SEARCH ON THE REPLAY BUFFER:
BRIDGING PLANNING AND REINFORCEMENT LEARNING

Benjamin Eysenbach
Carnegie Mellon University, Google Brain
beysenba@cs.cmu.edu

Ruslan Salakhutdinov
Carnegie Mellon University
rsalakhu@cs.cmu.edu

Sergey Levine
UC Berkeley, Google Brain
svlevine@eecs.berkeley.edu

ABSTRACT

The history of learning to control has been an exciting back and forth between two broad classes of algorithms: planning and reinforcement learning. Planning algorithms effectively reason over extraordinarily long horizons, but are limited by their dependence on a hand-crafted local policy and distance metric over collision-free paths. Reinforcement learning excels at learning policies and predicting reachability through value functions, but fails to plan over long horizons. Despite the successes of each method on various tasks, long horizon, sparse reward tasks with high-dimensional remain exceedingly challenging for both planning and reinforcement learning algorithms. Frustratingly, these sorts of tasks are potentially the most useful, as they are simple to design (a human only need to provide an example goal state) and avoid injecting bias through reward shaping. We introduce a general-purpose control algorithm that combines the strengths of planning and reinforcement learning to effectively solve these tasks. Our main idea is to decompose the task of reaching a distant goal state into a sequence of easier tasks, each of which corresponds to reaching a particular subgoal. Using graph search over our replay buffer, we can automatically generate this sequence of subgoals, even in image-based environments. Our framework, Search on the Replay Buffer (SoRB), enables agents to solve sparse reward tasks over hundreds of steps, and generalizes substantially better than standard RL algorithms.

1 INTRODUCTION

How should agents learn to solve complex, temporally extended tasks? How should agents acquire the ability to reach new goals specified at test time? Classically, planning algorithms give use one tool for learning such tasks. While planning algorithms work well for tasks where it is easy to determine distances between states and easy to design a local policy to reach nearby states, both of these requirements become roadblocks when applying planning to high-dimensional (e.g., image-based) tasks. Learning algorithms excel at handling high-dimensional observations, but reinforcement learning – learning for control – fails to reason over long horizons to solve temporally extended tasks. Moreover, existing RL algorithms cannot be repurposed learned policies to achieve new goals specified at test time.

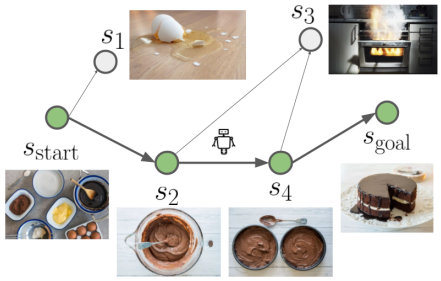

Figure 1: **SoRB**: In this cartoon illustration of SoRB, the robot makes a plan to bake a cake. Our algorithm, SoRB, automatically finds the subgoals via graph search over previously-seen images.

Recent work has introduced goal-conditioned RL (Schaul et al., 2015; Pong et al., 2018) in an attempt to acquire a single agent that can reach a wide range of goals. The overarching idea is that learning to reach a wide range of goals will allow the policy to generalize to new goals and be more data efficient

than learning a separate policy for each goal. In practice, goal-conditioned RL succeeds in reaching nearby goals but fail to reach distant goals; performance degrades quickly as the number of steps to the goal increases. Moreover, goal-conditioned RL often requires large amounts of reward shaping, which can limit the asymptotic performance of the policy by discouraging the policy from seeking novel solutions.

We propose to solve long-horizon sparse reward tasks by decomposing the task into a series of easier goal-reaching tasks. We learn a goal-conditioned policy for solving each of the goal-reaching tasks. *Our key insight is that we can reduce the problem of finding these subgoals to solving a shortest path problem over states that we have previous visited.* Using a distance metric extracted from our goal-conditioned policy, we solve the shortest path problem over previously visited states. We call this general framework Search on Replay Buffer (SoRB).

Our primary contribution is a framework (SoRB) that bridges planning and deep RL to solving long-horizon, sparse reward tasks. We develop a practical instantiation of this framework using ensembles of distributional value functions, which allow us to *robustly* learn distances and use them for *risk-aware* planning. Empirically, we find that our method generates effective plans to solve long horizon navigation tasks, even in image-based domains. Comparisons with state-of-the-art RL baselines show that SoRB is substantially more successful in reaching distant goals.

2 Bridging Planning and Reinforcement Learning

Planning algorithms must (1) sample valid states (i.e., in free space), (2) determine which pairs of states are reachable, and (3) estimate the distance between reachable pairs of states. These requirements are difficult to satisfy in complex tasks with high dimensional images. For example, consider a robot arm stacking blocks using image-based observations. Sampling states requires generating photo-realistic images, and determining reachability requires reasoning about dozens of interactions between blocks. Our method will obtain distance and reachability estimates using a reinforcement learning algorithm. To sample states, we will simply use a replay buffer of previously visited states as a non-parametric generative model. Our approach is outlined in Algorithm 2: after learning distances using off-policy RL, we do graph search to find a set of waypoints, and then use the learned policy to reach these waypoints.

Algorithm 1 Inputs are the current state s, the goal state g, the active set $\mathcal{T}$, the learned policy π and its value function V. Returns an action a.

function SEARCHPOLICY($s, g, \mathcal{T}, V, \pi$)
  $d_{\text{sp}}, w_1 \leftarrow$ SHORTESTPATH($s, g, \mathcal{T}, V$)
  $d_\pi \leftarrow -V(s, g)$
  if $d_{\text{sp}} < d_\pi$ or $d_\pi >$ MAXDIST **then**
    $a \leftarrow \pi(a, \mid s, w_1)$
  else
    $a \leftarrow \pi(a, \mid s, g)$
  return a

Assumptions: Unlike planning algorithms, we do not assume that we have access to a reachability oracle or a distance oracle. However, we do assume that we have a state identity oracle $id : \mathcal{S} \times \mathcal{S} \to \{0, 1\}$ that takes as input two states, and outputs 1 if those states are with nearly identical. One way to remove this assumption is to say two states are identical if their distance in some representation space is small. We leave this as future work.

2.1 Learning Distances with Reinforcement Learning

We consider an agent that maximizes its cumulative, *undiscounted* reward. We define the reward for s and action a with respect to goal g as

$$r(s, g, a) \triangleq \begin{cases} -1 & \text{if } id(s, g) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Crucially, we will terminate the episode as soon as the agent reaches the goal. With these ingredients in hand, we observe a close connection between the Q values and shortest paths. We define $d_{\text{sp}}(s, g)$ to be the shortest path distance from state s to state g. That is, $d_{\text{sp}}(s, g)$ is the expected number of steps to reach g from s under the optimal policy. The value of state s with respect to goal g is simply the negative shortest path distance: $V(s, g) = -d_{\text{sp}}(s, g)$. Overloading notation, we can define $d_{\text{sp}}(s, g, a)$ as the shortest path distance, conditioned on initially taking action a. Then Q values

also equal a negative shortest path distance: $Q(s, g, a) = -d_{\text{sp}}(s, g, a)$. Thus, we have reduced the problem of learning distances into a standard RL problem.

2.2 Distances via Distributional Reinforcement Learning

Standard Q-learning with the reward function in Equation 1 will fail to learn accurate distance estimates. The true value for a state and goal that are unreachable is $-\infty$, which cannot be represented by a standard, feed-forward Q-network. Simply clipping the Q-value estimates to be within some range avoids the problem of ill-defined Q-values, but results in very unstable learning dynamics. We adopt distributional Q-learning (Bellemare et al., 2017), noting that is has a convenient form when used with the reward function in Equation 1. In distributional RL, we discretize the possible value estimates into discretized set of bins $B = (B_1, B_2, \cdots, B_N)$. For learning distances, the bins B are simply distances, so B_i corresponds to the probability that the current state and goal are i steps away from one another. Our Q-function predicts a distribution $Q(s, g, a) \in \mathcal{P}^N$ over these bins, where $Q(s, g, a)_i$ is the predicted probability that states s and g are i steps away from one another. To avoid ill-defined Q-values, the final bin, B_N is a catch-all for predicted distances of at least N. Importantly, this gives us a well-defined method to represent large and infinite distances. Under this formulation, the targets Q^* for our Q-values have a simple form:

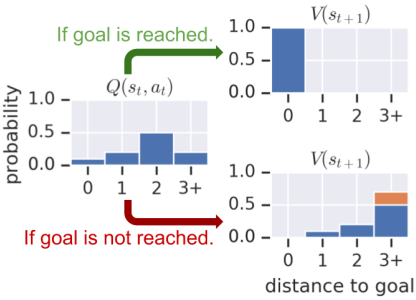

Figure 2: Bellman update for distributional distance learning.

$$Q^* = \begin{cases} (1, 0, \cdots, 0) & \text{if } id(s, g) = 1 \\ (0, Q_1, \cdots, Q_{N-2}, Q_{N-1} + Q_N) & \text{if } id(s, g) = 0 \end{cases}$$

As illustrated in Figure 2, if the state and goal are equivalent, then the target places all probability mass in bin 1. Otherwise, the targets are a right-shift of the current predictions. To ensure the target values sum to one, the mass in bin N of the targets is the sum of bins $N - 1$ and N from the predicted values. Following Bellemare et al. (2017), we update our Q function by minimizing the KL divergence between our predictions Q^θ and the target Q^*.

3 Experiments

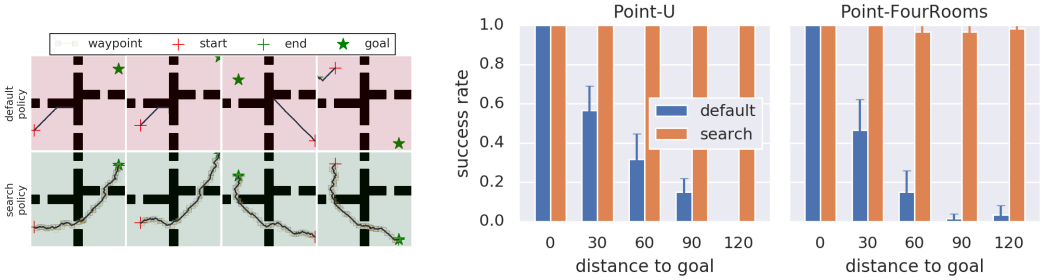

Figure 3: **Simple 2D Navigation**: An agent that combines a feedforward policy with search is substantially more successful at reaching distant goals than using the feedforward policy alone.

3.1 Didactic Experiment: Simple 2D Navigation

We start by gaining intuition for our method by applying it to two simple, 2D navigation tasks (details in Appendix B). The magnitude of the agent's actions is intentionally small, so reaching the goal can take over 100 steps, even for the optimal policy. Figure 3 visualizes the success rate as we vary the distance to the goal. We observe that the learned policy can reach nearby goals, but fails to generalize to distant goals. In contrast, the search policy successfully generalize in reaching these distant goals.

3.2 Searching over Images for Visual Navigation

We now examine how our method scales to high-dimensional observations. We apply SoRB to visual navigational in SunCG houses (Song et al., 2017), similar to the task described in Fu et al. (2019).

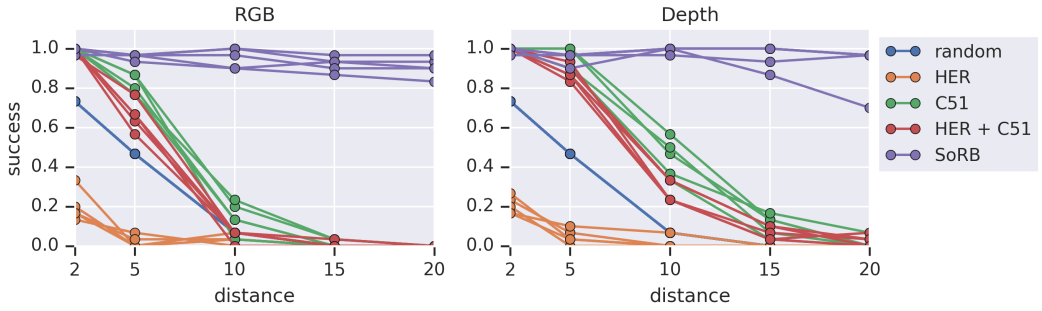

Figure 4: **Visual Navigation**: RL algorithms without search (HER, C51, and HER + C51) degrade quickly as the distance to the goal increases, while our method (SoRB) continues to succeed in reaching goals.

The agent receives either RGB or Depth images and takes actions to move North/South/East/West. To mitigate partial observability, we stitch four images into a panorama, so the resulting observation has dimension $4 \times 24 \times 32 \times C$, where C is the number of channels (3 for RGB, 1 for Depth).

We evaluate three state-of-the-art baselines: Hindsight Experience Replay (HER) (Andrychowicz et al., 2017), Distributional RL (C51) (Bellemare et al., 2017), and a combination of both baselines (HER + C51). Our method, SoRB, is obtained by performing search on top of the HER + C51 baseline. Figure 4 shows that while all baselines degrade quickly as the distance to the goal increases, our method continues to succeed in reaching goals with probability around 90%. We run another experiment to study how our search policy generalizes to new domains in Appendix C.

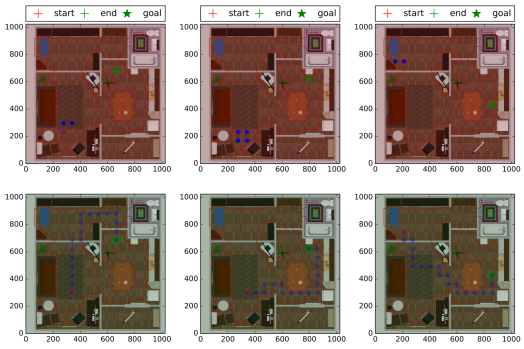

Figure 5: **Visual Navigation**: Rollouts of the baseline RL policy (top) and the search policy (bottom) on the same start and goals. The agent does not have access to the global (x, y) position.

4 Related Work

Planning Algorithms: Planning algorithms (LaValle, 2006; Choset et al., 2005) efficiently solve long-horizon tasks, including those that stymie reinforcement learning algorithms (see, e.g., Levine et al. (2011); Kavraki et al. (1996); Lau & Kuffner (2005)). However, these techniques assume that we can (1) efficiently sample valid states and (2) estimate the distance between two states, which make it challenging to apply these techniques to high-dimensional tasks (e.g., with image-based observations). Our method removes these assumptions by (1) sampling states from the replay buffer and (2) learning the distance metric with reinforcement learning. While the underlying motion planner we use is quite simple (namely, Dijkstra), we believe that the key idea of uses distance estimates obtained from RL algorithms for planning will open doors to incorporating more sophisticated motion planning techniques into reinforcement learning.

Goal-Conditioned Reinforcement Learning: Goal conditioned policies (Kaelbling, 1993b; Schaul et al., 2015; Pong et al., 2018) take as input the current state and a goal state, and predict a sequence of actions to arrive at the goal. Our algorithm learns a goal conditioned policy to reach waypoints along the planned path. Recent algorithms (Andrychowicz et al., 2017; Pong et al., 2018) combine off-policy RL algorithms with goal-relabelling to improve upon the sample complexity and robustness of goal conditioned policies. Similar algorithms have been proposed for visual navigation (Anderson et al., 2018; Gupta et al., 2017; Zhu et al., 2017; Mirowski et al., 2016). A common theme in recent work is learning distance metrics to accelerate reinforcement learning. While most methods (Florensa et al., 2019; Savinov et al., 2018; Wu et al., 2018) simply perform RL on top of the learned representation, our method explicitly performs search using the learned metric. We choose an off-policy RL algorithm (Lillicrap et al., 2015) with goal relabelling and distributional RL (Bellemare et al., 2017)) not only for improved data efficiency, but also to obtain good distance estimates.

Hierarchical RL: Hierarchical RL algorithms automatically learn a set of primitive skills to help an agent learn complex tasks. One class of methods (e.g., Kaelbling (1993a); Parr & Russell (1998);

Sutton et al. (1999); Precup (2000); Vezhnevets et al. (2017); Nachum et al. (2018); Frans et al. (2017); Bacon et al. (2017); Kulkarni et al. (2016)) jointly learn a low-level policy for performing each of the skills, and a high-level policy for sequencing these skills to complete a desired task. Another class of algorithms (e.g., Fox et al. (2017); Şimşek et al. (2005); Drummond (2002)) focus solely on automatically discovering these skills or subgoals. SoRB learns primitive skills that correspond to goal-reaching tasks, similar to Nachum et al. (2018). For sequencing these skills, SoRB directly relies on graph search, rather than learning an explicit, high-level policy. Learning the high-level policy is a often quite challenging because it is non-stationary: the actions for the high-level policy correspond to running certain low-level skills, but the meaning of the actions changes as the low-level skills are updated throughout training. Moreover, SoRB is more interpretable: it is easier to reason about shortest paths than deep neural networks. Whereas most hierarchical RL algorithms can be applied to any task, SoRB only works on goal-reaching tasks. We argue that the class of goal-reaching tasks is not overly restrictive, and provides rich structure in the form of shortest paths that algorithms like SoRB can exploit to obtain more efficient policies.

Model Based RL: Reinforcement learning methods are typically divided into model-free (Williams, 1992; Schulman et al., 2015b;a; 2017) and model-based (Watkins & Dayan, 1992; Lillicrap et al., 2015) approaches. Model-based approaches all perform some degree of planning, from predicting the value of some state (Silver et al., 2016; Mnih et al., 2013), obtaining representations by unrolling a learned dynamics model (Racanière et al., 2017), or learning a policy directly on a learned dynamics model (Sutton, 1990; Chua et al., 2018; Kurutach et al., 2018a; Finn & Levine, 2017; Agrawal et al., 2016; Oh et al., 2015; Nagabandi et al., 2018). One line of work (Amos et al., 2018; Srinivas et al., 2018; Tamar et al., 2016; Lee et al., 2018) embeds a differentiable planner inside a policy, with the planner learned end-to-end with the rest of the policy. Other work (Watter et al., 2015; Lenz et al., 2015) explicitly learns a representation for use inside a standard planning algorithm. Broadly, model-based approaches fall into three classes, learning a *state-space transition model* (Chua et al., 2018; Deisenroth & Rasmussen, 2011), a *latent-space models* (Hafner et al., 2018; Watter et al., 2015; Zhang et al., 2018; Kurutach et al., 2018b), or a *inverse-action models* (Pathak et al., 2017; Agrawal et al., 2016; Nair et al., 2017). In contrast, SoRB learns to predict the distances between states, which can be viewed as a high-level inverse model. SoRB predicts a scalar (the distance) rather than actions or observations, making the prediction problem substantially easier. By planning over previously visited states, SoRB does not have to cope with infeasible states that can be predicted by forward models in state-space and latent-space. While inverse action models consider a pair of consecutive states, SoRB considers distances between arbitrary pairs of states, allowing it to be readily applied to multi-step problems.

model	real states	multi-step	prediction dimension
state-space	✓	✓	1000s+
latent-space	✗	✗	10s
inverse	✓	✗	10s
SoRB	✓	✓	1

Figure 6: Four classes of model-based RL methods. Dimensions in the last column correspond to typical robotics tasks with image/lidar observations.

5 Future Work

In summary, we present SoRB as a method for combining planning algorithms with reinforcement learning. By exploiting the structure of goal-reaching tasks, we can obtain policies that generalize substantially better than those learned directly from RL. We believe that further work in this direction will facilitate sharing of ideas between planning and learning communities.

Acknowledgements: We thank Vitchyr Pong, Xingyu Lin, and Shane Gu for helpful discussions on learning goal-conditioned value functions, and Brian Okorn for feedback on connections to motion planning.

References

Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pp. 5074–5082, 2016.

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. In *Advances in Neural Information Processing Systems*, pp. 8289–8300, 2018.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 449–458. JMLR. org, 2017.

Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pp. 4759–4770, 2018.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Chris Drummond. Accelerating reinforcement learning by composing solutions of automatically identified subtasks. *Journal of Artificial Intelligence Research*, 16:59–104, 2002.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Carlos Florensa, Jonas Degrave, Nicolas Heess, Jost Tobias Springenberg, and Martin Riedmiller. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

Justin Fu, Anoop Korattikara, Sergey Levine, and Sergio Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *arXiv preprint arXiv:1902.07742*, 2019.

Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *arXiv preprint arXiv:1702.03920*, 3, 2017.

Josef Hadar and William R. Russell. Rules for ordering uncertain prospects. *The American Economic Review*, 59(1):25–34, 1969. ISSN 00028282. URL `http://www.jstor.org/stable/1811090`.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning*, volume 951, pp. 167–173, 1993a.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993b.

Lydia Kavraki, Petr Svestka, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on robotics and automation*, 12(4):566–580, 1996.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pp. 3675–3683, 2016.

Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *arXiv preprint arXiv:1802.10592*, 2018a.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pp. 8747–8758, 2018b.

Manfred Lau and James J Kuffner. Behavior planning for character animation. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 271–280. ACM, 2005.

Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

Lisa Lee, Emilio Parisotto, Devendra Singh Chaplot, Eric Xing, and Ruslan Salakhutdinov. Gated path planning networks. *arXiv preprint arXiv:1806.06408*, 2018.

Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

Sergey Levine, Yongjoon Lee, Vladlen Koltun, and Zoran Popović. Space-time planning with parameterized locomotion controllers. *ACM Transactions on Graphics (TOG)*, 30(3):23, 2011.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3307–3317, 2018.

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

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2146–2153. IEEE, 2017.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in neural information processing systems*, pp. 2863–2871, 2015.

Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pp. 1043–1049, 1998.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

Doina Precup. *Temporal abstraction in reinforcement learning*. University of Massachusetts Amherst, 2000.

Sébastien Racanière, Théophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Advances in neural information processing systems*, pp. 5690–5701, 2017.

Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274*, 2018.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pp. 1889–1897, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*, pp. 816–823. ACM, 2005.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1746–1754, 2017.

Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks. *arXiv preprint arXiv:1804.00645*, 2018.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pp. 216–224. Elsevier, 1990.

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

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pp. 2154–2162, 2016.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3540–3549. JMLR. org, 2017.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew J Johnson, and Sergey Levine. Solar: Deep structured latent representations for model-based reinforcement learning. *arXiv preprint arXiv:1808.09105*, 2018.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3357–3364. IEEE, 2017.

A EFFICIENT SHORTEST PATH COMPUTATION

Our policy solves a shortest path problem every time it recomputes a new waypoint. Naïvely running Dijkstra's algorithm to compute a shortest path among the states in our active set $\mathcal{T}$ requires $O(|\mathcal{T}|^2)$ queries of our value function. While the search algorithm itself is fast, it is expensive to evaluate the value function on each pair of states at every time step.

Algorithm 2 Inputs are the current state s, the goal state g, the active set $\mathcal{T}$, and the value function V. Returns the length and first waypoint of the shortest path.

function SHORTESTPATH($s, g, \mathcal{T}, V$)
  // Matrices: $D_\pi, D_{\mathcal{T}\to\mathcal{T}}, D_{s\to g} \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$
  // Vectors: $D_{s\to\mathcal{T}}, D_{\mathcal{T}\to g} \in \mathbb{R}^{|\mathcal{T}|}$
  $D_\pi \leftarrow -V(\mathcal{T}, \mathcal{T})$ ▷ cached
  $D_{\mathcal{T}\to\mathcal{T}} \leftarrow$ FLOYDWARSHALL(D_π) ▷ cached
  $D_{s\to\mathcal{T}} \leftarrow -V(s, \mathcal{T})$
  $D_{\mathcal{T}\to g} \leftarrow -V(\mathcal{T}, g)$
  $D_{s\to g} \leftarrow D_{s\to\mathcal{T}} + D_{\mathcal{T}\to\mathcal{T}} + (D_{\mathcal{T}\to g})^T$
  $d_{\text{sp}} \leftarrow \min_{u,v\in\mathcal{T}} D_{s\to g}[u, v]$
  $w_1, w_n \leftarrow \arg\min_{u,v\in\mathcal{T}} D_{s\to g}$
  return d_{sp}, w_1

In our implementation (Algorithm 2), we amortize this computation across many calls to the policy. We periodically periodically evaluate the value function on each pair of nodes in the active set, and then used the Floyd Warshall algorithm to compute the shortest path between all pairs. This takes $O(|\mathcal{T}|^3)$ time, but only $O(|\mathcal{T}|^2)$ calls to the value function. Let $D \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{T}|}$ be the resulting matrix storing the shortest path distances between all pairs of states in the active set. Now, given a start state s and goal state g, the shortest path distance is

$$d_{\text{sp}}(s, g) = \min\left(\min_{u,v\in\mathcal{T}} d(s, u) + D[u, v] + d(v, g), d(s, g)\right)$$

This computation requires $O(|\mathcal{T}|)$ calls to the value function, substantially better than the $O(|\mathcal{T}|^2)$ calls required with the naïve implementation.

The cost of this amortized computation is that we use stale distance estimates. In practice, we find that distance estimates vary slowly throughout training, so the stale distance estimates remain approximately correct.

B ENVIRONMENTS

B.1 2D NAVIGATION ENVIRONMENTS

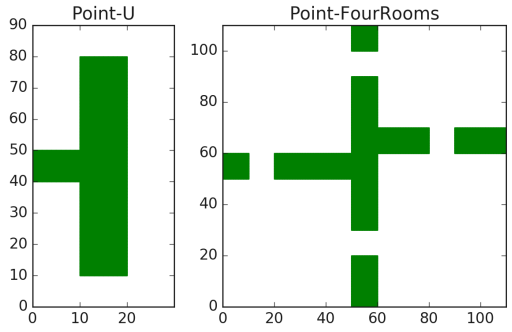

Figure 7: Navigation environments.

We used two simple navigation environments, Point-U and Point-FourRooms, shown in Figure 7. In both environments, the observations are the location of the agent, $s = (x, y) \in \mathbb{R}^2$. The agents actions $a = (dx, dy) \in [-1, 1]^2$ are added to the agents current position at every time step. We tuned the environments so that the baseline algorithm would perform as well as possible. Observing that the baseline agent would get stuck at corners, we modified the environment to automatically add

Gaussian noise to the agents action. The resulting dynamics were

$$s_{t+1} = \text{proj}(s_t + a_t + \epsilon_t) \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)$$

where proj() handles collisions with walls by projecting the state to the nearest free state. We used $\sigma^2 = 1.0$ for Point-U, and $\sigma^2 = 0.1$ for the (larger) Point-FourRooms environment.

B.2 Visual Navigation

For the visual navigation tasks, success is defined as arriving within 1 meter of the goal state in 100 steps. In Figure 4, each line corresponds to one of 5 random seeds. For each distance, each random seed was evaluated on 30 randomly-sampled (start, goal) pairs. Figure 5 shows example rollouts of the baseline HER + C51 (top) and out method (bottom). To avoid cherry-picking, we saved the first 3 rollouts of the first random seed when evaluating on the most distant set of goals ($\approx$ 20 steps).

C Generalization

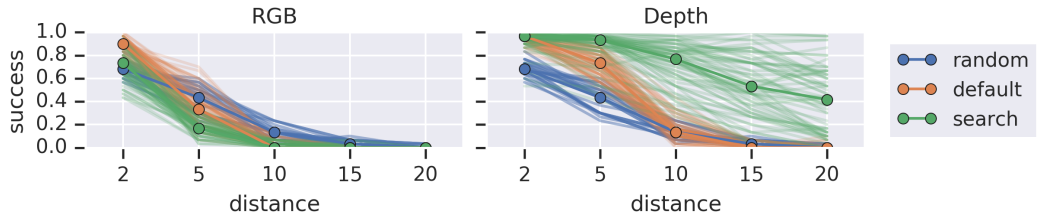

Figure 8: **Generalization in Visual Navigation**: After training on 100 environments, we collect data in a new environment to use for search. Opaque lines correspond to average success rate (across 30 start-goal pairs) across 3 training runs (with different random seeds) across 22 held-out houses

We study the generalization of our method on the visual navigation tasks. We train on a set of 100 houses. At test time, collect random data from the test environment and use that data to perform search. Figure 8 compares our method to the HER + C51 baseline. With RGB observations, neither method performs better than a random policy. Surprisingly, with Depth observation, our method reaches almost 80% of goals that are 10 steps away, and more than 40% of goals that are 20 steps away.

D Tricks for Learning Distances with RL

1. *Small learning rates*: Especially for the image-based tasks, we found that RL completely failed with using a critic learning rate larger than 1e-4. Smaller learning rates work too, but take longer to converge.
2. *Distributional RL*: The value function update for distributional RL has a particularly nice form when values correspond to distances. Additionally, distributional RL implicitly clips the values, preventing the critic to predict that unreachable states are infinitely far away.
3. *Termination Condition*: Carefully consider whether you set `done = True` at the end of each episode. In our setting the agent received a reward of -1 at each time step, so the value of each state was negative. An optimal agent therefore attempts to terminate the episode as quickly as possible. We only set `done = True` when the agent reached the goal state, not when the maximum number of time steps was reached or when it reached some other absorbing state.
4. *Ensembles of Value Functions*: Predicted distances from a single value function can be inaccurate for unseen (state, goal) pairs. When performing search using these predicted distances, these inaccurately-short predictions result in "wormholes" through the environment, where the agent mistakenly believes that two distant states are actually nearby. To mitigate this, we trained multiple, independent critics in parallel on the same data, and then aggregated predictions from each before doing search. Surprisingly, we found that taking

the average predicted distance over the ensemble worked as well as taking the maximum predicted distance. We tried accelerating training by using shared convolutional layers for all critics in the ensemble, but found that this resulted in highly-correlated distant predictions that exhibited the "wormhole" problem.

E FAILED EXPERIMENTS

1. *Goal Relabelling*: As mentioned above, we tried to combine our method with off-policy goal relabelling (Andrychowicz et al., 2017; Pong et al., 2018). Surprisingly, we found that this hurt performance of the non-search policy, and had no effect on the search policy.
2. *Lower-bounds on Q-values*: We attempted to use the search path to obtain a lower bound on the target Q-values during training. In the Bellman update, we replaced the distance predicted by the target Q-values with the minimum of (1) the distance predicted by the target Q-network and (2) the distance of the shortest search path. This can be interpreted as a generalization of the single-step lower bound from Kaelbling (1993b). Initial experiments showed this approach slowed down learning, and in some cases prevented the algorithm from converging. We hypothesize that Q-learning is must more sensitive to error in the *relative values* of two actions, rather than the *absolute value* of any particular action. While our lower-bound method likely decreased the absolute error, it did not decrease the relative error (and may have even increased it).
3. *TD3-style Ensemble Aggregation*: In our main experiments, we aggregated distance predictions from the ensemble of distributional critics by first computing the expected distance of each critic, and then averaging the predicted means. This approach ignores the fact that our critics are distributional. Inspired by the stability of TD3, we attempted to apply a similar approach to aggregating predictions from the ensemble of distributional critics. The naïve approach of taking the minimum for each atom does not work because the resulting distribution will not sum to one. Instead, we first compute the cumulative density function (CDF) of each critic and then take the pointwise maximum over the CDFs. Note that critics correspond to negative distance, so the maximum corresponds to being pessimistic. Finally, we convert the resulting CDF back into a PDF and return the corresponding expected distance. While this method has neat connections to second-order stochastic dominance and risk-averse expected utility maximizers (Hadar & Russell, 1969), we found that it worked quite poorly in practice.