Unsupervised Reinforcement Learning in Environments with Strong Priors

Samuel Triest [1]

Abstract

In practice, deep reinforcement learning algorithms have difficulty scaling to tasks with long time horizons and tend to require a large number of environment interactions in order to achieve acceptable performance. Unsupervised reinforcement learning addresses these issues by learning a set of skills that can be reused to accelerate learning of more complex downstream tasks. While prior work focuses on information-theoretic methods that are agnostic to both task and environment, we argue that environment-specific priors exist and that it is beneficial to incorporate them into the skill discovery process. Using the information-theoretic objectives of unsupervised RL, we provide a simple mechanism for incorporating priors into skill discovery and show that they have a positive impact on the skills learned by an RL agent, as well as performance on downstream planning tasks.

1. Introduction

Unsupervised reinforcement learning is an RL technique that has emerged in the last few years whose goal is to teach agents to learn a diverse set of skills in an environment without access to a reward function. This technique has promise in allowing agents to learn new tasks more quickly by using unsupervised RL as a pre-training step.

In practice, it is often the case that algorithm designers know some information about the agent's environment *a priori* (e.g. a cooking robot should hold its tools at certain orientations) or that certain actions in the environment may be undesirable, or even unsafe (e.g. an autonomous vehicle should not drive off the road). To address these potential issues, we propose augmenting the information-theoretic objectives used in traditional unsupervised RL with environment-specific priors in order to obtain a more focused set of skills to use for learning tasks at test-time.

[*]Equal contribution [1]Computer Science Department, University of Rochester, Rochester, NY, USA. Correspondence to: Samuel Triest <striest@u.rochester.edu>.

2. Related Work

Gregor et al. (Gregor et al., 2016) describe mutual information as a useful objective for unsupervised RL algorithms to learn skills that result in a diverse set of states between skills, while having each skill be precise. Mathematically, given a set of skills Z, they represent inter-skill diversity using the entropy $\mathcal{H}(s_f)$ of the final state distribution and measure skill precision as the log-probability of a final state s_f, given a skill $z \in Z$. When averaged over all states and skills, the resulting objective is the mutual information of final states s_f and skills z, or $I(Z, s_f|s_0) =$

$$\mathcal{H}(s_f|s_0) - \mathcal{H}(s_f|z, s_0) \tag{1}$$

$$= \mathcal{H}(z|s_0) - \mathcal{H}(z|s_f, s_0) \tag{2}$$

As noted by the authors, computing $p(s_f|s_0)$ (which is a requisite step in computing $\mathcal{H}(s_f|z, s_0)$) is impractical as it requires integration over the skill z. Instead, a computationally tractable solution is reached by using equation 2. The first term is easily computable, while relaxing mutual information to the variational bound (Mohamed & Rezende, 2015) allows the use of a learned function approximator $q(z|s_f, s_0)$ to estimate the second term. This results in a practical algorithm called Variational Intrinsic Control (VIC).

Diversity Is All You Need (DIAYN) (Eysenbach et al., 2018) uses a similar information-theoretic approach to learning skills by training a policy π conditioned on a latent variable z to maximize the following:

$$I(S; Z) + \mathcal{H}(A|S) - I(A; Z|S) = \tag{3}$$

$$\mathcal{H}(Z) - \mathcal{H}(Z|S) + \mathcal{H}(A|Z, S) \tag{4}$$

Additional terms are added to the objective of VIC as to minimize the direct effect of actions on the objective. The second term encourages the policy to take actions randomly, and the third term encourages the policy to keep actions from increasing the discriminability of Z. This ultimately results in an additional term in the practical objective (Equation 4), which is implicitly solved for by using Soft Actor-Critic (SAC) (Haarnoja et al., 2018a) on the objective of VIC. DIAYN also differs from VIC in its use of all states in the objective (instead of final states), as well as sampling uniformly from the distribution of skills, instead of learning it.

Eysenbach et al. discuss using a function of state $f(s)$ (i.e. train $q(z|f(s))$ instead of $q(z|s)$) to deal with priors. This inherently focuses the search on subspaces of the state space. They show improved results in high-dimensional tasks by encoding prior information as to what skills are useful via their choice of subspace.

Achaim et al. (Achiam et al., 2018) use an autoencoder-based approach to learning diverse skills. That is, they sample a context variable $z \sim Z$ and use it to condition a policy $\pi_\theta(\cdot|s, c)$. The trajectory τ produced by the policy is used as input to a decoder function D_ψ (also referred to as the discriminator by other work) that attempts to reconstruct the context variable z. To train the policy and decoder, they maximize the following objective:

$$J(\theta, \psi) = \mathop{\mathbb{E}}_{c\sim C}[\mathop{\mathbb{E}}_{\tau\sim\pi_\theta, c}[log(D_\psi(c|\tau))] + \alpha\mathcal{H}(\pi)] \quad (5)$$

Their practical algorithm, Valor, uses this objective to optimize a policy to learn distinct skills, as in prior work. However, they train their discriminator to predict context given entire trajectories, as opposed to states.

Florensa et al. (Florensa et al., 2017) use a hand-designed reward and mutual information regularization to learn a skill repertoire to improve exploration on downstream tasks.

Sharma et al. (Sharma et al., 2020) incorporate model-based reinforcement learning, which allows skills to be used in planning algorithms. They incorporate parameterized distribution of next state, given current state and latent code $q_\phi(s'|z, s)$ and train this, as well as a policy π_θ that is trained to maximize the following:

$$log(\frac{q_\phi(s'|s, z)}{\Sigma_{i=1}^{L} q_\phi(s'|s, z_i)}) + log(L)$$

where each z_i is sampled from the distribution of latent codes. Intuitively, this reward incentivizes the policy to transition to the same state for a given latent code z_i, and different states for different latent codes. This achieves the desired effect of creating reproducible skills z_i that are mutually distinct.

Green et al. (Green & Kelly, 2007) also address the issue of state-covering paths from a traditional planning perspective. Unlike the previous RL-based approaches, Green et al. work directly in path-space to minimize the dispersion (low dispersion is preferable) of a set of paths. These paths are then sampled from for downstream tasks.

3. Unsupervised Reinforcement Learning as Auto-Encoding

We would like to draw additional attention to the formulation of unsupervised reinforcement learning as variational auto-encoding (Kingma & Welling, 2013) to note that the notion of incorporating priors into unsupervised reinforcement learning algorithms falls out naturally from the theory provided in (Achiam et al., 2018). Achiam et al. draw a one-to-one correspondence between option discovery and the β-VAE objective of (Higgins et al., 2017):

$$\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[\mathop{\mathbb{E}}_{z\sim q_\phi(\cdot|x)}[log(p_\theta(x|z))] - \beta D_{KL}(q_\phi(z|x)||p(z))] \quad (6)$$

where z is the latent variable, x is the data, and p and q are the decoder and encoder respectively. Achiam et al. note that z corresponds to trajectories τ, x corresponds to skills c, the decoder p corresponds to the discriminator (which maps from trajectories to skills), and the policy π, along with the underlying MDP correspond to the encoder q.

Of particular interest is the regularization term in the objective that attempts to minimize the KL-divergence of the latent variable to some arbitrary distribution. Given that the latent variable in unsupervised RL actually corresponds to trajectories τ, there is a mechanism inherently present in biasing the trajectory distribution of our skills towards particular distributions. Achiam et al. show that traditional entropy regularization corresponds to minimizing the KL-divergence to the trajectory distribution produced by a policy that uniformly samples from the action space in every state (yielding Equation 5). We apply a similar proof in the following section to illustrate that we can formulate an objective that minimizes KL-divergence to the trajectory distributions of arbitrary policies.

4. Mathematical Derivation

We are able to incorporate priors into the unsupervised learning objective by training an RL agent to maximize the following reward:

$$r(s, a, s') = log(D_\psi(c|s')) + \beta log(\pi_{prior}(a|s)) \quad (7)$$

4.1. The Unsupervised Reward

The unsupervised reward $log(D_\psi(s'))$ follows directly from the implementation of DIAYN from Eysenbach et al. (Eysenbach et al., 2018). By training the agent to maximize the performance of the discriminator, the agent is encouraged to visit unique regions of the state space depending on the context variable.

4.2. The Prior Reward

To learn in the presence of environment specific priors, we add an additional term $\beta log(\pi_{prior}(a|s))$ that encourages the learned policy to minimize its KL-divergence to some prior, encoded as a policy. We note that the asymmetry of

KL-divergence will cause skills to exhibit mode-seeking behavior. That is, they will concentrate highly on peaks in the prior's trajectory distribution without attempting to cover it.

We want to minimize the KL-divergence of the trajectory distribution induced by $\pi_\theta(a|s,c)$ to the trajectory distribution of $\pi_{prior}(a|s)$. We can evaluate this $D_{KL}(p(\tau|\pi_\theta)||p(\tau|\pi_{prior}))$ as:

Definition of KL-divergence:

$$D_{KL}(p(\tau|\pi_\theta)||p(\tau|\pi_{prior})) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[log(\frac{p(\tau|\pi_\theta)}{p(\tau|\pi_{prior})})]$$

Expand probability of τ in terms of state transition model and policy:

$$= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[log(\frac{\mu_0(s_0)\Pi_{t=0}^{t=T-1}p(s_{t+1}|s_t,a_t)\pi_\theta(a_t|s_t,c)}{\mu_0(s_0)\Pi_{t=0}^{t=T-1}p(s_{t+1}|s_t,a_t)\pi_{prior}(a_t|s_t)})]$$

Cancel like terms:

$$= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[log(\frac{\Pi_{t=0}^{t=T-1}\pi_\theta(a_t|s_t,c)}{\Pi_{t=0}^{t=T-1}\pi_{prior}(a_t|s_t)})]$$

Log of quotient as difference of logs:

$$= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\Sigma_{t=0}^{t=T-1}log(\pi_\theta(a_t|s_t,c))-\Sigma_{t=0}^{t=T-1}log(\pi_{prior}(a_t|s_t))]$$

Split expectation:

$$= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\Sigma_{t=0}^{t=T-1}log(\pi_\theta(a_t|s_t,c)]-\mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[\Sigma_{t=0}^{t=T-1}log(\pi_{prior}(a_t|s_t))]$$

Definition of entropy and cross-entropy and definition of stationary policy:

$$= T(-\mathcal{H}(\pi_\theta(\cdot,|s,c)) + \mathcal{H}_{\pi_\theta}(\pi_{prior}(\cdot|s))) \quad (8)$$

Using this, we can see that if we want our learned skills to minimize the KL-divergence to a trajectory distribution induced by a prior, we need to minimize equation 8.

4.3. Deriving a Practical Objective

We can thus incorporate priors into unsupervised reinforcement learning by using the following as our objective:

$$J(\theta,\psi) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[log(D_\psi(c|\tau))+\alpha(-\mathcal{H}_{\pi_{prior}}(\tau)+\mathcal{H}(\pi|c))] \quad (9)$$

The optimization problem thus becomes:

$$\theta^*,\psi^* = \arg\max_{\theta,\psi} \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}[log(D_\psi(c|\tau))+\alpha(-\mathcal{H}_{\pi_{prior}}(\tau)+\mathcal{H}(\pi|c))]$$

$$= \arg\max_{\theta,\psi} \mathop{\mathbb{E}}_{s\sim\pi_\theta}[log(D_\psi(c|s)+\alpha log(\pi_{prior}(s))+\alpha\mathcal{H}(\pi(\cdot|s,c))] \quad (10)$$

As mentioned in other work, ψ can be solved for via a decoder network that is trained through standard supervised learning (as the context variables are provided). We can solve for θ by solving a reinforcement learning problem with the reward provided in equation 7. Note that we assume that some max-entropy RL algorithm such as SAC (Haarnoja et al., 2018a) is being used. As such, we omit the entropy term from the reward. Additionally, we include a baseline of $-log(p(c))$ described in (Eysenbach et al., 2018). Lastly, we deviate from the theory in our implementation and use a different, fixed coefficient β to scale the influence of the prior on learned skills.

We can thus arrive at Algorithm 1 for unsupervised RL with priors:

5. Experimental Setup

We provide several experiments to validate the efficacy of including priors in unsupervised reinforcement learning.

5.1. Algorithmic Implementation

5.1.1. Discriminator

Following the implementation of (Eysenbach et al., 2018) , our discriminator is implemented as a simple feed-forward neural network with two fully-connected layers of 300 neurons each (full details are provided in the appendix). The network accepts as input the agent's observation and outputs a categorical distribution over the context space. Like in (Eysenbach et al., 2018), out discriminator can be conditioned on subspaces of the observation space to decide which features of the observation space the agent should learn diverse skills over.

5.1.2. Policy

Like the discriminator, our policy is a two-layer feed-forward neural network with 300 neurons per layer. Per the implementation of SAC (Haarnoja et al., 2018a), the network accepts as input the agent's observation and a context variable, and outputs a squashed Gaussian over the action space. We train this network using the version of SAC provided in (Haarnoja et al., 2018b), which does not parameterize the V function of the underlying POMDP, and automatically tunes the value of α.

5.2. Environments

We consider two types of environments for this work:

1. Gridworld environments
2. Autonomous driving-based environments

Algorithm 1 Unsupervised RL with priors

Input: Untrained networks π, Q_1, Q_2, D with params $\theta, \phi_1, \phi_2, \psi$, learning rates $\lambda_\theta, \lambda_\phi, \lambda_\psi$. Prior π_{prior}, prior weight β
while *not converged* **do**
  $c \sim p(c)$, $s \sim p_0(s)$ ◁ Sample context and initial state
  for $k \leftarrow 1$ **to** *steps_per_iteration* **do**
    $s', t \sim \pi(\cdot|c, s_0)$ ◁ Roll out trajectory
    $\mathcal{D} = \mathcal{D} \cup \{s, a, s', t, c\}$ ◁ Add to replay buffer
    if *t* **then**
      $c \sim p(c)$, $s \sim p_0(s)$ ◁ sample new trajectory
    end
  end
  for $i \leftarrow 1$ **to** *update_steps_per_iteration* **do**
    $s, a, s', t, c \sim \mathcal{D}$ ◁ sample from replay buffer
    $\psi \leftarrow \psi + \lambda_\psi(\nabla_\psi J_\psi(D_\psi(s), c))$ ◁ update D
    $\hat{r} = log(D_\psi(c, s')) - log(p(c)) + \beta log(\pi_{prior}(a|s))$ ◁ pseudo-reward
    $\theta, \phi_1, \phi_2 \leftarrow SAC(s, a, s', \hat{r}, t)$ ◁ use supervised RL to maximize the pseudo-reward
  end
end

5.2.1. GRIDWORLDS

We include gridworld environments as a baseline to confirm that the theory provided in Section 4 translates smoothly to the practical algorithm. Our gridworld forms an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, d_0, \gamma)$, with the omission of a reward function R. Our state space is $\mathcal{S} = \{s|s \in \mathbb{R}^2, s \in [-10, 10] \times [-10, 10]\}$. That is, our agent is a point-mass restricted to states within a 20x20 box centered at the origin. Similarly, our action space is $\mathcal{A} = \{a|a \in \mathbb{R}^2, a \in [-1, 1] \times [-1, 1]\}$. The MDP transitions deterministically according to $s_{t+1} = s_t + a_t$, where a_t and a_{t+1} are projected to the closest point in $\mathcal{S}, \mathcal{A}$ if they fall outside the appropriate spaces. The MDP terminates after a certain number of steps t are taken (we choose $t = 40$ for our experiments). We also set our discount factor $\gamma = 0.95$.

5.2.2. DRIVING ENVIRONMENTS

In our experiments, we also consider learning a repertoire of skills for a series of environments based on highway driving. We consider highway driving in particular because highway driving has a number of easily explainable priors for which a number of well-understood controllers exist.

We formulate highway driving with the ego-vehicle controlled using RL as a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, O, \gamma, d_0)$. Our state space $\mathcal{S} = \{s|s \in \mathbb{R}^4\}$ consists of the vehicle's position, orientation and velocity in a highway-relative frame. Our action space $\mathcal{A} = \{a|a \in \mathcal{R}^2\}$ consists of an acceleration and turn rate command. The transition model $\mathcal{T}$ forward simulates the ego-vehicle's state according to the Ackermann-steered vehicle motion model. The vehicle receives as an observation its lateral position on the highway, as well as its heading, velocity, lateral position in-lane, and an integer corresponding to its current lane (1 for the rightmost lane, |lanes| for the leftmost, etc.). We set $\gamma = 0.99$, and terminate an episode if the vehicle drives off-road, or if it successfully drives 200m.

We consider highway driving to be an interesting environment to consider for unsupervised RL with priors because the general rules of highway driving impose a prior on which types of skills are more desirable. These priors include:

1. Vehicles should stay in their lane.
2. Vehicles should stay a safe distance behind other vehicles.

Controllers can be designed that let vehicles exhibit these behaviors (though a single controller cannot be used to learn options). These controllers can easily be represented as stochastic policies that can be incorporated into the unsupervised RL paradigm.

6. Experiments and Results

6.1. Gridworlds

To verify the efficacy of incorporating priors into unsupervised reinforcement learning, we first ran DIAYN as a baseline on the gridworld environment. We chose to learn ten skills. Figure 1 illustrates the resulting skill repertoire (each color represents a different skill) once the algorithm achieved maximum reward. As we can see, all skills follow the same general behavior of quickly moving to a particular point in the state space and staying there.

Given the low dimension of the state space in this environ-

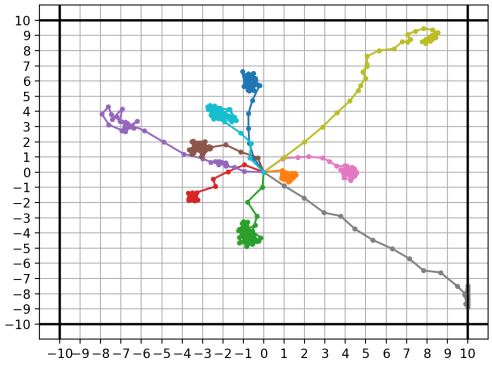

Figure 1. The set of skills learned from DIAYN (Eysenbach et al., 2018)

ment, taking subspaces of the state space to encode a prior is impractical (note that in the DIAYN paper, while they use the notation $f(s)$, they explicitly refer to subspaces of the observation space). We show the results of two different priors in this work. We will refer to these priors as the *diagonal prior* and *row prior*. The diagonal prior was designed as a simple prior on desirable types of behaviors. The row prior was designed as a simple prior on regions of the state space. The diagonal and row priors were represented as stochastic Gaussian policies given in Algorithms 2 and 3.

Algorithm 2 Diagonal Prior for Gridworld

Input: State x, y

$\mu_x = 0.5$**sign(x)**

$\mu_y = 0.5$**sign(y)**

$\sigma_x = 0.2$

$\sigma_y = 0.2$

return $\mathcal{N}(\begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \begin{bmatrix}\sigma_x & 0 \\ 0 & \sigma_y\end{bmatrix})$

Algorithm 3 Row Prior for Gridworld

Input: State x, y, Height bounds y_{upper}, y_{lower}

$\mu_x = 0.0$

$$\mu_y = \begin{cases} -0.5 & \text{if } y > y_{upper} \\ 0.5 & \text{if } y < y_{lower} \\ 0.0 & \text{otherwise} \end{cases}$$

$\sigma_x = 1.0$

$$\sigma_y = \begin{cases} 0.1 & \text{if } y > y_{upper} \text{ or } y < y_{lower} \\ 1.0 & \text{otherwise} \end{cases}$$

return $\mathcal{N}(\begin{bmatrix}\mu_x \\ \mu_y\end{bmatrix}, \begin{bmatrix}\sigma_x & 0 \\ 0 & \sigma_y\end{bmatrix})$

As we can see, the priors achieve the desired effect of altering the learned skill repertoire. In Figure 6.1, we can see that the diagonal prior caused all learned skills to move towards one of the four corners of the environment. Furthermore, though all skills terminate near a corner, the DIAYN objective encourages the skills to approach the corners at different angles. (e.g. the green, red and purple skills reach the top-right from roughly $60°$, $45°$ and $30°$ angles, respectively). In Figure 6.1, we can see that the learned skills maintain the behavior observed in the baseline experiment (find a location and stay there), but now do so within $y \in [-3, 3]$.

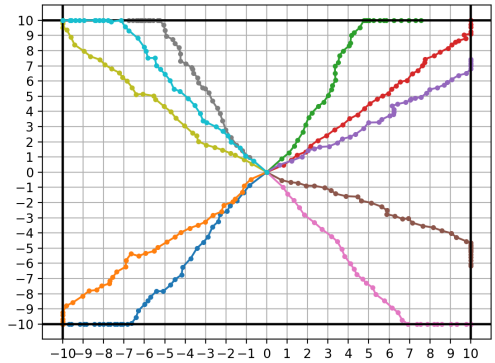

Figure 2. The set of skills learned from adding a diagonal prior to DIAYN

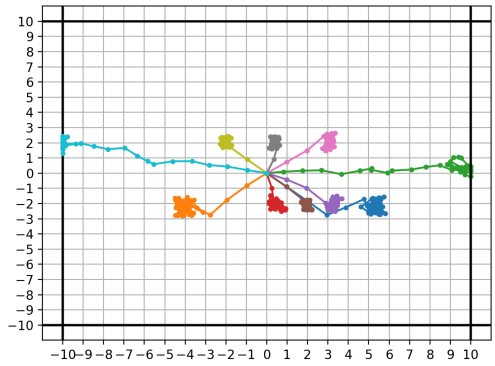

Figure 3. The set of skills learned from adding a row prior to DIAYN

6.2. Driving

We run a number of unsupervised RL experiments on driving environments, both with and without vehicles. We first report a series of experiments on trying to run unsupervised RL without priors in the absence of other vehicles. We first explored learning ten skills on a five-lane highway using various subspaces of the observation space. Figures 4, 5 and 6 illustrate the learned skill repertoire at convergence

when using the full observation space (Figure 4), velocity, lane id and lane boundaries (Figure 5) and velocity and lane id (Figure 6). Figures 5 and 6 illustrate a potential concern with not incorporating a prior. When we include the vehicle's lane position, the unsupervised RL algorithm will attempt to learn skills that partition along that dimension. As such, we get many skills that drive out of lane as a result of maximizing the objective (Figure 5). However, if we omit the lane position, it no longer has an effect on the reward, and trajectories oscillate within a lane as a result (Figure 6).

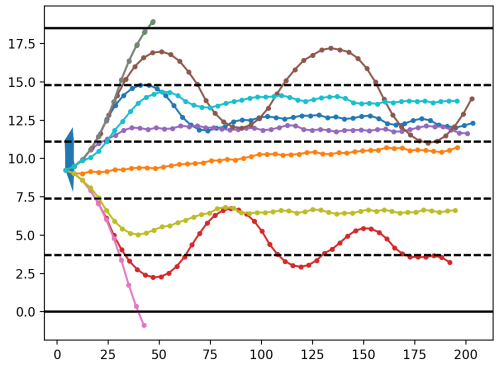

Figure 4. The set of skills learned from running DIAYN on the full observation space

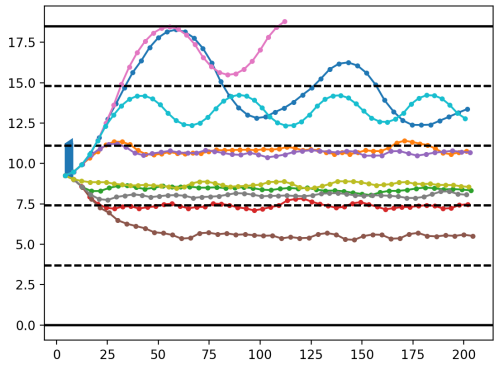

Figure 5. The set of skills learned from running DIAYN on the vehicle's velocity, current lane, and distances to lane line

To address this, we incorporate a lane-following prior based on pure pursuit tracking of the current lane's centerline. We center a Gaussian around the controller's output and feed it into the unsupervised RL algorithm as a prior. Figure 7 shows that doing so has the desired effect of straightening out the learned skills. One thing to note is that although running the pure pursuit controller as is would only result in keeping to the center lane, the unsupervised RL algorithm's

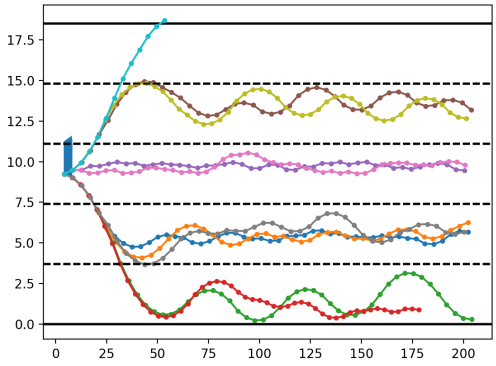

Figure 6. The set of skills learned from running DIAYN on the vehicle's velocity and current lane

objective causes the agent to break from the lane-following controller in the early steps. As a result, the agent learns to lane-change without a reward function.

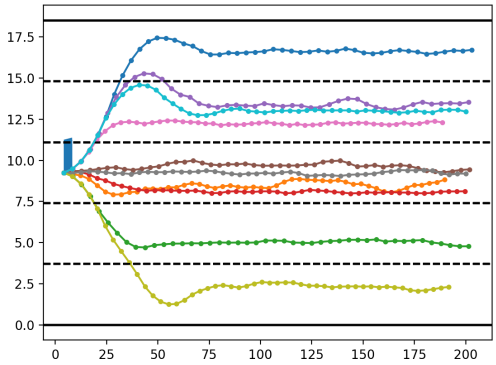

Figure 7. The set of skills learned from adding a lane-keeping prior to DIAYN

We also ran an experiment to learn skills in dense traffic. Specific details on the experiment setup are provided in the appendix. Overall, we found that these skills did not lane-keep as well as the traffic-free skills.

6.3. Evaluation on Downstream Tasks

To quantify the effect of learning skills with a prior on downstream tasks, we constructed a highway navigation scenario in which an agent is required to navigate to a random lane using a set of learned skills to plan.

6.3.1. Highway Navigation Task

To be more specific, we consider a fairly simplistic highway driving task in which the ego vehicle begins in a random

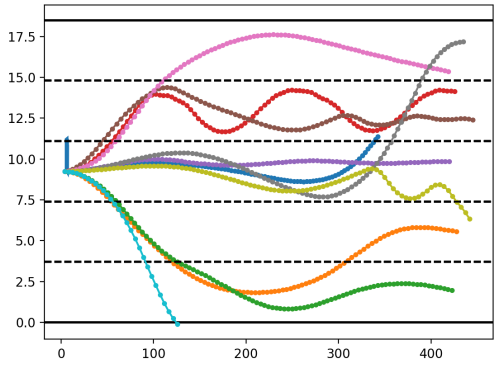

Figure 8. The set of skills learned from running DIAYN with a lane keeping prior in traffic

lane in a five-lane highway, and is expected to navigate to a random target lane through randomly generated traffic patterns. We observe that the analytical form of the Ackermann-steered motion model makes this environment amenable to traditional planning techniques, and choose to directly plan using various skill repertoires instead of learning a meta-policy.

Given the task, we use two primary metrics to evaluate the usefulness of various skill repertoires; safety and efficiency. We measure safety through the success rate of the planning algorithm (i.e. does the vehicle reach the goal lane without collision) and measure efficiency through the number of samples the planning algorithm takes to find a solution, as well as how long it takes to generate each sample.

For all experiments, we plan using a state-lattice planner, where the motion primitives we use are the various skill repertoires mentioned in the previous section. In particular we consider the following sets of skills:

1. Skills learned from the minimum-dispersion method of (Green & Kelly, 2007) (n=10)

2. Skills learned from DIAYN (Eysenbach et al., 2018) on a traffic-free environment where the discriminator is trained on lane id and velocity (n=10))

3. Skills learned from DIAYN with a lane-keeping prior on a traffic-free environment, also conditioned on lane id and velocity (n=10)

At each state, motion primitives are generated by forward-simulating each skill for two seconds. A complete version of the algorithm is provided in the appendix. We report the results of each set of primitives over 1000 simulations each in Table 1.

We note that the learned skill repertoires outperform the min-dispersion repertoire in terms of success rate. We suspect that this is due to the fact that the learned skills are adaptive (i.e. are dependent on initial state), while the min-dispersion skills are not. Since skills are trained to reach a particular state distribution, errors in state from a previous skill can be adjusted for, allowing for more reliable input sampling. It can also be observed that adding the prior significantly reduces the average amount of samples necessary. This is expected, as all the skills with the prior follow the priors in the environment, ultimately leading to improved exploration (and thus fewer samples necessary).

We also note that real-time planning speed is an important factor in the efficacy of these skills. In that regard, the min-dispersion skill set has a significant advantage in that sampling from it does not require the forward pass of a neural network. To account for the decreased performance of the min-dispersion set, we increased the amount of paths in the min-dispersion path set until its success rate approached the success rate of the learned skills. Results are shown in table 2.

7. Discussion and Future Work

We presented a potential method of incorporating priors into unsupervised reinforcement learning through the addition of controllers into the training process. We show that we can encourage skills to exhibit certain behaviors or bias skills to certain regions by adding an additional term to the pseudo-reward provided in conventional unsupervised RL. Overall, we find that while this approach performs well in simpler environments, we had some difficulty scaling to more complex ones (such as the dynamic obstacles in the driving environment with traffic).

A limitation of our method is the need to represent the prior as a policy. (Florensa et al., 2017) take a similar approach to learning skills with a reward function and mutual information. Arguably, while reward design is not a trivial problem, designing a reward function is generally simpler than designing a policy in environments where there are not existing controllers. However, a potential advantage of explicitly matching trajectory distributions is that it may be possible to extract a prior from trajectory samples. In environments such as highway driving, large corpora of unlabeled trajectories exist (ngs; Krajewski et al., 2018). If we view these trajectories as samples from an expert prior, we may be able to train an agent to learn diverse, expert-like options.

References

Next generation simulation (ngsim). URL https://ops.fhwa.dot.gov/trafficanalysistools/ngsim.htm.

Table 1. Performance of Planner under Different Skill Repertoires

Skills	Success Rate	Avg Num. Samples	Avg. Time per Sample
Min-Dispersion (1)	86%	140.5	5 μs
DIAYN (2)	96.1%	155.1	23 μs
DIAYN + prior (3) (Ours)	97.1%	90.2	23 μs

Table 2. Performance of the min-dispersion path sampler under different numbers of paths

Skills	Success Rate	Avg Num. Samples	Avg. Time per Sample
Min-Dispersion (n=10)	86%	140.5	5 μs
Min-Dispersion (n=50)	91.2%	341.5	5 μs
Min-Dispersion (n=100)	93.0%	523.2	5 μs

Achiam, J., Edwards, H., Amodei, D., and Abbeel, P. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

Coulter, R. C. Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Green, C. and Kelly, A. Toward optimal sampling in the space of paths. In *13th International Symposium of Robotics Research*, volume 3, pp. 1. Citeseer, 2007.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018a.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. *Iclr*, 2(5):6, 2017.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Krajewski, R., Bock, J., Kloeker, L., and Eckstein, L. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2118–2125. IEEE, 2018.

Mohamed, S. and Rezende, D. J. Variational information maximisation for intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pp. 2125–2133, 2015.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised skill discovery. In *Proceeding of the International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia*, pp. 26–30, 2020.

A. Experiment Details

Below, we list the hyperparameters for implementation of DIAYN with priors.

Table 3. Hyperparameters for SAC

Hyperparamter	Value
Network	$2 \times$ [Dense(300)]
Training Steps	3×10^6
Activation Function	Tanh
Optimizer	Adam (Kingma & Ba, 2014)
Learning Rate	5×10^{-4}
Replay Buffer Size	1×10^6
Soft Target Update	0.995
β	0.05
Target Entropy	-2

Return curves are provided for the driving experiments. Incorporating a prior has little effect on performance, while incorporating traffic into the training process affects the speed and convergence of learning significantly, regardless of the prior.

B. Driving Environment Details

We include more details about our driving environment below. We assume that the RL agent receives processed inputs from some perception system and outputs controls that are executed by a high frequency controller. Our agent outputs actions at 5hz.

B.1. Observations of Other Vehicles

We choose to represent observations of other vehicles as a fixed feature vector. For each vehicle observed, the agent receives a vector $[\Delta x, \Delta y, \theta, \Delta v]$ corresponding to the displacement from the ego-vehicle to the observed vehicle along the direction of the highway (Δx) and orthogonal to the direction of the highway (Δy), the heading of the observed vehicle, and the difference in the two vehicles' velocities. These observations are made for six vehicles, the vehicles immediately in front and behind the ego vehicle (or its projection) in the ego-vehicle's lane, the lane immediately to its left, or immediately to its right. If one of these vehicles does not exist, a placeholder observation is used instead.

B.2. Pure Pursuit Controller

For lane-following, we implement the pure pursuit controller (Coulter, 1992). Since the pure pursuit controller is deterministic, we center a Gaussian on its output with standard deviation 0.1. We also scale the lookahead with the ego-vehicle's velocity to improve the stability of the controller for high speeds.

B.3. Simulated Traffic

We simulated traffic patterns by randomly populating a window around the ego vehicle. This was accomplished by randomly placing a vehicle every $25m$ with probability 0.4. We then added an offset sampled from $[1, 25]m$ as the initial position of the vehicle. All vehicles moved forward at 21m/s.

B.4. Planning Task

We ran our planning task by first initializing a random traffic pattern, then assigning the ego vehicle to a random lane. We then assigned another random lane as the goal lane. We considered a state a goal state if the ego-vehicle's lateral position was within $\pm 0.5m$ of the goal lane's centerline and its angle to the centerline was within $\pm 0.05rad$.

B.5. Planning Algorithm

Our planning algorithm was essentially a heuristic-based search with an RRT using the set of skills as motion primitives. We forward-simulated each skill for a number of timesteps (10 timesteps at 5hz) and expanded the new state if none of the forward-simulated states were in a collision state. Our heuristic was the L_2 norm of heading deviation and lateral distance to the goal lane.

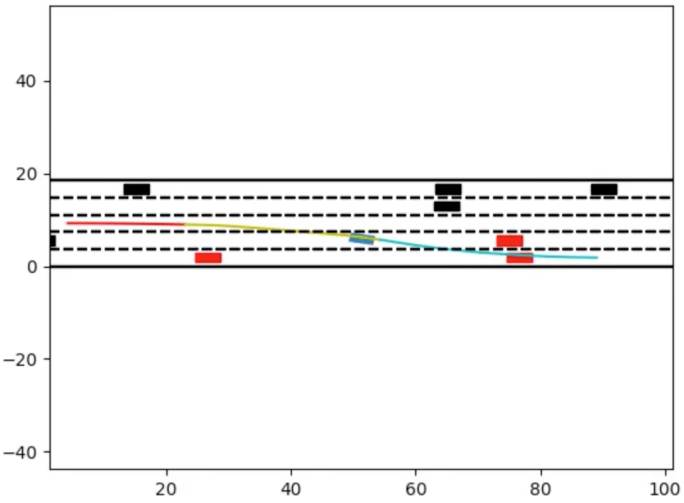

Figure 9. An example plan generated from learned skills with a prior. The vehicle follows the curve. Different colors represent different skills.

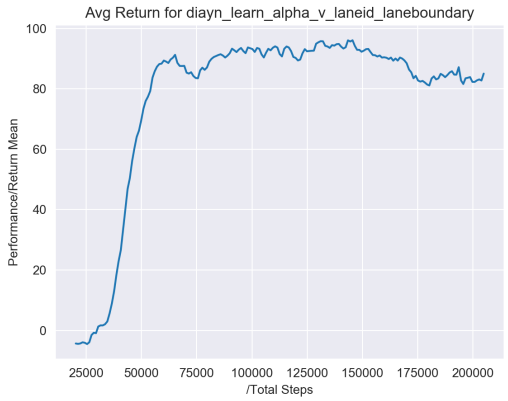

Figure 10. Return curve for DIAYN without prior on highway driving

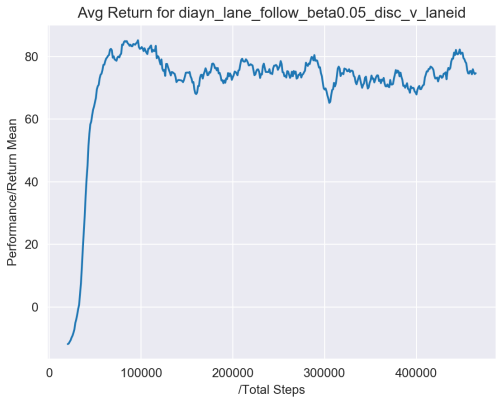

Figure 11. Return curve for DIAYN with prior on highway driving

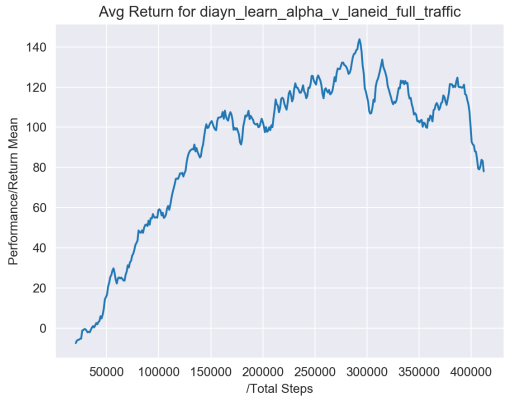

Figure 12. Return curve for DIAYN with prior on highway driving with traffic

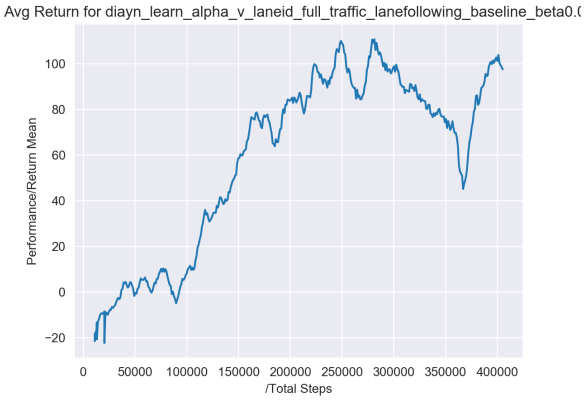

Figure 13. Return curve for DIAYN with prior on highway driving with traffic