THINK, ACT, AND LEARN ON AN EPISODIC MEMORY GRAPH

Ge Yang*†**, Amy Zhang***†**, Ari Morcos**†**, Joelle Pineau**†**, Pieter Abbeel**§**, Roberto Calandra**†
†Facebook AI Research, §University of California, Berkeley

ABSTRACT

Designing agents that can rapidly adapt to changing circumstances while mastering a wide variety of tasks remains an open challenge. Model-free methods require a large amount of trial-and-error, and struggle with learning universal value functions with deep neural networks. Search-based methods can accomplish a dynamic set of goals, but are sensitive to model deviation away from the true environment dynamics. 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 action-conditioned value-prediction 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 within the same agent to alleviate the deficiency of each.

1 INTRODUCTION

Recent reinforcement learning methods often take a delayed approach to learning, where learning occurs after batches of experience are collected. One bottleneck at the basis of this delayed learning approach is value-bootstrapping. In Q-learning, value-bootstrapping introduces learning instabilities that force one to adopt a slow-moving target network, and batched-replay (Mnih et al., 2013). These measures limit how quickly new rewards 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 (Silver et al., 2018; 2017). Finally, recent model-based methods that learn *latent, partial models* for planning is a promising direction (Oh et al., 2017; Schrittwieser et al., 2019; Farquhar et al., 2017), but the model learning typically occurs through stochastic gradient descent, limiting the test-time adaptability within a single episode.

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 as events occur (Botvinick et al., 2019), which relies on features provided by much slower learning processes. Work in meta-learning and episodic control have shown the importance of memory in building agents that can pick up new task instantaneously (Wang et al., 2019; Duan et al., 2016; Ritter et al., 2018; Pritzel et al., 2017; Blundell et al., 2016). Along 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 model-based unrolls for learning state representations (Yang et al., 2019). Nevertheless, how to take full-advantage of this type of structured model in a model-based policy learning scenario, and how to minimize modeling error in the face of nonstationarity, both remain under-explored problems.

In this paper, we study how to integrate value-learning with episodic memory graphs in a tight learning loop, to obtain an agent that can distill domain knowledge into reactive skills, yet retain 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 ever-evolving episodic memory graph that continuously incorporate new knowledge. UVPN extends aforementioned work that constructs graphs on the replay buffer, to treat the graph as the basis for a *value-prediction model* (Oh et al., 2017; Schrittwieser et al., 2019) that predicts generalized value estimates for long-term goals.

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 distilled estimator has less variance, enabling its use as a critic for learning a policy. Critical for online adaptation and learning a successful policy, the tabular transition model drives the exploratory behavior of the agent in a form of causal structure learning. The graph is updated as new experience occurs, instantaneously affecting the agent's behavior in non-stationary environments. This type of fast learning depends on components of the the rest 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 extends prior work that constructs a topological map on the replay buffer using a learned local metric (Savinov et al., 2018; Eysenbach et al., 2019). We refer to this graph as the *episodic memory graph* (EMG).

2.1 Value Distillation

Following Kaelbling (1993), we define the value function as $V^*(o, o_g) = -d(o, o_g)$, where d is the shortest path distance metric between o and o_g. Plan2vec (Yang et al., 2019) offers an objective to acquire a low-variance estimate by directly regressing towards the length of shortest paths τ^* found on the graph. Factoring $V = \|\phi(o) - \phi(o_g)\|_p$ allows the value network to generalize to new goals.

$$^{\text{(plan2vec)}}L_{\text{distill}} = \left| V_\phi(o, o_g) - \sum_{r_t \sim \tau^*} [r_0 + \gamma r_1 + \cdots] \right| . \tag{1}$$

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 for directing 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 estimator learned via Eq.1, in combination with a small amount of action-state transition data can be used to learn an advantage function that can reach long-range goals. Following the approach in dueling-network and policy gradient methods (Wang et al., 2015; Schulman et al., 2015), the optimal policy is the one that picks the most advantageous action

$$\underset{\text{discrete}}{\pi^*(a|s, g)} := \arg\max_a A(s, a, g) = \arg\max_a \frac{\exp \beta A(s, a, g)}{\sum_{a_i} \exp \beta A(s, a_i, g)} . \tag{2}$$

whereas for a sampled transition tuple $\langle s, \bar{a}, s' \rangle$ and a goal g, the advantage is

$$A(\bar{a}|s, g) = V(s, g) - [V(s, s') + V(s', g)] . \tag{3}$$

The regression objective in Q-learning only supervises the action $\bar{a}$ that is sampled at s. Hence the partition function of the action distribution $\forall a_i \in \mathcal{A}$ at state s is under-constrained. We therefore combine Eq.3 with an inverse model objective. For observation transition tuples $\langle o, a, o' \rangle$ and o_g,

$$L_{\text{UVPN}} = \|A(\bar{a}|o, o_g) - V(o, o_g) + V(o, o') + V(o', o_g)\|_1 - \log \frac{\exp \beta A(\bar{a}|o, o')}{\sum_{a \sim \mathcal{A}} \exp \beta A(a|o, o')} . \tag{4}$$

Reaching farther with goal-relabeling. Each transition tuple above is augmented with a randomly selected goal g. This is a form of goal-relabeling (Andrychowicz et al., 2017; Warde-Farley et al., 2018). This enables UVPN to learn longer-reach action value estimate than the duration of individual sampling episode.

05_methods/figures/v2/uvpn_main_v2.pdf

Figure 1: Universal Value Prediction Network.

2.3 Episodic Model Update and Causal Structure Learning

Fast learning requires focused update to an agent's world model or cost function (Tamar et al., 2016a). This is antithetical to the assumption that gradient-descent makes, that data are uniformly sampled from an i.i.d. For this reason, when we attempt to update the value of a single state via gradient descent, undesirable interference with other states generally occur. On the other hand, the episodic memory graph is a structured model of the environment that allows exactly this type of insular update. Each edge can be considered a *local model*, and if an agent fails to reach the next node, we can consider this edge invalidated by the data and permanently remove it from the graph as a form of causal structure learning that happens instantaneously. Because the agent is driven by plans made using the same model, the high-level planner prioritizes weaker edges, which can be considered a form of hypothesis-driven exploration where faulty shortcuts are more likely to be proposed. In classical planning literature, such bias towards optimistic estimate of *distance-to-goal* is what guarantees the discovery of shortest paths. In a sample-then-learn paradigm of policy search under modern reinforcement learning context, this mechanism guarantees model and policy improvement. Empirically, iteratively improving the episodic model in a tight *think, act, and learning* loop is critical to the performance of the parametric policy in contact-rich environments.

We additionally mark each (directed) edge as visited, preventing the agent from re-planning in a cycle. We reset this visitation mask over edges when the goal is reset.

2.4 Dynamic Episodic Memory Graph with Prioritized Cache Replacement

The space complexity of the graph grows quadratically with respect to the number of vertices. To allow the agent to learn continuously, we treat the graph as a cache, and maintain constant-space by imposing a local replacement rule that prioritizes vertex removal by the local density around that vertex. $w_i = \sum_{j \in N(v_i)} 1/e^p_{ij}$. Alternatively, one can maintain a fixed threshold and remove vertices

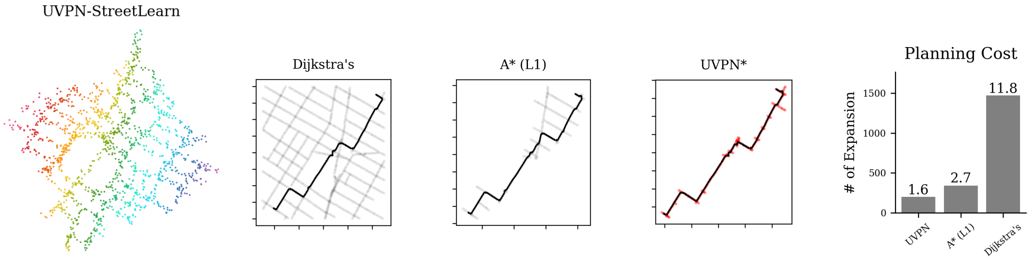

Figure 2: **Left**: Embedding Learned by UVPN on Street learn. **Middle**: Nodes Expanded During Search. Gray dots show the nodes in the graph that are covered during search. With a strong heuristic (L1 distance), A* is more economical than Dijkstra's. But with a learned heuristic UVPN has even lower 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.

before the graph reaches the size limit. Our efficient graph implementation achieves more than 1k insertions per second on a graph with 10k vertices.

2.5 Goal-relabeled Expert Distillation (GRED)

For completeness, we propose an alternative scheme for learning a parameterized policy that does not rely on learning a long-term value estimate. In goal-relabeled expert distillation, we use the inverse model to label plans made on the episodic memory graph with action proposals. We include details on this method and empirical evaluation in the appendix.

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 limitations. 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 Dijkstra's used by SPTM and SoRB 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 reactively 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 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 Broadway's triangular shape. UVPN*, using the learned value function, expands only a 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 $2500\times$[2].

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

[2] $1500/0.6 \approx 2500$

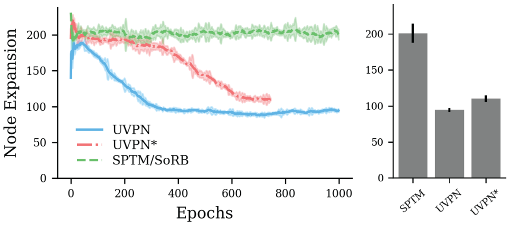

Figure 3: Planning cost during learning on the Maze domain. UVPN uses the learned *distance-to-goal* value (H) as a planning heuristic. UVPN* is similar to A* in that it uses path-length so-far (D) in addition to the distance estimate (H). One can see the planning cost goes down as learning progresses. Bar chart shows the final 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.

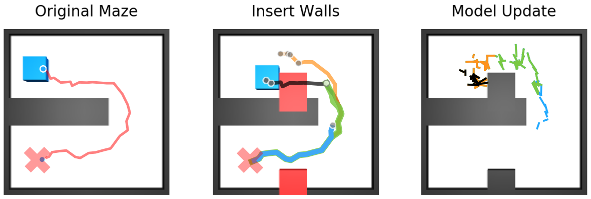

Figure 4: Adapting to Changes in the Environment. **Left**: Original plan made by the agent in Maze. **Middle**: we insert a vertical separation, shown as two red blocks. Colored lines shows the forward plan at 4 separate steps in the same episode. **Right**: colored segments showing the edges pruned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

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 memory-intensive 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 mastered the maze. To get to the goal on the other side, it has learned to move around the wall (Fig.6a). Now we insert a 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 updates the model by pruning edges from the graph that the agent's low-level policy is not able to implement (Fig.6c). The re-plan reflects these changes, and the agent is able to get 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.

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.

Bansal, S., Calandra, R., Chua, K., Levine, S., and Tomlin, C. Mbmf: Model-based priors for model-free reinforcement learning. September 2017.

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.

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.

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.

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.

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.

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

Farquhar, G., Rocktäschel, T., Igl, M., and Whiteson, S. Treeqn and atreec: Differentiable tree-structured models for deep reinforcement learning. October 2017.

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.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. Continuous deep q-learning with model-based acceleration. March 2016.

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.

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.

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.

Jurgenson, T., Groshev, E., and Tamar, A. Sub-goal trees – a framework for goal-directed trajectory prediction and optimization. June 2019.

Kaelbling, L. P. Learning to achieve goals. *IJCAI*, 1993. ISSN 1045-0823.

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.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Oh, J., Singh, S., and Lee, H. Value prediction network. July 2017.

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.

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.

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.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. June 2015.

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.

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.

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.

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.

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.

Yang, G., Zhang, A., Morcos, A., Pineau, J., Abbeel, P., and Calandra, R. Unsupervised representation learning by latent plans. 2019.

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 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 $V(s;\theta')$ to a goal-conditioned, *universal value function* $V(s,g;\theta)$. 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,* $\text{Search} : (G, s, g) \rightarrow [s_i]$, *where* G *is the state graph,* s *and* g *are the root node and the goal, which returns an ordered list of*

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

Figure 5: Heuristic Search

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)

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.

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 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 $V_{\pi^*}(s, g) \equiv Q^*(s, a, g)$,

$$Q^*(s, a, g) = V_{\pi^*}(s, g) + A(s, a^*, g) \Rightarrow A(s, a^*, g) = 0 \tag{5}$$

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

$$A(s, a, g) = V^*(s, g) - \big[V^*(s, s') + V^*(s', g)\big]. \tag{6}$$

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

$$L_{\text{baseline}} = \min_\theta \|A(s, a, g), V_\theta(s, g)\|_1. \tag{7}$$

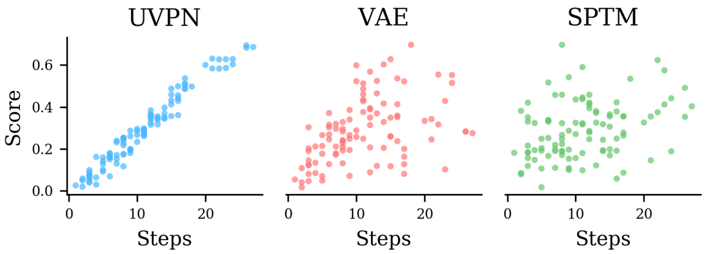

Figure 6: Distance Predicted by Various Methods in Visual Maze. This is an easy domain for VAE because the visual features are sufficient for learning a distance between near neighbors. VAE tend to under estimate distances, due to the embedding being crumpled. SPTM only learns a local metric for 1-step 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.

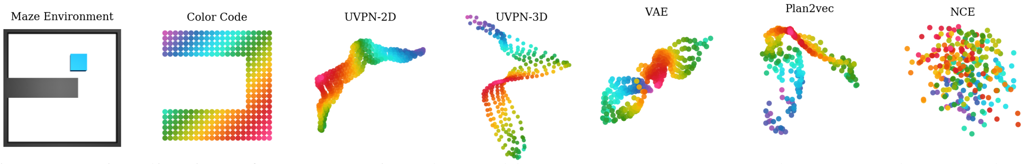

Figure 7: Visualization of representations learned by UVPN versus VAE, plan2vec and NCE. These results are selected randomly without cherry-picking. a) A rendering of the Maze domain. Blue square is the robot. The square arena is separated by a wall to the left. b) the color code for the sample images used for visualization. c) UVPN using a 2D projection. The C-shape of the domain is stretched, so that the L_2 distance reflects the path-length in-between. d) UVPN with a 3-D latent space.

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 target for 1-step neighbors for the local metric. We opt to project to a 2-dimensional latent space, and directly visualize the latent vectors. for UVPN, we also include a 3-dimensional projection. We compare the latent representation UVPN learns against variational autoencoder (VAE), plan2vec, and noise-contrastive embedding (NCE) used by SPTM in Fig.2. To generate the image observations, we sample possible robot positions on a color-coded grid shown in Fig.2b. Details on architecture and hyper parameters are in the appendix.

Interestingly, this shows that a distance *metric map* of the domain arises naturally from the *topological map*.

6.8 Hyper Parameter and Architecture

Value Estimation Experiments All baselines use a Siamese network, where the convolution trunk is a 18-layer ResNet with a coordinate convolution on the input Liu et al. (2018). the kernel is an L_2

norm on the latent vectors, which applies pressure on the learned embedding for the 2D case. For this reason we also supply 3D embedding result for UVPN. We omit the result for Q-learning, as it fails to converge with this architecture. Similar challenges with learning a UVFA from image inputs has been reported previously Jurgenson et al. (2019); Florensa et al. (2019); Hartikainen et al. (2019).

Hyperparameters We use a learning rate of $3e^{-5}$ and Adam optimizer for all training. We use a batch size of 16 for the value function and the local metric, and 256 for the advantage weighted regression. We clip and normalize the advantage weight so that the mazimum weight is 1.