Temporal Abstractions-Augmented Temporally Contrastive Learning: An Alternative to the Laplacian in RL (Supplementary material)

Akram Erraqabi[1,2] **Marlos C. Machado**[4,5,6] **Mingde Zhao**[1,3] **Sainbayar Sukhbaatar**[8]
Alessandro Lazaric[8] **Ludovic Denoyer**[8] **Yoshua Bengio**[1,2,7]

[1]Mila [2]Université de Montréal [3]McGill University [4]Amii
[5]University of Alberta [6]CIFAR AI Chair [7]CIFAR Fellow [8]Meta AI

A REPRESENTATION OBJECTIVE AUGMENTATION: ABLATION STUDY

A.1 BOREDOM AUGMENTATION HELPS EXPLORATION

In order to illustrate the importance of the boredom term $\mathcal{B}$ – in the final objective (6) we conducted the same representation learning experiments for the three gridworld domains in the non-uniform prior setting, but this time with the non-augmented representation learning objective ($\beta' = 0$).

Figure 9 shows how the agent failed at exploring the whole domain. In T-MAZE, it focuses only on one corridor without getting curious about the other one. Regarding U-MAZE and 4-ROOMS, the agent stops exploring after discovering the end of the first corridor and the second room, respectively. This is due to the lack of incentive to visit the yet unseen states, as they are less rewarding for π_{hi} (i.e., closer in the representation space, hence smaller R^{hi}) than the furthest explored state. The effect of the proposed augmentation compresses the representation of the explored area, say the first corridor in U-MAZE, which makes the rest of the environment more appealing to explore for π_{hi} (i.e. further in the representation space, hence larger R^{hi}).

(a) U-MAZE (b) T-MAZE (c) 4-ROOMS

Figure 9: Learned representations in the gridworld domains with the *non-augmented* objective. Without the boredom term, the agent fails to cover the state space (cf. Figure 3), and may settle for incomplete representations. The colors reflect the distances in terms of the dynamics. They can be seen as quantities proportional to the length of the shortest path from the s_0 (marked in red) to the represented state.

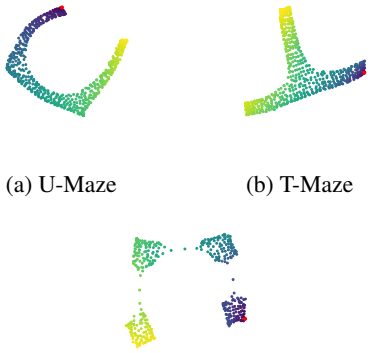
A.2 BOREDOM AUGMENTATION ENFORCES DYNAMICS-AWARENESS

To verify the benefit of the boredom term beyond helping exploration, we train the representation with the non-augmented objective ($\beta' = 0$) but this time in the uniform prior setting, so that to marginalize the exploration problem. Figure 10 illustrates the learned representations in the three gridworld domains. These representations have failed to capture the dynamics. For example, in the case of 4-ROOMS, the distances from the first room to the fourth and third rooms are comparable in the representation space, which indicates that the representation does not take into account the relative order in which the rooms should be visited, when moving from the first room to the last. Similarly, in U-MAZE, the end of the maze is closer to the initial area than the second corner is. However, in order to reach the former on must pass by the latter. This proves that the boredom term is not only important for the desired exploratory behavior (cf. Figure 9), but also enhances the dynamics-awareness of our representation.

B A GREEDY HIGH-LEVEL REWARD CAN HURT EXPLORATION

Our high-level reward term (4) can be seen as an instance of assigning credit to skills based on their potential of affording more exploration. The greedy version of this reward, e.g. of the form $R^{\text{hi}}_{\text{greedy}}(s^{\text{hi}}_k, \boldsymbol{\delta}_k) \triangleq \|\phi(s^{\text{hi}}_k) - \phi(s^{\text{hi}}_{k+1})\|_2$, would encourage each skill (and its direction) to take large steps. Taking large steps, even if it may seem intuitive, is not a good choice for exploration since the agent can get trapped in large oscillations: each skill can travel maximally, back and forth, e.g. along the diagonal of some initially explored area around s_0. These oscillations would still be very rewarding to the high-level policy in this greedy reward case without encouraging further exploration of the environment. Our high-level reward (4) does not fall in this limitation. Our reward choice does not force each of the intermediate skills

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).



(a) U-Maze (b) T-Maze

(c) 4-rooms

Figure 10: Learned representations when uniformly sampling over the state space. Without the boredom term, the representation does not reflect temporally-extended dynamics. The colors reflect the distances in terms of the dynamics. They can be seen as quantities proportional to the length of the shortest path from the s_0 (marked in red) to the represented state.

(within the sequence of L skills) to be individually "greedy". This allows picking skills that might seem sub-optimal, in the sense that they don't travel far on their own, but still afford/offer more promising exploration opportunities later on. For example, consider the case where the agent is in a room with a closed door and on the other side of this door there is a large space to explore. With our reward term, the skill of opening the door would be rewarded similarly to that of a skill able to travel far the other side of the door once this one is open. The high-level policy eventually learns that opening the door is valuable for exploring further areas in the environment.

C IMPLEMENTATION DETAILS

C.1 GRIDWORLD

The states are one-hot encoded such that no positional information is provided to the agent. The domains dimensions are: U-MAZE 30×30, T-MAZE 40×30, 4-ROOMS 21×21.

For all the experiments, we defined the representation network as an MLP of two hidden layers of size 128 with tanh activations and a linear output layer of the size of representation's dimensionality d. The high-level and the low-level policies are both MLPs of two hidden layers of size 128 with tanh activations and a logsoftmax output layer of the size of their respective action spaces: the environment's 4 actions for the low-level policy and 8 actions for the high-level policy corresponding to the 8 directions $\Omega = \{(\cos(2k\pi/n), \sin(2k\pi/n)) \mid k \in \{0, ..., 7\}\}$ that define diverse skills.

The policies were trained with vanilla A2C with MC returns from the collected trajectories (Monte-Carlo estimates), i.e. no bootstrapped values where used. The skills being of a fixed size they could be trained without any reward discount ($\gamma = 1$). The high-level and low-level policies were entropy-regularized with coefficients 0.3 and 0.1 respectively.

All of these networks were trained with RMSprop [Hinton et al., 2012] and a step size of 0.001. Environments specific hyperparameters are provided below.

C.1.1 Representation Learning

U-MAZE. Our representation is learned in the non-uniform prior setting with p_r=0.3, p_{rw}=0.4 and K=90 (around the number of steps between s_0 and the furthest state in the maze). We learn a 2-dimensional representation ($d = 2$) using the representation learning objective (6) with $\beta = 0.2$ and $\beta' = 2$. We fix the skills length to $c = 30$ steps (so $L = K/c = 3$), and jointly train the representation ϕ and the policies $(\pi_{\text{hi}}, \pi_{\text{low}})$ by collecting, for each update, a batch of $N = 32$ trajectories of length c to fill D_s and D_{π_μ} as described in Algorithm 1. We train them for 700 epochs where each epoch corresponds to 10 updates (convergence to the complete representation required around 500 epochs).

T-MAZE. Our representation is learned in the non-uniform prior setting with p_r=0.2, p_{rw}=0.4 and K=40 (around the number of steps between s_0 and the furthest state in the maze). We learn a 2-dimensional representation ($d = 2$) using the representation learning objective (6) with $\beta = 0.2$ and $\beta' = 2$. We fix the skills' length to $c = 20$ steps (so $L = K/c = 2$). and jointly train the representation ϕ and the policies $(\pi_{\text{hi}}, \pi_{\text{low}})$ by collecting, for each update, a batch of $N = 32$ trajectories of length c to fill D_s and D_{π_μ} as described in Algorithm 1. We train them for 700 epochs where each epoch corresponds to 10 updates (convergence to the complete representation required around 350 epochs).

4-ROOMS. Our representation is learned in the non-uniform prior setting with p_r=0.25, p_{rw}=0.5 and K=60 (around the number of steps between s_0 and the furthest state in the maze). We learn a 2-dimensional representation ($d = 2$) using the representation learning objective (6) with $\beta = 0.2$ and $\beta' = 2$. We fix the skills' length to $c = 20$ steps (so $L = K/c = 3$). and jointly train the representation ϕ and the policies $(\pi_{\text{hi}}, \pi_{\text{low}})$ by collecting, for each update, a batch of $N = 32$ trajectories of length c to fill D_s and D_{π_μ} as described in Algorithm 1. We train them for 700 epochs where each epoch corresponds to 10 updates (convergence to the complete representation required around 350 epochs).

The Laplacian representation LAP-REP was trained in the same environments' settings described above, for both

the uniform and non-uniform prior settings (of course no policy is trained here so $p_{rw} = 1$, and (s_0, p_r) are not relevant for the uniform prior setting). Besides the representation's dimension d, we used the training configuration and hyperparameters proposed by Wu et al. [2019]. For the uniform prior setting, our online data collection does not cause any discrepancy compared to the offline scheme used in Wu et al. [2019]. Indeed, for a minibatch size large enough, the stochastic minibatch based training of LAP-REP when using a uniform prior is agnostic to the data collection scheme (offline vs online) since in both cases the minibatches are sampled from the exact same state distribution.

C.1.2 Prediction and Control

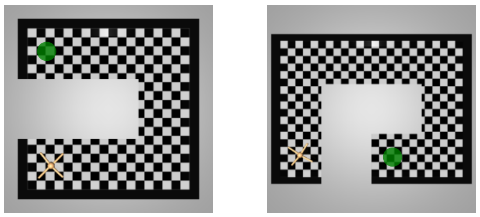
In the prediction and control experiments, we evaluate each pretrained representation by training an actor-critic agent to solve a goal-achieving task with a sparse reward ($r = 1$ upon reaching the goal, and $r = 0$ otherwise). Here are the set goal positions: $(1, 30)$ in U-MAZE, $(25, 30)$ in T-MAZE and $(1, 21)$ in 4-ROOMS. The episode size was set to 100 steps for all the gridworld domains.

For the prediction, the critic head is a linear function in the given representation, while the actor is an MLP with two hidden layers of size 64 and tanh activations, a logsoftmax output layer of size 4 (discrete gridworld actions), and the actor's input is the state one-hot code. For the control experiments, the actor-critic agent is defined on top of the representation as an MLP of two hidden layers of size 64 with tanh activations that feed two output heads: a linear critic head, and a logsoftmax action head for the 4 actions. The agent is trained with A2C with MC returns and a discount of $\gamma = 0.98$, a batchsize of 80 episodes, an entropy regularization with a 0.01 coefficient, and Adam optimizer [Kingma and Ba, 2014] with a learning rate of 0.001.

C.2 MUJOCO: ANTMAZE

The Ant agent has a 29 dimensional state space and a 8 dimensional action space (4 legs with 2 joints each to control). For the sake of simplifying the RL training algorithm, we mapped each action-dimension interval to a discrete set of 5 values equally spaced over this interval.

We used 2 mazes similar in shape to those from Wu et al. [2019] (see Figure 11): ANTMAZE-1 defines a 3D U-shaped corridor and ANTMAZE-2 is a 3D swirl-shaped corridor.



(a) ANTMAZE-1 (b) ANTMAZE-2

Figure 11: ANTMAZE domains. Goal positions of the evaluation tasks are shown in green.

We used the same architectures for the representation and the policies as for the gridworld, with the only difference that for the low-level policy, the action head was adapted to the discretization of the action space by having 8 logsoftmax output heads of size 5, one for each action dimension, and the corresponding 5 discrete values. This choice makes the training algorithm simpler as it allows using A2C here as well.

Our representation is learned in the non-uniform prior setting with $p_r = 0.2$, $p_{rw} = 0.3$ and $K = 500$. We learn a 2-dimensional representation ($d = 2$) using the representation learning objective 6 with $\beta = 0.2$ and $\beta' = 5$. We fixed their length to $c = 100$ steps (so $L = K/c = 5$). and jointly train the representation ϕ and the policies $(\pi_{\text{hi}}, \pi_{\text{low}})$ by collecting, for each update, a batch of $N = 32$ trajectories of length c to fill D_s and D_{π_μ}, as described in Algorithm 1. We train them for 1000 epochs where each epoch corresponds to 10 updates (convergence to the complete representation required around 650 epochs).

The policies were trained with the same A2C used in gridworld domains and the same RMSprop hyperparameters. The high-level and low-level policies were entropy-regularized with the coefficients 0.15 and 0.1, respectively.

For the reward shaping and skills evaluation experiments, the goal positions are shown in Figure 11, and success is defined as being within ϵ from the goal. Here, $\epsilon = 2$ which corresponds to half the size of the building blocks of the mazes.

C.2.1 Reward Shaping

For these experiments, LAP-REP was trained in both the uniform and the non-uniform prior setting.

For the uniform prior setting, we used $d = 2$ and followed the experimental framework of Wu et al. [2019]. Since our ANTMAZE environments are larger, we collected $500,000$ training samples (10 times more than in Wu et al. [2019]) from a uniformly random policy, then we trained the representation on this large dataset. For all other hyperparameters, we used those provided in Wu et al. [2019]. With $d = 20$,

our replication of LAP-REP did not succeed in reward shaping.

Regarding the non-uniform prior setting, we used the same setting configuration as for TATC, with the representation objective from Wu et al. [2019]. We have tested online (similar to TATC) and offline [Wu et al., 2019] data collection for the representation training, and $d \in \{2, 20\}$. Both schemes ended up performed the same way for the reward shaping task.

Now, for the reward shaping, we train a Soft Actor-Critic (SAC) [Haarnoja et al., 2018] agent to reach a goal area (neighbourhood around the goal state) with episodes of size 1000 steps. We use the following hyperparameters:

- Discount $\gamma = 0.99$
- Entropy coefficient (temperature) $\alpha = 0.1$
- Soft critic updates with smoothing constant $\tau = 0.005$
- Replay buffer of size $5 \cdot 10^6$ (equal to the number of training steps).
- Adam optimizer with step size of 0.0001

As SAC is sensitive to the reward scale [Haarnoja et al., 2018], we grid-searched this hyperparameter in $\{10^{-5}, 10^{-4}, \cdots, 1, 2, 10, 20\}$, and the best performing one for our representation was 1 in ANTMAZE-1 and 0.01 in ANTMAZE-2. Regarding LAP-REP, we found that 10 and 0.01 worked the best for these two mazes, respectively. All these coefficients correspond to the dense reward shaping setting. Their values were doubled for the half-half mix reward setting, to account for the 0.5 coefficient.

C.2.2 Skills Evaluation

To train DCO, we first collect a dataset to estimate the second eigenvector and then use the same dataset to train a policy – the option – using DDPG [Lillicrap et al., 2015]. Each DCO option is tied to its own eigenvector estimate and its own training set of size 500000 (10 times the size used in Jinnai et al. [2020]). As suggested by the authors of DCO [Jinnai et al., 2020], the remaining hyperparameters to estimate the eigenvectors and train their corresponding options were taken from Wu et al. [2019]. DIAYN skills were trained as recommended by Eysenbach et al. [2019]. For fair comparison, we train 8 skills for both DCO and DIAYN.

For the skills evaluation stage, we freeze the learned low-level policies and train a high-level policy to use the 8 skills as the only available actions to reach the goal g on the other end of the ANTMAZE-1 environment using a sparse reward $r_t = \mathbb{1}\left[\|s_{t+1} - g\|_2 \leq \epsilon\right]$ within a finite horizon of 1000 steps. Note that this tasks is quite challenging given the type of reward and the length of episode especially in a continuous state space. As our skills offer some flexibility in their execution (can be started everywhere and run for arbitrary number of steps), this episode length was decomposed to 5 skills of 200 steps each. The high-level policy was trained with A2C with MC returns (no discount) a batch size of 8 episodes, and RMSprop optimizer with a learning rate of 0.001.

D THE SWITCHING UTILITY OF THE BOREDOM TERM

Note that $\mathcal{D}_s$, in Equation (5), may contain trajectories from skills that are not yet duly trained; for example early in the training or in a freshly discovered area. Since at that stage, these skills' trajectories are close to random walks, their contribution in the boredom term is similar to the first attractive term, in Equation (2). This means that a new skill trajectory initially contributes to the temporal similarity term (attractive term) in training the representation, thus making the most out of the sampled skills' trajectories while these are still early in their training. The more a skill is trained, the more structured its trajectories become and the more they contribute to the intended "boredom" effect (Section 3.4), that is encouraging exploration and dynamics awareness (Appendix A).

E CONNECTION TO BEHAVIORAL MUTUAL INFORMATION

There are numerous methods in the *intrinsic control* literature that aim at maximizing the mutual information between the agent's behavior and a conditioning variable that encodes the available skills.

This type of intrinsic control is achieved by training a skill conditioned policy, π, to maximize the mutual information between the skill code, z, and some representation of the trajectory, τ, obtained from the conditioned policy $\pi(\cdot|z)$. This objective can be written as:

$$I(z; f(\tau)) = \mathcal{H}(z) - \mathcal{H}(z|f(\tau)), \tag{7}$$

where entropy is denoted by $\mathcal{H}$, and f is a function of the trajectory. It is also common to assume that z is sampled from a fixed prior [Eysenbach et al., 2019], which simplifies this policy training objective as a minimization of the conditional entropy term in Equation (7). In practice, the adopted training loss, can be derived as a lower bound of this quantity, using an approximate posterior, q:

$$L_q(\pi) = -\mathbb{E}_{z,\pi}[\log q(z|f(\tau))]. \tag{8}$$

Traditionally, the integrand of this expectation defines the intrinsic reward of the skill conditioned policy, while the approximate posterior q is trained to discriminate the correct z based on the observed behavior τ.

Inspired by the fast inference offered by Successor Features, Hansen et al. [2020] proposed to use a *log-linear* discriminator in the successor representation (SR), ϕ_{SR}. In other words, the skill rewards can be written as:

$$r(s) = \phi_{\text{SR}}(s)^\top \mathbf{w}, \tag{9}$$

with $\mathbf{w}$ playing the role of the skill-identifying variable (denoted by z in the general case above). Note that in this case, the function f maps, as it is commonly the case, to the final state of the trajectory. Now, consider the case where the trajectory is instead represented by the (normalized) latent direction of the final transition (s, s'). This reward would be

$$r(s, s') = \frac{(\phi_{\text{SR}}(s') - \phi_{\text{SR}}(s))^\top \mathbf{w}}{\|\phi_{\text{SR}}(s') - \phi_{\text{SR}}(s)\|_2}. \tag{10}$$

Let's recall that the SR[1] shares the same eigenvectors as the Laplacian [Stachenfeld et al., 2014, Machado et al., 2018]. This implies that it can also be approximated with a temporally-contrastive objective [Wu et al., 2019], and potentially replaced by our alternative TATC representation ϕ. Finally, we can rewrite Equation (10) as

$$r(s, s') = \frac{(\phi(s') - \phi(s))^\top \mathbf{w}}{\|\phi(s') - \phi(s)\|_2} \tag{11}$$

This reward corresponds to our skills intrinsic reward from Equation (3), with $\mathbf{w} \equiv \boldsymbol{\delta}$.

References

Benjamin Eysenbach, Julian Ibarz, Abhishek Gupta, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *ICLR*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *ICLR*, 2020.

Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6d - a separate, adaptive learning rate for each connection. 2012.

Yuu Jinnai, Jee Won Park, Marlos C. Machado, and George Konidaris. Exploration in reinforcement learning with deep covering options. In *ICLR*, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[1]The SR is encoded by the matrix $(I - \gamma T)^{-1}$, with T the MDP's transition matrix.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, 2015.

Marlos C. Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *ICLR*, 2018.

Kimberly L Stachenfeld, Matthew Botvinick, and Samuel J Gershman. Design principles of the hippocampal cognitive map. In *NeurIPS*, 2014.

Yifan Wu, George Tucker, and Ofir Nachum. The Laplacian in RL: Learning representations with efficient approximations. In *ICLR*, 2019.