ADAPTIVE ONLINE PLANNING FOR CONTINUAL LIFELONG LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

We study learning control in an online lifelong learning scenario, where mistakes can compound catastrophically into the future and the underlying dynamics of the environment may change. Traditional model-free policy learning methods have achieved successes in difficult tasks due to their broad flexibility, and capably condense broad experiences into compact networks, but struggle in this setting, as they can activate failure modes early in their lifetimes which are difficult to recover from and face performance degradation as dynamics change. On the other hand, model-based planning methods learn and adapt quickly, but require prohibitive levels of computational resources. Under constrained computation limits, the agent must allocate its resources wisely, which requires the agent to understand both its own performance and the current state of the environment: knowing that its mastery over control in the current dynamics is poor, the agent should dedicate more time to planning. We present a new algorithm, Adaptive Online Planning (AOP), that achieves strong performance in this setting by combining model-based planning with model-free learning. By measuring the performance of the planner and the uncertainty of the model-free components, AOP is able to call upon more extensive planning only when necessary, leading to reduced computation times. We show that AOP gracefully deals with novel situations, adapting behaviors and policies effectively in the face of unpredictable changes in the world – challenges that a continual learning agent naturally faces over an extended lifetime – even when traditional reinforcement learning methods fail.

1 INTRODUCTION

We consider agents in a human-like setting, where agents must simultaneously act and learn in the world continuously with limited computational resources. All decisions are made online; there are no discrete episodes. Furthermore, the world is vast – too large to feasibly explore exhaustively – and changes over the course of the agent's lifetime, like how a robot's actuators might deteriorate with continued use. There are no resets to wipe away past errors. Mistakes are costly, as they compound downstream. To perform well at reasonable computational costs, the agent must utilize its past experience alongside new information about the world to make careful, yet performant, decisions.

Non-stationary worlds require algorithms that are fundamentally robust to changes in dynamics. Factors that would lead to a change in the environment may either be too difficult or principally undesirable to model: for example, humans might interact with the robot in unpredictable ways, or furniture in a robot's environment could be rearranged. Therefore, we assume that the world can change unpredictably in ways that cannot be learned, and focus on developing algorithms that instead handle these changes gracefully, without using extensive computation.

Model-based trajectory optimization via planning is useful for quickly learning control, but is computationally demanding and can lead to bias due to the finite planning horizon. Model-free reinforcement learning is sample inefficient, but capable of cheaply accessing past experience without sacrifices to asymptotic performance. Consequently, we would like to distill expensive experience from an intelligent planner into neural networks to reduce computation for future decision making.

Deciding when to use the planner vs a learned policy presents a difficult challenge, as it is hard to evaluate the improvement the planner would give without actually using the planner. We tackle this as a problem of uncertainty. When uncertain about a course of action, humans use an elongated model-based search to evaluate long-term trajectories, but fall back on habitual behaviors learned with model-free paradigms when they are certain of what to do (Dayan & Berridge, 2014; Daw et al., 2005; Banks & Hope, 2014; Kahneman, 2003). By measuring this uncertainty, we can make informed decisions about when to use model-based planning vs a model-free policy.

Our approach combines model-based planning with model-free policy learning, along with an adaptive computation mechanism, to tackle this setting. Like a robot that is well-calibrated when first coming out of a factory, we give the agent access to a ground truth dynamics model that lacks information about future changes to the dynamics, like different settings in which the robot may be deployed. This allows us to make progress on finite computation continual learning without worrying about model learning. The dynamics model is updated immediately at world changes. However, as we show empirically, knowing the dynamics alone falls far short of success at this task.

We present a new algorithm, Adaptive Online Planning (AOP), that links Model Predictive Path Integral control (MPPI) (Williams et al., 2015), a model-based planner, with Twin Delayed DDPG (TD3) (Fujimoto et al., 2018), a model-free policy learning method. We combine the model-based planning method of iteratively updating planned trajectory with the model-free method of updating the network weights to develop a unified update rule formulation that is amenable to reducing computation when combined with a switching mechanism. We inform this mechanism with the uncertainty given by an ensemble of value functions. Access to the ground truth model is not sufficient by itself, as we show that PPO (Schulman et al., 2017) and TD3 perform poorly, even with the ground truth model. We demonstrate empirically that AOP is capable of integrating the two methodologies to reduce computation while achieving and maintaining strong performance in non-stationary worlds, outperforming other model-based planning methods and avoiding the empirical performance degradation of policy learning methods in changing world scenarios.

Our contributions include the proposal of a new algorithm combining model-based planning and model-free learning, the introduction of evaluation environments that target the challenges of lifelong reinforcement learning in which traditional methods struggle, and experiments showing the usefulness of utilizing both model-based and model-free methods in this setting. Code to run all algorithms and experiments is available at `www.github.com/` (link redacted for anonymity).

2 Background

We consider the world as an infinite-horizon Markov Decision Process (MDP), defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma\}$, where $\mathcal{S}$ is the continuous state space, $\mathcal{A}$ is the continuous action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the deterministic reward function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ are the transition probabilities, and $\gamma \in (0, 1)$ is the discount factor. The world can change over time; i.e. the transitions and rewards $(\mathcal{T}, \mathcal{R})$ may change to some new $(\mathcal{T}', \mathcal{R}')$. Unlike what is commonly done in reinforcement learning, the agent's state is not reset at these world changes. The agent is able to generate rollouts using the current $(\mathcal{T}, \mathcal{R})$ starting from its current state, but not for future $(\mathcal{T}', \mathcal{R}')$, or from arbitrary start states. The agent's goal is to execute a future sequence of actions, summarized as a policy $\pi(a_t|s_t)$, that maximizes the expected future reward $R(t) = \mathbb{E}_{\tau\sim\pi}[\sum_{k=0}^{\infty} \gamma^k r(s_t, a_t)]$.

2.1 Continual Online Learning

In our work, we consider online learning in the same style as Lowrey et al. (2018), where both acting and learning must occur on a per-timestep basis, and there are no episodes that reset the state. We desire agents that are equipped, like humans, to handle different tasks in various environments. Continual learning is difficult, as agents must learn to perform well in new tasks while preserving the ability to perform well in old tasks (backward transfer). Previous experience from old tasks should guide learning for new tasks (forward transfer). In addition to the these difficulties, there is also the challenge of avoiding failure sink states that prevent future learning progress. We augment this task with a world where the dynamics continually change, creating a difficult setting for agent learning.

2.2 Model-Based Planning

Online model-based planning uses a model to evaluate future sequences of actions, developing a projected future trajectory over some time horizon, and then executing the first action of that trajectory, before repeating. We specifically focus on Model Predictive Control (MPC), which iteratively applies Gaussian noise to the prior predicted trajectory, evaluates them using the dynamics model, and combines them with an update rule. When the update rule is a softmax weighting, this procedure is called Model Predictive Path Integral (MPPI) control (Williams et al., 2015). In general, these algorithms are very expensive, but perform well when used on the ground truth model.

2.3 Model-Free Policy Optimization

Model-free algorithms seek to encode the agent's past experiences in a function dependent only on the current state, often in the form of a value function critic and/or policy actor. As a result, such algorithms can have difficulty learning long-term dependencies and struggle early on in training; deep exploration is difficult. In exchange, they attain high asymptotic performance, having shown successes in a variety of tasks for the traditional offline setting (Mnih et al., 2013; Schulman et al., 2017). As a consequence of their compact nature, once learned, these algorithms tend to generate cyclic and regular behaviors, whereas model-based planners have no such guarantees.

We run online versions of TD3 (Fujimoto et al., 2018) and PPO (Schulman et al., 2017) as baselines to AOP. While there is no natural way to give a policy access to the ground truth model, we allow the policies to train on future trajectories generated via the model, in similar fashion to algorithms that learn a model for this purpose (Kurutach et al., 2018; Buckman et al., 2018), in order to help facilitate fair comparisons to model-based planners.

2.4 Update Rule Perspective on Planning vs Policy Optimization

From a high-level perspective, the model-based planning and model-free policy optimization procedures are very similar (see Appendix A for a side-by-side comparison). Where the planner generates noisy rollouts to synthesize a new trajectory, the model-free algorithm applies noise to the policy to generate data for learning. After an update step, either an action from the planned trajectory or one call of the policy is executed. These procedures are only distinct by their respective update rules.

The primary contribution of AOP is unifying both update rules to compensate for their individual weaknesses. AOP distills the learned experience from the planner into the off-policy learning method of TD3 and a value function, so that planning and acting can be done cheaper in the future.

3 Adaptive Online Planning

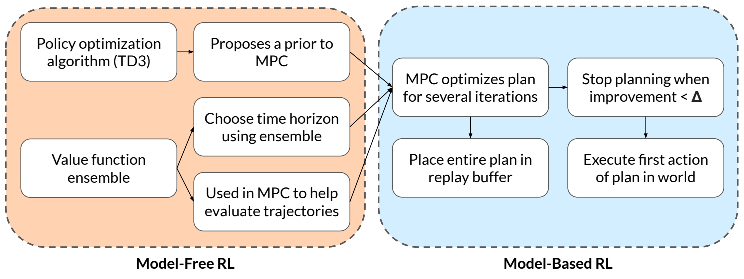

Policy optimization algorithm (TD3)
Proposes a prior to MPC
Choose time horizon using ensemble
Value function ensemble
Used in MPC to help evaluate trajectories
MPC optimizes plan for several iterations
Stop planning when improvement < Δ
Place entire plan in replay buffer
Execute first action of plan in world
Model-Free RL
Model-Based RL

Figure 1: Schematic view of Adaptive Online Planning (AOP).

AOP combines model-based planning with model-free learning using an adaptive switching mechanism, summarized above in Fig. 1, and written with more detail in Appendix A in Alg. 3.

3.1 Model-Based Planning with Terminal Value Approximation

For the model-based component, AOP uses MPPI, as described in Section 2.2, with a terminal value function $\hat{V}$, where trajectories are evaluated in the form of Eq. 1. This process is repeated for several iterations to improve the plan, and then the first action is executed in the environment.

$$R(\tau) = \sum_{k=0}^{H-1} \gamma^k r(s_k, a_k) + \gamma^H \hat{V}(s) \tag{1}$$

$\hat{V}$ is generated by an ensemble of n value functions (see Eq. 2), as proposed in POLO (Lowrey et al., 2018) for MPC. The value ensemble improves the exploration ability of the optimization procedure (Osband et al., 2016; 2018). The log-sum-exp function serves as a softmax, enabling exploration while preserving a stable target for learning. The $\log n$ term normalizes the estimate to lie between the mean and the max of the ensemble, determined by the temperature hyperparameter κ, which ensures that the approximation is still semantically meaningful for estimating the value of the state.

$$\hat{V}(s) = \frac{1}{\kappa} \log \sum_{i=1}^{n} e^{\kappa V_i(s) - \log n} \tag{2}$$

3.2 Early Planning Termination

Past model-based planning procedures (Chua et al., 2018; Wang & Ba, 2019) run a fixed number of iterations of MPC per timestep before executing an action in the environment. However, this is often wasteful. Within a particular timestep, later planning iterations often improve the planned trajectory less than earlier iterations, and may not even improve the trajectory at all.

We propose to decide on the number of planning iterations on a per-timestep basis. After generating a new trajectory τ_{k+1} from the k-th iteration of planning, we measure the improvement $\Delta(\tau_{k+1}|\tau_k)$ against the trajectory τ_k of the previous iteration, as in Eq. 3. When this improvement decreases below a threshold Δ_{thres}, we terminate planning for the current timestep with probability $1 - \epsilon_{plan}$. Using a stochastic termination rule allows for robustness against local minima where more extensive planning may be required, but not obvious from early planning iterations, in order to escape.

$$\Delta(\tau_{k+1}|\tau_k) = \frac{R(\tau_{k+1}) - R(\tau_k)}{|R(\tau_k)|} \tag{3}$$

3.3 Adaptive Planning Horizon

Since planning over a long time horizon is expensive, it would also be desirable to plan over a shorter time horizon when the planner is confident in achieving long-term success with only short-term planning. We represent the planner's uncertainty with the value ensemble from Section 3.1; the mean and standard deviation of the ensemble represent the epistemic uncertainty of the value of the state (Osband et al., 2018). When deciding to use a reduced time horizon, we require that the standard deviation σ of the value ensemble on the current state be lower than some threshold σ_{thres}.

A problem with only considering the standard deviation is that the metric only considers uncertainty in the state with respect to past experiences – it does not immediately measure uncertainty in a changing dynamics setting, which is only observed when considering experiences in the future. Therefore, we fine-tune the horizon length using the Bellman error, as in Eq. 4. The time horizon that will be used for planning is given by the largest $H \leq H_{full}$ such that $\epsilon(H|\tau_k) > \epsilon_{thres}$. Note that when $\sigma > \sigma_{thres}$, the full time horizon is always used, regardless of the Bellman error.

$$\epsilon(H|\tau_{s_t,\ldots,s_{t+H_{full}}}) = (R(\tau_{s_{t+H},\ldots s_{t+H_{full}}}) - \frac{1}{n}\sum_{i=1}^{n} V_i(s_{t+H}))^2 \tag{4}$$

3.4 Off-Policy Model-Free Prior

We use TD3 as a prior to the planing procedure, with the policy learning off of the data generate dby the planner during planning, which allows the agent to recall past experience quickly. Similarly to past work (Rajeswaran et al., 2017a; Zhu et al., 2018), we found that imitation learning caps the

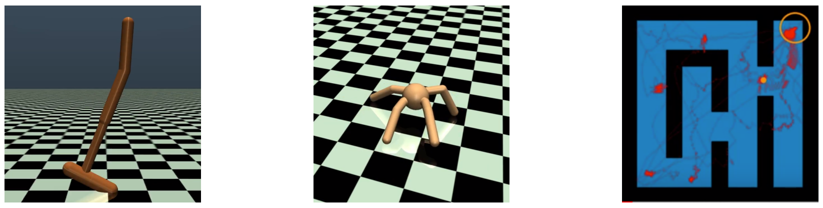


Figure 2: Pictures of lifelong learning environments: Hopper, Ant, and Maze (from left to right).

asymptotic performance of the learned policy. As a baseline, we also run behavior cloning (BC) as a prior, and refer to the resulting algorithms as AOP-TD3 and AOP-BC, respectively.

We note that MPC and policy optimization are both special cases of AOP. MPC is equivalent to AOP with a constant setting for the time horizon that always uses full planning iterations (i.e. a threshold of 0). Policy optimization is equivalent to AOP with one planning iteration, since the first plan is a noisy version of the policy, acting as the data collection procedure in standard policy learning.

4 Empirical Evaluations

We investigate several questions empirically:

1. What are the difficulties of the continual lifelong learning setting?
2. How does AOP perform when acting from well-known states, novel states, and in changing worlds? How do traditional on-policy and off-policy methods fare in these situations?
3. Are the variance and the Bellman error of the value ensemble suitable metrics for determining the planning computational budget?

4.1 Lifelong Learning Environments

We propose three environments to evaluate our proposed algorithm in the continual lifelong learning setting: Hopper, Ant, and Maze. While these environments are not overly complex, they crisply highlight the difficulties of lifelong continual learning in MDPs.

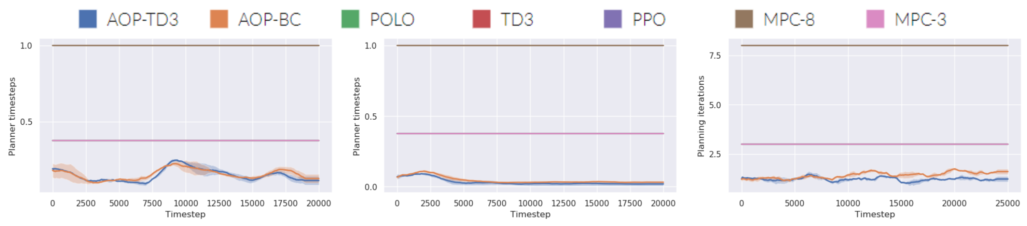
Hopper: First, we consider the OpenAI Gym environment Hopper (Brockman et al., 2016). The agent is rewarded based on how closely it matches an unobserved target velocity. Every 4000 timesteps, this target velocity changes. The environment is tough, as it can be difficult to get up if the agent falls down in a strange position, and momentum from past actions affect the state greatly, which makes it easy for the agent to fall over. We consider three versions of the Hopper environment: (1) a standard Hopper with an unchanging target velocity, (2) a novel states Hopper with the target velocity in the observation (thus new target velocities correspond to the agent seeing a new state), and (3) a changing worlds Hopper, where the target velocity is not in the observation.

Ant: We also consider the Ant from Gym. The agent seeks to maximize its forward velocity, but a joint at random is disabled every 2500 timesteps. Once flipped over, getting back up is extremely difficult, which makes this environment harshly unforgiving. We consider two versions: (1) a standard Ant with no disabled joints, and (2) a changing worlds Ant with one changing disabled joint.

Maze: Like in POLO, we test in a 2D point mass maze. The agent seeks to reach a goal. The observation is $(x_{point}, y_{point}, x_{goal}, y_{goal})$. Every 500 timesteps, the walls of the maze change, and the goal swaps locations. The difficulty lies in adapting quickly to new mazes, while avoiding the negative transfer of old experience. We consider two versions: (1) a novel states Maze, where the walls of the maze remain constant, but new goals are introduced after 20 goal changes in the original positions, and (2) a changing worlds Maze, which is as described above. We also test both versions in a dense reward and a sparse reward setting, where the reward is either the negative L2 distance or a boolean value, respectively. In the sparse reward Maze, exploration can be particularly challenging.

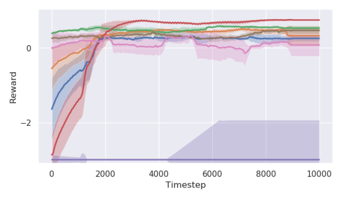
4.2 Baselines and Ablations

We run AOP-BC, POLO, MPC, TD3, and PPO as baselines against AOP-TD3; they can be seen as ablations/special cases of our proposed algorithm (see Section 3.4). We consider two versions of MPC, with 8 and 3 planning iterations, henceforth referred to as MPC-8 and MPC-3, respectively.

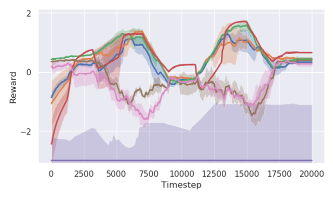


Changing worlds Hopper   Changing worlds Ant   Changing worlds Maze (D)

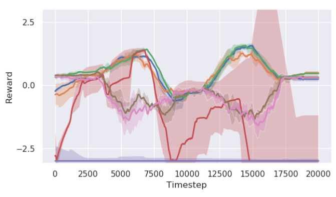
Figure 3: Number of timesteps rolled out by planner per timestep as a percentage of MPC-8 (MPC-based methods only). For Maze, (D) and (S) denote the mazes with dense and sparse rewards, respectively. Results are averaged over 3 seeds each, curves are smoothed for visual clarity, and the shaded area shows one standard deviation above and below the mean.



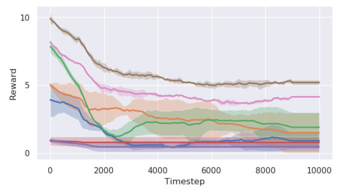
(a) Standard Hopper



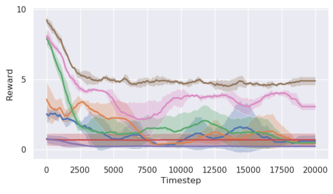
(b) Novel states Hopper



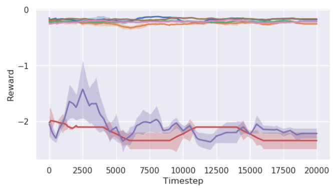
(c) Changing worlds Hopper



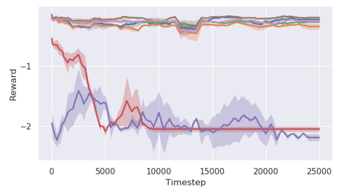
(d) Standard Ant



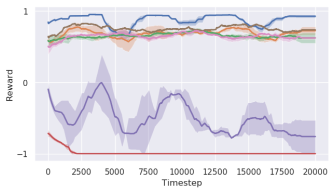
(e) Changing worlds Ant



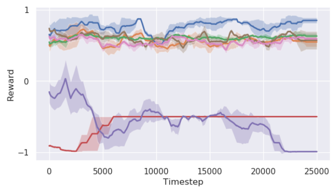
(f) Novel states Maze (D)



(g) Changing worlds Maze (D)



(h) Novel states Maze (S)



(i) Changing worlds Maze (S)

Figure 4: Reward curves for lifelong learning tasks. Rewards are for a single timestep, not over an episode. Note that in the changing worlds environments (esp. Hopper, Ant), some worlds may be more difficult than others, and yield a naturally lower reward. We clip the lower bound of the plotted rewards to be -3 so that the rewards are easier to view.

4.3 Difficulties in Continual Online Learning

Planner usage for the changing world environments is shown in Fig. 3, and rewards for all environments are in Fig. 4. AOP uses only $0 - 25\%$ of the number of timesteps as MPC-8, but achieves generally comparable or stronger performance. In Appendix D, we show the planner usage for all environments in Fig. 9 and compute times for the changing worlds experiments in Fig. 8.

Reset-free Setting: We observe that, even with model access, these environments are challenging for the algorithms to learn. In the standard offline reinforcement learning setting, long-term action dependencies are learned from past experience over time, and this experience can be utilized when the agent resets to the initial state. However, in the online setting, these dependencies must be learned on the fly, and if the agent falls, it must return to the prior state in order to use that information. In the Ant environment, such falling is catastrophic, as it takes a complex action sequence to return to standing. Much of the difficulty of the Hopper and Ant settings is due to this lack of resetting.

Vast Worlds: The performance gain of MPC-8 over MPC-3 shows that, even with the ground truth model, achieving strong performance is difficult with constrained computation. In the sparse mazes (h,i), MPC is significantly outperformed by AOP-TD3, and the model-free algorithms struggle to make any progress at all, showing their lackluster exploration abilities. Even POLO – the exploration mechanism of AOP – faces weaker performance, indicating that AOP-TD3 has not only correctly identified when planning is important, but is able to effectively leverage additional computation to increase its overall performance whilst still using less overall computation.

Policy Degradation: TD3's performance significantly degrades in the changing worlds settings (c,g), and PPO's degrades as well, as seen in (g,h,i). In (b), the variant where the policy is capable of directly seeing the target velocity, TD3 performs very well, even learning to outperform MPC. However, without the help of the observation, in (c), TD3's performance quickly suffers after world changes. The model-based planning methods do not suffer this degradation, and AOP-TD3 is able to maintain its performance and computational savings, even through many world changes (g,i), despite its reliance on model-based components.

4.4 Behavior in Well-Known States/Novel States/Changing Worlds

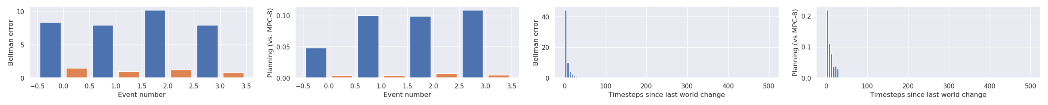


Figure 5: Graphs for Maze (D). From left to right: (1-novel states): average Bellman error of the the first time a goal is presented (blue) vs the last time it is presented (orange). (2-novel states) average number of planning steps. Events denote the 4 times the set of goals switch during the agent's lifetime. (3-changing worlds): average Bellman error by the time since the last world change. (4-changing worlds): average number of planning steps. Both quickly decrease to ≈ 0.

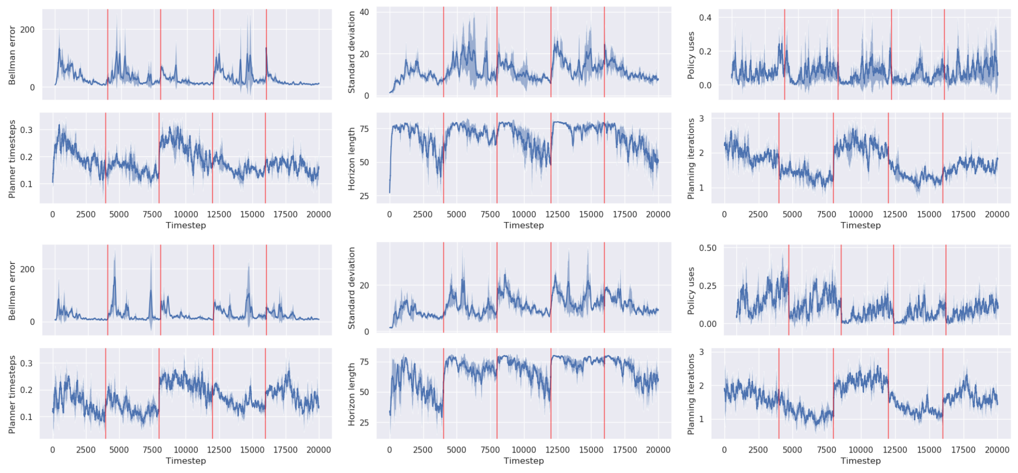


Figure 6: Graphs for Hopper. Red lines denote world changes. Policy uses is the percent of the time that the policy was used instead of the initial plan (see line 3 of Alg. 3). The top two rows show the changing worlds results and the bottom two rows show the corresponding novel states results.

Fig. 5 shows AOP behavior in Maze (D). When encountering novel states, Bellman error is high, but as time progresses, when confronted with the same states again, Bellman error becomes low. The number of planning timesteps matches this: AOP correctly identifies a need to plan early on, but greatly saves computation later, when it is immediately able to know the correct action with almost no planning. The same effect occurs when measuring the time since the world changed for the changing worlds. At the beginning of a new world, the amount of planning is high, before quickly declining to nearly zero, almost running with the speed of a policy: $> 100\times$ faster than MPC-8.

We plot the standard deviation and Bellman error over time of AOP for the changing worlds and novel states Hoppers in Fig. 6. After each world change, the Bellman error spikes, and then decreases as time goes on. We observe a similar overall trend for the standard deviation, especially in the novel states setting, where the spike can be immediate, due to the change being in the observation. These trends are reflected in the time horizon (bottom center), which decreases as the agent trains in each world, and indicate that the standard deviation and Bellman error are suitable metrics for determining planning levels. The same effect also occurs for the number of planning iterations.

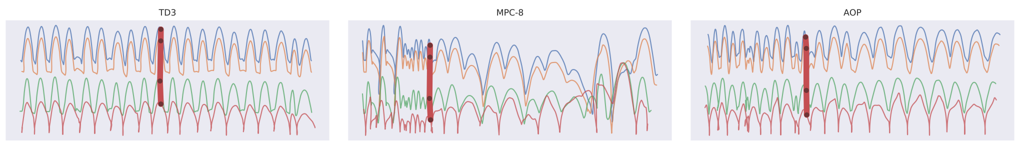


Figure 7: Example trajectory traces of Cartesian hopper positions for different algorithms. An unobserved change in target speed is encountered at the timestep marked by red outline.

In Fig. 7, we show qualitatively how TD3, MPC, and AOP handle world changes in the changing worlds Hopper setting. An unobserved change in the target velocity from 2.5 to 1 is encountered during the agent's lifetime. The purely model-free TD3 produced cyclic and regular behavior, but adapted slowly to the target speed, still moving at 2.5. MPC quickly adapted to the new target speed, but moved in an irregular fashion. AOP was both able to rapidly adapt and do so in a regular manner.

5 Related Work and Future Directions

Continual online learning: Much of past lifelong learning work (Goodfellow et al., 2013; Parisi et al., 2018) has focused on catastrophic forgetting, which is useful in our novel states setting, but does not directly apply to changing worlds. Kearns & Singh (2002) considers MDPs with various reward functions, using them for to make decisions about exploration or exploitation and gathering new experience, similar to our framework. Finn et al. (2019) uses meta-learning to perform continual online learning, where the tasks are considered in episodes. Nagabandi et al. (2018) learn multiple models to represent different tasks, which contrasts with our single policy/set of value functions.

Augmented planners: Algorithms that combine planning with learning have previously been studied in both discrete and continuous domains (Chua et al., 2018; Anthony et al., 2017). Recent work (Guez et al., 2018) generalizes the MCTS algorithm and proposes to learn the algorithm instead, which leads to the efficient selection of promising branches; further asking the algorithm to set computation levels could lead to effective results in our setting as well. Levine & Koltun (2013); Mordatch et al. (2015) propose to use priors that make trajectory planner stay close to policy outputs, which is problematic in changing worlds.

Learned dynamics: Many previous works (Azizzadenesheli et al., 2018; Nagabandi et al., 2017) have learned dynamics models and then performed MPC optimization on them. Kurutach et al. (2018); Clavera et al. (2018) utilized model ensembles to reduce performance degradation due to model overfitting, and (Janner et al., 2019) investigates model uncertainty. Integrating AOP with a learned uncertainty-aware dynamics model would be interesting future work.

6 Conclusion

We proposed AOP, which incorporates model-based planning with model-free learning, and introduced environments for evaluating algorithms in the continual lifelong learning setting. We empirically analyzed the performance of and signals from the the model-free components, and showed experimentally that AOP was able to successfully reduce computation while achieving high performance in difficult tasks, often competitive with a much more powerful MPC procedure.

REFERENCES

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *CoRR*, abs/1705.08439, 2017. URL `http://arxiv.org/abs/1705.08439`.

Kamyar Azizzadenesheli, Brandon Yang, Weitang Liu, Emma Brunskill, Zachary C. Lipton, and Animashree Anandkumar. Sample-efficient deep RL with generative adversarial tree search. *CoRR*, abs/1806.05780, 2018. URL `http://arxiv.org/abs/1806.05780`.

Adrian Banks and Christopher Hope. Heuristic and analytic processes in reasoning: An event-related potential study of belief bias. *Psychophysiology*, 51, 03 2014. doi: 10.1111/psyp.12169.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL `http://arxiv.org/abs/1606.01540`.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *CoRR*, abs/1807.01675, 2018. URL `http://arxiv.org/abs/1807.01675`.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *CoRR*, abs/1805.12114, 2018. URL `http://arxiv.org/abs/1805.12114`.

Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *CoRR*, abs/1809.05214, 2018. URL `http://arxiv.org/abs/1809.05214`.

Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704, 2005.

Peter Dayan and Kent C Berridge. Model-based and model-free pavlovian reward learning: Revaluation, revision, and revelation. 2014.

Chelsea Finn, Aravind Rajeswaran, Sham M. Kakade, and Sergey Levine. Online meta-learning. *CoRR*, abs/1902.08438, 2019. URL `http://arxiv.org/abs/1902.08438`.

Scott Fujimoto, Herke van Hoof, and Dave Meger. Addressing function approximation error in actor-critic methods. *CoRR*, abs/1802.09477, 2018. URL `http://arxiv.org/abs/1802.09477`.

Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013.

Arthur Guez, Théophane Weber, Ioannis Antonoglou, Karen Simonyan, Oriol Vinyals, Daan Wierstra, Rémi Munos, and David Silver. Learning to search with mctsnets. *CoRR*, abs/1802.04697, 2018. URL `http://arxiv.org/abs/1802.04697`.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019. URL `http://arxiv.org/abs/1906.08253`.

Daniel Kahneman. Maps of bounded rationality: Psychology for behavioral economics . *American Economic Review*, 93:1449–1475, 02 2003. doi: 10.1257/000282803322655392.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, Nov 2002. ISSN 1573-0565. doi: 10.1023/A:1017984413808. URL `https://doi.org/10.1023/A:1017984413808`.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *CoRR*, abs/1802.10592, 2018. URL `http://arxiv.org/abs/1802.10592`.

Sergey Levine and Vladlen Koltun. Guided policy search. In *International Conference on Machine Learning*, pp. 1–9, 2013.

Kendall Lowrey, Aravind Rajeswaran, Sham M. Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *CoRR*, abs/1811.01848, 2018. URL http://arxiv.org/abs/1811.01848.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.

Igor Mordatch, Kendall Lowrey, Galen Andrew, Zoran Popovic, and Emanuel V Todorov. Interactive control of diverse complex characters with neural networks. In *Advances in Neural Information Processing Systems*, pp. 3132–3140, 2015.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *CoRR*, abs/1708.02596, 2017. URL http://arxiv.org/abs/1708.02596.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based RL. *CoRR*, abs/1812.07671, 2018. URL http://arxiv.org/abs/1812.07671.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. *CoRR*, abs/1602.04621, 2016. URL http://arxiv.org/abs/1602.04621.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8617–8629. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8080-randomized-prior-functions-for-deep-reinforcement-learning.pdf.

German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569, 2018. URL http://arxiv.org/abs/1802.07569.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017a.

Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. *CoRR*, abs/1703.02660, 2017b. URL http://arxiv.org/abs/1703.02660.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Tingwu Wang and Jimmy Ba. Exploring model-based planning with policy networks. *CoRR*, abs/1906.08649, 2019. URL http://arxiv.org/abs/1906.08649.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *CoRR*, abs/1509.01149, 2015. URL http://arxiv.org/abs/1509.01149.

Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.

A Algorithm Pseudocode

Algorithm 1: Model-Based Planning

```
1 Initialize action trajectory $\tau_{plan}$
2 while alive do
3     Generate $n$ rollouts based on $\tau_{plan}$
4     Use rollouts to update $\tau_{plan}$
5     Execute first action of $\tau_{plan}$
6 end
```

Algorithm 2: Policy Optimization

```
1 Initialize policy $\pi_\phi$
2 while alive do
3     Generate $n$ rollouts based on $\pi_\phi$
4     Use rollouts to update $\pi_\phi$
5     Execute an action from $\pi_\phi$
6 end
```

Algorithm 3: Adaptive Online Planning

```
1  Initialize value ensemble $\{V_i\}_{i=1}^n$, policy $\pi$, replay buffers $\mathcal{D}_V, \mathcal{D}_\pi$
2  while alive do
3      Set $\tau_{plan}$ to $\arg\max_{\tau_\pi, \tau_{plan}} R(\tau)$, generating $\tau_\pi$ from the policy $\pi_\theta$
4      Select time horizon $H_t$ for planning, described in Section 3.3
5      for $k \leftarrow 1$ to $max\_iters$ do
6          Run MPC planning to generate $\tau_{plan}$, described in Section 3.1; add trajectories to $\mathcal{D}_\pi$
7          If $\Delta(\tau_{k+1}|\tau_k) < \Delta_{thres}$, stop planning with probability $1 - \epsilon_{plan}$
8      end
9      If it's time to update: update value ensemble $\{V_i\}_{i=1}^n$ and policy $\pi_\theta$ with $\mathcal{D}_V$, $\mathcal{D}_\pi$, resp.
10     Step once in the environment with first action of $\tau_{plan}$; add $(s_t, a_t, s'_t, r_t)$ to $\mathcal{D}_V$
11 end
```

B Hyperparameters

For AOP, we set $\sigma_{thres} = 8$, $\epsilon_{thres} = 25$ and $\epsilon_{plan} = 0.2$. We did not tune these hyperparameters much, and similarly do not believe that the algorithm is overly sensitive to the thresholds in dense reward environments. However, in the sparse Mazes, we set $\sigma_{thres} = \epsilon_{thres} = 0$, in order to avoid early termination of exploration (we do not change the hyperparameters determining the number of planning iterations). Tuning over these hyperparameters (for both dense and sparse rewards) could lead to better performance, if desired.

For the first planning iteration we set $\Delta_{thres} = 0.01$, and for the later planning iterations, $\Delta_{thres} = 0.05$. We found that having a lower threshold for the first iteration helps the agent to avoid getting stuck in poor trajectories (i.e. avoid only using the policy), alongside the stochastic decision rule.

In all MPC-based methods, our MPPI temperature is set to $\lambda = 0.01$. The other planning hyperparameters vary as follows (constant across all environments):

Table 1: Planning hyperparameters

Parameter	AOP-TD3/AOP-BC	POLO	TD3	PPO	MPC
Planning horizon	$1-80$	80	256	120	80
Planning iterations per timestep	$0-8$	3	1	2	$3, 8$
Trajectories per iteration	40	40	1	16	40
Noise standard deviation	0.1	0.1	0.2	-	0.1

For our value ensembles, we use an ensemble size of 6 and $\kappa = 10^{-2}$. For the randomized priors, we use $\beta_{prior} = 1$ and a noise of 0. The value functions are updated in batches of size 32 for 32 gradient steps every 4 timesteps. All networks use tanh activations and a learning rate of 10^{-3}, trained using the Adam optimizer with default parameters.

Our TD3 uses the same hyperparameters as the original authors (Fujimoto et al., 2018). Our PPO uses $\epsilon = 0.2$, $KL_{targ} = 3 \times 10^{-3}$, $\lambda = 0.95$, batch sizes of 256, and 2 actor steps/40 critic steps per

Table 2: Network sizes

Environment	Network	AOP-TD3/AOP-BC	POLO	TD3	PPO
Hopper	Critic	$V : (64, 64), Q : (400, 300)$	$(64, 64)$	$(400, 300)$	$(128, 128)$
Hopper	Actor	$(400, 300)$	-	$(400, 300)$	$(128, 128)$
Ant	Critic	$V : (64, 64), Q : (400, 300)$	$(64, 64)$	$(400, 300)$	$(128, 128)$
Ant	Actor	$(400, 300)$	-	$(400, 300)$	$(128, 128)$
Maze	Critic	$(64, 64)$	$(64, 64)$	$(64, 64)$	$(64, 64)$
Maze	Actor	$(64, 64)$	-	$(64, 64)$	$(64, 64)$

iteration. For behavior cloning, we run 400 gradient steps on batches of size 64 every 4 timesteps; for the policy in AOP-TD3, we run 128 gradient steps on batches of size 100 every 4 timesteps.

Our implementations are available at `www.github.com/` (link redacted for anonymity).

C Environment Details

In the online setting, the agent receives no signal from termination states, i.e. it becomes more difficult to know not to fall down in the cases of Hopper and Ant. To amend this, and achieve similar behavior as the standard reinforcement learning setting, we set the reward functions as the following for our environments, based off of Rajeswaran et al. (2017b):

Environment	Reward Function
Hopper	$\lvert x\_vel - x\_vel_{targ}\rvert + 5(z - 1.8)^2 + .1 \left\lVert a \right\rVert_2^2 + x\_vel_{targ}$
Ant	$x\_vel + 3(z - .9)^2 + .01 \left\lVert a \right\rVert_2^2$
Maze (Dense)	$- \left\lVert (x, y) - (x, y)_{goal} \right\rVert_2 - \mathbf{1}\{\text{contact with wall}\}$
Maze (Sparse)	$\mathbf{1}\{\text{inside goal}\} - \mathbf{1}\{\text{contact with wall}\}$

D Extra Graphs

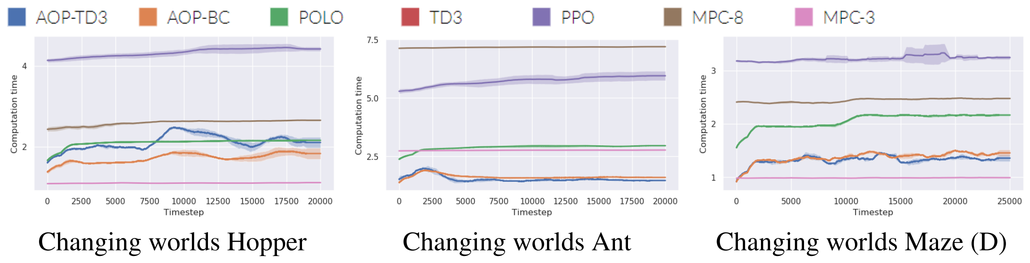
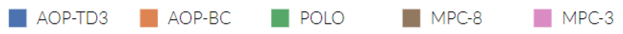
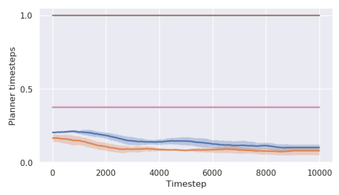
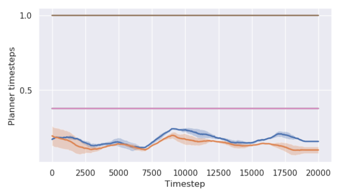


Figure 8: Wall-clock computation time per timestep for changing worlds environments. All experiments (except TD3) were run on CPU only for consistency; TD3 is not shown, as it is too slow with only access to CPU.

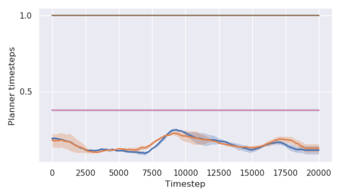




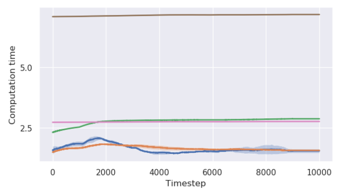
(a) Standard Hopper



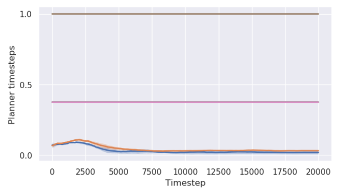
(b) Novel states Hopper



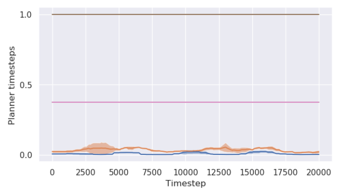
(c) Changing worlds Hopper



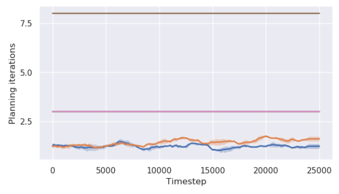
(d) Standard Ant



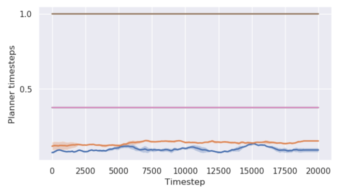
(e) Changing worlds Ant



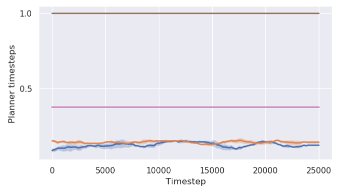
(f) Novel states Maze (D)



(g) Changing worlds Maze (D)



(h) Novel states Maze (S)



(i) Changing worlds Maze (S)

Figure 9: Number of timesteps rolled out by planner per timestep as a percentage of MPC-8.