THINK, ACT, AND LEARN ON AN EPISODIC MEMORY GRAPH

Anonymous authors
Paper under double-blind review

ABSTRACT

Designing agents that adapt quickly in non-stationary environments remains an open challenge. Value-bootstrapping limits how quickly Q-learning converges, while search-based methods require a large number of simulations at decision time. In this work, we study how to obtain the best of both approaches when data is scarce and the environment is changing. We introduce the *Universal Value Prediction Network*, an approach that learns a partial model by distilling a distance metric from searches on an episodic memory graph. Results show that the learned value function contains an accurate metric map of the state space; the learned heuristic search have lower planning cost at inference time than exhaustive graph search methods; and that the learning system is quick at adapting to changes in the environment. Our method is a way to bring model-free, model-based, and episodic control together within the same agent to alleviate the deficiency of each.

1 INTRODUCTION

In standard Markov decision processes (MDPs), an agent is expected to maximize its cumulative return in each episode. Recent reinforcement learning methods often take a delayed approach to learning, where learning occurs after experience is collected. A close examination reveals the speed bottlenecks at the basis of this delayed learning approach: in value-bootstrapping methods such as Q-learning, the bootstrapping objective introduces learning instabilities that limits the size of the gradient update, and how quickly new reward can propagate through (Blundell et al., 2016; Pritzel et al., 2017). Search-based methods avoid such problems by generating value estimates at decision time, but they rely on a hard-coded model of the environment and prohibitive amounts of training (Silver et al., 2018; 2017). Finally, recent model-based methods that learn *latent, partial models* for planning appear to be a promising compromise, but the model learning typically occurs through stochastic gradient descent, again limiting the test-time adaptability within a single episode Oh et al. (2017); Schrittwieser et al. (2019).

Being able to quickly adapt to changing situations without having to pause to learn is a desirable feature in general intelligence. This is a type of *fast* learning that happens instantaneously Botvinick et al. (2019). Work in meta-learning and episodic control have shown the importance of memory in building agents that can pick up new knowledge instantaneously (Wang et al., 2019; Duan et al., 2016; Ritter et al., 2018; Pritzel et al., 2017; Blundell et al., 2016). On a parallel thread, recent works in navigation introduce a topological graph structure on top of episodic memory. This graph acts as a tabular model for high-level planning Savinov et al. (2018); Eysenbach et al. (2019) and for model-based unrolls to learn state representation Yang et al. (2019). Nevertheless, adapting this type of model at test time so that the agent can succeed in a single trial in the face of domain change remains an under-explored problem.

In this paper, we study how to integrate value-learning with episodic memory graphs, to obtain an agent that can distill domain knowledge into reactive skills, yet retaining the flexibility to adapt when the environment changes. Our main contribution is *Universal Value Prediction Networks* (UVPN), a sample-efficient approach that combines model-free value approximation with model-based search methods on an episodic memory graph. UVPN extends aforementioned work that construct graphs on the replay buffer, to treat the graph as the basis for a *partial model* (Oh et al., 2017; Schrittwieser et al., 2019) that predicts local rewards. The structured nature of the episodic memory graph enables fast generation of value targets that can be distilled into a neural network value estimator. The

Figure 1: Universal Value Prediction Network.

distilled estimator has less variance, enabling its use as a critic for learning a policy. Critical for online adaptation, the tabular transition model is updated at decision time when changes occur, instantaneously affecting the agent's behavior in changing environments. This type of fast learning depends on components of the learning system that learns slowly (Botvinick et al., 2019). Our method does not rely on a extrinsic reward, and is a way to master the environment without supervision.

We include related works and technical background in the appendix.

2 The Universal Value Prediction Network

The universal value prediction network constructs a graph on the replay buffer using a learned local metric (Savinov et al., 2018). We refer to this graph as the episodic memory graph (EMG). UVPN generate value targets directly by running heuristic search on this graph, and distills these value target into a neural network function approximator. To learn a policy, UVPN extends the advantage weighted regression (AWR Peng et.al 2019). At decision time, UVPN actively updates the episodic memory graph to mark edges that have been visited, and to remove edges that are infeasible.

2.1 Value Distillation

Following Kaelbling (1993), we de ne the value function as $V(s;g) = d(s;g)$, where d is a distance metric on the domain that represents the shortest path-length between s and g. UVPN learns the optimal goal-conditioned value between s and g, by directly regressing towards the path-length of a plan as the value target similar to Oh et al. (2017)

$$^{(MCMC)}L_{\text{distill}} = \quad V \ (s;g); \ r_t + \ r_{t+1} + \qquad {}_1 : \qquad (1)$$

2.2 Learning How to Act: The Advantage

Past work in this area treat the episodic memory graph in combination with Dijkstra's shortest path algorithm as a high-level planner that is used at test time to direct a low-level policy over long-horizon tasks. The low-level policy is either trained separately as an inverse kinematics model (Savinov et al., 2018), or a policy restricted to short-range goals (Eysenbach et al., 2019; Huang et al., 2019). When evaluated alone without guidance from the planner, these local policies fail.

Our key insight is that the value function Plan2Vec acquires in Sec. 6.5 contains suf cient long-range information of the state space to help obtain a policy with only a small additional amount of action-state transition data. Following the approach in advantage actor-critic and policy gradient methods (Wang et al., 2015; Mnih et al., 2016; Schulman et al., 2015), the optimal policy is the one that picks the most advantageous action

$$_{\text{discrete}}(a|s;g) := \arg\max_{a} A(s;a;g) = \arg\max_{a} \frac{\exp A(s;a;g)}{\sum_{a_i} \exp A(s;a_i;g)} : \qquad (2)$$

Q-learning uses a regression objective that only supervises the action that is sampled a. The partition function of the action distribution $\forall a_i \in A$ at state s is therefore under-constrained. For this reason we extend advantage weighted regression (AWR Peng et.al 2019) to a goal-conditioned policy, in combination with a inverse model objective:

$$L = \; E \log (a|s;g) \exp (A(s;a;g) \; b(s;g)) \quad E \log (a|s;s') \; ; \qquad (3)$$

b is the goal-conditioned value baseline. a is the sampled action for the state transition (s, s'). We found this maximum likelihood based objective much more stable than Q-learning.

Reaching farther with goal-relabeling The advantage is defined with respect to a goal. We augment each transition with randomly selected goals from memory. This is a form of goal-relabeling (Andrychowicz et al., 2017; Warde-Farley et al., 2018), and is the primary mechanism that enables Plan2Vec to learn a policy that is longer in reach in comparison to prior methods.

2.3 Episodic Model Update

The episodic memory graph offers a way for the agent to instantaneoulsy incorporate knowledge into its planning. We incoporate two types of episodic model update: The first type marks each edge as visited, preventing the agent from re-planning in a cycle. The second type removes edges permanently, when the agent fails to move far enough towards the goal that is set by the planner.

3 Value Learning Experiments

First, we investigate the quality of the value estimation that UVPN learns from a very small amount of data. We found that the regression objective allows UVPN to quickly learn an accurate value estimate. We include the results in the appendix due to space limit. Then we scale UVPN up to Street Learn, a challenging real-world navigation dataset where value-bootstrapping methods struggle (Yang et al., 2019), in Fig.2a[1]. UVPN is able to learn a metric map of this large domain on a single GPU machine with 8 CPU cores, within 45 minutes.

3.1 Reactive Planning

We visualize the planning cost during training in the maze domain in Fig.5. We compare two variants of UVPN with the Dijkstra's used by SPTM and SoRM in Figure 3. The first variant (UVPN) uses the learned value estimate to prioritize node expansion. This is a purely heuristic-based search. The second variant (UVPN*) is similar to the A* search algorithm in that it uses a sum of the planning distance that has been covered so far D (in table.2), in addition to the value estimate which is a surrogate for the expected distance to the goal.

Reactive Planning The contrast between planning reactive versus searching exhaustively over all memory can be substantial on a large dataset. On the Street Learn domain for example, the reduced dataset contains 1500 street views. We pick two points on the map in Figure 2 that lay at the opposite corners of the map area. Dijkstra's exhaustive search expands all 1500 nodes. A* using a “Manhattan distance” does better, but gets thrown off by Broad Way's triangular shape. UVPN* using the learned value function expand few points along the path. On average, UVPN* expands 1.6 nodes per step. If we exclude the correct node itself, the contrast in planning overhead is 2500[2] times

4 Robust Episodic Control

UVPN is distinguished from prior methods in that it contains all three types of control: model-based, model-free, and episodic controls. The model-based control using search is the slowest, and the most

[1]Additional results can be found at https://sites.google.com/view/uvpn

[2]$1500/0.6 \approx 2500$

Figure 2: Left: Embedding Learned by UVPN on Street learn. Middle: Nodes Expanded During Search. Gray dots show the nodes in the graph that are covered during search. With a strong heuristic (L1 distance), A* is more economical than Dijkstra's. But with a learned heuristic UVPN has even lower search cost. We color the expanded nodes with UVPN in red to make the few expanded nodes more obvious. Right: Bar chart shows the number of nodes expanded for these plans. The plans span about 1.2 kilometers each. Number on top of bars show the average cost per planning step. UVPN approaches 1.

Figure 3: Planning cost during learning on the Maze domain. UVPN uses the learned distance-to-goal value (H) as a planning heuristic. UVPN* is similar to A* in that it uses path-length so-far (D) in addition to the distance estimate (H). One can see the planning cost goes down as learning progresses. Bar chart shows the nal planning cost at the end of training. Both SPTM and SoRB uses Dijkstra's, which has a constant search cost that does not go down.

memory-hungry for decision-making, because the agent needs to reason steps ahead into the future. The model-free control is the fastest at reaction time, but it is much slower to learn, requiring many gradient steps of distillation. Finally, the episodic memory is a fast learning mechanism that can pick up knowledge instantaneously, but it lacks access to integrated quantities such as the expected distance-to-goal.

In the following maze experiment, we start with a pre-trained agent that has become familiar with the maze. To get to the goal on the other side, it has learned to move around the wall (Fig.6a). Now we place a complex vertical separation to split the maze into 4 rooms. The agent acts according to how it has been acting in the past, but it quickly runs into the new wall colored in red (Fig.6b). As the agent fails to move according to its plan, it removes edges from the graph that the agent's low-level policy is not able to implement (Fig.6c). The agent then quickly re-plans on this updated episodic model. It gets around the wall step-by-step, reaching the goal.

5 CONCLUSIONS

We have presented universal value prediction network, which integrates value-learning with model-based planning on an episodic memory graph. The structured nature of the episodic memory graph allows for fast generation of value targets for distillation into a neural network estimator. We show in various experiments that UVPN achieves accurate long-range value estimate, can learn a low-level policy, plan reactively, while also being robust to changes in the environment.

Figure 4: Adapting to Changes in the Environment. Left: Original plan made by the agent in Maze. Middle: we insert a vertical separation, shown as two red blocks. Colored lines shows the forward plan at 4 separate steps in the same episode. Right: colored segments showing the edges pruned at each step.

REFERENCES

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Anthony, T., Tian, Z., and Barber, D. Thinking fast and slow with deep learning and tree search. pp. 5360–5370, 2017. URL http://papers.nips.cc/paper/7120-thinking-fast-and-slow-with-deep-learning-and-tree-search.pdf.

Bansal, S., Calandra, R., Chua, K., Levine, S., and Tomlin, C. Mbmf: Model-based priors for model-free reinforcement learning. September 2017. URL http://arxiv.org/abs/1709.03153.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. Model-free episodic control. June 2016. URL http://arxiv.org/abs/1606.04460.

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., and Hassabis, D. Reinforcement learning, fast and slow. *Trends Cogn. Sci.*, 23(5):408–422, May 2019. ISSN 1364-6613, 1879-307X. doi: 10.1016/j.tics.2019.02.006. URL http://dx.doi.org/10.1016/j.tics.2019.02.006.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S., and Levine, S. Combining model-based and model-free updates for trajectory-centric reinforcement learning. March 2017. URL http://arxiv.org/abs/1703.03078.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural Comput.*, 7(5):889–904, September 1995. ISSN 0899-7667. doi: 10.1162/neco.1995.7.5.889. URL http://dx.doi.org/10.1162/neco.1995.7.5.889.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., and Abbeel, P. RL^2: Fast reinforcement learning via slow reinforcement learning. November 2016. URL http://arxiv.org/abs/1611.02779.

Eysenbach, B., Salakhutdinov, R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint*, 2019.

Florensa, C., Degrave, J., Heess, N., Springenberg, J. T., and Riedmiller, M. Self-supervised learning of image embedding for continuous control. *arXiv preprint arXiv:1901.00943*, 2019.

Gallo, G. and Pallottino, S. Shortest path methods: A unifying approach. pp. 38–64, 1986. doi: 10.1007/BFb0121087. URL https://doi.org/10.1007/BFb0121087.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. March 2016. URL http://arxiv.org/abs/1603.00748.

Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Pfaff, T., Weber, T., Buesing, L., and Battaglia, P. W. Combining q-learning and search with amortized value estimates. September 2019. URL https://openreview.net/forum?id=SkeAaJrKDS.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 2168-2887. doi: 10.1109/TSSC.1968.300136. URL http://dx.doi.org/10.1109/TSSC.1968.300136.

Hartikainen, K., Geng, X., Haarnoja, T., and Levine, S. Dynamical distance learning for unsupervised and semi-supervised skill discovery. *arXiv preprint arXiv:1907.08225*, 2019.

Huang, Z., Liu, F., and Su, H. Mapping state space using landmarks for universal goal reaching. In *Advances in Neural Information Processing Systems 32*, pp. 1940–1950. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/8469-mapping-state-space-using-landmarks-for-universal-goal-reaching.pdf.

Jurgenson, T., Groshev, E., and Tamar, A. Sub-goal trees – a framework for goal-directed trajectory prediction and optimization. June 2019. URL http://arxiv.org/abs/1906.05329.

Kaelbling, L. P. Learning to achieve goals. *IJCAI*, 1993. ISSN 1045-0823. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.51.3077&rep=rep1&type=pdf.

Liu, R., Lehman, J., Molino, P., Such, F. P., and others. An intriguing failing of convolutional neural networks and the coordconv solution. *Adv. Neural Inf. Process. Syst.*, 2018. ISSN 1049-5258. URL http://papers.nips.cc/paper/8169-an-intriguing-failing-of-convolutional-neural-networks-and-the-coordconv-solution.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pp. 1928–1937. jmlr.org, June 2016. URL http://www.jmlr.org/proceedings/papers/v48/mniha16.html.

Oh, J., Singh, S., and Lee, H. Value prediction network. July 2017. URL http://arxiv.org/abs/1707.03497.

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

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2827–2836, Sydney, NSW, Australia, 2017. JMLR.org. URL http://dl.acm.org/citation.cfm?id=3305890.3305973.

Ritter, S., Wang, J. X., Kurth-Nelson, Z., Jayakumar, S. M., Blundell, C., Pascanu, R., and Botvinick, M. Been there, done that: Meta-learning with episodic recall. May 2018. URL http://arxiv.org/abs/1805.09692.

Savinov, N., Dosovitskiy, A., and Koltun, V. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*, 2018.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. Mastering atari, go, chess and shogi by planning with a learned model. November 2019. URL http://arxiv.org/abs/1911.08265.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. June 2015. URL http://arxiv.org/abs/1506.02438.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550(7676): 354–359, October 2017. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature24270. URL http://dx.doi.org/10.1038/nature24270.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, December 2018. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.aar6404. URL http://dx.doi.org/10.1126/science.aar6404.

Sutton, R. S. and Barto, A. G. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998. ISBN 0262193981.

Tamar, A., Thomas, G., Zhang, T., Levine, S., and Abbeel, P. Learning from the hindsight plan – episodic mpc improvement. September 2016a. URL http://arxiv.org/abs/1609.09001.

Tamar, A., WU, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 2154–2162. Curran Associates, Inc., 2016b.

Wang, A., Kurutach, T., Liu, K., Abbeel, P., and Tamar, A. Learning robotic manipulation through visual planning and acting. *arXiv preprint arXiv:1905.04411*, 2019.

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. Dueling network architectures for deep reinforcement learning. November 2015. URL http:==arxiv:org=abs=1511:06581.

Warde-Farley, D., Van de Wiele, T., Kulkarni, T., Ionescu, C., Hansen, S., and Mnih, V. Unsupervised control through non-parametric discriminative rewards. November 2018. URL http:==arxiv:org=abs=1811:11359.

Yang, G., Zhang, A., Morcos, A., Pineau, J., Abbeel, P., and Calandra, R. Unsupervised representation learning by latent plans. 2019. URL https:==openreview:net=forum?id=Bye6weHFvB.

Zhang, A., Wu, Y., and Pineau, J. Natural environment benchmarks for reinforcement learning. CoRR, abs/1811.06032, 2018. URL http:==arxiv:org=abs=1811:06032.

6 Appendix

6.1 Related Works

This work extends recent proposals that construct topological graphs on an exemplar ensemble for navigation (Savinov et al., 2018; Eysenbach et al., 2019) to fully integrate high-level planning with learning a behavior policy. In search on replay buffer (SoRB, Eysenbach et al. (2019)), the graph structure is only used to make high-level plans. The goal-conditioned Q-function learns via 1-step value-bootstrapping from traditional linear replay. Plan2vec Yang et al. (2019) introduces model-based unrolls on the graph, but the learning is again hampered by 1-step value-iteration. UVPN improves upon these methods by directly generating value targets with n-step plans made on the graph and learning by supervision. This value estimator has lower variance, and is used to supervise an advantage function for the policy.

UVPN also learns to plan reactively using this value function as opposed to the exhaustive search used in SPTM and SoRB. This is closely related to TD-search, expert iteration (ExIt), and searching with amortized value estimates (SAVE) in the MCTS literature Oh et al. (2017); Anthony et al. (2017); Hamrick et al. (2019). UVPN applies these techniques on a learned, topological model of the environment, in a high-dimensional, continuous state space.

UVPN is a member of the recently proposed *partial models* Oh et al. (2017); Schrittwieser et al. (2019). A partial model focuses on predicting the reward and return value, as opposed to the raw observation. Methods proposed so far produce features robust against task-irrelevant details, but they rely on the assumption that a reward is available *a priori*, and only a *single task* needs to be learned. UVPN extends the state-only value estimate $V(s;\ ^{0})$ to a goal-conditioned, universal value function $V(s; g; \ )$. The challenge of learning to generalize over *goals* in addition to states is well documented (Schaul et al., 2015; Jurgenson et al., 2019; Hartikainen et al., 2019). Value iteration network (VIN Tamar et al. 2016b) tackles a similar problem on a grid world, whereas universal planning networks (UPN) require expert demonstrations to supervise both the latent model and the reward for visual motor control.

The way UVPN represents the environment is inspired by *neural episodic control* (NEC Pritzel et al. 2017; Blundell et al. 2016), where the reward and return value for recent experience are kept in a tabular record. This record also allows UVPN to plan without action data, as it only needs to decide what states are *close* to the current one in recent memory. In addition, UVPN takes advantage of the instantaneity of this non-parametric approach to update the graph at test time, when goals generated by the planner turn out to be non-reachable. This graph-improvement scheme is similar to hindsight iterative MPC (HIMPC) explored by Tamar et al. 2016a, where a robot applies focused short-term correction to its cost function before replan. UVPN improves upon HIMPC which is episodic, to apply changes after each time step.

Finally, UVPN takes inspiration from a long history of methods at the intersection between model-based and model-free approaches Pong et al. (2018); Chebotar et al. (2017); Bansal et al. (2017); Gu et al. (2016).

6.2 Technical Background

Episodic Memory as A Graph. Recent work in navigation Savinov et al. (2018); Eysenbach et al. (2019); Yang et al. (2019) explores the construction of a state-graph with samples inside the replay buffer. We refer to both semi-parametric topological memory (SPTM) and search on replay buffer (SoRB) as constructing an *episodic memory graph*.

Graph Planning Algorithms such as breadth-first search (BFS), Dijkstra's shortest-path algorithm, heuristic-based search, and A* can be unified under a simple formulation, where the only difference is the priority weight used to rank nodes for expansion (Hart et al., 1968; Gallo & Pallottino, 1986). The particular choice of expansion priority directly affects the search efficiency of a method, and whether there is a guarantee of finding the shortest path (see Sec. 6.5).

When a planner is combined with an episodic memory graph, an agent can act by retrieving states from recent memory that match the current observation, as a form of *episodic control* (Blundell et al., 2016). Under this light, plan2vec is a way to transfer from episodic control to a reactive universal policy via model-based distillation.

6.3 Heuristic Search Algorithms

In comparison to proximal dynamic programming techniques such as Q-learning, graph planning algorithms require no training, and execute the Bellman relaxation explicitly at decision time (Sutton & Barto, 1998). It also requires less compute and is numerically more appealing than the optimization based relaxation that proximal methods use, which in combination with a neural network function approximator can often fail to converge on universal value approximation tasks (Jurgenson et al., 2019; Hartikainen et al., 2019). The latter, however, has the benefit of acquiring a reactive plan as a closed-form policy, which by itself takes less memory and compute.

Definition 1 *A heuristic search algorithm can be defined as a function,* $\text{Search} : (G, s, g) \to [s_i]$, *where* G *is the state graph,* s *and* g *are the root node and the goal, which returns an ordered list of nodes. Common variants can be formulated as the following procedure, where the only difference is the priority that is used to draw nodes from the queue for expansion.*

1. init priority queue H with root node and weight 0
2. **selection** pick the top node from the queue $n \leftarrow H$
3. **expansion** queue n's neighbor set N with weights W_i
4. if $g \in N$ go to next, otherwise go to 2.
5. **backtrack** from g to s to generate the plan $[s_i]$.

Method	**Priority W_i**
BFS	path steps S
Dijkstra's	path length D
Heuristic	distance-to-goal H
A*	$D + H$
UVPN	V (learned)
UVPN*	$D + V$ (learned)

6.4 Episodic Memory Graph as A Partial Model

Decoding all pixel values of a high-dimensional observation can be unrealistic and wasteful, especially if there is natural noise in the background Zhang et al. (2018). Reward signals and the value of a state are important aspects of the agent's day-to-day operation, hence they are also good measures on the relevancy of the compressed feature set. Under this light, we extend value prediction networks and *partial models* (Oh et al., 2017; Schrittwieser et al., 2019) to the episodic regime Edges in an episodic memory graph are the *rewards*. The value estimator we introduce in Sec.2.1 captures long-term returns. The graph thus becomes a *universal value prediction network*.

6.5 Amortized Search with Learned Heuristic

This constructed episodic memory graph now enables us to apply exact dynamic programming methods to generate latent plans. Search algorithms usually discard the distance value collected during backtrack (step 5), whereas value-based methods specifically learn these as the *distance-to-goal* (Kaelbling, 1993; Sutton & Barto, 1998). With UVPN, we need a way to quickly distill plans found by search into a neural network that can generalize. This procedure is referred to as *expert iteration*, and is a form of *amortized planning* in the MCTS literature (Dayan et al., 1995; Anthony

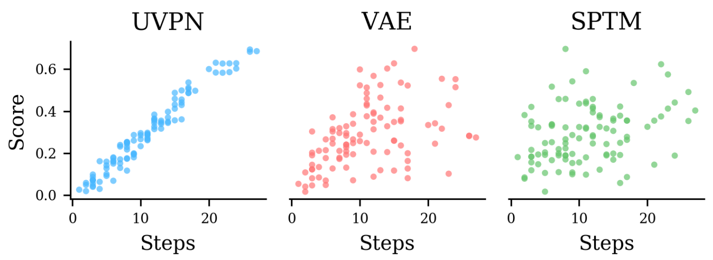

Figure 5: Distance Predicted by Various Methods in Visual Maze. This is an easy domain for VAE because the visual features are sufficient for learning a distance between near neighbors. VAE tend to under estimate distances, due to the embedding being crumpled. SPTM only learns a local metric for 1-step whereas we show prediction up to 30 planning steps in these plots. Therefore linear dependency is not visible. UVPN learns an accurate distance metric linear w.r.t path length. We use the number of discrete steps to provide intuition on the length of the paths.

et al., 2017; Hamrick et al., 2019). It can also be interpreted as augmenting an otherwise space-hungry graph search algorithm with a lossy long-term memory with constant space overhead.

6.6 The Advantage

We focus on the discrete action set in this exposition, but similar considerations apply to other types of parameterization for continuous and hybrid action spaces. The value estimate UVPN learns allows us to compute directly the advantage for a particular transition $(s;a;s')$ for a given goal g. UVPN treats $V(s;g)$ as the *negative shortest path-length* between s and g. This is the expected value for an optimal policy $V_*(s;g) \quad Q(s;a;g)$,

$$Q(s;a;g) = V_*(s;g) + A(s;a;g) \;\Rightarrow\; A(s;a;g) = 0 \tag{4}$$

where $a = \arg\max_a Q(s;a;g)$. This is because the advantage describes the *"detour"* of going through a neighbor s'

$$A(s;a;g) = V(s;g) \quad V(s;s') + V(s';g) \,: \tag{5}$$

Under this formulation, the maximum of 0 is achieved when s' is on the shortest path between s and g (Wang et al., 2015).

The value baseline we use in policy gradient methods are the *behavior state values* conditioned on an underlying sampling process. There is a gap between the value estimate for this value baseline, versus those for the optimal value that UVPN learns. To learn the behavior policy, we learn the value baseline with an L_1 loss towards the average advantage

$$L_{\text{baseline}} = \min \|A(s;a;g); V(s;g)\|_1 : \tag{6}$$

6.7 A Dissection: Accurate Value Estimate With Few Data

Having inspected the metric map, we can now look at the accuracy that UVPN predicts in comparison to VAE and the local metric function learns. Each datapoint in Fig.3. is the predicted distance between two randomly sampled points, versus the shortest path-length in between. Note that in the Maze environment this is different from the Euclidean distance because of the wall. Not surprisingly, UVPN shows good linear dependency, whereas VAE tend to under estimate the path-distance. This occurs because the embedding VAE learns is warped, making the L_2 distance on this embedding less than the actual length of the geodesics. The local metric that SPTM uses is only trained up to 1 step, failing to exhibit linear dependency. We use the number of planning steps for the x-axis to give an intuition of how far out these pairs are w.r.t. the agent's step size.

On the Maze domain (Fig.2a), a blue robot is asked to reach specific locations in a square arena separated by a wall to the left. We collect 400 pairs of transitions uniformly sampled in this domain to train a local metric function using the noise-contrastive objective from Yang et al. (2019). Then we use the same batch to build the graph. We set the threshold for local neighborhood to 1, the regression