TRACE: Topologically Constrained Reward Aware Action Embeddings for Life-Long Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Lifelong Reinforcement Learning with dynamic action spaces is a fundamentally challenging problem because, in such environments, it is hard to learn a general policy across multiple action spaces. Recent approaches are based on assuming an underlying structure and vector space for the set of all possible actions and constant transition state dynamics in this space. Based on these assumptions, they formalize a policy parameterization that generates universal action embeddings and acts in that space of embeddings. While current approaches are able to learn a complete policy, they have the following drawbacks: a) the action embeddings generated using state transition dynamics ignore the reward accrued during each transition, b) there is a non-stationarity between the policy and existing actions when the embeddings are re-learned in different action space. In this work, we address these problems by augmenting the action embedding objective with a *reward-aware* term and a *topology-preserving* term. We test our approach across three different environments - a simple grid world environment and two environments simulating a recommendation platform for a user, using artificial and real-world data simultaneously, and demonstrate a significant increase in average return and other transfer metrics.

1 Introduction

Lifelong learning is a well-explored problem that deals with systems that continually learn new tasks over the course of time without losing information learned in the initial stages of learning. It is very relevant for real-world problems where the system ingests tons of data every day. In order to utilize this data, it is important to use it to update the knowledge stored in the system while retaining the existing learnings of the system.

We address the problem of lifelong learning for sequential decision-making. For example, consider the recommendation system of an Over The Top (OTT) platform (a platform that streams content over the internet directly to the viewers by bypassing cable, broadcast, and satellite television platforms [1] that is designed to maximize the watch time of a user by serving a series of movie recommendations. However, new movies are added to the platform by humans managing the platform every day leading to a change in the action space. Retraining the model with new actions is not just expensive but also leads to less overall return due to the opportunity cost incurred while retraining the model. This example highlights the importance of lifelong learning architecture. We need a framework that can handle this problem in a principled way.

[1] https://en.wikipedia.org/wiki/Over-the-top_media_service

Reinforcement Learning (RL) is typically used to build models used in the example above. In more general terms, RL is used to build sequential decision-making systems. However, lifelong learning framework in reinforcement learning is an under-explored problem, and Chandak et al. [2020] (LAICA) is one of the few works which have attempted to solve this problem. The authors define an underlying action embedding space that does not depend on the cardinality of the action space. The policy function is then parameterized, leveraging the structure of the latent action space, making it invariant to the cardinality of the action space.

Even though LAICA presents a robust framework for lifelong learning, there are certain shortcomings that lead to inefficient training and a drop in performance. When new actions are added, the embeddings of all the actions in the latent space, including the existing actions are retrained, leading to a significant drop in the initial performance and longer training times. The agent takes significant time just to reach the previous best performance, that is, the best performance, before adding new actions.

Moreover, the action embeddings are dependent on just the transition dynamics, ignoring the reward dynamics. This leads to inconsistencies as two actions that are very similar can lead to very different outcomes, and hence these two actions should be distant in the latent action space. To demonstrate this case, consider the following example. Two movies with the same storyline but different endings can lead to huge differences in the number of views. Thus, reward-aware action representations would lead to better performance.

We propose a method to address these drawbacks. We build upon the LAICA's framework and add two additional losses, namely the topology preservation loss and reward prediction loss. The former helps in stabilizing the action embeddings of the existing actions by maintaining the correlation between relative distances between action nodes before and after new action addition. The latter helps in training reward-aware action embeddings, thus enabling us to learn more optimal policies. To summarize, our contributions are:

1. A topology preserving loss that maintains the topology of the latent action space leading to less distortion in the embeddings of existing actions and faster learning.
2. Introducing exponential decay in the topology-preserving loss to ensure that the action embeddings are not too rigid and adapt to the new action space with miniaml distortion.
3. Adding a reward prediction loss to learn embeddings that are not just dependent on transition dynamics but also aware of the reward dynamics.

Our algorithm outperforms the baseline on all the metrics by a significant margin tested on three different environments.

2 Related Work

Over the years, there has been a rich interest in lifelong learning Thrun [1998], Ruvolo and Eaton [2013], Silver et al. [2013]. Past work has worked on dealing with the problem of catastrophic forgetting French [1999], Kirkpatrick et al. [2017], Lopez-Paz and Ranzato [2017], Zenke et al. [2017], where these methods leverage prior knowledge for new tasks. [Neu, 2013, Gajane et al., 2018, Auer et al., 2019, Barreto et al., 2017] address the problem of learning online with changing transition dynamics or reward functions.

Recent works also introduced and dealt with the problem of *dynamic action spaces*, where the size and members of the action set available to the agent in the environment are stochastic i.e., they are continuously changing Chandak et al. [2020], Mandel et al. [2017], Boutilier et al. [2018]. [Chandak et al., 2020] introduced a learning objective and framework to learn action embeddings Chandak et al. [2019a] in such lifelong MDPs Boutilier et al. [2018], where the number of actions changes, so the agent has to deal with changing dynamics and actions that it has never seen before. The goal of the work was to learn action embeddings in a continuous latent space, and a policy that acts in that latent space. This makes the policy invariant to the cardinality of the action set. However, a key challenge in this learning process is the efficient transfer of the learned policy across the set of changed actions and action embeddings. Past works in meta-reinforcement learning, [Xu et al., 2018, Gupta et al., 2018, Wang et al., 2017, Duan et al., 2016, Finn et al., 2017], have worked on transfer learning and few-shot shot adaption to new tasks after training on the distribution of similar tasks.

Castro et al. [2018], Yan et al. [2021], Tao et al. [2020b], Wu et al. [2019], Tao et al. [2020a], Iscen et al. [2020], Mazumder et al. [2021], Shankarampeta and Yamauchi [2021] deal with the problem of catastrophic forgetting and transfer learning in a class incremental setting. In this problem, the works deal with training and transfer learning on tasks such as image classification, where the model receives samples from new target classes gradually. The model has to learn efficiently on the new classes while preserving knowledge from the previously seen ones. [Tao et al., 2020a,b, Chang et al., 2021] deal with this problem through building topological representations of the feature space and adding constraints to the learning process to avoid catastrophic forgetting of previous knowledge, while simultaneously learning on the incremental classes.

In this work, we employ methods from class incremental learning to deal with the problem of *increments* and changes in the action space of the agent. We compare our work to Chandak et al. [2019b], which is the closest work to us in this problem.

3 Preliminaries

Sequential decision-making problems in the real world can be modeled as Markov decision processes (MDPs).

Formally, an MDP is defined as $\mathcal{M}_0 = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma \rangle$, where:

- State space, $\mathcal{S}$ is the state space of the environment. At each time t, the agent observes state $S_t \in \mathcal{S}$.
- Action space, $\mathcal{A}$ is the discrete set of actions available to the agent to interact with the environment. At each time t, the agent takes an action $A_t \in \mathcal{A}$.
- Transition function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, which represents the dynamics of the environment in response to the agent's actions.
- Reward function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which represents the per-step reward function and $R_t = r(S_t, A_t)$ represents the per-step reward the agent receives from the environment at time t, given the state S_t, and the action A_t. As is standard in infinite horizon reinforcement learning, we further assume that the rewards are uniformly bounded, i.e., $R_t \in [R_{\min}, R_{\max}]$, for all t.

The objective of the agent is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes its expected cumulative discounted return given by:

$$J^\pi = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_{t+1} \sim \mathcal{P}(S_t, A_t); A_t \sim \pi(S_t); R_t = r(S_t, A_t), S_0 \sim \mathbb{P}(S_0)\Big], \tag{1}$$

where $\mathbb{P}(S_0)$ denotes the distribution of the starting state.

3.1 Lifelong Learning

In most real-world problems. the set of actions available to the agents varies over time. Standard MDP frameworks are not flexible enough to model such lifelong learning problems where the action set size changes over time. To this end, [Chandak et al., 2020] introduced a new formalism for dealing with dynamic action spaces, assuming a universal vector space $\mathcal{E} in \mathbb{R}^d$ for all possible actions and leveraging the underlying substructure in the set of actions available. The lifelong MDP (LMDP) is defined as $\mathscr{L} = (\mathcal{M}_0, \mathcal{E}, \mathcal{D}, \mathcal{F})$, which extends a base MDP $\mathcal{M}_0$ as defined above. $\mathcal{M}_0 = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, d_0)$.

- $\mathcal{E}$: Set of all possible actions, including those that are available at one point in time and those that might become available at any time in the future.
- $\mathcal{D}$: Set of distributions $\mathcal{D}_\tau$, from which the random variable, corresponding to the set of actions added at episode τ, is sampled.
- $\mathcal{F}$: Probability distribution over the episodes from which we sample episodes in which new actions are added.

For $\mathcal{M}_0$ we define $\mathcal{A}$ to be empty, i.e., $\mathcal{A} = \emptyset$. When the set of available actions changes and the agent observes a new set of discrete actions, $\mathcal{A}_k$, then $\mathcal{M}_{k-1}$ transitions to $\mathcal{M}_k$, such that $\mathcal{A}$ in $\mathcal{M}_k$ is the set union of $\mathcal{A}$ in $\mathcal{M}_{k-1}$ and $\mathcal{A}_k$. Apart from the available actions, other aspects of the L-MDP remain the same throughout. An illustration of the framework is provided in Figure 1. We use $S_t \in \mathcal{S}, A_t \in \mathcal{A}$, $R_t \in \mathbb{R}$ same as a standard MDP, and the reward function $\mathcal{R}$ is defined to be only dependent on the state such that $\mathcal{R}(s) = \mathbf{E}\left[R_t \mid S_t = s\right]$ for all $s \in \mathcal{S}$. $\mathcal{P}$ is the state transition function, such that $\forall s, a, s', t$, $\mathcal{P}\left(s, a, s'\right)$ denotes the transition probability $P\left(s' \mid s, e\right)$, where $a = \phi(e)$ and $e \in \mathbb{R}^d$ is the underlying structure of the action.

In the most general case, new actions could be completely arbitrary and have no relation to the ones seen before. In such cases, there is very little hope of lifelong learning by leveraging past experience. To make the problem more feasible, we resort to a notion of smoothness between actions. Formally, we assume that transition probabilities in an L-MDP are ρ-Lipschitz in the structure of actions, i.e., $\exists \rho > 0$ s.t. $\forall s, s', e_i, e_j$,

$$\left\|P\left(s' \mid s, e_i\right) - P\left(s' \mid s, e_j\right)\right\|_1 \leq \rho \left\|e_i - e_j\right\|_1 \tag{2}$$

For any given MDP $\mathcal{M}_k$ in $\mathscr{L}$, an agent's goal is to find a policy, π_k, that maximizes the expected sum of discounted future rewards. For any policy π_k, the corresponding state value function is,

$$v^{\pi_k}(s) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid s, \pi_k\right] \tag{3}$$

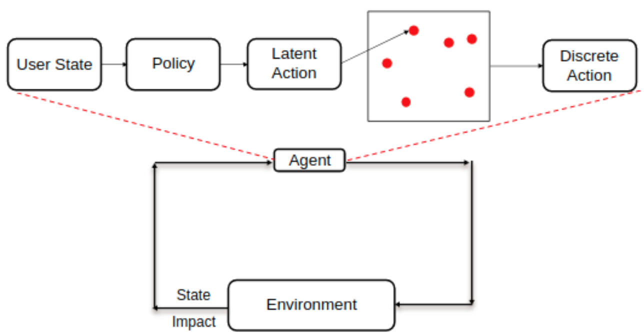
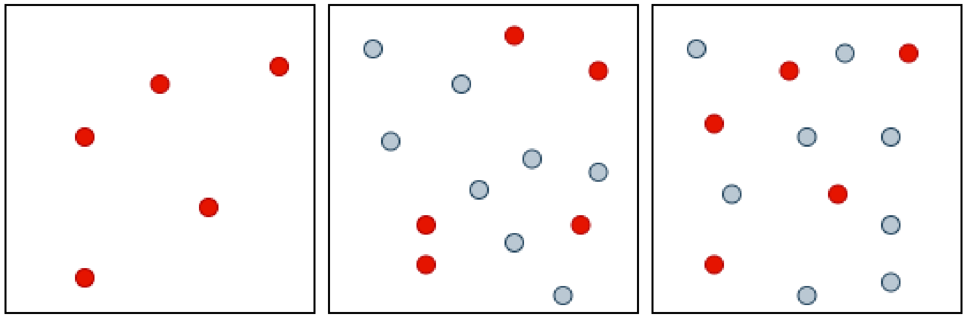


Figure 1: Life Long Learning Architecture

Instead of having the policy, π, act directly in the observed action space, $\mathcal{A}$, Chandak et al. [2020] proposes an approach wherein the policy is invariant to the action set size, and is parameterized into two components. The first component corresponds to the state conditional continuous policy which acts in the universal action space, $\beta : \mathcal{S} \times \hat{\mathcal{E}} \to [0, 1]$, where $\hat{\mathcal{E}} \in \mathbb{R}^d$. The second component corresponds to $\hat{\phi} : \hat{\mathcal{E}} \times \mathcal{A} \to [0, 1]$, an estimator of the relation ϕ, which is used to map the output of β to an action in the set of available actions. In this parameterization, $E_t \in \hat{\mathcal{E}}$ is sampled from $\beta\left(S_t, \cdot\right)$ and then $\hat{\phi}\left(E_t\right)$ is used to obtain the action A_t. Together, β and $\hat{\phi}$ form a complete policy, and $\hat{\mathcal{E}}$ corresponds to the inferred structure in action space. In case of a set of discrete actions $\mathcal{A}$, phi is formalised as a set of action embeddings $\boldsymbol{e} = e_0, e_1, e_2 \ldots e_{|\mathcal{A}|}$, where $e_i \in \hat{\mathcal{E}} \forall i \in [0, \mathcal{A}]$. Here, $\hat{\phi} \sim$ Softmax($\|e_i - E_t\|$) i.e. the action selection distribution is proportional to euclidean distance between the discrete action embedding and the output from β.

$$\mathcal{L}^{\mathrm{lb}}(\hat{\phi}, \varphi) := \mathbf{E}\left[\log \hat{\phi}\left(e_{\hat{A}_t} \mid \hat{E}_t\right) \mid \varphi\left(\hat{E}_t \mid S_t, S_{t+1}\right)\right] - \lambda \mathrm{KL}\left(\varphi\left(\hat{E}_t \mid S_t, S_{t+1}\right) \| P\left(\hat{E}_t \mid S_t, S_{t+1}\right)\right) \tag{4}$$

where, $\mathcal{P}$ is prior over the inverse dynamic outputs $\sim \mathcal{N}(0, 1)$, $\hat{E}_t$ is the action vector from inverse dynamic function, $\hat{A}_t$ is the action taken at time t, and $\hat{S}_t$ is the state at time t. The policy β is learned using the RL objective as denoted earlier. For ϕ, Chandak et al. [2020] derived an objective function Eq. 4, that learns φ: the inverse dynamics of the environment and the action embeddings $\boldsymbol{E}$ in an



(a) $\boldsymbol{e}$ at t=0.

(b) $\boldsymbol{e}$ at t=T.

(c) $\boldsymbol{e}$ with ℓ_{TPL} at t=T.

Figure 2: Comparing action embeddings with and without topology preserving loss. Fig 2a represents actions at t=0, Fig 2b actions with topology preserving loss and Fig 2c represents actions without topology preservation at t=T.

unsupervised manner. The objective is very similar to that of a variational autoencoder, where the output from the encoder (i.e., the inverse dynamics function φ) is reconstructed to be the same as the action embeddings learned, with a regularisation term.

4 Our Approach

In a given MDP $\mathcal{M}_k$, the parameterization in Sec:3.1 learns a continuous policy β_k and a set of action embeddings $\boldsymbol{e}^k = e_0, e_1, e_2 \ldots e_{|\mathcal{A}_k|}$, where $\mathcal{A}_k$ is the set of actions available in $\mathcal{M}_k$. However, as actions are added i.e. $\mathcal{A}_{k+1} = \mathcal{A}_k \bigcup \mathcal{A}'$, then, $\mathcal{M}_k \rightarrow \mathcal{M}_{k+1}$, and $\mathcal{A}'$ is the set of new actions added. On this transition, the embeddings for all actions $a \in \mathcal{A}_k$ are trained further, along with the added actions A'. This basically represents an update the function $\phi : \phi_k \rightarrow \phi_{k+1}$ In addition to this, the policy β_k has to be trained to adapt to the newer set of actions and their action embeddings $\boldsymbol{e^{k+1}}$. So, there is a non-stationarity between the learned policy β_k and ϕ_{k+1}, which may cause poor performance in the environment unless β_k is finetuned, which may be costly or very difficult.

Topological Invariance from Hebbian Theory: In this work, we build upon the framework to augment the learning objective to control the non-stationarity and to generate efficient transfer learning across the action spaces. We adopt a notion of topological invariance from Tao et al. [2020a], between the embeddings of existing actions A_k in $\boldsymbol{e}^k$ and $\boldsymbol{e}^{k+1}$. Our key insight is that, given sufficient experience, the transition dynamics function around the existing actions are relatively constant, so $\boldsymbol{e}^k$ is a reasonable representation for the actions in $\hat{\mathcal{E}}$. To this end, while updating the action embeddings $\boldsymbol{e}^k$, the change in embeddings should be minimized. Furthermore, having a stable embedding space across MDPs will efficiently transfer the performance of the old policy β_k, which further makes learning β_{k+1} efficient.

We thus propose an extension to Chandak et al. [2020] by adding a topology preserving loss. Using the loss function from Tao et al. [2020a] we maintain the topology of the action embeddings across action space changes. In order to use this loss, $\boldsymbol{e}^k$ is modelled using Hebbian graphs which is constructed using *Competitive Hebbian Learning* Martinetz [1993]. Topology is preserved by maintaining the similarity between the nodes i.e. $e_1^k, e_2^k \ldots e_{\mathcal{A}_k}^k$. Figure 2 shows the impact of preserving topology on the action embeddings.

The Topology Preserving Loss or ℓ_{TPL} is the correlation of distance between different nodes across the different action spaces. Figure 2 represents the effects of using and not using ℓ_{TPL}, where Figure 2c shows less change in existing actions due to loss as compared to Figure 2b. Intuitively, the loss forces the distances between $\boldsymbol{e}^k$ before and after the change to be highly correlated, which preserves the structure i.e., the topology of the space. Formally, we define a fully connected graph G^{k+1}, with nodes as $e_1^{k+1}, e_2^{k+1}, \ldots e_{\mathcal{A}_k}^{k+1}$. To simplify notation, here $\boldsymbol{e}^{k+1}$ represents the updated embeddings of only the existing actions $\mathcal{A}_k$, where $N = |\mathcal{A}_k|$. The loss is defined as:

$$\ell_{TPL}\left(G^{k+1}; \boldsymbol{e}^{k+1}\right) = -\frac{\sum_{i,j}^{N}\left(s_{ij} - \frac{1}{N^2}\sum_{i,j}^{N} s_{ij}\right)\left(\tilde{s}_{ij} - \frac{1}{N^2}\sum_{i,j}^{N}\tilde{s}_{ij}\right)}{\sqrt{\sum_{i,j}^{N}\left(s_{ij} - \frac{1}{N^2}\sum_{i,j}^{N} s_{ij}\right)^2}\sqrt{\sum_{i,j}^{N}\left(\tilde{s}_{ij} - \frac{1}{N^2}\sum_{i,j}^{N}\tilde{s}_{ij}\right)^2}} \tag{5}$$

where, $S = \{s_{ij} \mid 1 \leq i, j \leq N\}$ and $\tilde{S} = \{\tilde{s}_{ij} \mid 1 \leq i, j \leq N\}$ are the sets of the initial and updated edges' weights in $\boldsymbol{e}^k$, and $\boldsymbol{e}^{k+1}$ respectively. The active value $\tilde{s}_{ij}$ is estimated by,

$$\tilde{s}_{ij} = \left\| e_i^{k+1} - e_j^{k+1} \right\|_2 \tag{6}$$

Adaptive Topology using Decaying Weight: The topology-preserving loss helps to improve the transfer of performance; however, it may lead to a complete loss of adaptation to new actions. Further, it restricts the exploration of embeddings in case of inexperienced agents. So, we deal with these issues by adding an exponential decay to τ. It conserves the topology of the existing actions in the initial stages of learning to avoid large deviations in the initial exploratory phase, and then, it decays to allow small changes in those action embeddings to adapt to the new action space. Using the loss defined in 5, we formulate augmented objective $\mathcal{L}_{TPL}$,

$$\mathcal{L}_{TPL} = \mathcal{L}^{lb} + \tau * \ell_{TPL} \tag{7}$$

where τ represents the coefficient of topology preservation.

Reward Augmentation: In the objective 4, the action embeddings $\boldsymbol{e}$ are generated using the underlying substructure $\hat{\mathcal{E}}_t$, which is extracted from the state inverse dynamics. This existing method of lifelong learning works well in scenarios where the actions have distinct transitions and a unique reward is present for an action. But it fails to generalize in case of actions with similar transitions but very different rewards. To mitigate this problem, we propose to embed the reward information in the embedding space leading to reward aware latent action representations.

$$\ell_{reward} = \|f(S_t, \hat{A}_t) - r_t\|_2 \tag{8}$$

where f is a function which predicts the reward, and $S_t, \hat{A}_t, r_t$ represent the state, action and reward at timestep t. With this improvement, two actions with different reward structures are well separated in the embedding space. This disentanglement allows for better representations which accelerates policy optimization.

Additionally, we also observe that the reward prediction loss acts like a regularizer, thereby further reinforcing the learning and resulting in improved performance.

5 Experiments and Results

To demonstrate the robustness of our algorithm, We test it on three different environments:

1. **Gridworld or 2d Maze Environment**: A standard RL environment for benchmarking with dynamic actions being steps in different directions. This environment is used for basic sanity checks of our algorithm.
2. **Recsim** [2]: A standard environment proposed by google research where the changing action space is a dynamic set of videos that can be recommended to the user. We use the interest evolution environment of Recsim as it is suitable for our use case. We use this environment to test our algorithm on a state-of-the-art sequential decision-making framework
3. **Recsim with clickstream data**: We use real clickstream data in the third environment to show the practicality of our algorithm. We have click logs of the sequences of pages visited by the user in a session. We represent the pages as available actions $t_1, t_2 \cdots t_n$. Considering each page as a token, we generate embeddings for each of them using the

[2] `https://github.com/google-research/recsim/`

word2vec algorithm.

Once we obtain the features for these pages, we use these pages as the objects to be recommended to the user in the recsim environment, and corresponding embeddings are used as features of the document. The remaining environment is the same, where we uniformly sample the initial user features, namely satisfaction and net kale exposure, and read time for each tutorial, sampled from a log-normal distribution. The user features are not present in the data and are inherently sampled in the Recsim environment, but in the presence of some features, we can use those features instead of sampling. The objective is to make such recommendations to maximize the total read time i.e., user engagement, which can be achieved by selecting pages with larger read times (which are sampled from a log-normal distribution) and also by maximizing more meaningful and useful pages which may lead to higher utility and lower time budget drop. **This environment is very generic and can be used with any clickstream data.**

5.1 Transfer Learning Metrics

We define the following metrics from [Taylor and Stone, 2009] to evaluate the performance of our models (**TRACE w/o Reward & Decay**: Only with topological invariance, **TRACE w/o Reward**: With adaptive topology, **TRACE**: With topological invariance and reward augmentation) against the baseline model (**LAICA**) Chandak et al. [2020]:

- Total Return R_t = Total return received by the agent during training i.e. area under the training performance curve.
- Total Regret R_g = Sum of difference between the best episode return received by the agent and the actual performance during training, i.e., the area between the best performance line and training performance curve. Lower regret implies better performance.
- Jump Start J = Mean performance over the first N training episodes after new actions are added, and the action representations are updated. We need a better jump start as the exploration is costly, and the agent should adapt to the new actions as quickly as possible. J_i represents the jump start after i^{th} action addition.
- Gradient of Learning L = Analogous to velocity; where distance is the difference between the performance after adding new actions and the previous best performance, and the time taken is the number of steps needed to cross the previous best performance after new actions are added. L_i represents the gradient of learning after i^{th} action addition.

Figures 3, 4, 5 show the rewards accumulated by the baseline and all our models during training across different environments. The reward, which is the bold line in the graph, is the running mean of instantaneous reward averaged across seeds. The shaded region in the graph represents the standard deviation across 5 seeds. We outperform the baseline in all the environments. The performance improves with every incremental improvement to the algorithm. As expected, the final algorithm with decaying topological loss and reward-aware embeddings shows the best performance.

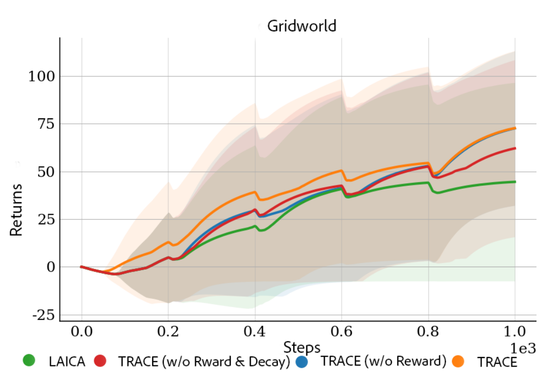


Figure 3: Returns of different agents in the Gridworld environment. Our best-performing model shows a 46.49% improvement in the performance over the baseline model

Agent	Total Return	Total Regret	J1	J2	J3	J4	L1	L2	L3	L4
Gridworld										
LAICA	25.21	72.03	3.96	19.08	36.60	39.33	0.037	0.024	0.018	0.010
TRACE (w/o Reward & Decay)	30.28	66.82	4.07	27.21	38.16	47.16	0.056	0.0002	0.034	0.024
TRACE (w/o Reward)	31.69	65.40	3.98	26.43	37.18	47.68	**0.068**	**0.037**	**0.047**	**0.098**
TRACE	**36.93**	**60.18**	**11.47**	**35.19**	**45.37**	**49.09**	0.062	0.032	0.027	0.090
Recsim										
Agent	Total Return	Total Regret	J1	J2	J3	J4	L1	L2	L3	L4
LAICA	184.34	56.13	200.99	207.50	191.50	192.14	0.014	0.027	**0.078**	0.000
TRACE (w/o Reward & Decay)	199.64	**42.42**	**201.30**	220.19	213.89	212.02	0.037	0.012	0.017	**0.040**
TRACE (w/o Reward)	197.83	45.53	201.14	**223.38**	215.43	205.85	0.036	**0.035**	0.012	0.000
TRACE	**206.72**	42.88	198.48	221.23	**225.34**	**224.79**	**0.040**	0.027	0.020	0.023
Recsim with real-world data										
Agent	Total Return	Total Regret	J2	J4	J6	J8	L2	L4	L6	L8
LAICA	175.46	45.72	180.66	202.93	**204.97**	187.94	1.662	0.148	0.039	-0.001
TRACE (w/o Reward & Decay)	**182.10**	42.97	**182.78**	200.03	199.26	**203.38**	1.544	**0.309**	**0.081**	**0.228**
TRACE (w/o Reward)	179.63	**40.94**	181.52	**208.32**	203.53	197.90	1.663	0.093	0.043	0.066
TRACE	177.89	41.22	180.70	200.84	202.34	194.05	**1.684**	0.130	0.042	0.000

Table 1: Transfer learning metrics for all the environments. We outperform the baseline on almost all the metrics. Our model with just topology loss (TRACE(w/o Reward & Decay)) outperforms other models in the third environment indicating the minimal impact of augmenting reward and using decaying topological weight. We achieve 46.49%, 12.14%, and 3.78% increase in the overall performance (defined in terms of the total return metric) on the three environments, respectively

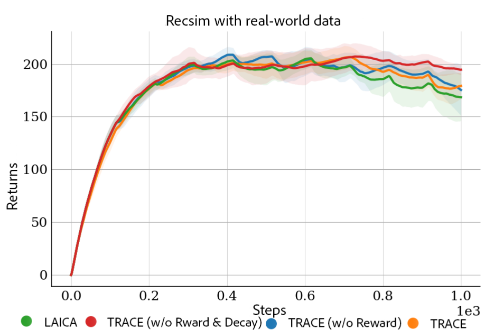


Figure 4: Returns of different agents in the Recsim environment with real-world data. Our best-performing model shows a 3.78% improvement in the performance over the baseline model

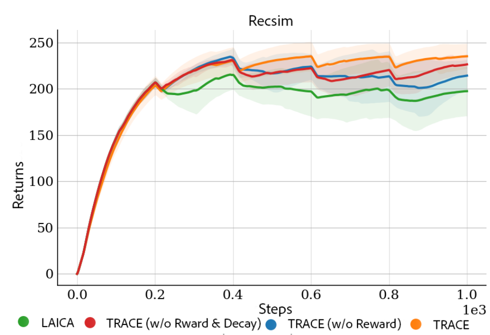


Figure 5: Returns of different agents in the Recsim environment. Our best-performing model shows a 12.14% improvement in the performance over the baseline model

6 Discussion

In this work, we build upon the existing lifelong learning architecture to develop more robust action embeddings in the universal embedding space. With topological invariance and reward augmentation, we outperform the baseline on all the metrics in three different environments. The new action embeddings are semantically consistent; two actions with similar state transitions and reward dynamics are close to each other in the embedding space. With reward-aware action embeddings, the applicability of this architecture extends beyond regular applications in the real world, as it can be used to identify gaps in the embedding space to generate new actions with high rewards. For example, a content platform manager can curate new content after analyzing the embedding space and identifying content that can lead to a specific state transition and yield high rewards.

References

Peter Auer, Pratik Gajane, and Ronald Ortner. Adaptively tracking the best bandit arm with an unknown number of distribution changes. In *Conference on Learning Theory*, pages 138–158. PMLR, 2019.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

Craig Boutilier, Alon Cohen, Amit Daniely, Avinatan Hassidim, Yishay Mansour, Ofer Meshi, Martin Mladenov, and Dale Schuurmans. Planning and learning with stochastic action sets. In *IJCAI*, 2018.

Francisco Manuel Castro, Manuel J. Marín-Jiménez, Nicolás Guil Mata, Cordelia Schmid, and Alahari Karteek. End-to-end incremental learning. *ArXiv*, abs/1807.09536, 2018.

Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip S. Thomas. Learning action representations for reinforcement learning. *International Conference on Machine Learning*, 2019a.

Yash Chandak, Georgios Theocharous, Blossom Metevier, and Philip S Thomas. Reinforcement learning when all actions are not always available. *arXiv preprint arXiv:1906.01772*, 2019b.

Yash Chandak, Georgios Theocharous, Chris Nota, and Philip Thomas. Lifelong learning with a changing action set. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3373–3380, 2020.

Xinyuan Chang, Xiaoyu Tao, Xiaopeng Hong, Xing Wei, Wei Ke, and Yihong Gong. Class-incremental learning with topological schemas of memory spaces. *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 9719–9726, 2021.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl$\hat{2}$: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

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

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3 (4):128–135, 1999.

Pratik Gajane, Ronald Ortner, and Peter Auer. A sliding-window algorithm for markov decision processes with arbitrarily changing rewards and transitions. *arXiv preprint arXiv:1805.10066*, 2018.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pages 5302–5311, 2018.

Ahmet Iscen, Jeffrey O. Zhang, Svetlana Lazebnik, and Cordelia Schmid. Memory-efficient incremental learning through feature adaptation. *ArXiv*, abs/2004.00713, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

Travis Mandel, Yun-En Liu, Emma Brunskill, and Zoran Popović. Where to add actions in human-in-the-loop reinforcement learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Thomas Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *International conference on artificial neural networks*, pages 427–434. Springer, 1993.

Pratik Mazumder, Pravendra Singh, and Piyush Rai. Few-shot lifelong learning. In *AAAI*, 2021.

Gergely Neu. Online learning in non-stationary markov decision processes. *CoRR*, 2013.

Paul Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *International conference on machine learning*, pages 507–515. PMLR, 2013.

Abhilash Shankarampeta and Koichiro Yamauchi. Few-shot class incremental learning with generative feature replay. In *ICPRAM*, 2021.

Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

Xiaoyu Tao, Xinyuan Chang, Xiaopeng Hong, Xing Wei, and Yihong Gong. Topology-preserving class-incremental learning. In *European Conference on Computer Vision*, pages 254–270. Springer, 2020a.

Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. Few-shot class-incremental learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12180–12189, 2020b.

Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, dec 2009. ISSN 1532-4435.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

JX Wang, Z Kurth-Nelson, D Tirumala, H Soyer, JZ Leibo, R Munos, C Blundell, D Kumaran, and M Botivnick. Learning to reinforcement learn. arxiv 1611.05763, 2017.

Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Raymond Fu. Large scale incremental learning. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 374–382, 2019.

Zhongwen Xu, Hado P van Hasselt, and David Silver. Meta-gradient reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3013–3022, 2021.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.