Policy Transfer in Lifelong Reinforcement Learning through Learning Generalizing Features

by

Zhiyuan Zhou

A Thesis submitted in partial fulfillment of the requirements for Honors

in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2023

This thesis by Zhiyuan Zhou is accepted in its present form by
the Department of Computer Science as satisfying the research requirement
for the awardment of Honors.

Date _____ _____
George Konidaris, Advisor

Date _____ _____
Michael L. Littman, Reader

Abstract of "Policy Transfer in Lifelong Reinforcement Learning through Learning Generalizing Features" by Zhiyuan Zhou
Brown University, May 2023.

The ability to rapidly generalize is crucial for reinforcement learning to be practical in real-world tasks. However, generalization is complicated by the fact that, in many settings, some state features reliably support generalization while others do not. We consider the problem of learning policies that can be reused, or *transferred*, by identifying state features that generalize across tasks drawn from the same distribution. We propose an ensemble approach, where we use an attention mechanism to learn an ensemble of minimally overlapping state features, leading to an ensemble of policies. We then use a bandit algorithm to learn to identify the generalizing feature in the ensemble and capitalize on that to learn a transferable policy. We show that our approach leads to fast policy generalization for eight tasks in the Procgen benchmark and successful policy transfer in the Atari game Montezuma's Revenge.

Acknowledgements

I am immensely grateful to my advisors for their guidance and encouragement during my research journey at Brown, thankful to my collaborators for their help and ideas, and delighted at the support of my dear friends and family. I would not be where I am today without any of them, and words barely do justice to my gratitude towards all of them. But I will try.

Advisors

First, to my thesis advisor professor George Konidaris. George, thank you for guiding me through my undergraduate research journey and always being there to support me. I joined your lab as a sophomore with little research experience, yet you welcomed me and spent time with me individually brainstorming ideas and showing me how to do rigorous research. You are endlessly humorous, and I will forever remember sitting in your office during group research meetings listening to you crack one joke after another. I will also fondly remember and benefit from the many life advice you shared with me. I have such admiration for your vision and enthusiasm, and it has deeply inspired me during my research journey. Talking with you gave me clarity and encouragement every time I felt stuck. Thank you, George, for advising me and cultivating my interest in RL!

To my advisor professor Michael Littman: Michael, it has truly been a pleasure working with you. I still remember the first time we met, in that taco store in East Providence, and it's been such a fun journey ever since. Your passion for research is contagious, and I have been deeply inspired by it. Talking with you is enjoyable every single time, and you have such an incredible superpower to guide me towards interesting directions and to empower me to figure out the research problems I was chewing. I have very much enjoyed all our conversations, and I look forward to more in the future. You were never one to shy away from the technicalities of research, and I am constantly amazed by your attentiveness to details. Meanwhile, you have such a humorous energy that always lights up our conversations. Thanks for showing me the way into AI research, and I have learned a great deal from you. Thank you, Michael, for mentoring me and believing in me!

Collaborators

Anita de Mello Koch, this thesis wouldn't have came together if it weren't for your collaboration along the way. Thanks for the many hours of brainstorming, chatting about portable skills, and for running the experiments on Montezuma's Revenge presented in this thesis. I won't forget the cheerful energy you always bring into the room.

To Akhil Bagaria, thank you for teaching me all about hierarchical reinforcement learning and always being there to support me when I was at a lost what to do. You have such a natural talent for mentoring—you always had a way of clearly explaining to me the eluding concepts in RL, and were willing to support me even at the lowest levels of debugging. I have greatly enjoyed our collaborations, and look forward to more collaborations and discussion in the future.

To Cameron Allen and Kavosh Asadi, thank you both for guiding me on my first-ever RL research project. You both were incredibly patient with me, and showed me first-hand the various stages of research. Thank you for introducing me to AI research. Thank you, Cam, for always being meticulous with me and for showing me how to write production-quality code.

And to Haotian Fu, thank you for all the conversations about portable skills and the laughters we shared.

Families and Friends

To my parents, thank you for supporting me emotionally and financially throughout my journey at Brown. Thanks for encouraging me to follow my passion, and I am forever grateful for all that you do for me! You guys have always been my unwavering supporters ever since 22 years ago, and I am so fortunate to have you.

To all my dearest friends at Brown who have been there for me through thick and thin, I am so incredibly lucky to have you all and I am immensely grateful for the joy you have brought to my life. I will always carry with me the good memories and conversations we have had together. In no particular order, thanks to Xiao "Sean" Zhan, Edward Xing, Caroline Cheng, Preston Cheng, Yingzhe "Alex" Guo, Kevin Nguyen, Nova Chen, Andrew Yuan, Ivo Su, Michael Xu, Alan Gu, Yuchen "Juliana" Han, David Moon, Joseph Namkung, Luca Fonstad, Vanessa Chang, Sahil Bansal, Jessica Y Li, Linghai Liu, and Yuanhao "Harry" Wang.

Finally, I'd like to thank all my friends at Brown who have made my four years here forever memorable and enjoyable.

Contents

1 Introduction

In recent years, Reinforcement Learning (RL) has outperformed humans in many Atari games [Mnih et al., 2015], learned to play world champion level Go [Silver et al., 2017] and mastered many robotic manipulation tasks [Levine et al., 2016, 2018]. While these achievements are undeniably impressive, they focus on solving tasks in a single controlled environment. The learned policy would often fail when the environment takes on even minor changes. For example, RL agents are known to fail to solve video games when the visual input changes, even though the game dynamics stay exactly the same [Cobbe et al., 2019, 2020]. In the real world, environments constantly change and the task is drawn from some task distribution. For RL to realize its promise in real-world applications, the learned policy must be able to solve all tasks drawn from that distribution. For example, an RL agent should be able to use the same policy to open doors, regardless of the door's color and material. In this work, I aim to use RL to learn a single policy that can solve a distribution of similar tasks.

To mimic learning in the real world, RL agents should not have access to the task distribution a priori. Instead, the agent will encounter tasks drawn from the distribution sequentially, and learn how to bootstrap from prior tasks to efficiently solve the new task with minimal re-learning. This is usually framed as the lifelong or continual RL [Khetarpal et al., 2022] problem. The core difficulty here is that, in practice, the policy will be first learned in a small number of specific tasks instances—possibly just one—without foreknowledge of the circumstances under which it will be applied again in the future. In such cases, there may be many state features over which the initially learned policy could be defined, but not all of which will support reuse. Therefore, the learned policy will overfit to the initial task [Zhang et al., 2018, Justesen et al., 2018, Dubey et al., 2018]. For example, a single policy to open a door might be equally well-defined using features describing the door's location in a global map or features describing the location of its handle relative to the agent, but only the latter will generalize to new doors.

To avoid this overfitting problem, I propose to learn policies that can be reused, or *transferred*, across a distribution of tasks by learning to identify which state features can be generalized and reused. I consider the lifelong learning setting, where the agent is trained sequentially on a series of similar tasks drawn from the same distribution, and aim to efficiently solve the later tasks with minimum re-training by transferring the previously learned policy. I introduce a method where the RL agent learns a collection of diverse state features using an attention-based ensemble [Kim et al., 2018]. When the agent is trained on the very first task, the RL agent will use each of the individual features to learn a policy to maximize the performance (in terms of discounted return), resulting in an ensemble of features. As the agent encounters more and more tasks from the same distribution, it uses a bandit algorithm to identify the single feature that supports policy learning across all tasks, deeming it the *generalizing feature*. We evaluate the algorithm empirically

and show that it leads to better generalization and fast learning on seven out of eight games from the Procgen generalization benchmark [Cobbe et al., 2020] and the Atari game Montezuma's Revenge. The code for this thesis can be found at https://github.com/zhouzypaul/policy-transfer-lifelong-rl.

2 Background and Related Work

2.1 Reinforcement Learning

In the episodic RL setting, an agent operates in a Markov Decision Process with state space S and action space A. $p(s_{t+1}|s_t, a_t)$ denotes the probability of transitioning from state s_t to state s_{t+1} with action a_t. At each time step, the agent receives a scalar reward defined by the reward function $r(s_t, a_t)$. The agent's goal is to maximize the cumulative discounted reward over an episode by finding a policy $\pi(a|s)$ that selects actions at each step.

2.2 Learning from a Distribution of Tasks

There have been many lines of work aimed at learning tasks drawn from similar distributions. In transfer learning [Taylor and Stone, 2009] the goal is to learn on a source task and perform well on a similar target task. However, it has limited real-world applications because it only models two types of tasks, and the real world offers a much wider distribution of similar tasks. Meta Learning [Finn et al., 2017] bears the goal of quickly solving a new task using a small number of example after training on a distribution of tasks. However, it makes the assumption that the distribution of training task is known before hand, which do not always hold true for real-world application. Finally, continual or lifelong learning [Khetarpal et al., 2022] studies the incremental development of complex behavior over a sequence of tasks. Central to lifelong RL is the ability to build on previous behavior and adapt quickly to new situation. It is the natural framework in which to study generalization in RL when the sequence of tasks comes from the same distribution that can be solved with a single policy. Lifelong RL does not assume access to the distribution of tasks, and learns to adapt and generalize previously learned behavior to new tasks. It is the most accurate framework of how RL can operate in the real world. Therefore, we adopt this framework to study policy transfer.

2.3 Policy Transfer in Reinforcement Learning

In this work, I use "policy transfer" and "policy generalizaiton" interchangeably, referring to the policy's ability to be reused in similar tasks from the same distribution. There have been various methods on policy transfer in RL [Taylor and Stone, 2011]. Abel et al. [2018] proposed a method to transfer policy and value functions as a way to initialize a new jump-started policy in a new task, but can only deal with simple tabular

domains. Successor features and the generalized policy improvement framework proposed by Barreto et al. [2018] can successfully transfer policies across tasks, but can only deal with tasks with the same transition dynamics but different reward functions. Sherstov and Stone [2005] have shown successful action transfer by building an action prior from learned task to prune the large action spaces in new task, but cannot deal with changes in the observation space. Policy transfer has also been shown in the context of hierarchical RL under the options framework. Konidaris and Barto [2007] showed that an agent-centric representation, analogous to an egocentric space, would be sufficient for porting options across tasks. However, this requires a hand-designed agent-centric observation space that is not available for all tasks. Gupta et al. [2017] learn skills in an invariant feature space which enables generalization across morphologically different robots. However, the invariant feature space cannot generalize across tasks. This work differs from all previous approaches by using diversity to learn transferable policies. I propose to learn a diverse set of state feature, and as the agent encounters more and more tasks, learn to identify the feature that supports generalization across tasks.

2.4 Diversity through Ensembling

One way of constructing diverse sets of features is by building an ensemble. Ensemble methods train multiple learners on the same task, resulting in a combined model that outperforms each individual model. The quality of the ensemble depends on the quality of each individual and the diversity among them. The attention-based ensemble proposed by Kim et al. [2018] is a deep learning framework, originally intended for deep metric learning, that encourages learning a diverse ensemble of features. The objective for the ensemble is to minimize a combination of the training objective loss and a divergence loss:

$$L(\{x_i\}) = \sum_m L^{(m)}_{\text{train}}(\{x_i\}) + \lambda_{\text{div}} L_{\text{div}}(\{x_i\}), \tag{1}$$

where $\{x_i\}$ is a set of training samples and $L^{(m)}_{\text{train}}$ is the training loss for the m-th learner. In RL, the training loss could be the temporal difference loss, for example. The divergence loss, L_{div}, is weighted by λ_{div} and defined as:

$$L_{\text{div}}(\{x_i\}) = \sum_i \sum_{p,q} \max(0, m_{\text{div}} - d_{\mathcal{Y}}(B_p(x_i), B_q(x_i))^2) \tag{2}$$

where $B_p(x_i), B_q(x_i)$ are the embedding outputs of learners p and q, $d_{\mathcal{Y}}$ is a distance metric in the embedding space, and m_{div} is a margin. Optimizing $L(\{x_i\})$ incentivizes learners to individually perform well on the training task while learning maximally different feature embeddings.

3 Policy Transfer through Generalizing Features

The main challenge to learning a transferable policy is that policies usually overfit to the tasks they are trained on, and cannot generalize to new situations. Ideally, a single policy should be able to be reused across similar tasks. The key to transfer is learning state features that generalize across the task distribution. For example, when learning a transferable open–door policy, the RL agent should learn to ignore state features such as the color and texture of the door, and only focus on the knob position—the generalizing feature. In this section, I will show how policy transfer can be achieved through learning the generalizing feature using an ensemble. By learning an ensemble of diverse features, we increase the chance of finding a feature that generalizes to new tasks. I will first introduce the ensemble of policies method, and then show how a bandit algorithm can be used to learn the generalizing feature.

3.1 Ensemble of Policies

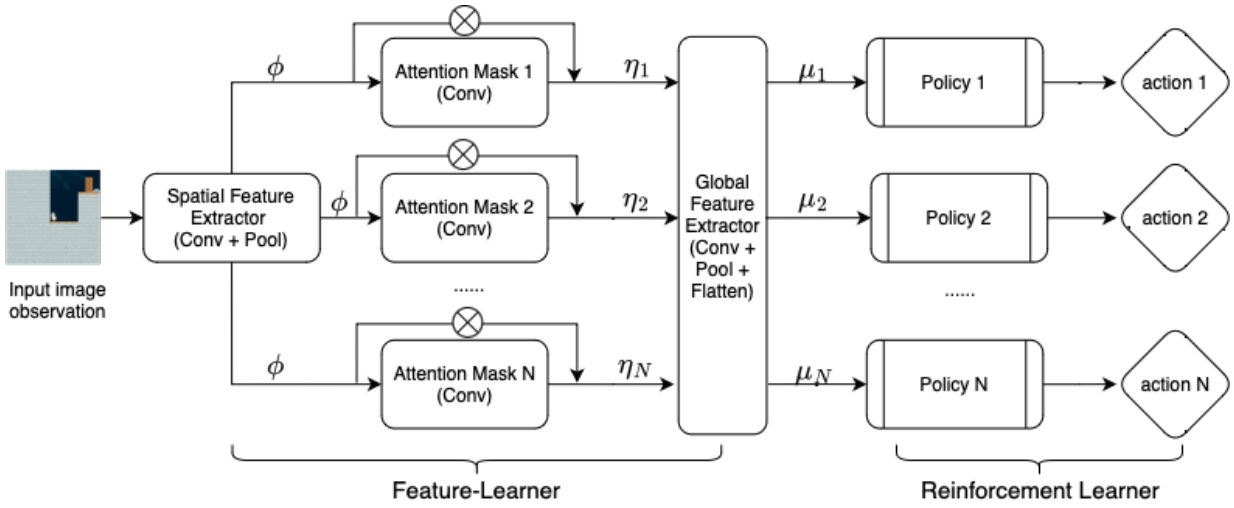

Figure 1: Agent architecture for learning an ensemble of policies using an attention-based ensemble.

Figure 1 shows the agent architecture I use to learn an ensemble of policies. It utilizes an attention-based ensemble [Kim et al., 2018] as a feature-learner to learn N distinct feature embeddings. These embeddings are then inputted to N individual reinforcement learners to learn N candidate policies.

Each feature-learner consists of three components: a spatial feature extractor, an attention module, and a global feature extractor. The spatial and global feature extractor are shared among all learners, but each learner has an individual attention module. The attention module encourages each learner to attend to a different aspect of the input through optimizing a divergence loss, thereby promoting diversity. The input is first passed through the shared spatial feature extractor to obtain the spatial feature embedding ϕ. This embedding is then passed to each attention module to obtain the attention mask, which is multiplied with

the spatial feature embedding to form an attention embedding η_i. Finally, each attention embedding is passed through the global feature extractor to get the final feature embedding μ_i. The μ_i's are optimized to be maximally different from each other through the divergence loss L_{div} in Equation 2. Each feature embedding μ_i is fed into the corresponding reinforcement learner. This method does not put any constraints on the reinforcement learner, and therefore it can be any RL algorithm. The ensemble minimizes the loss defined in Equation 1 with L_{train} as the RL loss. Minimizing this loss incentivizes the agent to learn a good policy using each of the feature sets, while keeping feature sets different from one another.

3.2 Bandit Algorithm

At the beginning of each episode a single ensemble member, known as the *leader*, is selected to govern action selection and all members are trained on the experiences collected. The choice of leader has a major impact on the ensemble performance when transferring to new tasks. If the leader is transferable, then the ensemble will perform well during transfer. Otherwise, the ensemble will likely fail. Since we repeatedly select a new leader every episode, it is natural to frame this choice as an N-armed bandit [Sutton and Barto, 2018] where each policy is an action and the discounted cumulative return is the reward. We select the leader using UCB [Auer, 2002] to balance exploration and exploitation. We select leader L_t at time step t as follows:

$$L_t = \arg\max_i \left[R_t^{\text{cum}}(i) + c\sqrt{\frac{\ln t}{N_t(i)}} \right]. \tag{3}$$

$R_t^{\text{cum}}(i)$ is the cumulative reward of ensemble member i at time t, $N_t(i)$ denotes the number of times that i has been elected leader prior to time t, and constant c controls the degree of exploration. The first part of Eq. 3 can be understood as exploiting good arms, and the second part exploring others.

4 Experiments

4.1 Procgen Experiments

We showcase the effect of the attention-based ensemble on Procgen [Cobbe et al., 2020], a popular benchmark for evaluating generalization in reinforcement learning. Procgen is a suite of procedurally generated game environments, each offering many levels. Each level is a unique instance of the game, differing in appearance—for example background color or object shape—but represents the same underlying game mechanism. We evaluate our approach on 8 games: Bigfish, Climber, Dodgeball, Heist, Jumper, Maze, Miner and Ninja.[1] We choose these 8 games because Cobbe et al. [2020] showed that these are the 8 hardest games for generalization across different levels—They exhibit the largest gaps in learning curves when

[1] To reduce the computation resources required, we set the difficulty level to easy.

trained and tested on different levels. Following Cobbe et al. [2020], we used Proximal Policy Optimization (PPO) [Schulman et al., 2017] as the learning algorithm, but our method is compatible with any RL algorithm. The training loss L_{train} is the PPO loss detailed in Appendix A.2.

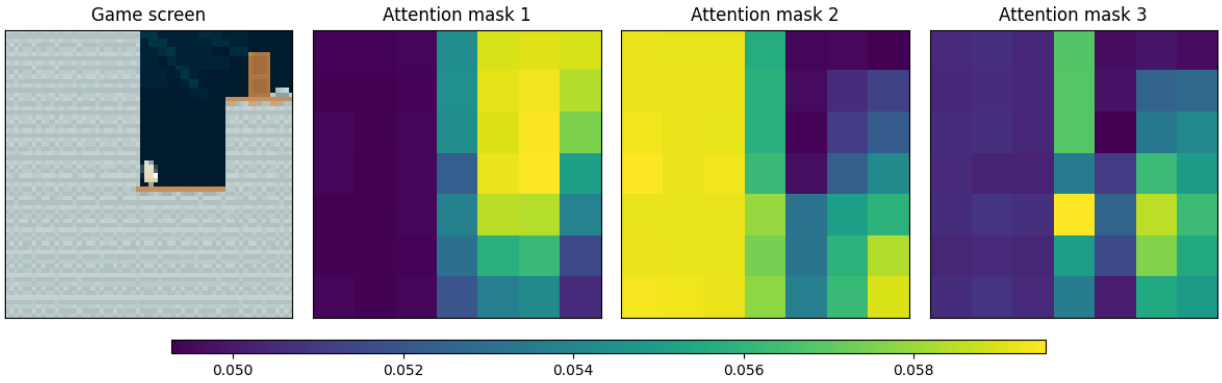

Figure 2: Visualization of the attention masks of a 3-policy ensemble trained on Coinrun, with intensity represented by color. Each attention mask focuses on a different set of features.

4.1.1 Visualizing Attentions

To provide intuition on the effect of the attention modules, we visualize the attention masks at a single time-step of a 3-policy ensemble trained on the game Coinrun[2] in Figure 2. Areas of high attention (yellow) are accentuated by the attention module, while areas with low attention (dark purple) have little effect on the final feature embedding space. Each ensemble member attends to different aspects of the image, and each represents a different set of features. Attention mask 1 focuses on the game background, while attention mask 2 focuses on the way and attention mask 3 focuses on the game character and its surroundings. For other levels of Coinrun where the walls are of different shapes and the background of different colors, the third feature is generalizing feature.

4.1.2 Lifelong Learning on Procgen

We adapt the training scheme of Procgen to better align with the lifelong learning paradigm, as follows. For each game, we randomly select 20 training levels. The ensemble trains for 500k steps on each level sequentially, maintaining knowledge obtained in previous levels. To evaluate, we compare the average within-level performance across the 20 levels. (In other words, we average the learning curves of the 20 sequential levels into one succinct learning curve.) Here, performance is quantified by the mean episodic reward when evaluated on random levels drawn from the selected 20. In Figure 3, we plot the discounted episodic reward

[2]We choose Coinrun instead of the other games we experiment on because its game screen is clean and easy to interpret.

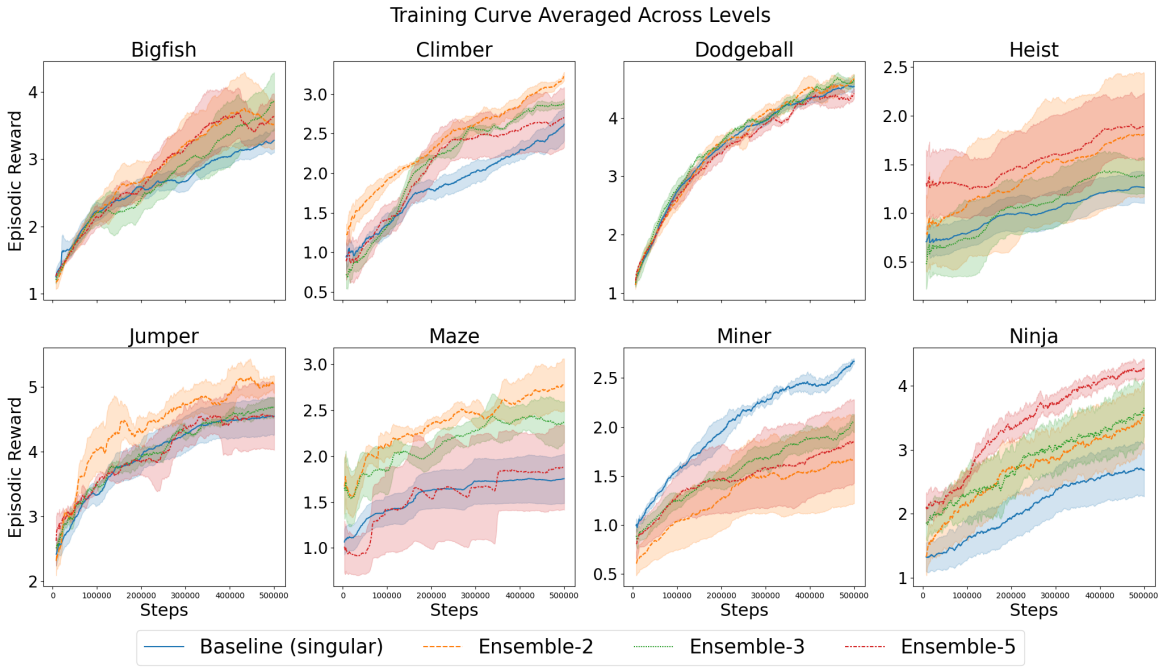

Figure 3: Learning curves of different ensembles of policies on 8 Procgen environments. The 4 different agents have an ensemble of 1, 2, 3, and 5 policies, respectively. Multi-member ensemble agents outperform the baseline agent on six of the environments. (3 random seeds; error bars denote standard error.)

achieved on each game, comparing a baseline (singular) PPO agent to three ensemble agents with ensemble sizes 2, 3, and 5. To maintain comparability, we implement the PPO agent using the same ensemble architecture with only 1 learner in the ensemble, making the divergence loss in Equation 2 zero and hence equivalent to a standard PPO agent.

For seven out of the eight games, ensemble outperforms or performs as well as the PPO baseline, showing better skill transfer across levels. Across these seven games, all ensemble sizes perform better or as well as the baseline. This demonstrates that we can always choose to use an ensemble agent over the baseline RL agent, without the need to tune the ensemble size, to achieve performance gain in expectation. All ensemble agents perform worse on the single game Miner, and we hypothesize that it is because of Miner needs a different exploration–exploitation tradeoff in the bandit algorithm. We further hypothesize that ensemble can perform better than the baseline agent on Miner with more fine-grained bandit hyperparameter tuning or different bandit algorithms. Overall, these results demonstrate that our method can learn transferable policies.

We note that the optimal ensemble size varies for each game. This is expected because differing complexities of the games result in different numbers of possible useful features. When the ensemble size is too small, the learned feature representation may not be fine-grained enough to generalize. This can be seen in Figure 3 in Bigfish and Ninja, where a larger ensemble size leads to increasing performance. On the other

hand, an excessive number of ensemble members can also degrade performance by spending too much time exploring policies that do not generalize, and not enough time exploiting ones that do. An excessive number of ensemble members can also split the generalizing feature up among a few learners in the ensemble, and therefore no single learner will be able to learn the generalizing feature. We can see this degradation in performance in games Climber, Jumper, and Maze.

4.1.3 Perturbation Study on Bandit Algorithms

To justify choosing Upper Confidence Bound algorithm in Equation 3 as the bandit algorithm, we compare UCB's performance to three other bandit algorithms:

1. Sliding-window UCB: Badia et al. [2020] proposed a simple non-stationary bandit algorithm based on UCB to adapt to the change of reward distribution over time. It is a slight modification of the classic UCB algorithm, adding a sliding time window to deal with reward distribution of a limited time frame and adding ϵ-greedy exploration.

2. Exp3: Auer et al. [2002] proposed Exponential-weight algorithm for Exploration and Exploitation (Exp3) to deal with non-stationary bandit problems. Exp3 assigns weights to each arm and use these weights to determine the probability of selecting each arm. The weights are updated based on the received rewards, with higher rewards leading to increased weights. However, to ensure sufficient exploration, the algorithm also introduces uniform random action selection.

3. UCB with gestation: We propose to modify the classic UCB algorithm with an initial gestation period, when the bandit chooses actions uniform randomly. After the gestation period, it chooses actions based on the UCB algorithm. The gestation period is designed to give time for the ensemble-agent to learn good state features before judging their generalization ability.

We evaluate the four different bandit algorithms on the game Bigfish, and the learning curves are shown in Figure 4. Both sliding-window UCB and Exp3 makes ensemble agents learn slower. This is because they alternate exploiting different learner as leaders and cannot identify the generalizing feature. This is perhaps expected as they are both non-stationary bandit formulations, and they spend much more time exploring the different learners over time rather than exploiting them, leading to slow learning. Both the stationary bandit algorithms, UCB and UCB with gestation, helps ensemble agents outperform the baseline, and have similar performances. For simplicity and to reduce the need to tune the gestation time hyperparameter, we choose to use UCB as the bandit algorithm for all experiments.

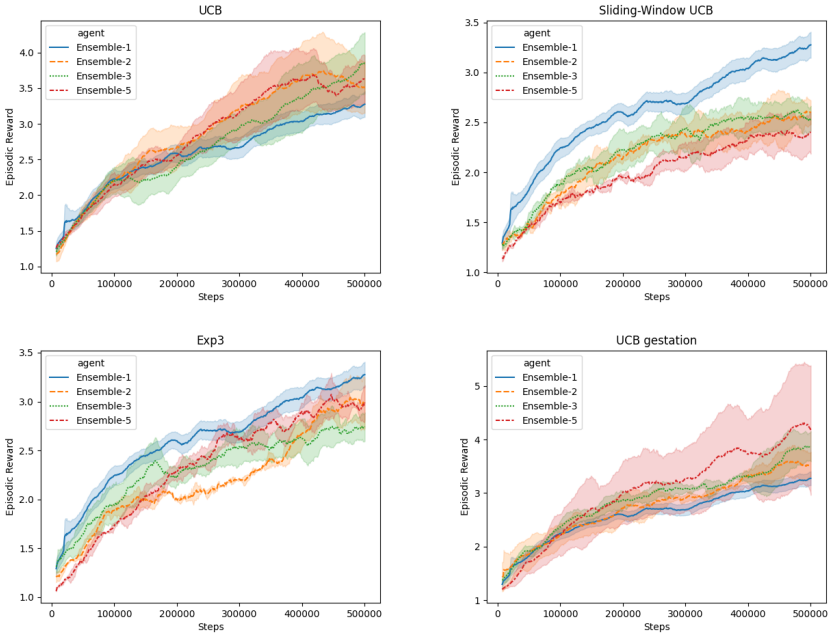

Figure 4: Learning curves of four different bandit algorithms on the game Bigfish. (3 random seeds; error bars denote standard error.)

4.1.4 Perturbation Study on Bandit Exploration

We also draw attention to the importance of the exploration constant c in the UCB algorithm (Equation 3). If c is too small, there is insufficient exploration when learning to choose the leader, essentially collapsing the ensemble agent to a 1-member ensemble. If c is too big, then the ensemble agent will learn slowly because it spends too much time exploring the different leaders and not enough time exploiting them. The effect of different exploration constant can be seen in Figure 5.

4.2 Montezuma Experiments

We further test the capabilities of our method in the Atari game Montezuma's Revenge, where the agent must learn to control a character through multiple rooms filled with enemies, obstacles, and treasures. Montezuma's Revenge is a natural environment for testing policy transfer in lifelong RL because local policies learned in one place can be naturally reused in other places. Here, we consider the `ClimbDownLadder` policy, in which the agent starts on or above a ladder and climbs down the ladder to the next horizontal platform. This policy can be utilized in multiple ladders across the game, even though they are visually different. Figure 6 depicts four such ladders.

We selected 9 ladders in the game, and trained the RL agent sequentially on those ladders. When training on ladder k, the policy is initialized with the learned policy from training on ladder $k-1$. Here, we use an ensemble size of $N = 3$, and use Deep Q-Network [Mnih et al., 2015] as the RL algorithm, as is common for

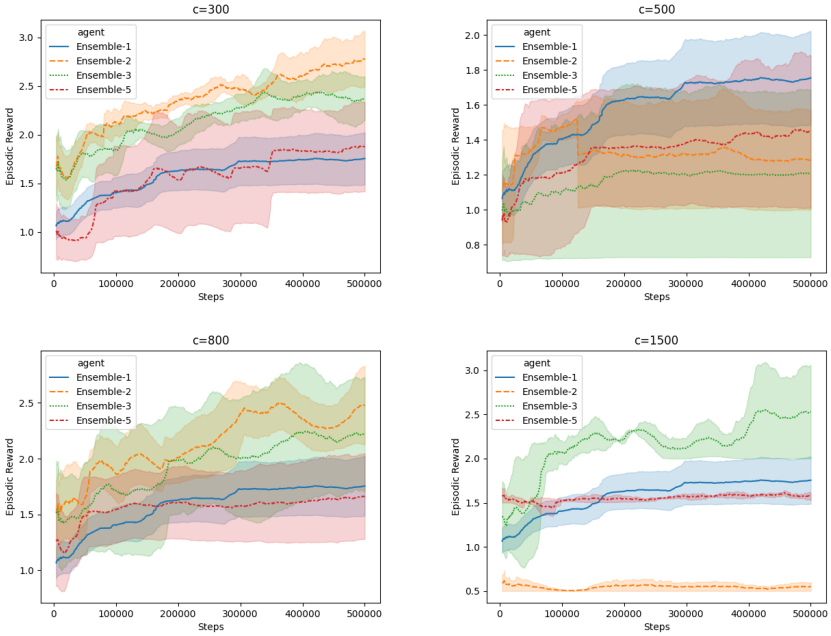

Figure 5: Learning curves of different exploration constant c in the UCB algorithm on game Maze. (3 random seeds; error bars denote standard error.)

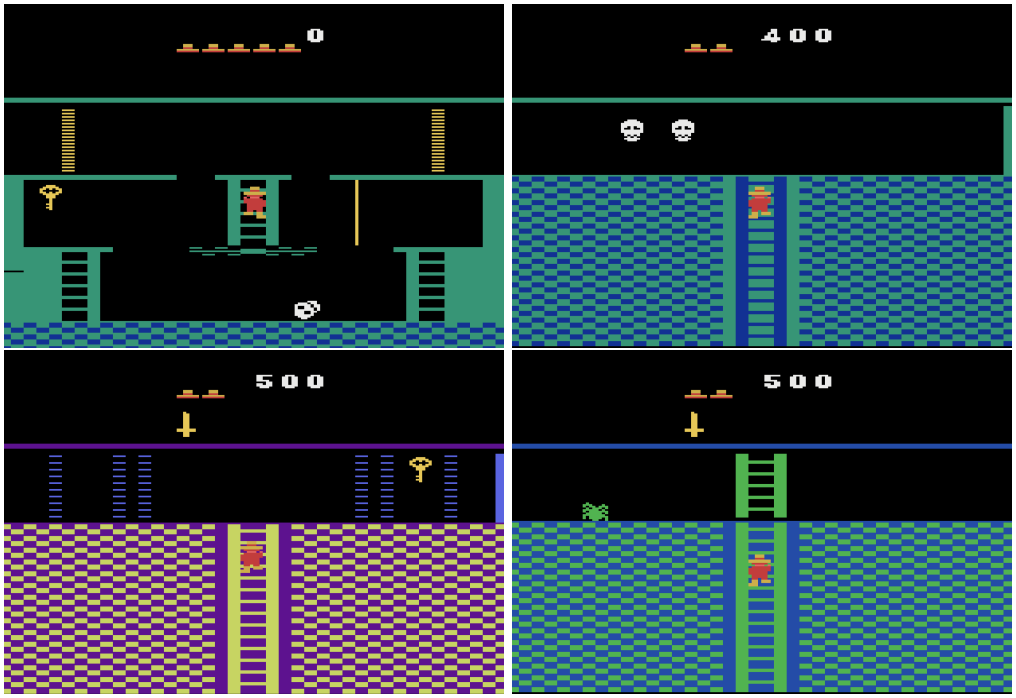

Figure 6: Four ladders in Montezuma's Revenge

Atari games. Figure 7 shows the number environment steps required for the policy to reach a 90% success rate for each room. It shows that the number of steps requires to reach a high success rate drastically decreases as more and more ladders are encountered, demonstrating effective policy transfer. Notice in Figure 6 that the ladders are visually very different. This demonstrates that our method is still able to see past the visual differences and find the generalizing feature for ladders. It is also worth noting that, as in this experiment, transferable policy can be viewed as a "smart initialization method" when encountering a new task from the same task distribution.

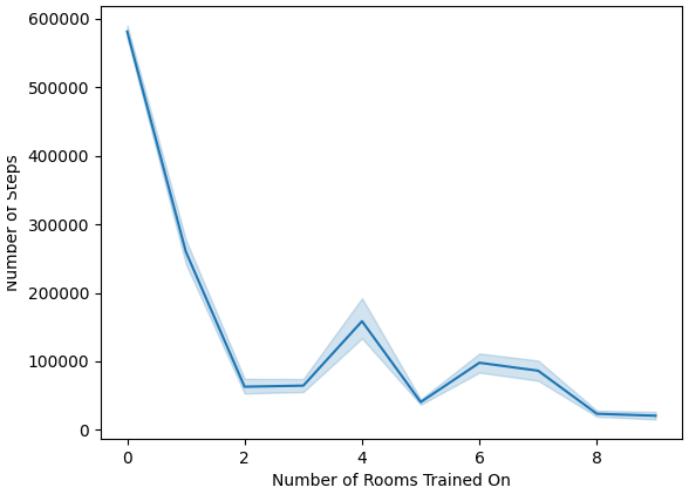

Figure 7: Number of steps taken to reach a success rate of 0.9 for the `ClimbDownLadder` policy (averaged over 10 seeds).

5 Conclusion

We tackle the problem of policy transfer in lifelong reinforcement learning. We propose to use an attention-based ensemble to learn multiple minimally-overlapping state features, and use a bandit algorithm to learn to pick out the generalizing feature from the ensemble. We validate this approach on the Procgen benchmark and found faster generalization performance on seven out of eight games. We also experiment on Montezuma's Revenge and found our method can successfully transfer policy across tasks and work well as a policy–initialization method.

One interesting direction for future work is to make the optimal ensemble size remain the same across different environments. Another fascinating direction is to extend the policy–ensemble method into the options framework in hierarchical RL to learn transferable options.

A Architecture and Hyperparameters

A.1 Attention-Based Ensemble Architecture

The attention-based ensemble has three components, a spatial feature extractor, global feature extractor and attention module. The spatial feature extractor consists of a convolutional layer with 32 channels, kernel size of 3 and stride 1 followed by a max pooling layer with a kernel size of 2. Each attention module is a convolutional layer with 32 channels and has a kernel size of 1 and stride 1. We do not include bias in this layer. Furthermore, the attention outputs are normalized to between 0 and 1 with min-max normalization. The global feature extractor consists a convolutional layer, with 64 channels, kernel size of 3, and stride 2, followed by a max pooling layer with a kernel size of 2. The output will also be flattened into a one-dimensional feature embedding.

For experiments on Montezuma's Revenge, the global feature extractor has an appended Gated Recurrent Unit [Cho et al., 2014] layer with a hidden size of 128. We found having this layer to improve generalization performance.

A.2 PPO

Our PPO policy network consists of two linear layers, where the output feature size is 256 and 15, respectively. The weights of the second layer are initialized with a (semi) orthogonal matrix, and the bias to 0. The output of the policy is fed into the value network to predict the value. The value network is simply a linear layer with output feature size 1.

The PPO agents are optimized according to the loss function proposed by Schulman et al. [2017]. At time step t, the loss of the network θ is the combination of three parts

$$L_t^{\text{CLIP+VF+S}}(\theta) = -\mathbb{E}_t[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t)].$$

The clipped objective is

$$L_t^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta, 1-\epsilon, 1+\epsilon))\hat{A}_t)],$$

where $\hat{A}_t$ is an advantage estimator. The value function loss is

$$L_t^{\text{VF}} = (V_\theta(s_t) - V_t^{\text{targ}})^2.$$

Finally, S denotes an entropy bonus.

A.3 Q Network

For experiments on Montezuma's Revenge, the ensemble of policies has the same structure of an attention-based ensemble followed by policy networks. Each policy network is a q function, parameterized by a neural

network with two linear layers. The first layer outputs embeddings of dimension 64, and the second layer outputs q values of dimension 18. The policy selects actions using an epsilon-greedy scheme, where the epsilon parameter is being decayed linearly through time.

The updates to the q function uses samples collected and stored in a replay buffer. We weight each sample with Prioritized Experience Replay [Schaul et al., 2015].

B Environment Settings

B.1 Procgen Environment

For all the experiments in Procgen, we use 8 parallel environments to speed up learning. The discount factor is set to $\gamma = 0.999$. We always center the image observations on the game character.

We normalize the reward function using the Parallel Algorithm Chan et al. [1982].

The results in Figure **??** report the discounted episodic return for every time step. Furthermore, the episodic rewards are smoothed over a sliding window of 100 episodes.

B.2 Montezuma's Revenge Environment

We use a frame stack of four previous time steps and standard Atari wrappers by DeepMind.

C Experiment Hyperparameters

Table 1: PPO ensemble of policies hyperparameters.

HYPERPARAMETER	VALUE
c_1	0.5
c_2	0.01
Clip (likelihood ratio & value function)	0.2
Maximum L_2 norm for gradient clipping	0.5
Agent update period	256
Batch size	1024
Number epochs per gradient update	3
Learning rate	5×10^{-4}
λ-return	0.95
Replay buffer size	10^5
Bandit exploration weight c	500
Embedding distance function	Euclidean distance
m_{div}	1000

Table 2: Q-function ensemble of policies hyperparameters.

HYPERPARAMETER	VALUE
Hidden layer output size	128
Discount rate	0.9
Target update interval	10
Exploration algorithm	Linear decay epsilon greedy
Initial epsilon	1.0
Final epsilon	0.01
Epsilon decay steps	10^6
Ensemble size	3
Replay buffer size	10^5
Batch size	32
Learning rate	2.5×10^{-4}
Bandit exploration weight c	100
Embedding distance function	Euclidean distance
m_{div}	1

Table 3: Bandit Algorithms hyperparameters.

HYPERPARAMETER	**VALUE**
exploration constant c (UCB)	300 for Maze, 500 otherwise
exploration weight γ (Exp3)	0.1
window size (sliding-window UCB)	90
β (sliding-window UCB)	1
ϵ (sliding-window UCB)	0.5
gestation period (UCB with gestation)	$1,000,000$ steps

References

D. Abel, Y. Jinnai, S. Y. Guo, G. Konidaris, and M. Littman. Policy and value transfer in lifelong reinforcement learning. In *International Conference on Machine Learning*, pages 20–29. PMLR, 2018.

P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, Z. D. Guo, and C. Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

A. Barreto, D. Borsa, J. Quan, T. Schaul, D. Silver, M. Hessel, D. Mankowitz, A. Zidek, and R. Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

T. F. Chan, G. H. Golub, and R. J. LeVeque. Updating formulae and a pairwise algorithm for computing sample variances. In *COMPSTAT 1982 5th Symposium held at Toulouse 1982*, pages 30–41. Springer, 1982.

K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

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

R. Dubey, P. Agrawal, D. Pathak, T. L. Griffiths, and A. A. Efros. Investigating human priors for playing video games. *arXiv preprint arXiv:1802.10217*, 2018.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

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.

K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.

W. Kim, B. Goyal, K. Chawla, J. Lee, and K. Kwon. Attention-based ensemble for deep metric learning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 736–751, 2018.

G. D. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, pages 895–900, 2007.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

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

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

A. A. Sherstov and P. Stone. Improving action selection in mdp's via knowledge transfer. In *AAAI*, volume 5, pages 1024–1029, 2005.

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

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

M. E. Taylor and P. Stone. An introduction to intertask transfer for reinforcement learning. *Ai Magazine*, 32(1):15–15, 2011.

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