Combining Spatial and Temporal Abstraction in Planning for Better Generalization

Mingde Zhao[1,3], **Safa Alver**[1,3], **Harm van Seijen**[4], **Romain Laroche**, **Doina Precup**[1,3,5], **Yoshua Bengio**[2,3]
[*1]McGill University, [2]Université de Montréal, [3]Mila, [4]Sony AI, [5]Google DeepMind
{mingde.zhao,safa.alver}@mail.mcgill.ca, harm.vanseijen@sony.com
romain.laroche@gmail.com, dprecup@cs.mcgill.ca, yoshua.bengio@mila.quebec

Abstract

Inspired by human conscious planning, we propose ***Skipper***, a model-based reinforcement learning agent that utilizes spatial and temporal abstractions to generalize learned skills in novel situations. It automatically decomposes the task at hand into smaller-scale, more manageable subtasks and hence enables sparse decision-making and focuses its computation on the relevant parts of the environment. This relies on the definition of a high-level proxy problem represented as a directed graph, in which vertices and edges are learned end-to-end using hindsight. Our theoretical analyses provide performance guarantees under appropriate assumptions and establish where our approach is expected to be helpful. Generalization-focused experiments validate ***Skipper***'s significant advantage in zero-shot generalization, compared to existing state-of-the-art hierarchical planning methods.

1 Introduction

By making use of imagination and intuition, human conscious planning breaks down long-horizon tasks into more manageable abstract steps, each of which can be narrowed down further. This type of planning attends to important decision points [43] and relevant environmental factors linking the decision points [44], thus operating abstractly both in time and in space [14]. In contrast, existing RL agents either operate solely based on intuition (model-free methods) or are limited to reasoning over mostly shortsighted plans (model-based methods). The intrinsic limitations constrain the application of RL in real-world under a glass ceiling formed by challenges of longer-term generalization, below the level of human conscious reasoning.

In this paper, we leverage these intuitions to develop a planning agent that automatically decomposes the complex task at hand into smaller subtasks, by constructing high-level abstract "proxy" problems. A proxy problem is represented as a graph where 1) the vertices consist of states proposed by a generative model, which represent sparse decision points; and 2) the edges, which define temporally-extended transitions, are constructed by focusing on a small amount of relevant information from the states, using an attention mechanism. Once a proxy problem is constructed and the agent solves it to form a plan, each of the edges defines a new sub-problem, on which the agent will focus next. This divide-and-conquer strategy allows constructing partial solutions that generalize better to new situations, while also giving the agent flexibility to construct abstractions necessary

[*]Work largely done during Mingde, Harm and Romain's time at MSR Montreal. Source code of experiments available at https://github.com/PwnerHarry/Skipper

for the problem at hand. Our theoretical analysis establishes guarantees on the quality of the solution to the overall problem.

We also examine empirically whether out-of-training-distribution generalization can be achieved through our method after using only a few training tasks. We show through detailed controlled experiments that the proposed agent, which we name ***Skipper***, performs significantly better in terms of zero-shot generalization, compared to the baselines and to state-of-the-art Hierarchical Planning (HP) methods [33, 18].

2 Preliminaries

Reinforcement Learning & Problem Setting. An RL agent interacts with an environment through a sequence of actions to maximize its cumulative reward. The interaction is usually modeled as a Markov decision process (MDP) $\mathcal{M} \equiv \langle \mathcal{S}, \mathcal{A}, P, R, d, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the set of states and actions, $P : \mathcal{S} \times \mathcal{A} \to \text{Dist}(\mathcal{S})$ is the state transition function, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function, $d : \mathcal{S} \to \text{Dist}(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1]$ is the discount factor. The agent needs to learn a policy $\pi : \mathcal{S} \to \text{Dist}(\mathcal{A})$ that maximizes the value function, *i.e.* the expected discounted cumulative reward $\mathbb{E}_{\pi,P}[\sum_{t=0}^{T_\perp} \gamma^t R(S_t, A_t, S_{t+1}) | S_0 \sim d]$, where $T_\perp$ denotes the time step at which the episode terminates. A value estimator $Q : \mathcal{S} \times |\mathcal{A}| \to \mathbb{R}$ can be used to guide the search for a good policy. However, real-world problems are typically partially observable, meaning that at each time step t, after taking an action $a_t \in \mathcal{A}$, the agent receives an observation $x_{t+1} \in \mathcal{X}$, where $\mathcal{X}$ is the observation space. The agent then needs to infer the state from its sequence of observations, which is usually done through a state encoder.

One important goal of RL is to achieve high (generalization) performance on evaluation tasks after learning from a limited number of training tasks, where the evaluation and training distributions may differ; For instance, a policy for a robot may need to be trained in a simulated environment for safety reasons, but would need to be deployed on a physical device, a setting called sim2real. Discrepancy between task distributions is often recognized as a major reason why RL agents are yet to be applied pervasively in the real world [21]. To address this issue, in this paper, agents are trained on a small set of fixed training tasks, then evaluated in unseen tasks, where there are environmental variations, but the core strategies needed to finish the task remains consistent, for example because of the existence of causal mechanisms. To generalize well, the agents need to build learned skills which capture the consistent knowledge across tasks.

Deep Model-based RL. Deep model-based RL uses approximations to the transition and reward functions in order to guide the search for a good policy [39, 38]. Intuitively, rich models, expressed by Neural Networks (NNs), have the ability to capture generalizable information and possibly to infer latent causal structure. *Background* planning agents *e.g.*, Dreamer [19] use their model as a data generator to improve their value estimators and policies, which they execute while interacting with the environment [41]. Thus, these agents usually do not improve on the trained policy at test time. In contrast, *decision-time* planning agents *e.g.*, MuZero [38] actively use their model at decision time to make better decisions. Recent work suggests that this approach may provide better generalization [1].

Options & Goal-Conditioned RL. Temporal abstraction allows RL agents to use sub-policies, and to model the environment over extended time scales, in order to enable both better generalization and solving larger problems. Options and their models provide a formalism for temporal abstraction in RL [43]. Each option consists of an initiation condition, a policy, and a termination condition. For any set of options defined on an MDP, the decision process that selects only among those options, executing each to termination, is a Semi-MDP (SMDP) [35], consisting of the set of states $\mathcal{S}$, the set of options $\mathcal{O}$, and for each state-option pair, an expected return, and a joint distribution of the next state and transit time. In this paper, we focus on goal-conditioned options, where the initiation set covers the whole state space $\mathcal{S}$. Each such option is a tuple $o = \langle \pi, \beta \rangle$, where $\pi : \mathcal{S} \to \text{Dist}(\mathcal{A})$ is the (intra-)option policy and $\beta : \mathcal{S} \to \{0, 1\}$ indicates when a goal state is reached. Hindsight Experience Replay (HER) [3] is often used to train goal-conditioned options by sampling

a transition $\langle x_t, a_t, r_{t+1}, x_{t+1}\rangle$ together with an additional observation $x^{\odot}$ from the same trajectory, which is re-labelled as a "goal".

3 *Skipper*: Spatially & Temporally Abstract Planning

In this section we describe the main ingredients of ***Skipper*** - an agent that formulates a simplified **proxy** problem for a given task, solves this problem, and then proceeds to "fill in" the details of the plan.

3.1 Proxy Problems

Proxy problems are finite graphs constructed at decision-time, whose vertices are states and whose directed edges are estimated possible transitions between the vertices, as shown in Fig. 1. We call the states selected to be vertices of the proxy problems *checkpoints* to differentiate them from other states are not involved in the decision time planning process. The current state is always one of the vertices. The checkpoints are proposed by a generative model and represent a finite subset of states that the agent might experience in the current episode. Each edge is annotated with estimates of the duration and reward associated with the transition between the checkpoints it connects; these estimates are learned over the relevant aspects of the environment and depend on the agent's capability. As the low-level policy that implements checkpoint transitions improves, the edges strengthen, in the sense that shorter or higher-reward paths may be discovered. Planning in a proxy problem is temporally abstract, because the checkpoints represent sparse decision points. Estimating each checkpoint transition is spatially abstract, as an option corresponding to such a task would base its decisions only on some aspects of the environment state [6], in order to improve generalization as well as computational efficiency [49].

Note that a proxy problem can be viewed as a deterministic SMDP, where each edge will be implemented as a goal-conditioned option, aiming to reach the end checkpoint of the edge. Thus, it can be fully described by the discount and reward matrices, Γ^π and V^π, where γ^π_{ij} and v^π_{ij} are defined as:

$$\gamma^\pi_{ij} \doteq \mathbb{E}_\pi \left[\gamma^{T_\perp} | S_0 = s_i, S_{T_\perp} = s_j\right] \tag{1}$$

$$v^\pi_{ij} \doteq \mathbb{E}_\pi \left[\textstyle\sum_{t=0}^{T_\perp} \gamma^t R_t | S_0 = s_i, S_{T_\perp} = s_j\right]. \tag{2}$$

By planning with Γ^π and V^π, *e.g.* using SMDP value iteration [43], we can solve the proxy problem, and form a jumpy plan to travel between states in the original problem. If the proxy problems can be estimated well, the obtained solution will be of good quality, as established in the following theorem:

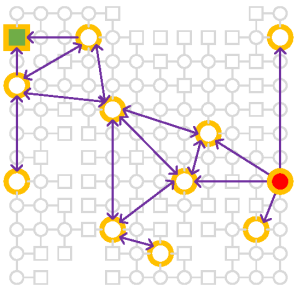


Figure 1: **A Proxy Problem on a Grid-World Navigation Task**: the MDP of the original problem is in gray; terminal states are marked with squares. An agent needs to get from the (filled red) position, to the goal (filled green). The proxy problem has 12 checkpoints (enlarged and outlined orange). The agent's plan corresponds to reaching a series of distant states to get closer to the goal.

Theorem 1 *Let μ be the SMDP policy (high-level) and π be the low-level policy. Let $\hat{V}^\pi$ and $\hat{\Gamma}^\pi$ denote learned estimates of the SMDP model. If the estimation accuracy satisfies:*

$$|v^\pi_{ij} - \hat{v}^\pi_{ij}| < \epsilon_v v_{max} \ll (1-\gamma) v_{max} \qquad \textbf{\textit{and}} \tag{3}$$

$$|\gamma^\pi_{ij} - \hat{\gamma}^\pi_{ij}| < \epsilon_\gamma \ll (1-\gamma)^2 \qquad \forall i, j.$$

Then, the estimated value of the composite $\hat{v}_{\mu\circ\pi}(s)$ is accurate up to error terms linear in ϵ_v and ϵ_γ:

$$\hat{v}_{\mu\circ\pi}(s) \doteq \sum_{k=0}^{\infty} \hat{v}_\pi(s^\odot_k | s^\odot_{k+1}) \prod_{\ell=0}^{k-1} \hat{\gamma}_\pi(s^\odot_\ell | s^\odot_{\ell+1}) = v_{\mu\circ\pi}(s) \pm \frac{\epsilon_v v_{max}}{1-\gamma} \pm \frac{\epsilon_\gamma v_{max}}{(1-\gamma)^2} + o(\epsilon_v + \epsilon_\gamma)$$

where $\hat{v}_\pi(s_i|s_j) \equiv \hat{v}^\pi_{ij}$ and $\hat{\gamma}_\pi(s_i|s_j) \equiv \hat{\gamma}^\pi_{ij}$, and v_{max} denotes the maximum value of $v_{\mu\circ\pi}$, the true value of $\mu \circ \pi$.

The theorem indicates that once the agent achieves high accuracy estimation of the model for the proxy problem and a near-optimal lower-level policy π, its performance becomes close to the optimal $v_{\mu \circ \pi}$ (proof in Appendix I.2). Although the theorem is general, in the experiments, we limit ourselves to navigation tasks with sparse rewards for reaching goals, where the goals are included as permanent vertices in the proxy problem. This is a case where the accuracy assumption can be met non-trivially, *i.e.*, while avoiding degenerate proxy problems whose edges involve no rewards. The theorem also makes no assumption on π because it would likely be difficult to learn a good π for far away targets. Following Thm. 1's guidance, we train estimators for v_π and γ_π and refer to this task as *edge estimation*.

3.2 Design Choices

To implement planning over proxy problems, our framework embraces the following design choices:

Spatio-temporal abstraction: temporal abstraction allows us to break down the given task into smaller ones, while spatial abstraction over the state features through an attention mechanism is used to improve local learning and generalization;

Decision-time planning is employed due to its ability to improve the policy in novel situations;

Learning end-to-end from hindsight, off-policy: to maximize sample efficiency and the ease of training, we propose to use auxiliary (off-)policy methods for edge estimation, and learn a context-conditioned checkpoint generation, both from hindsight experience replay;

Higher quality proxies: we introduce pruning techniques to improve the sparsity of the proxy problems, which leads to better quality;

Delusion suppression: we propose a delusion suppression technique to minimize the behavior of chasing non-existent outcomes. This is done by exposing the edge estimators to targets that would otherwise not exist in experience.

3.3 Problem 1: Edge Estimation

First, we discuss how to estimate the edges of the proxy problem, given a set of already generated checkpoints. Taking inspiration from conscious information processing in brains, we introduce a local perceptive field selector, σ, consisting of a learned attention bottleneck that (soft-)selects the top-k local segments of the full state (*e.g.* a feature map by a typical convolutional encoder); all segments of the state compete for the k attention slots, *i.e.* relevant aspects of states are promoted, and irrelevant ones discarded, to form a partial state representation [29, 44, 49, 2]. We provide an example in Fig. 2 (see purple parts). On top of σ, the auxiliary estimators, to be discussed soon, force the bottleneck mechanism to promote aspects relevant to the local estimation of connections between the checkpoints. The rewards and discounts are then estimated on top of the partial state $\sigma(S)$, based on the agent's behavior.

3.3.1 Basis for Connections: Checkpoint-Achieving Policy

The low-level policy π maximizes an intrinsic reward, *s.t.* the target checkpoint $S^\odot$ can be reached. The choice of intrinsic reward is flexible; for example, one could use a reward of +1 when S_{t+1} is within a small radius of $S^\odot$ according to some distance metric, or use reward-respecting intrinsic rewards that enable more sophisticated behaviors, as in [42]. In the following, for simplicity, we will denote the checkpoint-achievement condition with equality: $S_{t+1} = S^\odot$.

3.3.2 Estimate Connections

The following estimates are learned with using distributional RL, where the output of each estimator takes the form of a histogram over scalar support [13].

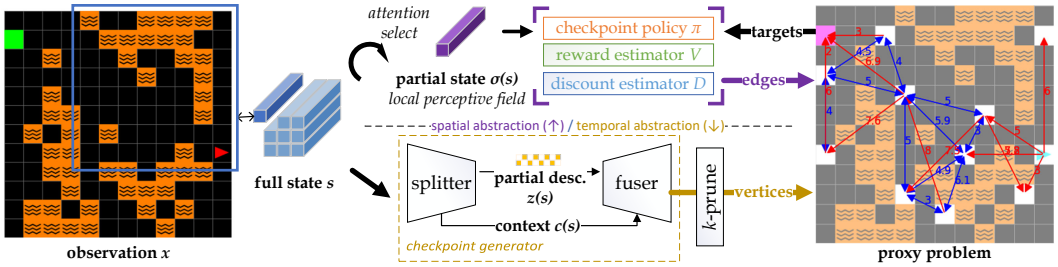


Figure 2: ***Skipper* Framework**: 1) Partial states consist of a few local fields, soft-selected via top-k attention [17]. ***Skipper***'s edge estimations as well as low-level behaviors π are solely based on the partial states. 2) The checkpoint generator learns by splitting the full state into context and partial descriptions, and fusing them to reconstruct the input. It generates checkpoints by sampling partial descriptions and combines them with the episodic contexts; 3) We prune the vertices and edges of the denser graphs to extract sparse proxy problems. Once a plan is formed, the immediate checkpoint target is used to condition the policy, which then guides the actions. In the proxy problem example, blue edges are estimated to be bidirectional and red edges are unidirectional (with the other direction pruned).

Cumulative Reward. The cumulative discounted task reward v^{π}_{ij} is learned using policy evaluation on an auxiliary reward that is the same as the original task reward everywhere except when reaching the target. Given a hindsight sample $\langle x_t, a_t, r_{t+1}, x_{t+1}, x^{\odot}\rangle$ and the corresponding encoded sample $\langle s_t, a_t, r_{t+1}, s_{t+1}, s^{\odot}\rangle$, we train V_π with KL-divergence as follows:

$$\hat{v}_\pi(\sigma(s_t), a_t|\sigma(s^{\odot})) \leftarrow \begin{cases} R(s_t, a_t, s_{t+1}) + \gamma \hat{v}_\pi(\sigma(s_{t+1}), a_{t+1}|\sigma(s^{\odot})) & \text{if } s_{t+1} \neq s^{\odot} \\ R(s_t, a_t, s_{t+1}) & \text{if } s_{t+1} = s^{\odot} \end{cases} \tag{4}$$

where $\sigma(s)$ is the spatially abstracted partial state from the full state s and $a_{t+1} \sim \pi(\cdot|\sigma(s_{t+1}), \sigma(s^{\odot}))$.

Cumulative Distances / Discounts. Similarly to V_π, we would want to know the cumulative discount leading to the target $s_{\odot}$ under π. Unfortunately, this quantity is difficult to learn, since the prediction would be heavily skewed towards 1 if $\gamma \approx 1$. Yet, we can instead effectively estimate cumulative (truncated) distances (or trajectory length) under π. Such distances can be learned with policy evaluation, where the auxiliary reward is +1 on every transition, except at the targets:

$$D_\pi(\sigma(s_t), a_t|\sigma(s^{\odot})) \leftarrow \begin{cases} 1 + D_\pi(\sigma(s_{t+1}), a_{t+1}|\sigma(s^{\odot})) & \text{if } s_{t+1} \neq s^{\odot} \\ 1 & \text{if } s_{t+1} = s^{\odot} \\ \infty & \text{if } s_{t+1} \text{ is terminal and } s_{t+1} \neq s^{\odot} \end{cases}$$

where $a_{t+1} \sim \pi(\cdot|\sigma(s_{t+1}), \sigma(s^{\odot}))$. The cumulative discount can then be recovered by replacing the support of the quantized distribution of distances with the corresponding discounts. The learned distance is also used to prune unwanted checkpoints in order to simplify the proxy problem, as well as prune far-fetched edges. The details of pruning will be presented shortly.

Please refer to the Appendix I.1 for a discussion of the properties of the proposed learning rules for $\hat{v}_\pi$ and $\hat{\gamma}_\pi$.

3.4 Problem 2: Vertex Generation

The checkpoint generator aims to directly model the possible future states *without needing to know how exactly the agent might reach them*. The details of checkpoint transitions will be abstracted by the connection estimates instead.

To make the checkpoint generator generalize well across diverse tasks, while still being able to capture the underlying causal mechanisms in the environment (a tall order for existing model-based methods), we propose that the checkpoint generator learns to split the state representation into two parts: an episodic context and a partial description. In a navigation problem, for example, as in Fig. 2, a context could be a representation of the map of a gridworld, and the partial description be the 2D-coordinates of the agent's location. In

different contexts, the same partial description could correspond to very different states. Yet, within the same context, we should be able to recover the same state given the same partial description.

As shown in Fig. 2, this information split is achieved using two functions: the *splitter* $\mathcal{E}_{CZ}$, which maps the input state S into a representation of a context $c(S)$ and a partial description $z(S)$, as well as the *fuser* $\bigoplus$ which, when applied to the input $\langle c, z \rangle$, recovers S. In order to achieve consistent context extraction across states in the same episode, at training time, we force the context to be extracted from other states in the same episode, instead of the input.

We sample in hindsight a diverse distribution of target encoded (full) states $S^{\odot}$, given any current S_t. Hence, we use as generator a conditional Variational AutoEncoder (VAE) [40] which learns a distribution $p(S^{\odot}|C(S_t)) = \sum_z p(S^{\odot}|C(S_t), z)p(z|C(S_t))$, where $C(S_t)$ is the extracted context from S_t and zs are the partial descriptions. We train the generator by minimizing the evidence lower bound on $\langle S_t, S^{\odot} \rangle$ pairs chosen with HER.

Similarly to [19], we constrain the partial description encoding to a bundle of binary variables and train them with the straight-through gradient estimator [7]. These discrete latents can be easily sampled or composed to generate checkpoints.

Compared to existing models such as in Director [18], which generates intermediate goals given the on-policy trajectory, our method generates a diverse distribution of states, which can be used to plan in novel scenarios.

3.4.1 Pruning

In this paper,we limit ourselves only to checkpoints from a return-unaware conditional generation model, leaving the question of how to improve the quality of the generated checkpoints for future work. Without learning, the proxy problem can be improved by making it more sparse, and making the proxy problem vertices more evenly spread in state space. To achieve this, we propose a pruning algorithm based on k-medoids clustering [22], which requires pairwise distance estimates between states. During proxy problem construction, we first sample a larger number of checkpoints, and then cluster them and select the centers (which are always real states).

Notably, for sparse reward tasks, the generator cannot guarantee the presence of the rewarding checkpoints in the proposed proxy problem. We could remedy this by explicitly learning the generation of the rewarding states with another conditional generator. These rewarding states should be kept as vertices (immune from pruning).

In addition to pruning the vertices, we also prune the edges according to a distance threshold, *i.e.*, all edges with estimated distance over the threshold are deleted from the complete graph of the pruned vertices. This biases potential plans towards shorter-length, smaller-scale sub-problems, as far-away checkpoints are difficult for π to achieve.

3.4.2 Safety & Delusion Control

Model-based HRL agents can be prone to blindly optimizing for objectives without understanding the consequences. We propose a technique to suppress delusions by exposing the edge estimators to potentially delusional targets that do not exist in the experience replay buffer. Details and examples are provided in the Appendix.

3.5 Overall Framework & Training

The overall method and its implementation details are presented in Appendix H. The estimators are trained with KL-divergence and equal weighting of all terms, and the generator is trained with a standard VAE loss. The overall training loss is a simple sum of the KL-divergences and the VAE loss. Details of ***Skipper*** can be found in the Appendix.

4 Experiments

As introduced in Sec. 2, our first goal is to test the zero-shot generalization ability of trained agents. In order to fully understand the results, it is necessary to have precise control of the difficulty of the training and evaluation environments. Also, to validate if the empirical performance of our agents matches the formal analyses (Thm. 1), we need to know how close to the (optimal) ground truth our edge estimations and checkpoint policies are. These goals give rise to the need to use environments whose ground truth information (optimal policies, true distances between checkpoints, etc) can be computed. Thus, we base our experimental setting on the MiniGrid-BabyAI framework [10, 9, 20]. Specifically, we build on the environments and experiment settings used in [49, 1]: the agent needs to navigate to the goal from its initial state in gridworld environments filled with terminal lava traps generated randomly according to a difficulty parameter, which controls their density. During evaluation, the agent is always spawned at the opposite side from the goals; During training, the agent's position is uniformly initialized to speed up training. We provide results for non-uniform training initialization in the Appendix.

These fully observable tasks focus on the challenge of reasoning over causal mechanisms instead of representation learning from observations, which is not the priority of this work. Across all experiments, we sample training tasks from an environment distribution of difficulty 0.4: each cell in the field has probability 0.4 to be filled with lava while guaranteeing a viable path from the initial position to the goal. The evaluation tasks are sampled from increasing OOD difficulties of 0.25, 0.35, 0.45 and 0.55, where the difficulty of training tasks acts as a median. In order to step up the long(er) term generalization difficulty compared to existing work, we showcase experiments done on large, 12×12 maze sizes, (see the visualization in Fig 2). The agents are trained for 1.5×10^6 agent-environment interactions.

We compare ***Skipper*** against two state-of-the-art Hierarchical Planning (HP) methods: LEAP [33] and Director [18]. The comparative results include:

Skipper*-once**: A ***Skipper agent that generates one proxy problem at the start of the episode, and the replanning (choosing a checkpoint target based on the existing proxy problem) only triggers a quick selection of the immediate checkpoint target;

Skipper*-regen**: A slower ***Skipper agent that generates a new proxy problem every time replanning is triggered;

modelfree: A model-free baseline agent that serves as the basis of the architecture for the ***Skipper*** variants, with a prioritized distributional Double DQN [37, 13, 46];

Director: A tuned Director agent [18] fed with simplified visual inputs. Since Director discards trajectories that are not long enough for training purposes, we make sure that the same amount of training data is gathered as for the other agents;

LEAP: A re-implemented LEAP for discrete action spaces. Due to low performance, we replaced the VAE and the distance learning mechanisms with our own counterparts. We waived the agent-environment interaction costs for its generator pretraining stage, only showing the second stage of RL pretraining.

Please refer to the Appendix for more details on these agents, and additional experimental insights.

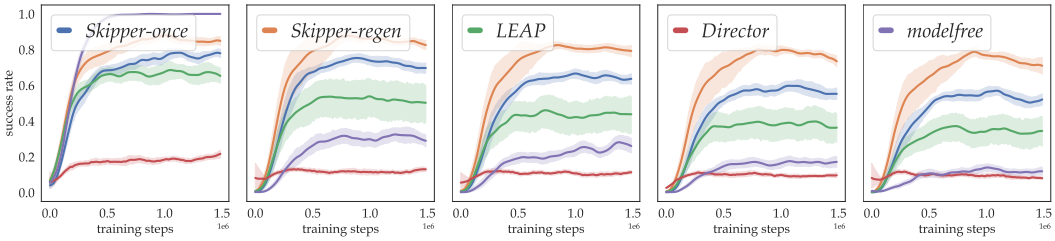
4.1 Generalization Performance

Fig. 3 shows how the agents' generalization performance evolves during the training process. These results are obtained with 50 fixed and sampled training tasks, a representative configuration of different numbers of training tasks including $\{1, 5, 25, 50, 100, \infty\}$[2], whose results can be found in the Appendix. In Fig. 3 a), we can observe how well an agent performs on its training tasks. If an agent performs well here but badly in b), c), d) and

[2] ∞ training tasks mean that an agent is trained on a different task for each training episode. In reality, this may lead to prohibitive costs in creating the training environment.

e), *e.g.* the **modelfree** baseline, then we conclude that it overfitted on training trajectories, likely an indicator of reliance on memorization.

In these experiments, we consistently observe a significant advantage (*i.e.* non-overlapping confidence intervals) in the generalization performance of the ***Skipper*** agents throughout the training process. The **regen** variant dominates all the other methods. This is because the frequent reconstruction of the graph makes the agent less prone to errors in the estimation and provides extra adaptability in novel scenarios. The ***Skipper*** agents behave less optimally than expected on training tasks, despite the strong generalization on evaluation tasks. As our ablation results and theoretical analyses consistently show, such a phenomenon is a composite outcome of inaccuracies both in the proxy problem and the checkpoint policy. One of the major symptoms of an inaccurate proxy problem is that the agent would chase over delusional checkpoint targets. We address this behavior with the delusion suppression technique, whose results can be found in the Appendix.



(a) training, diff 0.4 (b) OOD, diff 0.25 (c) OOD, diff 0.35 (d) OOD, diff 0.45 (e) OOD, diff 0.55

Figure 3: **Generalization Performance of Agents During Training**: the x-axes correspond to training progress, while the aligned y-axes represent the success rate of episodes (optimal is 1.0). Each agent is trained with 50 tasks. Each data point is the average success rate over 20 evaluation episodes, and each error bar (95% confidence interval) is processed from 20 independent seed runs. Training tasks performance is shown in (a) while OOD performance is shown in (b), (c), (d), (e).

Better than the **modelfree** baseline, LEAP obtains reasonable generalization performance, despite the extra budget it needs for pretraining. In the Appendix, we show that LEAP benefits largely from the delusion suppression technique. This indicates that optimizing for a path in the latent space is prone to errors caused by delusional subgoals. Lastly, we see that the Director agents suffer in these experiments despite their good performance in the single environment experimental settings reported by [18]. We present additional experiments in the Appendix to show that Director is ill-suited for our generalization-focused setting: Director still performs well in single environment configurations, but its performance deteriorates fast with more training tasks. This indicates poor scalability in terms of generalization, a limitation to its application in real-world scenarios.

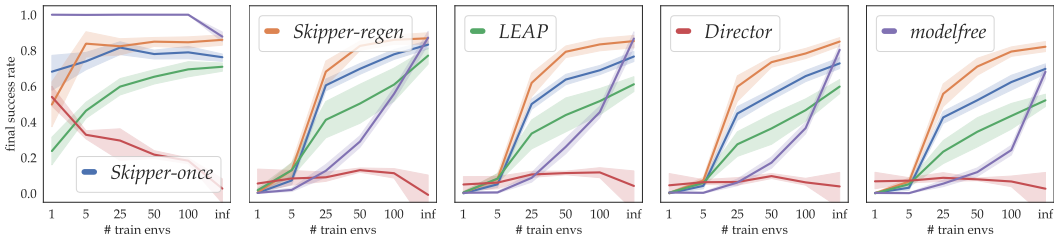
4.2 Scalability of Generalization Performance

Like [11], we investigate the scalability of the agents' generalization abilities across different numbers of training tasks. To this end, in Fig. 4, we present the results of the agents' final evaluation performance after training over different numbers of training tasks.

With more training tasks, the ***Skipper*** variants and the baseline show consistent improvements in generalization performance (significant advantage with a finite number of training tasks greater than 5). While both LEAP and Director behave similarly to the previous subsection, notably, the **modelfree** baseline can reach similar performance as ***Skipper***, but only when trained on a different task in each episode, which is generally not feasible in real-world applications beyond simulation.

4.3 Ablation & Sensitivity Studies

We present ablation results in the Appendix, where we confirm the effectiveness of delusion suppression, k-medoids checkpoint pruning and the local perception field, etc. In the Appendix, we also provide sensitivity study for the number of checkpoints in each proxy problem.



(a) training, diff 0.4 (b) OOD, diff 0.25 (c) OOD, diff 0.35 (d) OOD, diff 0.45 (e) OOD, diff 0.55

Figure 4: **Generalization Performance of Agents on Different Numbers of Training Tasks**: each data point and corresponding error bar (95% confidence interval) are based on the final performance from 20 independent seed runs. Training task performance is shown in (a) while OOD performance is shown in (b), (c), (d), (e). Notably, the ***Skipper*** agents as well as the adapted LEAP behave poorly during training when being trained on only one task, as the split of context and partial information cannot be achieved. Training on one task invalidates the purpose of the proposed generalization-focused checkpoint generator.

4.4 Summary of Experiments

Within the scope of the experiments, we conclude that:

- The proposed ***Skipper*** framework provides benefits for generalization;
- ***Skipper*** achieves better generalization when exposed to more training tasks;

From the content presented in the Appendix, we deduce additionally that:

- Spatial abstraction based on the local perception field is crucial for the scalability of the agents;
- ***Skipper*** performs well by reliably decomposing the given tasks, and achieving the sub-tasks robustly. Its performance is bottlenecked by the accuracy of the estimated proxy problems as well as the checkpoint policies. This matches well with our theory. The proposed delusion suppression technique (in Appendix) is effective in suppressing plans with non-existent checkpoints as targets, thereby increasing the accuracy of the proxy problems;
- LEAP fails to generalize well within its original form and can generalize better when combined with the ideas proposed in this paper; Director may generalize better only in domains where long and informative trajectory collection is possible;
- We verified empirically that, as expected, ***Skipper*** is compatible with stochasticity.

5 Related Work

Option Modelling. Learning options whose outcome can be reliably predicted has been studied in [16, 28], where the authors use unsupervised pretraining to discover intrinsic options via the empowerment objective. Similarly, our framework utilizes the models to learn the "outcomes" of options first, and then learn corresponding options constrained to achieve these outcomes [43, 33]. Thus, it dodges the difficulty of option collapse [5]. In fact, we trade difficulties in option modelling to those of generator learning. This is likely beneficial in tasks where states are easy to learn and generate, and / or in stochastic environments where the outcomes of unconstrained options are difficult to learn.

HP Frameworks. [31] uses generative models to imagine subgoals. In [23], promising states to explore are generated and plans are formed with shortest-path algorithms. Similar ideas have been attempted for guided exploration [15, 25]. Similar to [18], [12] generate k-th step ahead subgoals for complex reasoning tasks. In [8], the agent proposes halfway subgoals to the task goals. LEAP explicitly plans a chain of subgoals towards the task goal [33]. Checkpoints can be seen as sub-goals that generalize the notion of "landmarks" from [43], while proxy problems can be seen equivalent to a special case of subMDPs from [47]. A framework of using latent landmark graphs as high-level guidance has been explored by

[48]. These landmarks are selected via a sparsification procedure that uses a weighted sum in the latent space to compose subgoals. On the other hand, our checkpoint pruning selects a subset of generated states, which is less prone to issues created by weighted sums.

HP Estimates. [48] propose a distance estimate with an explicit regression. With TDMs [34], LEAP [33] uses a sparse intrinsic reward based on distance information to the goal when the time budget is depleted. Policy-aware estimators are investigated by [30].

Decision-Time / Background HP Methods. Prior to LEAP [33], path planning with evolutionary algorithms was investigated by [32]; [18, 27] propose world models to assist temporally abstract background planning.

Spatial Abstraction. Attention mechanisms have been investigated to construct state representations for model-free agents in [29, 26, 44]. In [49], the first form of spatial abstraction in planning was attempted. The authors proposed an attention-based bottleneck to dynamically select a subset of environmental entities during the atomic-step forward simulation during decision-time planning. ***Skipper*** is a step-up from their approach, where we identify that the previously overlooked aspect of spatial abstraction is as crucial for longer-term planning as temporal abstraction.

6 Conclusion & Future Work

We proposed, analyzed and validated our HP framework, ***Skipper***, which provides better generalization compared to other HP methods, because of its spatio-temporal abstraction abilities.

In this work, we generated checkpoints at random by sampling the partial description space. Despite the pruning mechanisms, the generated checkpoints do not prioritize the predictable, important states that matter the most to form a meaningful plan. This is likely why similar existing frameworks have limited applicable scenarios and experience trouble in environments with more complex reward structures. We would like to continue investigating the possibilities along this line. Additionally, we would like to explore other environments where the accuracy assumption (in Thm. 1) can meaningfully hold, *i.e.* beyond sparse reward cases.

References

[1] S. Alver and D. Precup. Understanding decision-time vs. background planning in model-based reinforcement learning. *arXiv preprint arXiv:2206.08442*, 2022.

[2] S. Alver and D. Precup. Minimal value-equivalent partial models for scalable and robust planning in lifelong reinforcement learning. *arXiv preprint arXiv:2301.10119*, 2023.

[3] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[4] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[5] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31, 2017.

[6] Y. Bengio. The consciousness prior. *arXiv*, 1709.08568, 2017. `http://arxiv.org/abs/1709.08568`.

[7] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint:1308.3432*, 2013.

[8] E. Chane-Sane, C. Schmid, and I. Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International Conference on Machine Learning*, pages 1430–1440. PMLR, 2021.

[9] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *International Conference on Learning Representations*, 2018. `http://arxiv.org/abs/1810.08272`.

[10] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for openai gym. *GitHub repository*, 2018. `https://github.com/maximecb/gym-minigrid`.

[11] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[12] K. Czechowski, T. Odrzygóźdź, M. Zbysiński, M. Zawalski, K. Olejnik, Y. Wu, Łukasz Kuciński, and P. Miłoś. Subgoal search for complex reasoning tasks. *arXiv preprint:2108.11204*, 2021.

[13] W. Dabney, M. Rowland, M. Bellemare, and R. Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[14] S. Dehaene, H. Lau, and S. Kouider. What is consciousness, and could machines have it? *Science*, 358, 2020.

[15] A. Erraqabi, M. Zhao, M. C. Machado, Y. Bengio, S. Sukhbaatar, L. Denoyer, and A. Lazaric. Exploration-driven representation learning in reinforcement learning. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

[16] K. Gregor, D. J. Rezende, and D. Wierstra. Variational intrinsic control. *arXiv preprint:1611.07507*, 2016.

[17] A. Gupta, G. Dar, S. Goodman, D. Ciprut, and J. Berant. Memory-efficient transformers via top-k attention. *arXiv preprint:2106.06899*, 2021.

[18] D. Hafner, K.-H. Lee, I. Fischer, and P. Abbeel. Deep hierarchical planning from pixels. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[19] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint:2301.04104*, 2023.

[20] D. Y.-T. Hui, M. Chevalier-Boisvert, D. Bahdanau, and Y. Bengio. Babyai 1.1, 2020.

[21] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. *Advances in neural information processing systems*, 32, 2019.

[22] L. Kaufman and P. Rousseeuw. *Partitioning Around Medoids (Program PAM)*, chapter 2, pages 68–125. John Wiley & Sons, Ltd, 1990.

[23] J. Kim, Y. Seo, and J. Shin. Landmark-guided subgoal generation in hierarchical reinforcement learning. *arXiv preprint:2110.13625*, 2021.

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014.

[25] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[26] A. Manchin, E. Abbasnejad, and A. Van Den Hengel. Reinforcement learning with attention that works: A self-supervised approach. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part V 26*, pages 223–230. Springer, 2019.

[27] R. Mendonca, O. Rybkin, K. Daniilidis, D. Hafner, and D. Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[28] N. Modhe, P. Chattopadhyay, M. Sharma, A. Das, D. Parikh, D. Batra, and R. Vedantam. IR-VIC: Unsupervised discovery of sub-goals for transfer in RL. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, jul 2020.

[29] A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in neural information processing systems*, 32, 2019.

[30] O. Nachum, S. S. Gu, H. Lee, and S. Levine. Data-efficient hierarchical reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[31] A. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. *arXiv preprint:1807.04742*, 2018.

[32] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. In *International Conference on Learning Representations*, 2020.

[33] S. Nasiriany, V. Pong, S. Lin, and S. Levine. Planning with goal-conditioned policies. *Advances in Neural Information Processing Systems*, 32, 2019.

[34] V. Pong, S. Gu, M. Dalal, and S. Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint:1802.09081*, 2018.

[35] M. L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[36] R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

[37] T. Schaul, J. Quan, I. Antonoglou, and D. Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[38] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[39] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[40] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[41] R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2(4):160–163, 1991.

[42] R. S. Sutton, M. C. Machado, G. Z. Holland, D. S. F. Timbers, B. Tanner, and A. White. Reward-respecting subtasks for model-based reinforcement learning. *arXiv preprint:2202.03466*, 2022.

[43] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[44] Y. Tang, D. Nguyen, and D. Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020.

[45] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[46] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[47] Z. Wen, D. Precup, M. Ibrahimi, A. Barreto, B. Van Roy, and S. Singh. On efficiency in hierarchical reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6708–6718. Curran Associates, Inc., 2020.

[48] L. Zhang, G. Yang, and B. C. Stadie. World model as a graph: Learning latent landmarks for planning. In *International Conference on Machine Learning*, pages 12611–12620. PMLR, 2021.

[49] M. Zhao, Z. Liu, S. Luan, S. Zhang, D. Precup, and Y. Bengio. A consciousness-inspired planning agent for model-based reinforcement learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 1569–1581. Curran Associates, Inc., 2021.