}$ respectively.
• Train world models $f_{\boldsymbol{\theta}_s}$ and $f_{\boldsymbol{\theta}_r}$ by performing deterministic gradient descent with step size α for E epochs using K trajectories.
• Compute optimal actions, a_t^* for each state in K trajectories using any RL method.
• Add $(\boldsymbol{\theta}_{\boldsymbol{s}}^*, \boldsymbol{\theta}_{\boldsymbol{r}}^*, s_t, a_t^*)$ to $\mathcal{D}$ for all states in trajectories.
end for
• Randomly initialize meta learner parameters $\boldsymbol{\Theta}$.
• Update meta parameters $\boldsymbol{\Theta}$ by minimizing error in predicting optimal actions as in eq 2.
Return $\boldsymbol{\Theta}$

Test Phase

Input: An MDP, $\mathcal{M}$, generated from any distribution with action and state space compatible with training phase.
Output: Optimal policy over $\mathcal{M}$.
• Sample K trajectories from $\mathcal{M}$.
• Initialize parameters of $f_{\boldsymbol{\theta}_s}$ and $f_{\boldsymbol{\theta}_r}$ to $\hat{\boldsymbol{\theta}}_{\boldsymbol{s}}$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{r}}$ respectively.
• Train world models $f_{\boldsymbol{\theta}_s}$ and $f_{\boldsymbol{\theta}_r}$ by performing deterministic gradient descent with step size α for E epochs using K trajectories.
Return $\boldsymbol{\Pi}_{\boldsymbol{\Theta}}(a|s; \boldsymbol{\theta}_{\boldsymbol{s}}^*, \boldsymbol{\theta}_{\boldsymbol{r}}^*)$

aim to understand whether DKBA is able to transfer knowledge between dynamically uncorrelated environments.

To keep the number of parameters tractable, in this proof of concept work, we evaluate our model on low-dimensional domains. We trained DKBA on uniformly sampled MDPs with $\mathrm{S} = 8, 10, 15$ and two actions, $\mathrm{A} = 2$. The generated MPDs are structured as follows: The states are represented as one-hot vectors and one state is selected as goal state, s_g. Also for simplicity, we choose a fixed reward function with $r(s_t, a_t) = \delta_{s_{t+1}, s_g} - 1$, i.e. reward is -1 on all transitions except those into goal state where it is zero. In order to generate dynamics of environments, we randomly select one state s_{t+1} as the resulting state of applying action a_t in state s_t. Then we train action-conditioned forward models to learn both transition and reward function.

The architecture of $f_{\boldsymbol{\theta}_s}$ and $f_{\boldsymbol{\theta}_r}$ consist of two fully-connected layer with $\tanh$ activation followed by a softmax layer. The number of nodes in the first and second layers of $f_{\boldsymbol{\theta}_s}$ are $(5, 10), (5, 10), (8, 12)$ for $\mathrm{S} = 8, 10, 15$, respectively. The corresponding numbers for reward forward model are $(5, 5), (5, 5), (8, 8)$. In total, we have 388, 452 and 876 parameters in forward models for $\mathrm{S} = 8, 10, 15$. In all our experiments we set number of epochs and learning rate to 300 and 0.01.

We also use a deep multilayer perceptron with ten layers and leaky-relu activation function (Maas et al., 2013) for representing inference engine. The number of neurons are $1200, 1000, 800, 600, 400, 200, 100, 50, 20$ and are kept fixed in all experiments.

The model was evaluated in two different ways. In the first scenario, we generated new test MPDs and computed the average return gained by agent, G, over them. More specifically, we generated 200 test environments and measured average return of agent on these in 10 episodes. In order to compare different setups, it would be beneficial to normalize the average returns as follows:

$$\hat{G} = \frac{G - G_{\text{random}}}{G_{\text{optimal}} - G_{\text{random}}} \tag{3}$$

where G_{random} and G_{optimal} are average returns of random and optimal agents, respectively. Here by optimal agent we mean an agent which is trained on individual environments to perform optimally. We may use any standard RL method to train this agent. However, for our low-dimensional spaces we used brute force search to find it. It is worth mentioning optimal agent does not have any ability to transfer knowledge and should be trained on each MDP from scratch. Fig 1 depicts the normalized returns in terms of the size of training data or seen environments, N. These results highlight the strength of DKBA in terms of sample efficiency. Indeed the total number of possible worlds are around $10^{14}, 10^{20}, 10^{35}$ for $\mathrm{S} = 8, 10, 15$, respectively. Therefore DKBA is able to perform close to optimal agent in new sampled environments without any further training by just utilizing the knowledge encoded in model parameters.

In the second scenario, we examine our trained agent on real environments. To do so, we employed a set of environments with different dynamics including: grid world, cliff walking, windy world and chip game. Figure ... shows the performance of DKBA on these real tasks. Interestingly, the results confirm the ability of DKBA in transforming

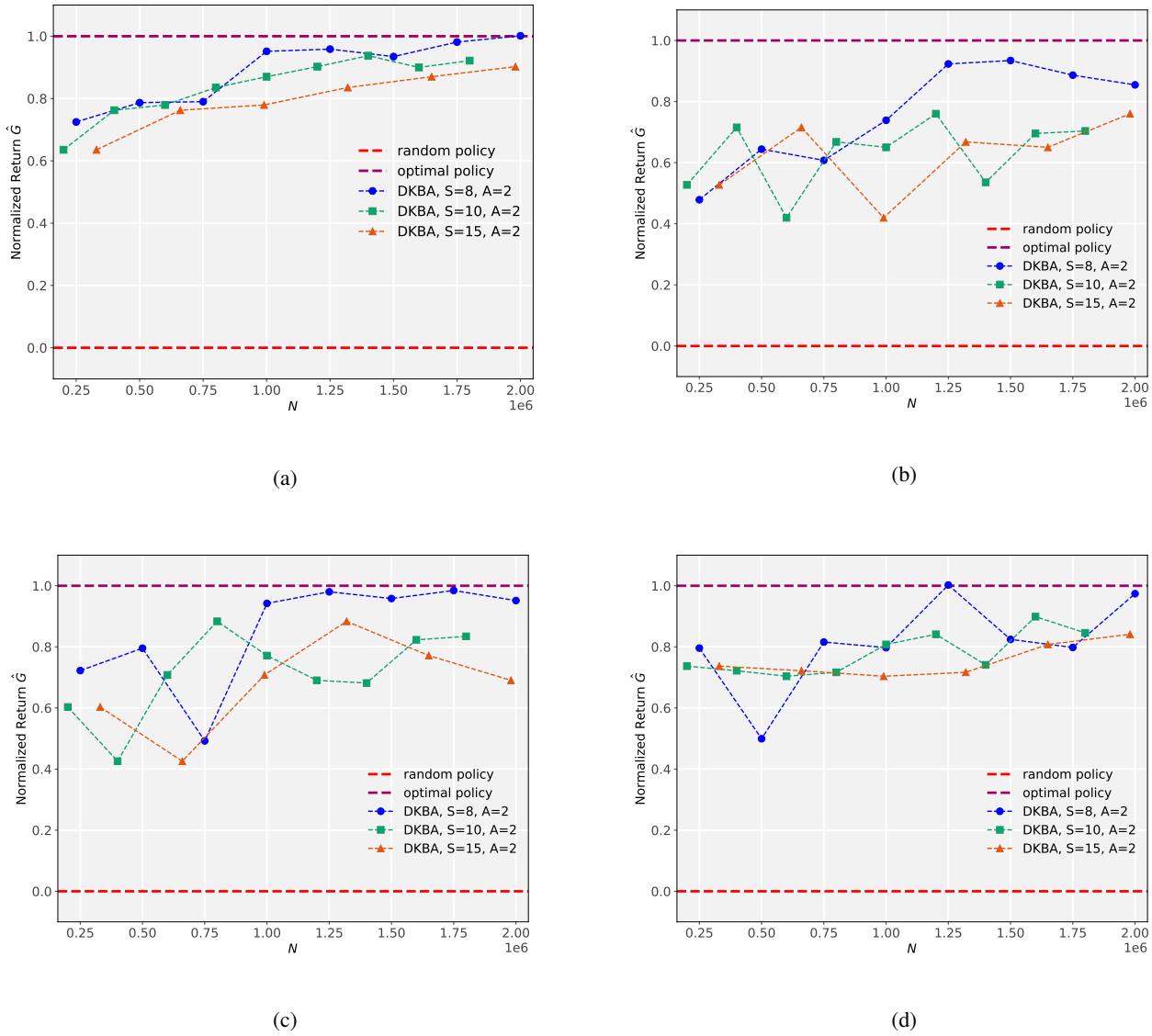

Figure 1. performance of DKBA on (a) test environments, (b)grid world,(c) cliff walking and (d) windy grid world.

knowledge from randomly sampled environments to real ones, as promised.

6. Conclusion and future directions

Adopting quickly to new tasks by operating on prior biases is a key feature of an intelligent system. Current meta-lifelong approaches provide such kind of agents on very correlated and similar tasks. In the present work, we propose a novel method for transferring knowledge between completely unrelated and unstructured tasks by learning the logic behind of reasoning. Our approach is general and can be used in conjunction with model based RL techniques without requiring any hand-specified task designing.

Our experimental results demonstrate that the parameters of trained world models can provide an expressive knowledge-base if the corresponding networks are trained in a deterministic way. Since learning the dynamics of complex domains requires utilizing very deep networks with lots of parameters, a challenge for applying our model in such domains is to keep the number of trained parameters under control. To this end, we may use an auto-encoder to compress the knowledge-base. Based on our results, this step should be also done in a deterministic way. Another solution would be distilling the knowledge in knowledge-base by replacing forward models with more smaller networks (Hinton et al., 2015). We leave these extensions to the future works.

Acknowledgments

TBA

References

Ammar, H. B., Tutunov, R., and Eaton, E. Safe policy search for lifelong reinforcement learning with sublinear regret. In *International Conference on Machine Learning*, pp. 2361–2369, 2015.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pp. 3981–3989, 2016.

Bengio, Y., Bengio, S., and Cloutier, J. *Learning a synaptic learning rule*. Université de Montréal, Département d'informatique et de recherche , 1990.

Chen, Z., Liu, B., Brachman, R., Stone, P., and Rossi, F. *Lifelong Machine Learning: Second Edition*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2018. ISBN 9781681733036. URL https://books.google.ro/books?id=JQ5pDwAAQBAJ.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., and Levine, S. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Kurutach, T., Clavera, I., Duan, Y., Tamar, A., and Abbeel, P. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3, 2013.

Minker, J. *Logic-based artificial intelligence*, volume 597. Springer Science & Business Media, 2012.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Nagabandi, A., Clavera, I., Liu, S., Fearing, R. S., Abbeel, P., Levine, S., and Finn, C. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 2018a.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018b.

Ring, M. B. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

Russell, S. J. and Norvig, P. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

Schmidhuber, J. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Schweighofer, N. and Doya, K. Meta-learning in reinforcement learning. *Neural Networks*, 16(1):5–9, 2003.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Tanaka, F. and Yamamura, M. An approach to lifelong reinforcement learning through multiple environments. In *6th European Workshop on Learning Robots*, pp. 93–99, 1997.

Thrun, S. and Pratt, L. *Learning to learn*. Springer Science & Business Media, 2012.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. Learning to reinforcement learn. *CoRR*, abs/1611.05763, 2016. URL http://arxiv.org/abs/1611.05763.

Wilson, A., Fern, A., Ray, S., and Tadepalli, P. Multi-task reinforcement learning: a hierarchical bayesian approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 1015–1022. ACM, 2007.

Xu, Z., van Hasselt, H. P., and Silver, D. Meta-gradient reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2396–2407, 2018.