Learning When to Trust the Expert for Guided Exploration in RL

Felix Schulz [*]
Neurorobotics Lab
University of Freiburg
Germany
felix.schulz@student.hpi.de

Jasper Hoffmann [*]
Neurorobotics Lab
University of Freiburg
Germany
hoffmaja@informatik.uni-freiburg.de

Yuan Zhang
Neurorobotics Lab
University of Freiburg
Germany
yzhang@informatik.uni-freiburg.de

Joschka Bödecker
Neurorobotics Lab
University of Freiburg
Germany
jboedeck@informatik.uni-freiburg.de

Abstract

Reinforcement learning (RL) algorithms often rely on trial and error for exploring environments, leading to local minima and high sample inefficiency during training. In many cases, leveraging prior knowledge can efficiently construct expert policies, e.g. model predictive control (MPC) techniques. However, the expert might not be optimal and thus, when used as a prior, might introduce bias that can harm the control performance. Thus, in this work, we propose a novel RL method based on a simple options framework that only uses the expert to guide the exploration during training. The exploration is controlled by a learned high-level policy that can decide to follow either an expert policy or a learned low-level policy. In that sense, the high-level skip policy learns when to trust the expert for exploration. As we aim at deploying the low-level policy without accessing the expert after training, we increasingly regularize the usage of the expert during training, to reduce the covariate shift problem. Using different environments combined with potentially sub-optimal experts derived from MPC or RL, we find that our method improves over sub-optimal experts and significantly improves the sample efficiency.

1 Introduction

Exploration is one of the key challenges in reinforcement learning (RL). Model-fee RL methods often treat the environment as a black box getting information about the environment by only observing transitions. This generality comes with the cost of inefficient exploration, leading to high sample inefficiency and local optima [11]. However, in many cases, we have access to an expert in the form of a policy that at least partially knows how to navigate the environment: Model predictive control (MPC) uses explicit knowledge of the environment dynamics but can be sub-optimal when the model is inaccurate, however, it can efficiently calculate a control by using numerical optimization. In the case of autonomous driving, the dynamics of the controlled car can be sufficiently described for an MPC formulation [25] wherein the uncertain behavior of the surrounding traffic is often predicted with significant simplifications [19]. Still, the MPC policy can be seen as an expert that at least partially knows how to drive safely and comfortably.

[*]Equal contribution

In such scenarios, using imitation learning can limit the performance of the final policy, as the sub-optimal behavior of the expert is reproduced. Instead, in this work we want to leverage the expert only to guide the exploration while training a policy with RL. To achieve this goal, we address two challenges in this work: Firstly, the expert might not always be helpful and can even mislead the exploration process. Secondly, as we only want to use the expert during training, we must ensure that the final learned policy works independently of the expert.

Our method is motivated by the work of [4], where a high-level skip policy learns how often an action proposed by a low-level action policy should be repeated, which can be seen as an option in the options framework of RL [22]. Furthermore, as the skip policy avoids predicting new actions with the low-level policy for a few steps, value backups can be done over multiple steps, which the authors introduce as skip connections. This allows RL methods to propagate reward signals faster to earlier states [4, 22]. Finally, for the approach to work, the underlying, potentially restricting assumption is that repeating actions helps for exploration.

In this work, we do not repeat the proposed action but generalize it by querying an expert for further actions. We explicitly learn a value function to decide how long we should follow the expert after taking one-step action from the low-level action policy. This addresses the first mentioned challenge as we learn when to trust the expert for exploration. The second challenge arises from a covariate shift problem: As the high-level skip policy generates the state distribution encountered during training by a mixture of expert and low-level policy actions, the training distribution might differ from the one experienced during deployment where we only follow the low-level action policy. In this case, we enforce that the expert is gradually used less and less during training, ensuring that the support of the training state distribution also covers the one during testing [20].

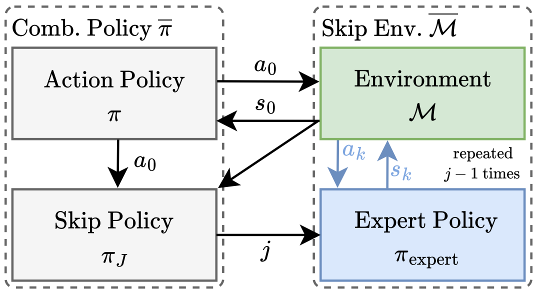
The main contributions of our work are: (1) We propose a simple options framework for utilizing queryable experts. With our approach we learn when to trust an expert and when to explore with the low-level learned action policy. Similar to [4] we introduce skip connections that allow us to speed up the backpropagation of reward signals, potentially allowing for faster learning progress. (2) We gradually regulate the sampling of the expert during training to reduce the covariate shift of the state distribution that would be encountered by the learned low-level action policy during deployment. (3) We validate our approach experimentally by finding that our method outperforms the sub-optimal MPC and RL experts and improves over baselines in terms of sample efficiency. We further find that regulating the usage of the expert is key to achieving good performance without the expert.

2 Related Work

Exploration in Reinforcement Learning The problem of exploration in RL has been worked on in various aspects [11]: Actions are repeated to allow for more efficient exploration and faster propagation of values [15]. This idea was further extended in [6], where the concept of an ϵ-greedy policy was extended by sampling how often the action should be repeated. Other approaches introduce a high-level policy that learns how often an action should be repeated [4] using the concept of skip-connections, where a repeated action is treated as one action, allowing faster propagation of the reward signal. The approach we follow in this work can be seen as an extension of the latter, where we replace repeating the same actions with expert actions. Further, curiosity-based approaches give the RL agent an additional reward based on how surprising the transition was [3, 17]. Further work used Ornstein-Uhlenbeck processes [8] to introduce temporally correlated exploration noise. The choice of different stochastic processes for exploration was investigated in [7].

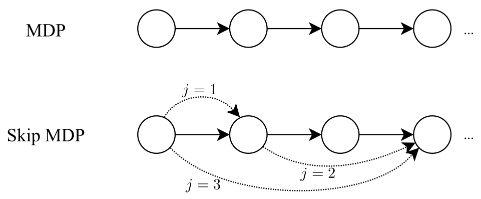
Guided Policy Search Guided policy search uses an MPC expert to gradually guide the learned policy to better trajectories [12–14, 16]. Here, the predicted trajectory of the expert is restricted by how far the expert can deviate from the current predictions of the learned policy [16]. Different from our approach, guided policy search mainly focuses on imitating the expert. However, our approach could further benefit from the idea that the expert takes into account the approximation error of the low-level learned policy.

Exploration via the Options Framework The options framework was proposed in [22] to include more high-level abstract actions in the form of a semi-Markov decision process that extends the action set of the original Markov decision process (MDP) with options. When an option is triggered, a low-level policy is applied closed-loop until a termination criterion is fulfilled. The option-



Comb. Policy $\overline{\pi}$
Action Policy π
Skip Policy π_J
Skip Env. $\overline{\mathcal{M}}$
Environment $\mathcal{M}$
Expert Policy π_{expert}
a_0
s_0
a_0
j
a_k
s_k
repeated $j-1$ times

(a) Diagram of our exploration method.



(b) Visualization of the additional transitions of Skip MDPs over traditional MDPs.

Figure 1: In (a), we highlight how the different components of the proposed method interact for exploring the environment. The low-level action policy π observes the current state s_0 by the environment, sending its current action a_0 to the high-level skip policy π_J. The skip policy then decides how many expert actions it adds to the action a_0 by choosing a skip length j. The skip connection is executed by letting the expert interact with the environment for $j-1$ steps. After that, the action policy chooses an action again. In (b), we highlight that using the expert in that way can be reformulated as an MDP with additional skip connections. A skip length of $j = 1$ means that no expert actions are executed.

critic architecture [1] learns options by interacting with the environment while simultaneously learning when to choose which option. They manage to create abstraction autonomously, and several works have used this architecture as their foundation: [10] aim to improve the interpretability and reusability of the learned options by using interest functions to identify where specific options are more advantageous. With a similar goal in mind, [5] used an attention-based mechanism to learn diverse options and mitigate problems of the option-critic architecture, which would often learn to tackle an environment primarily with one option or by switching options frequently. Note, that a key difference of our approach is, that we do not assume access of the expert or options during deployment, which can, as explained before, come with a covariate shift problem. To the best knowledge of the authors, this is not addressed in previous work on options.

Exploration with an Expert An approach that is very related to our work is [9], where multiple experts guide the exploration of a Q-learning agent. For the behavior policy, at each step - the experts together with the agent - vote on the next action by using Boltzman sampling of the sum of the value functions of the agent and the experts. Our approach differs by deciding on how many steps we want to follow an expert, which additionally allows faster value propagation via skip connections. Furthermore, the approach in [9] is derived only for discrete actions.

3 Approach

We begin this chapter by introducing skip connections to MDPs by using a queryable expert. Next, we detail how the high-level skip policy can be learned using the extended MDP formulation. Finally, we introduce a modified version of DAGGER sampling [20], a method from imitation learning that ensures that the states encountered with the expert - during training - and by the learned policy itself - during testing - are distributed more similarly.

3.1 Skip MDPs with Expert-Actions

Given an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ with a state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P \in \Delta_{\mathcal{S}}^{\mathcal{S}\times\mathcal{A}}$, a reward function R and a discount factor $\gamma \in (0, 1]$ [18], we extend the concept of an MDP to a skip MDP, see Figure 1b. Notably, this definition is an extension of the skip MDP introduced in [4], where we additionally incorporate an expert instead of repeating the action.

Let us now assume that we have access to an expert policy $\pi_{\text{expert}} \in \Delta_{\mathcal{A}}^{\mathcal{S}}$. The high-level idea is that after executing the first action a we further execute actions generated by the expert policy π_{expert}.

More specifically, we derive a new transition kernel P_{expert} via

$$P_{\text{expert}}(s'|a,j,s) := P(s_1|a_0,s_0)\prod_{k=1}^{j-1}\pi_{\text{expert}}(a_k|s_k)P(s_{k+1}|a_k,s_k) \tag{1}$$
$$a_k \sim \pi_{\text{expert}}(\cdot|s_k), s_{k+1} \sim P(\cdot|a_k,s_k), s_0 = s, s_j = s', a_0 = a,$$

by following the expert policy for $j-1$ steps. This allows us to do multiple steps in the original MDP in a single step, thus the name skip connection, see Figure 1b. We denote with $\mathcal{J} = \{1,\dots,J\}$ the set of all possible skip lengths, which will be later selected by the high-level skip policy. Similar to (1), we define the reward R_{expert} accumulated during the skip connection with

$$R_{\text{expert}}(s,a,j,s') := \sum_{k=0}^{j-1}\gamma^k R(s_k,a_k,s_{k+1}). \tag{2}$$

The action space of the Skip MDP is extended to $\mathcal{A}\times\mathcal{J}$. Thus, the final Skip MDP is an MDP given by $\overline{\mathcal{M}} = (\mathcal{S},\mathcal{A}\times\mathcal{J},P_{\text{expert}},R_{\text{expert}},\gamma)$. The expert skip MDPs can be seen as a simple options framework [22], where an option means following the expert policy for j steps after executing an action of the low-level policy first.

3.2 Learning When to Trust the Expert

In our approach, we are interested in learning an optimal low-level action policy $\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}$ and a high-level skip policy $\pi_J \in \Delta_{\mathcal{J}}^{\mathcal{S}\times\mathcal{A}}$. As can be seen in Figure 1a, given a current state s, the low-level action policy first predicts an action $a \sim \pi(s)$ and then the high-level skip policy based on the selected action a and the current state s predicts the skip length $j \sim \pi_J(s,a)$. We further denote the resulting combined policy with $\overline{\pi} \in \Delta_{\mathcal{A}\times\mathcal{J}}^{\mathcal{S}}$.

We start by discussing the standard approach of learning a value function for a given policy π for the original MDP $\mathcal{M}$ and then extend it to the skip MDP $\overline{\mathcal{M}}$. The action-value function Q^π of a given policy is uniquely defined via the Bellman equation with

$$Q^\pi(s,a) = \mathbb{E}[R(s,a,s') + \gamma Q^\pi(s',a')]$$
$$s' \sim P(\cdot|s,a), a' \sim \pi(\cdot|s').$$

From the Bellman equation, we can derive a typical temporal difference objective for a parameterized Q-function Q^θ [15] by

$$\mathcal{L}^Q(\theta) = \mathbb{E}_{(s,a,s')\sim\mathcal{D},a'\sim\pi(\cdot|s')}\left[\left(R(s,a,s') + \gamma Q^{\tilde{\theta}}(s',a') - Q^\theta(s,a)\right)^2\right], \tag{3}$$

where $\mathcal{D}$ denotes a replay buffer and $\tilde{\theta}$ indicates the parameters of a target network that are updated using Polyak averaging [8].

We extend the approach to learning the value function of the combined policy $\overline{\pi}$ on the skip MDP $\overline{\mathcal{M}}$. Given the extended transition kernel P_{expert} and reward function R_{expert}, the action-value function $\overline{Q}^{\overline{\pi}}$ is uniquely defined via the Bellman equation

$$\overline{Q}^{\overline{\pi}}(s,a,j) = \mathbb{E}\left[R_{\text{expert}}(s,a,j,s') + \gamma^j \overline{Q}^{\overline{\pi}}(s',a',j')\right],$$
$$s' \sim P_{\text{expert}}(\cdot|s,a,j), (a',j') \sim \overline{\pi}(\cdot|s').$$

A similar definition allows us to introduce the optimal action-value function $\overline{Q}^\star$ [21], which is the action-value function for an optimal policy $\overline{\pi}^\star$ of $\overline{\mathcal{M}}$. Another interesting observation is that different from the standard definition of MDPs, in skip MDPs, the discount factor depends on the skip length j.

Similar to (4), if we want to approximate $\overline{Q}^{\overline{\pi}}$ with a function approximator $\overline{Q}^\theta$ we derive the following objective

$$\mathcal{L}^{\overline{Q}}(\theta) = \mathbb{E}_{(s,a,j,s')\sim\overline{\mathcal{D}},\ (a',j')\sim\overline{\pi}(\cdot|s')}\left[\left(R_{\text{expert}}(s,a,j,s') + \gamma^j \overline{Q}^{\tilde{\theta}}(s',a',j') - \overline{Q}^\theta(s,a,j)\right)^2\right]. \tag{4}$$

In the context of this work, we use an actor-critic method [21]. Note, learning a critic with Equation (4) allows us to criticize the low-level action policy independently of the high-level skip policy. The following lemma justifies this by showing that the optimal action-value functions of $\mathcal{M}$ and $\overline{\mathcal{M}}$ align for a skip length of $j = 1$:

Lemma 1. *Given the unique optimal action-value function $Q^\star$ of the original MDP $\mathcal{M}$ and $\overline{Q}^\star$ of the skip MDP $\overline{\mathcal{M}}$, we have that*

$$\overline{Q}^\star(s, a, j) \le Q^\star(s, a) \quad \forall j \in \mathcal{J}, \tag{5}$$

$$\overline{Q}^\star(s, a, 1) = Q^\star(s, a), \tag{6}$$

for $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

For a proof see Appendix C. A general derivation for the options framework can be found in [22].

In conclusion, using this approach allows us to learn when to trust the expert, as we learn the Q-value of following the expert in the context of the skip MDP for different skip lengths. Furthermore, as noted in [4], using skip connections allows us to propagate values faster, leading to potentially faster learning. Finally, while collecting transition samples for the skip MDP in the replay buffer $\overline{\mathcal{D}}$, shorter skip connections can be observed from longer skip connections. For example, if a skip length of $j = 3$ is executed, we also observe two skip lengths with $j = 2$ and three skip lengths of $j = 1$.

3.3 Mitigating the Distribution Problem

Our goal is to have a standalone low-level action policy after training. Thus, the action policy should learn how to act without access to the expert. An improper action during evaluation can quickly lead to states that the policy has not encountered during training, resulting in poor performance. To address this issue, we take inspiration from the DAGGER algorithm from Ross et al. [20], which was designed for imitation learning. At each time step when we would decide on the skip length, we sample from a Bernoulli variable $\alpha_t \sim \text{Bern}(\beta_t)$, where $\alpha_t = 0$ means that the skip length is set to $j = 1$ and the expert will not be used and $\alpha_t = 1$ means that we can normally sample from $j \sim \pi^J$. The schedule β_t controls the sample mechanism during training. To give another example, if we choose a constant schedule $\beta_t = 1$, the expert usage is not regularized, whereas if we choose $\beta_t = 0$, the expert is not used.

4 Experiments

In the experimental part of this work, we evaluate the performance of our proposed method against other baselines with the following questions: (1) Can our method effectively leverage the expert to improve the training in terms of sample efficiency? (2) What happens when the expert is imperfect? More specifically, do we learn when to trust the expert, and improve over suboptimal experts?

Baselines and Variations We compare our method to two baselines: TD3 [8] and TEMPORL [4]. Even though Biedenkapp et al. [4] used DDPG as a policy gradient method for deep learning with TEMPORL, we implemented it using TD3 for a better comparison with our method. All methods, including ours, are based on TD3 and we conclude the shared hyperparameters in the Appendix B. Additionally, for TEMPORL we used a maximum skip length of 4 for the CartPole task and 10 for the LunarLander task as they performed best. Finally, we tested our method with two different configurations: (1) *Ours* uses the recommended β-schedule from Ross et al. [20] adapted to the RL framework, which performed best compared to linear, decaying, and sinusoidal β-schedules. Ross et al. [20] recommend the parameter-free β-function $\beta_i = I(i = 1)$, where the expert is queried as much as possible during the first iteration of training and not at all afterward. To recreate the parameter-free schedule for RL and since they used 20 iterations of training, we use $\beta_t = \mathbb{1}(t < T/20)$ with the addition of a small offset to allow for some expert actions through training. T denotes the maximum amount of time steps and t the current time step. (2) *Ours$_O$* does not use DAGGER sampling, by keeping a constant β value of 1 resulting in no regularization. All methods are trained for 10^5 steps for the CartPole task and 10^6 steps for the LunarLander task with 10 random seeds.

4.1 Learning from a Nearly Optimal Expert

Setup We evaluated our method on CartPole [2] with additional constraints and on the LunarLander environment from the GYMNASIUM project [23]. Details on both environments can be found in the Appendix A. The expert for the CartPole environment is derived from the Model Predictive Control (MPC) method. This expert is approximately optimal, as the model used to generate the expert actions in MPC is the same as the environment to generate the next state. In the LunarLander environment, the expert is a trained RL agent using PPO without the optional wind and turbulence.

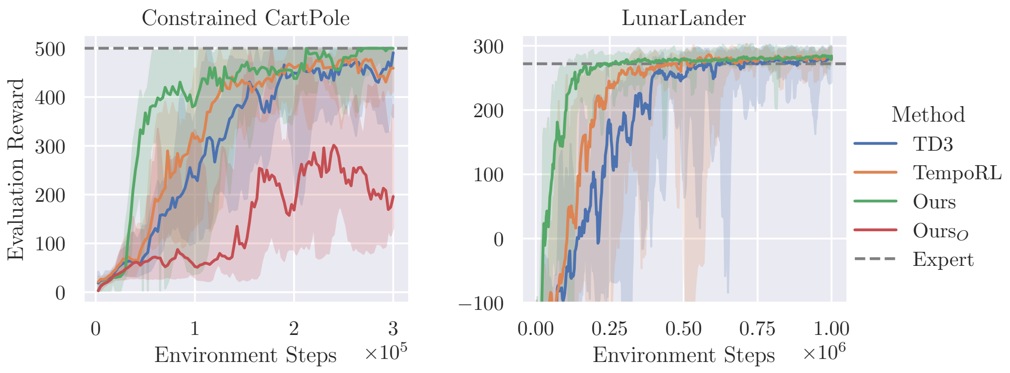


Figure 2: Evaluation performance during training of the baselines and our methods with nearly optimal experts. The usage of the expert is regularized for *Ours* while it is not for *Ours*$_O$. The performance of each expert is visible as the dotted line. Each configuration's median of 10 runs is plotted, with the minimum and maximum values creating the shaded areas.

Results In the CartPole environment TEMPORL performs similarly to the regular TD3 in the training curves plotted in Figure 2. This might be because of the specific environment used; repeating the same action multiple times is likely to bring the pole out of balance, similar to the *Pendulum* environment mentioned by Biedenkapp et al. [4]. Our approach shows promising results: the evaluation reward during training rises more quickly and achieves a perfect median score by the end of training. Similar results are found in the LunarLander environment. The policy in our method manages to learn faster and solve the environment, by reaching a reward of 200, in less than a third of the number of timesteps compared to the vanilla TD3 algorithm and about twice as fast as TEMPORL. The expert is matched after 300k timesteps, while TD3 and TEMPORL require about 600k. Our method learns to outperform the expert consistently, although only by a slight margin.

Regulating Expert Usage Including expert regularization is crucial, as can be seen when comparing the plots with regularization *Ours*and without *Ours*$_O$ in the CartPole environment. Without DAGGER sampling, even if the training reward is high (visible in Appendix A.1 Figure 6), the policy has a poor evaluation performance without the expert. The expert can fix mistakes made by the policy during training. However, during evaluation, the policy can not rely on the expert and quickly encounters states not represented well in its training distribution of mostly perfect runs.

4.2 Learning from a Suboptimal Expert

Setup In this section, we evaluate the method with suboptimal experts. For the CartPole task, the pole's length parameter of the MPC expert's model is modified to 0.3m, compared with the 0.5m in the real environment, leading to imperfect controls by the MPC expert. For the LunarLander task, the environment is made more difficult by enabling wind and turbulence parameters, which were disabled in the training environment of the RL expert. The wind and turbulence parameters are forces and torques applied to the spacecraft that slowly change over an episode, making controlling the lander much more tricky. Further details on the environments can be found in the Appendix A.

Results As shown in Figure 3, our method can learn to outperform the suboptimal experts in both CartPole and LunarLander environments. Additionally, our method can learn to solve the tasks

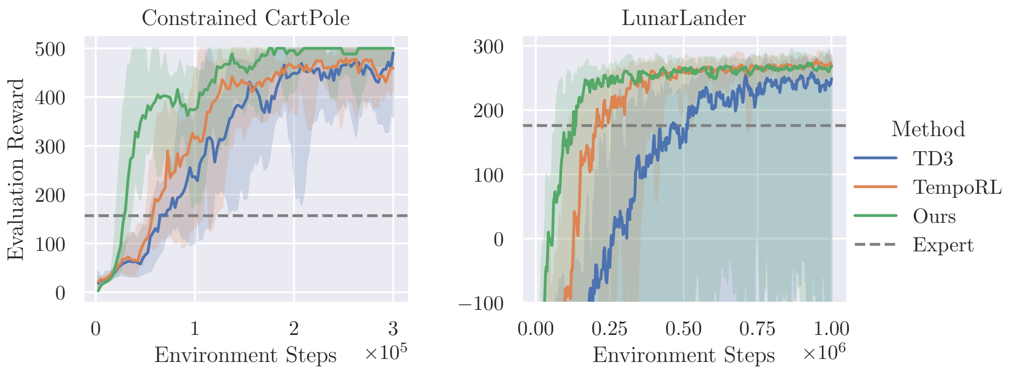


Figure 3: Evaluation performance during training of the baselines and our methods with suboptimal experts. Each expert's performance is visible as the dotted line. Similarly to the CartPole results, each configuration's median of 10 runs is plotted, with the minimum and maximum values creating the shaded areas.

in about the same amount of timesteps as with a nearly optimal expert. Meaning that the gains in sample efficiency are still realized even with suboptimal experts. The median performance in the CartPole environment reaches a perfect score of 500 earlier than with an optimal expert. This might be because the expert is worse, leading to our approach exploring by itself earlier. In the LunarLander environment, similar to our method, TEMPORL is unaffected by the increased difficulty while TD3 is not. We further tested imitation learning using DAGGER [20], achieving a similar evaluation reward as the expert. Compared to imitation learning methods, our method can achieve better performance beyond the suboptimal expert.

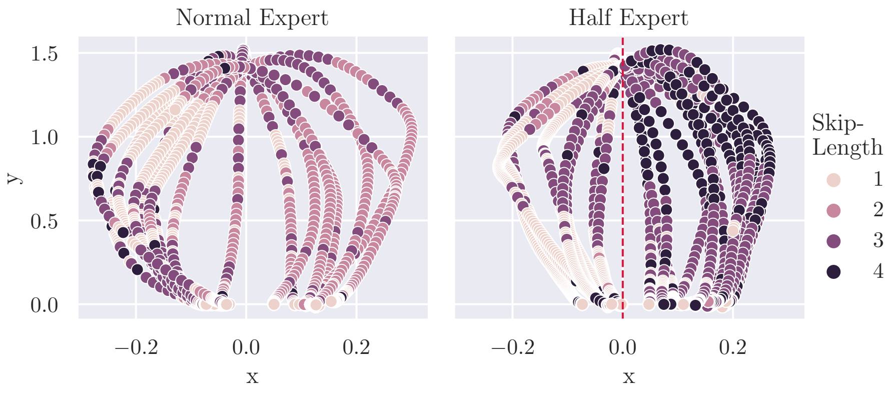


Figure 4: Comparing sample trajectories of two agents, one trained with the normal expert and the other one with an expert that outputs random actions when $x < 0$.

Analysis on Skip Policies Furthermore, we visualize the high-level skip policy to better understand its impacts on the low-level action policy. We additionally introduce the "half"-expert to the LunarLander environment. It is based on the original RL expert but outputs random actions in states with $x < 0$. The learned skip policies under both "half"-expert and normal expert are visualized in Figure 4. It indicates that the policy chooses longer skips when the expert is knowledgeable but sometimes selects a skip length higher than one where the actions are random. The skip policy trained with the normal expert also shows a slight tendency to choose higher skip lengths on the right but to a much lesser degree. In conclusion, we can see that our approach is able to learn when to trust the expert.

5 Conclusion

We found that our proposed framework guides the RL algorithm by learning when to trust the expert for exploring the environment leading to improved sample efficiency. An important next step could be validating the approach in more challenging environments with higher state or action dimensions, sparse rewards or a high degree of uncertainty. On such environments the advantage of our method might even further increase, as exploration becomes significantly harder, thus potentially showing that using the expert avoids local minima. An interesting practical direction could be to apply the method in the context of autonomous driving. Further directions to combine RL with control algorithms, could be to not only predict the skip length with the high-level policy but instead also predict a parameter for a parameterized MPC expert. This would allow the high-level policy to learn how to improve the expert policy, similar to the option-critic framework [1].

Acknowledgments and Disclosure of Funding

This work is supported by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 953348 (ELO-X). This research was further funded by the German Federal Ministry for the Environment, Nature Conservation and Nuclear Safety (BMU) on the basis of a resolution of the German Bundestag as part of the 'KI-Leuchtturm' project 'Intelligence for Cities' (I4C).

References

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017. Issue: 1.

[2] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, September 1983. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[3] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[4] André Biedenkapp, Raghu Rajan, Frank Hutter, and Marius Lindauer. TempoRL: Learning when to act. In *Proceedings of the 38th international conference on machine learning (ICML 2021)*, July 2021.

[5] Raviteja Chunduru and Doina Precup. Attention Option-Critic. In *4th Lifelong Machine Learning Workshop at ICML 2020*, 2020.

[6] Will Dabney, Georg Ostrovski, and Andre Barreto. Temporally-Extended -Greedy Exploration. In *International Conference on Learning Representations*, 2020.

[7] Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022.

[8] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pp. 1587–1596. PMLR, 2018.

[9] Ashwin Khadke, Arpit Agarwal, Anahita Mohseni-Kabir, and Devin Schwab. Exploration with Expert Policy Advice, 2019. ri.cmu.edu/app/uploads/2019/10/Exploration_with_Expert_Policy_Advice_ICAPS.pdf.

[10] Khimya Khetarpal, Martin Klissarov, Maxime Chevalier-Boisvert, Pierre-Luc Bacon, and Doina Precup. Options of interest: Temporal abstraction with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4444–4451, 2020. Issue: 04.

[11] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[12] Sergey Levine and Vladlen Koltun. Guided Policy Search. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 1–9. PMLR, May 2013.

[13] Sergey Levine and Vladlen Koltun. Variational Policy Search via Trajectory Optimization. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[14] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015.

[16] Igor Mordatch and Emo Todorov. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems*, volume 4, pp. 23, 2014.

[17] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V. Hafner. Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2): 265–286, April 2007. Conference Name: IEEE Transactions on Evolutionary Computation.

[18] Martin L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley series in probability and statistics. Wiley-Interscience, Hoboken, NJ, 2005.

[19] Rudolf Reiter, Jasper Hoffmann, Joschka Boedecker, and Moritz Diehl. A Hierarchical Approach for Strategic Motion Planning in Autonomous Racing. In *European Control Conference (ECC)*, pp. 1–8, June 2023.

[20] Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, volume 15 of *Proceedings of machine learning research*, pp. 627–635, Fort Lauderdale, FL, USA, April 2011. PMLR.

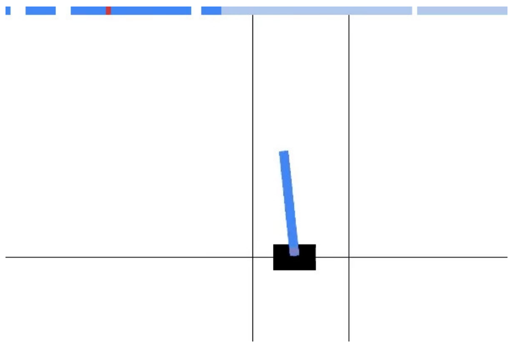
[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning series. The MIT Press, Cambridge, Massachusetts, second edition edition, 2018.

[22] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2): 181–211, August 1999.

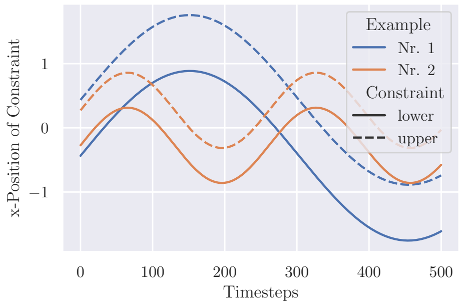
[23] Mark Towers, Jordan K. Terry, Ariel Kwiatkowski, John U. Balis, Gianluca de Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Arjun KG, Markus Krimmel, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Andrew Tan Jin Shen, and Omar G. Younis. Gymnasium, March 2023.

[24] Robin Verschueren, Gianluca Frison, Dimitris Kouzoupis, Jonathan Frey, Niels van Duijkeren, Andrea Zanelli, Branimir Novoselnik, Thivaharan Albin, Rien Quirynen, and Moritz Diehl. acados—a modular open-source framework for fast embedded optimal control. *Mathematical Programming Computation*, 14(1):147–183, March 2022.

[25] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a Frenét Frame. In *2010 IEEE International Conference on Robotics and Automation*, pp. 987–993, May 2010. ISSN: 1050-4729.



(a) Visualization of the CartPole environment with constraints as vertical lines which move over time.



(b) Lower and upper bound of the sin-based constraint function, with two different sets of parameters.

Figure 5: Visualization of the additional constraints in the modified CartPole Environment.

A Details on the Environments

A.1 Modified CartPole

This environment is derived from the classic CartPole enironment and uses an MPC expert based on the formulation of the *pendulum_on_cart* example from the ACADOS library [24]. We modified the environment to be more challenging to solve: instead of fixed thresholds for how far the cart is allowed to travel from the center before the episode is terminated, we implemented moving constraints, forcing the cart to move slowly from side to side while balancing the pole. The constraints move based on a sinus function with parameters for speed, amplitude, and width. After each episode, these parameters are randomized by sampling from a set of configurations that the MPC is able to solve. Meaning that the policy is forced to react differently in every episode and cannot simply memorize where to move. The constraints are visible in the visualizations of the environment in Figure 5a, and a plot of their change over time can be seen in Figure 5b.

The training performance of the different methods in the modified CartPole environment can be seen in Figure 6. With a nearly perfect expert (Ours, Ours$_I$), our method quickly learns to trust the expert. This results in a perfect training reward early on. The evaluation reward is close to zero since the low-level action policy has not learned to balance the pole yet. When the regularization starts to take effect, the training performance drops down since the expert can only rarely be queried. The reward quickly rises together with the evaluation reward, as the low-level action policy trains almost on its own with expert demonstrations in its replay buffer. With a suboptimal expert (Ours$_M$), there is no such spike in the beginning but the method still benefits from the expert demonstrations compared to the baselines.

A.2 LunarLander

In the LunarLander environment, the task is to land a 2D spacecraft on the surface of a moon by modulating one main engine and two side engines effectively. The agent can learn to accomplish this task based on eight observations and two control variables at each timestep. Wind and turbulence parameters can be enabled to change the difficulty and randomness of the environment. As the expert, we trained an RL agent on the environment using PPO without wind and turbulence.

Changing the difficulty of this environment is easily accomplished by enabling and tweaking its wind and turbulence parameters. In this case, wind and turbulence refer to a force and torque that is applied to the spacecraft, respectively. Both slowly change over an episode, making controlling the lander much more difficult. Table 1 shows how the performance of the RL-expert changes under different conditions. Note that the expert was trained without wind or turbulence.

An agent can learn to land the spacecraft based on eight observations and two control variables at each timestep: its position and velocity in x- and y-direction, its angle and angular velocity, as well as two booleans that indicate the ground contact of each leg. In our continuous cast, the thruster output can be determined with two values. The main engine will only turn on when its control value is above

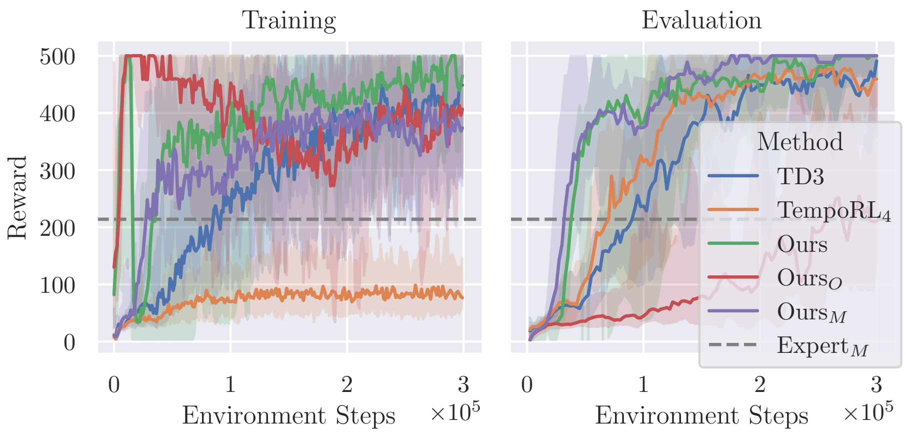


Figure 6: Evaluation and training performance in the CartPole environment of the baselines TD3 and TEMPORL as well as our method with an optimal expert with regularization (Ours), without regularization (Ours$_O$), and with a mismatched expert (Ours$_M$). The mismatched expert (Expert$_M$) is displayed, while the normal expert achieves a perfect reward of 500. Each configuration's median of 10 runs is plotted, with the minimum and maximum values creating the shaded areas. Note that we evaluated the method during training and not in an additional evaluation run.

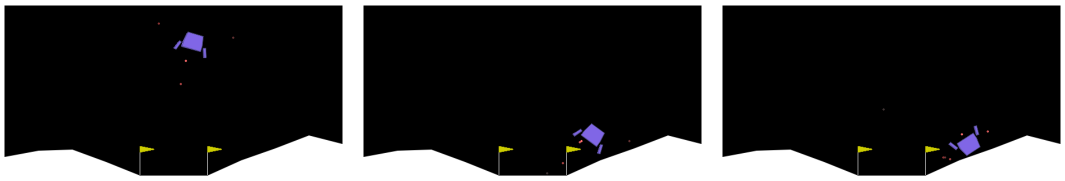


Figure 7: LunarLander Environment.

Description	**Wind**	**Turbulence**	**Mean Reward**
Max wind	20	2	176
Default wind	15	1.5	249
No wind	0	0	272

Table 1: Effect of wind and turbulence on the RL expert of the LunarLander environment.

0 with a thrust of 50% and scale to 100% thrust as the value reaches 1. The lateral thrusters will not turn on for a value between -0.5 and 0.5 while scaling similarly from 50% to 100% from ± 0.5 to ± 1 for the right and left thrusters, respectively.

B Hyperparameters

For both actor and critic we used a linear neural network with two hidden layers consisting of 400 and 300 nodes, using ReLU activation. More hyperparameters can be seen in Table 2. The models were trained on a workstation with an AMD 5950X 16-Core Processor, 64 GB of RAM and an NVIDIA GeForce RTX 3080 Ti. Usually, 3 experiments were started in parallel and would take less than an hour.

Hyperparameter	Value
Batch Size	256
γ	0.99
τ	0.005
ϵ	0.1
Learning Start	1000
Policy Noise	0.2
Noise Clip	0.5
Policy Delay	2
Loss Function	MSE Loss
Optimizer	Adam
Learning Rate	0.0003
β_1	0.9
β_2	0.999
Replay Buffer Sizes	10^6

Table 2: Hyperparameters used for all methods.

C Proof of Lemma 1.

First, we show $\overline{Q}^\star(s,a,j) \leq Q^\star(s,a)$. Let $\overline{\pi}^\star$ be an optimal policy for $\overline{\mathcal{M}}$, which can be seen as a history-dependent policy that depends at maximum on the last J transitions. Due to the Markov property of $\overline{\mathcal{M}}$, history-dependent policies can not perform better than Markovian policies. Thus, we conclude by

$$\overline{Q}^\star(s,a,j) \leq \max_{j \in \mathcal{J}} \overline{Q}^\star(s,a,j) = Q^{\overline{\pi}^\star}(s,a) \leq Q^\star(s,a).$$

Second, we show that $\overline{Q}^\star(s,a,1) \geq Q^\star(s,a)$ which given (5) is enough to show (6). For this let $\pi^\star$ be an optimal policy with respect to the original MDP. We can see $\pi^\star$ as a policy in the skip MDP that always chooses a constant skip length $j = 1$. From there we conclude by

$$\overline{Q}^\star(s,a,1) \geq \overline{Q}^{\pi^\star}(s,a,1) = Q^\star(s,a)$$

.