Enhanced Scene Specificity with Sparse Dynamic Value Estimation

Jaskirat Singh & Liang Zheng
College of Engineering and Computer Science
Australian National University
Canberra, Australia
jaskirat.singh,liang.zheng@anu.edu.au

Abstract

Multi-scene reinforcement learning involves training the RL agent across multiple scenes / levels from the same task, and has become essential for many generalization applications. However, the inclusion of multiple scenes leads to an increase in sample variance for policy gradient computations, often resulting in suboptimal performance with the direct application of traditional methods (e.g. PPO, A3C). One strategy for variance reduction is to consider each scene as a distinct Markov decision process (MDP) and learn a joint value function dependent on both state s and MDP $\mathcal{M}$. However, this is non-trivial as the agent is usually unaware of the underlying level at train / test times in multi-scene RL. Recently, Singh *et al.* [1] tried to address this by proposing a dynamic value estimation approach that models the true joint value function distribution as a Gaussian mixture model (GMM). In this paper, we argue that the error between the true scene-specific value function $V(s, \mathcal{M})$ and the predicted dynamic estimate $\hat{V}(s, \mathcal{M})$ can be further reduced by progressively enforcing sparse cluster assignments once the agent has explored most of the state space. The resulting agents not only show significant improvements in the final reward score across a range of OpenAI ProcGen environments, but also exhibit increased navigation efficiency while completing a game level.

1 Introduction

Training on environments comprising of multiple scenes / variations from the same domain task (*e.g.* different levels from a video game), has become a powerful strategy for countering over-fitting in deep reinforcement learning [2–8]. However, such an approach comes at the price of increased sample variance in policy gradient computations [9, 10]. The high variance necessitates using more samples [11], and thus, training high performance agents on these environments [9, 12–15] invariably involves increasing the sample size per update step through the use of multiple parallel actors [16–18]. While parallel sample collection helps in stabilizing the learning process, the obvious disadvantages of lower sample efficiency and higher hardware constraints, suggest the need for specialized variance reduction techniques in multi-scene RL.

One such strategy is to replace the traditionally used scene-generic value function $V(s)$ with a scene-specific estimate $V(s, \mathcal{M})$ while computing the advantage function [1]. However, in the absence of information about the operational scene at train / test times, learning the scene-specific value function presents a challenging problem. Recently, Singh *et al.* [1] showed that while a fine estimation of the joint value function is not feasible, a coarse approximation can be obtained by dividing the value function distribution into multiple clusters and then using episode trajectories to predict the assignment of the current state to each cluster (refer Section 2.2 for details).

Preprint. Under review.

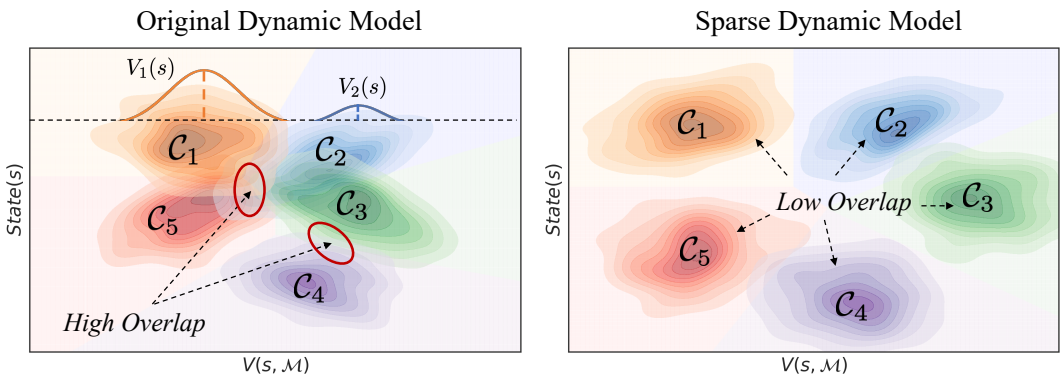

Figure 1: (**Left**) The clusters $\{\mathcal{C}_1, \mathcal{C}_2 ... \mathcal{C}_N\}$ originating from the original dynamic value estimation [1] can be approximated as multi-variate gaussian functions in $(s, V(s, \mathcal{M}))$ space. The overlap regions correspond to points of high confusion and are usually characterized by critical states / tricky obstacles [1]. We claim that such an overlap constitutes a waste of information available for joint value estimation and can be avoided by increasing the spread of learned cluster means (**Right**).

In this paper, we show that the scene-specificity of the dynamic value estimates can be further enhanced by enforcing sparse cluster assignments, once the agent has explored most of the state space and thus learned a good enough approximation of the cluster parameters. The sparse cluster probabilities along with application of traditional value function loss, have a combined effect of spreading out the learned clusters in $(s, V(s, \mathcal{M}))$ space. We claim that the adjustment of dynamic clusters in this manner reduces the overall value function prediction error and support it with extensive testing on OpenAI ProcGen [9] environments. Fig. 1 provides an overview of our method.

The main contributions of this paper are summarized as follows.

- We introduce a novel *confusion-contribution* loss for improving dynamic value estimation (DVE) [1]. The proposed loss decreases the overlap between learned dynamic clusters by progressively enforcing sparse cluster assignments.
- We demonstrate that the sparse clusters divide the overall state space into distinct sets of game skills. The collection of these skills represents a curriculum that the agent must master for effective game play.
- By comparing the game level trajectories for the non-sparse and sparse dynamic models, we show that the high navigation efficiency of our method and its tendency to limit unnecessary exploration, presents an effective alternative to explicit reward-shaping [19–21], for penalizing longer episode-lengths / reward-horizons in multi-scene reinforcement learning.

2 Relevant Background

2.1 Problem Setup

The multi-scene learning problem is characterized by a set of MDPs $\boldsymbol{\mathcal{M}} : \{\mathcal{M}_1, \mathcal{M}_2 ... \mathcal{M}_N\}$. Each MDP $\mathcal{M}$ is defined by state space $\mathcal{S}_{\mathcal{M}}$, transition probabilities $\mathcal{P}_{\mathcal{M}}(s_{t+1}|s_t, a_t)$, reward function $r_{\mathcal{M}}(s_t, a_t, s_{t+1})$, discount factor γ and the common action space $\mathcal{A}$. The agent with policy $\pi(a|s)$ then interacts with a randomly chosen MDP to generate a trajectory $\tau : \{s_0, a_0, s_1, a_1, ...s_T\}$ with total discounted reward $\mathcal{R}_\tau = \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t, s_{t+1})$. We aim to learn a policy π^* such that the expected reward over the tuples $(\mathcal{M}, \tau)$ is maximized, *i.e.*, $\pi^* = \arg\max_\pi \mathbf{E}_{\tau,\mathcal{M}} [\mathcal{R}_{\tau,\mathcal{M}}]$.

2.2 Revisiting Dynamic Value Estimation

Singh *et al.* [1] show that the true value function distribution across different scenes resembles a Gaussian Mixture Model and thus can be divided into clusters. The main idea of dynamic value estimation is to enforce muti-modal distribution learning by modelling the scene-specific value function as weighted sum over the mean value estimates for these clusters. Mathematically,

$$\hat{V}(s_t, \mathcal{M}) = \sum_{i=1}^{N_b} \alpha_i(s_t, \tau^{t-})\, \hat{V}_i(s_t) \qquad s.t. \quad \alpha_i > 0, \quad \sum_{i}^{N_b} \alpha_i = 1, \tag{1}$$

where τ^{t-} is the trajectory till time $(t-1)$, N_b is the number of clusters and α_i, $\hat{V}_i(s)$ represent the cluster assignments and the value function mean for the i^{th} cluster, respectively.

From a qualitative perspective, [1] also show that the distribution of cluster assignments (α_i) provides important intuition about the nature of states, and define two metrics to analyse the same, *confusion* and *contribution*. Confusion (δ) is a measure of uncertainty as to which cluster, the current state-trajectory pair $\{s_t, \tau^{t-}\}$ belongs to. On the other hand, contribution (ρ), as the name suggests, determines the 'contribution' of a cluster in the overall value function estimation across a general trajectory sequence $\tau : \{s_0, a_0, s_1, a_1, ...s_T\}$. Formally, confusion and contribution are defined as,

$$\delta(s_t, \tau^{t-}) = \frac{1}{N_b . \sum_i \alpha_i^2(s_t, \tau^{t-})}, \qquad \rho_i(\tau) = \frac{1}{T} \sum_{t=1}^{T} \delta(s_t, \tau^{t-})\, \alpha_i(s_t, \tau^{t-}). \tag{2}$$

3 Motivation

3.1 Minimizing Cluster Overlap

As shown in Fig. 1, we note that the original dynamic model leads to clusters with high overlap (high confusion) at critical states [1]. The high confusion states are usually characterized by presence of tricky obstacles / scenarios and are critical to the final episode reward. Given the value estimation model from Eq. 1, it is understandable that the use mean squared error critic loss drives multiple cluster centers towards the true value of these critical states. However, such a behavior is undesirable as it reduces the range of value estimates $\hat{V}(s, \mathcal{M})$ covered through interpolation among cluster means in Eq. 1. Fig. 2 explains how the spread of dynamic cluster means affects the prediction error $||V(s, \mathcal{M}) - \hat{V}(s, \mathcal{M})||$ across $\mathcal{M} \in \boldsymbol{\mathcal{M}}$. Consequently, we conjecture that the overall prediction error can be reduced by minimizing the overlap between the learned dynamic clusters.

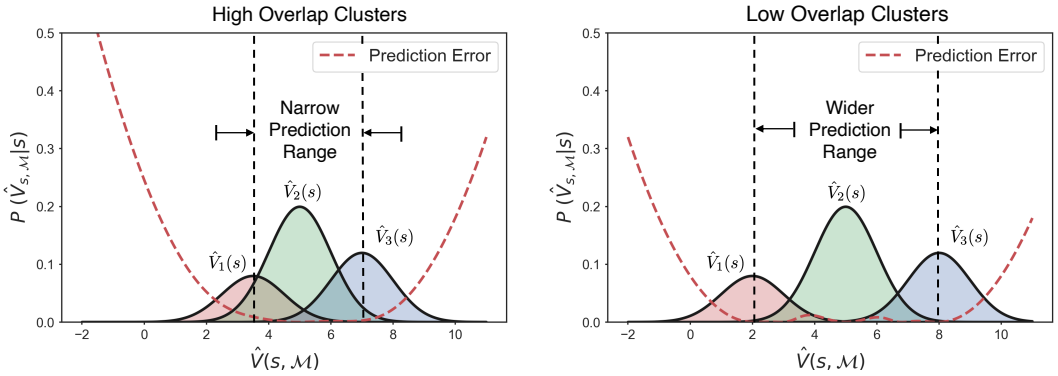

Figure 2: **Qualitative Analysis.** The dynamic value prediction model from Eq. 1 can be interpreted as the interpolation (using α_i) across the learned cluster means $\hat{V}_i(s)$. Thus, as seen above, the prediction error is usually low in the range covered by the cluster centers. The original dynamic clusters with their high overlap are similar to the distribution shown on left, and have a very narrow range of low prediction error. As shown on the right, this region of low prediction error can be expanded by increasing the spread of learned dynamic cluster means.

3.2 Correlation Analysis

As the clusters move far apart from each other, the cluster assignments α_i for a given tuple (s, τ^-) tend towards a one-hot encoding, with the one corresponding to the closest cluster. This implies

that a higher spread in cluster means corresponds to a sparser cluster assignment distribution and can be measured using the confusion δ (refer Eq. 2). Hence, to test the initial validity of the above analysis, we compute the correlation between final model performance and inverse confusion $(1/\delta)$, while training on OpenAI's ProcGen [9] environments. The samples for this testing are collected randomly during the first 50M timesteps of training across 4 distinct runs with the original dynamic model. The Pearson correlation [22] coefficients for various ProcGen games are shown in Fig. 3. The results clearly corroborate our analysis from section 3.1 and show a high correlation between reduced confusion and improved model performance.

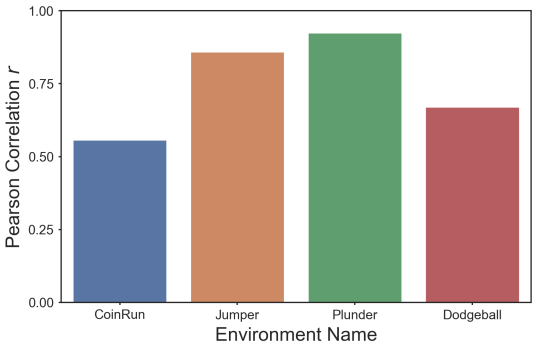

Figure 3: Results showing Pearson correlation [22] coefficient (r) between inverse confusion $(1/\delta)$ and total reward score $(\mathcal{R})$. For most games the correlation coefficient is greater than 0.5, which points to the statistical significance of the analysis done in section 3.1. We next demonstrate how the original dynamic training can be modified to achieve lower confusion in cluster assignments.

4 Sparse Dynamic Value Estimation

Given the analysis from Section 3.2, we note that increasing the inter-cluster mean variance leads to sparser cluster assignment distribution. We claim that the reverse is also true, *i.e.*, an appropriate spread in learned cluster means can be obtained by progressively enforcing sparse cluster assignments followed by adjustment of cluster means. Fig. 4 illustrates this process on a sample GMM distribution.

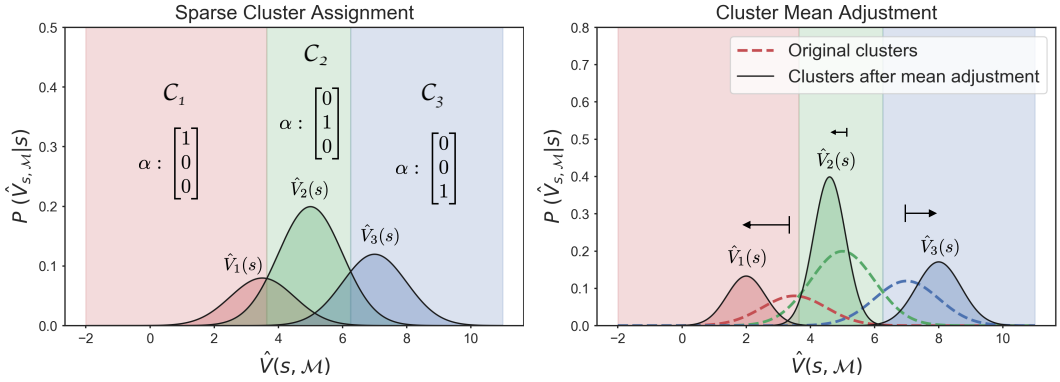

Figure 4: **Illustration.** Example showing how sparse cluster assignments help in reducing the overlap between clusters learned by the dynamic model. (**Left**) Each point $(s, \mathcal{M})$ is allocated to the most probable cluster based on the cluster assignments α_i. (**Right**) The means of each cluster adjust to reflect the expected value estimate of all $(s, \mathcal{M})$ pairs in the modified cluster assignments. Note that the hard sparse assignment is for illustration purposes only. In practice, the sparsity is introduced progressively as the new cluster means are learned through the value function loss.

4.1 Enforcing Sparsity

The sparsity condition is equivalent to maximization of the L2 norm for cluster assignments $\{\alpha_1, \alpha_2, \ldots \alpha_{N_b}\}$ and thus using Eq. 2, corresponds to minimal confusion (δ). However, we note that a mere enforcement of sparsity may encourage convergence to solutions where only one of the clusters is active. We also want to ensure that each cluster contributes equally in the $(s, \mathcal{M})$ space.

To achieve this, we propose the following *confusion-contribution loss*,

$$L^{CC} = k_1 \, \mathbf{E}_{s_t, \tau^{t-}} \left[\log \delta(s_t, \tau^{t-}) \right] + k_2 \, \mathbf{E}_{\tau} \left[\log \left(\sum_{i}^{N_b} \rho_i^2(\tau) \right) \right]. \tag{3}$$

We must emphasize that the state space must have already been well explored by the agent, prior to the application of confusion-contribution loss. If applied prematurely, due to the continuous nature of neural networks, the sparse cluster assignment is incorrectly generalized across the entire state space. This would lead to a detrimental impact on value function estimation for the currently unexplored states. Also, such a mistake is hard to recover from, because for any state s, the sparse assignment ensures that the gradients for all but one cluster are approximately zero.

5 Evaluation on OpenAI ProcGen

5.1 Experimental Design

Training Details. The network design for the dynamic model is quite similar to the one described in [1]. The states are fed through an IMPALA-CNN [23] + LSTM [24] network to output a joint latent representation, used for learning both the policy and the value function. The critic network uses these latent representations to predict cluster assignments α_i and mean value estimates $\hat{V}_i(s)$. Finally, the predicted value function $\hat{V}(s, \mathcal{M})$ is computed using Eq. 1. Similar to [9], the agent is trained using Proximal policy optimization (PPO) [25] with 4 parallel workers. The only point of difference with the original dynamic model is the application of confusion-contribution loss (L_{CC}) at a suitable stage in the training process. The loss coefficients (k_1, k_2) determine the balance between confusion and contribution, and are chosen through extensive hyper-parameter search for each environment.

We test our method on 8 ProcGen [9] environments: CoinRun, CaveFlyer, Climber, Jumper, Plunder, Dodgeball, FruitBot and StarPilot. Note that each game is characterized by a different rate of state exploration and training trajectories. Thus, depending upon the type of environment, we adopt the following strategies for obtaining sparse boosts.

Pre-boost. For games allowing rapid state space exploration at the beginning, the confusion-contribution loss can be applied quite early to promote sparsity. In fact, because the policy gradient and value function loss dominate the initial training updates, we apply the confusion-contribution loss from the start. However, the coefficients (k_1, k_2) are kept moderately small so as to encourage the network to progressively converge to a sparse cluster assignment over the first quarter timesteps. CoinRun, CaveFlyer, Climber and Jumper belong to this set and are labelled as *class-1* environments.

Post-boost. In contrast, other games display a much more gradual expansion of explored state space, exhibiting a positive correlation between episode lengths and the total reward. Sparse-boosting for such environments, can only be applied after the rate of increase of average episode length has declined. Thus, the application of confusion-contribution loss is usually preceded by pre-training with the original dynamic model for 50M timesteps (per worker). Games like Plunder, Dodgeball, FruitBot and StarPilot are part of this set and are labelled as *class-2* environments.

We also train the vanilla-LSTM based RL^2 [24] and non-sparse dynamic models from [1], to show a comprehensive comparison between model performances. For consistency reasons, a pre-training procedure same as the one described above is followed, for all class-2 environments. All results are reported as an average across 4 distinct runs using 500 levels for training.

5.2 Results

Class-1. The sparse model leads to consistent performance improvements in all 4 class-1 environments. Fig. 5 shows the total reward and average episode length curves during the training process for the Caveflyer environment. We clearly see that sparse training leads to significant gains in both final reward and sample efficiency over the regular dynamic model. For instance, we report an increase of 28.3% and 22.4% in the final episode reward, over the non-sparse dynamic model, for the games of CaveFlyer and Climber respectively (refer Table 1).

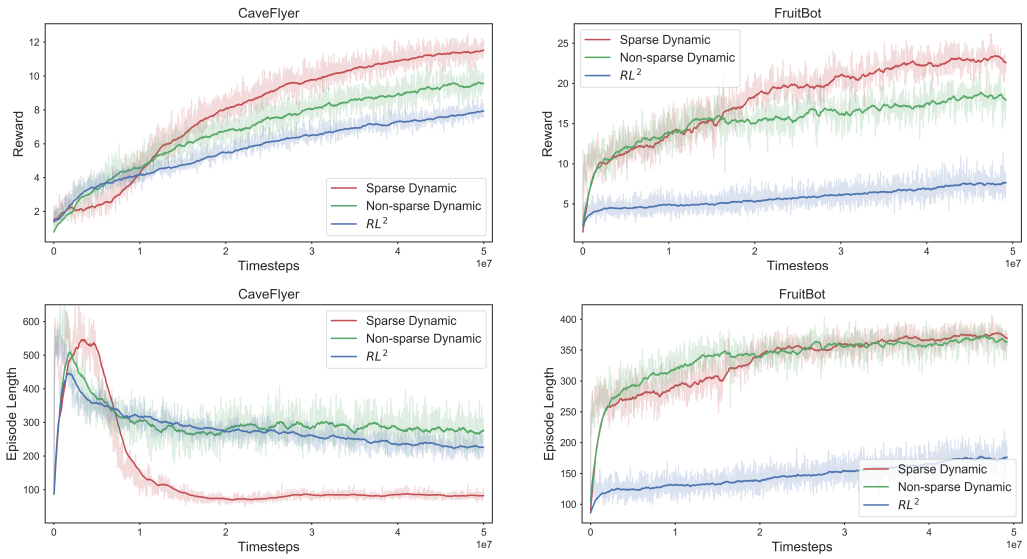

Figure 5: Learning curves for RL^2, sparse and non-sparse dynamic models, illustrating differences in sample efficiency, total reward and and episode lengths.

		Total Reward			Episode Length		
Class	Environment	RL^2	DVE	Sparse DVE	RL^2	DVE	Sparse DVE
Class 1	CoinRun	7.75	9.16	**9.62**	126	78.99	62.1
	CaveFlyer	6.82	9.02	**11.57**	225.6	275.1	75.2
	Climber	7.50	8.14	**10.17**	178.1	226.6	170.5
	Jumper	6.61	6.52	**6.65**	236.4	180.3	78.9
Class 2	Plunder	7.13	17.16	**18.42**	495.1	780.6	739.4
	DodgeBall	10.98	11.25	**12.76**	401.5	459.9	285.8
	FruitBot	7.33	18.32	**23.08**	172.1	364.5	374.2
	StarPilot	17.94	18.08	**19.81**	327.6	342.9	320.2

Table 1: Performance comparison in final reward and average episode length for both class 1 and 2 environments. Our method achieves higher total rewards while needing much shorter episode lengths.

Furthermore, as shown in Fig. 5 and Table 1, the sparse model leads to better reward scores while on average, using much fewer timesteps per episode[1]. We call this phenomenon as enhanced navigation efficiency and delve into it in detail in Section 7.

Class-2. Fig. 5 reports the results for the FruitBot (class-2) environment using the post-boost strategy. While the baseline RL^2 and non-sparse dynamic models show a saturation in model performance with the extended training protocol, the sparse model loss leads to continued gains in reward scores. Interestingly, we also see that a saturation in rate of state space exploration is necessary for getting gains with the sparse model. This is illustrated through the training curves for the game of FruitBot (refer Fig. 5), where relative gains over the non-sparse model occur only after a decline in the rate of increase of average episode length.

For sake of completeness, we report the results for all class-1 and class-2 environments in Table 1.

6 What are Clusters Made of?

Given the non-sparse nature of original cluster probability distribution, Singh *et al.* [1] use the normalized contribution scores as a measure of similarity between the basis MDPs [1] and a particular

[1]Note that the ProcGen environments have no explicit penalty for longer episode lengths.

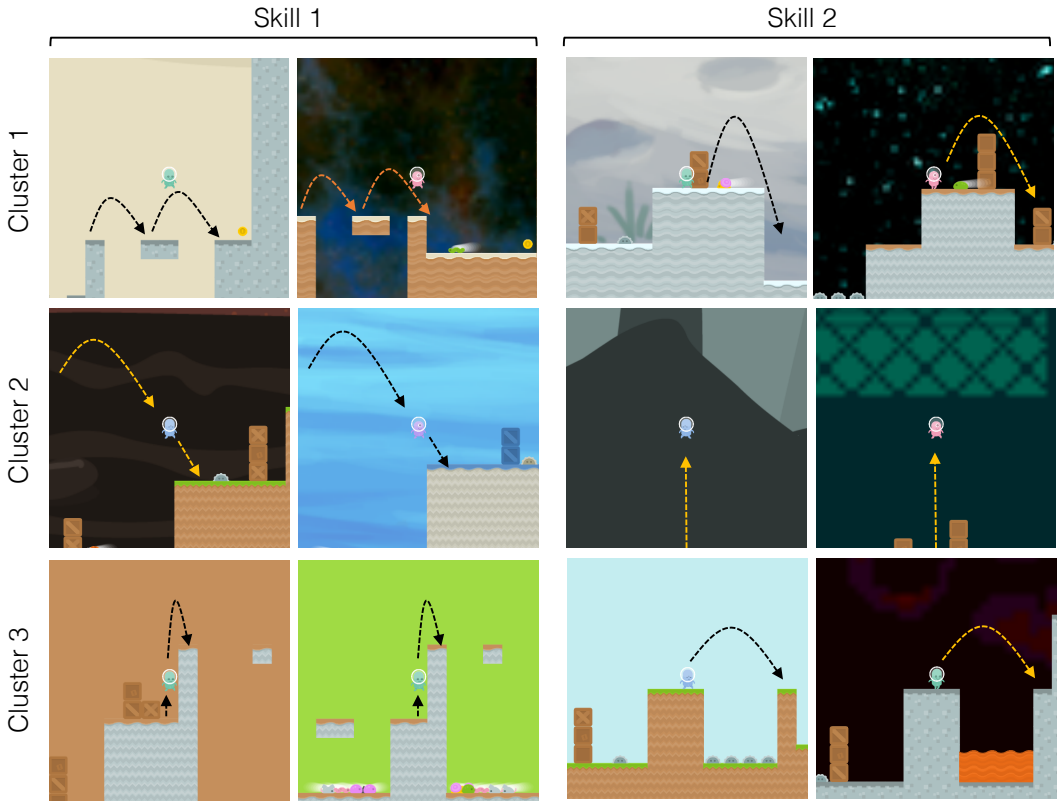

Figure 6: Examples of key obstacles types learned by the each cluster in the CoinRun Environment. We note that the sparse training divides the overall state space into a distinct sets of game skills.

game level. However, such a comparison is not helpful in understanding the key features that differentiate each basis cluster. In this section, we use the sparse property of our method to visualize different obstacle types characteristic of each cluster in the CoinRun Environment.

To visualize the distinguishing features for each cluster, we first extract the set of states $\mathcal{S}_i$ for which each cluster is active. The latent representations (output of the LSTM network) for these states are used to map each $s \in \mathcal{S}_i$ to a two dimensional embedding space using TSNE [26]. This embedding is then manually analysed for clusters to the identify the salient obstacle classes.

Fig. 6 shows some key obstacle types for each cluster. We observe that each cluster is responsible for predicting the value function on a distinct set of obstacles / skills. For instance, cluster-1 is responsible for value estimation in cases like double-jump from one side to another (skill-1) and crossing over moving enemies (skill-2). On the other hand, cluster-2 handles landing after jumps from higher ground (skill-1) and high jumps with very limited visibility of the coming obstacles (skill-2). Finally, cluster-3 takes care of precision climbs (skill-1) and jumps over wide valleys (skill-2).

Thus, we see that each disjoint state space set $\mathcal{S}_i,\ i \in [1, N_b]$ represents a distinct curriculum of game skills that must be learned for mastering the overall multi-scene game environment. This division is analogous to human learning where it is quite common to break down a complex task into a set of manageable skills before attempting the complete task.

7 Navigation Efficiency

A peculiar feature arising as a result of applying sparse boosting can be seen in terms of improved *navigation efficiency*. That is, the agent on average uses fewer time-steps per episode while achieving similar or better reward score. This massive difference in time-steps results from two reasons:

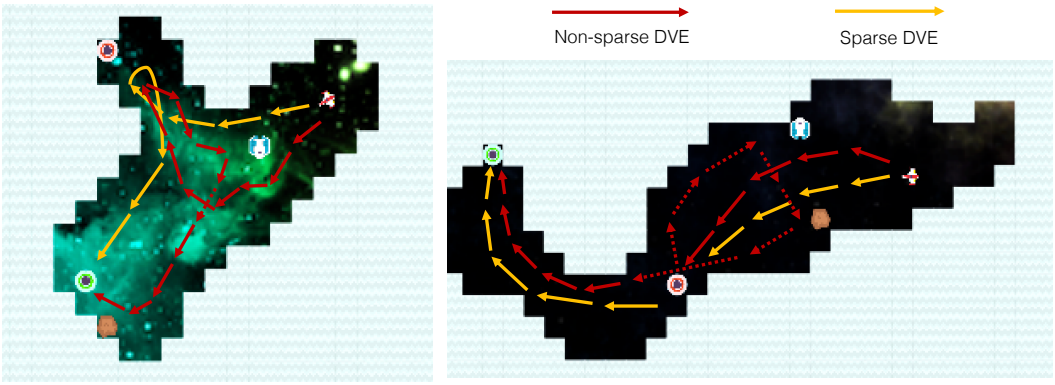

Figure 7: Demonstrating qualitative difference between trajectories for sparse and non-sparse dynamic agents. Our method shows higher efficiency in navigating to the final goals (red & green spheres).

- The tendency to use fewer time-steps is a direct consequence of optimizing the discounted reward function with $\gamma < 1$ [11]. As a result, the agent is incentivized to minimize the number of steps between the current state and the next reward. Hence, more accurate policy updates (lower sample variance) should lead to fewer timesteps.
- As explained in Section 4, the expansion in state space after application of confusion-contribution loss can lead to potential errors in value function estimation. Thus, the sparse dynamic agent learns to maximize the utilization of already explored state space.

In this section, we will analyse the first point in greater detail. We first note that not all critical game states correspond to a high overlap region in the non-sparse dynamic model. At a critical state with low overlap (and possibly high value prediction error), the computation of suboptimal value function, can lead the agent to underestimate the *advantage* of choosing an action leading to a faster route to the final destination / goal. We next try to identify these critical states by comparing episode trajectories for the sparse and non-sparse dynamic agents on the CaveFlyer environment.

Game Description. The goal of the Caveflyer environment is to destroy the red spheres and finally reach the green sphere while avoiding intermediate obstacles. The agent receives a small reward of +3 on destroying a red sphere and an end of episode reward of +10 on successfully reaching the green one. Direct collisions with an obstacle or the red sphere cause immediate episode termination.

Trajectories for both sparse and non-sparse dynamic agents are shown in Fig. 7. We see that the non-sparse agent after destruction of the red sphere (critical state), effectively restarts its search for the next target, while often revisiting already encountered states. In contrast, the sparse agent with its more accurate value estimates, realizes that the expected value for exploring unseen parts of the cave is much higher than revisiting previous states. Doing so not only helps the sparse agent in reaching the end goals much faster, but also eliminates the need for evading obstacles it has already crossed.

We also note that, the balance between the sparse model's reluctance towards state space expansion and maximization of total reward can be modulated through the coefficients of the confusion-contribution loss. In this regard, the high navigation efficiency of our method provides an effective alternative to designing explicit reward shaping [19] penalties for promoting reduced episode lengths.

8 Conclusion

This paper introduces a novel *confusion-contribution* loss which improves the efficiency of the recently proposed dynamic value estimation method, by progressively learning sparser cluster assignments. The resulting dynamic clusters *contribute* equally to the overall value function estimation and display minimal inter-cluster overlap. The proposed approach consistently outperforms the vanilla-LSTM based RL^2 and non-sparse dynamic models on a range of OpenAI ProcGen environments, while on average using much fewer timesteps per episode to complete a game level. Additionally, the sparse training divides the overall state space into disjoint subsets. We show that each subset focuses on a distinct set of game-skills, which draws a strong parallel with the human learning paradigm.

Broader Impact

While this work is largely theoretical, we believe that in the long term, it will have major impact in the upcoming area of AI-inspired learning [27]. Recent years have seen the field of deep reinforcement learning demonstrate tremendous success in achieving super-human performance in complex game play. Deepmind's Alphazero [28], Alphastar [29] and OpenAI's Dota-2 [30] are some salient examples. Each such milestone is followed by an increased public interest to analyse and break down the policy of the trained RL agent into a set of simple skills than can be consumed by a human learner [27, 31]. This process is often manual and involves painstaking analysis across hundreds of game runs. As shown in Section 6, our method does this automatically by dividing the possible game scenarios (states) into distinct sets of game skills. While each set can be composed of other mini-skills, the broad division achieved by our method promises great potential in the development of semi-automatic, AI-inspired teaching tools for human players.

References

[1] J. Singh and L. Zheng, "Dynamic value estimation for single-task multi-scene reinforcement learning," 2020.

[2] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," *arXiv preprint arXiv:1812.02341*, 2018.

[3] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization in deep reinforcement learning through procedural level generation," *arXiv preprint arXiv:1806.10729*, 2018.

[4] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *arXiv preprint arXiv:1806.07937*, 2018.

[5] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *arXiv preprint arXiv:1804.06893*, 2018.

[6] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, "Generalization in reinforcement learning with selective noise injection and information bottleneck," in *Advances in Neural Information Processing Systems*, pp. 13956–13968, 2019.

[7] C. Packer, K. Gao, J. Kos, P. Krähenbühl, V. Koltun, and D. Song, "Assessing generalization in deep reinforcement learning," *arXiv preprint arXiv:1810.12282*, 2018.

[8] F. Sadeghi and S. Levine, "Cad2rl: Real single-image flight without a single real image," *arXiv preprint arXiv:1611.04201*, 2016.

[9] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," *arXiv preprint arXiv:1912.01588*, 2019.

[10] X. Song, Y. Du, and J. Jackson, "An empirical study on hyperparameters and their interdependence for rl generalization," *arXiv preprint arXiv:1906.00431*, 2019.

[11] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[12] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," *arXiv preprint arXiv:1712.05474*, 2017.

[13] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman, "Gotta learn fast: A new benchmark for generalization in rl," *arXiv preprint arXiv:1804.03720*, 2018.

[14] A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, and D. Lange, "Obstacle tower: A generalization challenge in vision, control, and planning," *arXiv preprint arXiv:1902.01378*, 2019.

[15] C. Beattie, J. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, *et al.*, "Deepmind lab. arxiv 2016," *arXiv preprint arXiv:1612.03801*, 2016.

[16] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, *et al.*, "Learning to navigate in complex environments," *arXiv preprint arXiv:1611.03673*, 2016.

[17] M. Wortsman, K. Ehsani, M. Rastegari, A. Farhadi, and R. Mottaghi, "Learning to learn how to learn: Self-adaptive visual navigation using meta-learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6750–6759, 2019.

[18] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, pp. 1928–1937, 2016.

[19] A. D. Laud, "Theory and application of reward shaping in reinforcement learning," tech. rep., 2004.

[20] A. Laud and G. DeJong, "The influence of reward on the speed of reinforcement learning: An analysis of shaping," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 440–447, 2003.

[21] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *ICML*, vol. 99, pp. 278–287, 1999.

[22] J. Benesty, J. Chen, Y. Huang, and I. Cohen, "Pearson correlation coefficient," in *Noise reduction in speech processing*, pp. 1–4, Springer, 2009.

[23] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *International Conference on Machine Learning*, pp. 1407–1416, 2018.

[24] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779*, 2016.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[26] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[27] M. Sadler and N. Regan, *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. New in Chess, 2019.

[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *arXiv preprint arXiv:1712.01815*, 2017.

[29] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[30] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[31] Deepmind, "Alphago teach." `https://alphagoteach.deepmind.com/`, 2017.