Improving Zero-shot Generalization in Offline Reinforcement Learning using Generalized Similarity Functions

Bogdan Mazoure[*]
McGill University, Quebec AI Institute
`bogdan.mazoure@mail.mcgill.ca`

Ilya Kostrikov
UC Berkeley

Ofir Nachum
Google Brain

Jonathan Tompson
Google Brain

Abstract

Reinforcement learning (RL) agents are widely used for solving complex sequential decision making tasks, but still exhibit difficulty in generalizing to scenarios not seen during training. While prior online approaches demonstrated that using additional signals beyond the reward function can lead to better generalization capabilities in RL agents, i.e., using self-supervised learning (SSL), they struggle in the offline RL setting, i.e., learning from a static dataset. We show that performance of online algorithms for generalization in RL can be hindered in the offline setting due to poor estimation of similarity between observations. We propose a new theoretically-motivated framework called Generalized Similarity Functions (GSF), which uses contrastive learning to train an offline RL agent to aggregate observations based on the similarity of their expected future behavior, where we quantify this similarity using *generalized value functions*. We show that GSF is general enough to recover existing SSL objectives while also improving zero-shot generalization performance on two pixel-based offline RL benchmarks.

1 Introduction

Reinforcement learning (RL) is a powerful framework for solving complex tasks that require a sequence of decisions. The RL paradigm has allowed for major breakthroughs in various fields, e.g. outperforming humans on video games [1, 2], controlling stratospheric balloons [3] and learning reward functions from robot manipulation videos [4]. More recently, RL agents have been tested in a generalization setting, i.e. in which training involves a finite number of related tasks sampled from some distribution, with a potentially distinct sampling distribution during test time [5, 6, 7]. The main issue for designing generalizable agents is the lack of on-policy data from tasks not seen during training: it is impossible to enumerate all variants of a real-world environment during training and hence the agent must extrapolate from a (limited) training task collection onto a broader set of problems. Since the learning agent is given no training data from test-time tasks, this problem is referred to as zero-shot generalization. In our work, we are interested in the problem of zero-shot generalization where the difference between tasks is predominantly due to perceptually distinct observations. An example of this setting is any environment with distractor features [8, 9], i.e. features with no dependence on the reward signal nor the agent's decisions. This generalization setting has recently received much attention [10, 11, 12], due to its particular relevance to real-world scenarios, for example deploying the same autonomous driving agent at day or at night.

---

[*] Work done while at Google Brain.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Generalization capabilities of an agent can be analyzed through the prism of *representation learning*, under which the agent's current belief about a rich and high-dimensional environment are summarized in a low-dimensional entity, called a representation. Recent work in online RL has shown that learning state representations with specific properties such as disentanglement [13] or linear separability [14] can improve zero-shot generalization performance. Achieving this with limited data (i.e. offline RL) is challenging, since the representation will have a large estimation error over regions of low data coverage. A common solution to mitigate this task-specific overfitting and extracting the most information out of the data consists in introducing auxiliary learning signals other than instantaneous reward [15]. As we show later in the paper, many such signals already contained in the dataset can be used to further improve generalization performance. For instance, the generalization performance of PPO on Procgen remains limited even when training on 200M frames, while generalization-oriented agents [15, 12] can outperform it by leveraging additional auxiliary signals. However, a major issue with the aforementioned methods is their exorbitant reliance on online access to the environment, an impractical restriction for real-world scenarios.

In contrast, in many real-world scenarios access to the environment is restricted to an offline, fixed dataset of experience [16, 17]. A natural limitation for generalization from offline data is that policy improvement is dependent on dataset quality. Specifically, high-dimensional problems such as control from pixels require large amounts of training experience: a standard training of PPO [18] for 25 million frames on Procgen [8] generates more than 300 Gb of data, an impractical amount of data to share for offline RL research. Improving zero-shot generalization performance from an offline dataset of high-dimensional observations is therefore a hard problem due to limitations on dataset size and quality.

In this work, we are interested in improving zero-shot generalization across a family of Partially-Observable Markov decision processes [POMDPs, 19] in an offline RL setting, i.e. by training agents on a fixed dataset. We hypothesize that in order for an RL agent to be able to generalize across perceptually different POMDPs without adaptation, observations with similar future behavior should be assigned to close representations. We use the generalized value function (GVF) framework [20] to capture future behavior with respect to any instantaneous signal (called cumulant) at a given state. Specifically, the choice of cumulant determines the nature of the behavioral similarity that is induced into state representations. For example, using reward similarity leads to learning bisimulation metrics [21, 22, 23, 24], while using future state-action visitation counts encourages reward-free behavioral similarity [25, 10, 11, 12].

Our main contributions are as follows:

1. We propose Generalized Similarity Functions (GSF), a novel self-supervised learning algorithm for reinforcement learning that aggregates latent representations of observations by their future behavior (or generalized value function).
2. Existing offline RL benchmarks [26, 27] are not well-suited to test zero-shot generalization, and so we devise two new benchmarks: offline Procgen and Distracting Control Suite. The first consists of 5M transitions from 200 related levels of 16 distinct games; the second consists of 1M transitions from 3 variations of 4 distincts tasks.
3. We evaluate performance of GSF and other baseline methods on both benchmarks, and show that GSF outperforms both previous state-of-the-art offline RL and representation learning baselines on the entire distribution of levels.
4. We analyze the theoretical properties of GSF and describe the impact of hyperparameters and cumulant functions on empirical behavior in both offline Procgen and the offline Distracting Suite benchmarks.

2 Related Works

Generalization in reinforcement learning Generalizing a model's predictions across a variety of unseen, high-dimensional inputs has been extensively studied in the static supervised learning setting [28, 29, 30, 31]. Generalization in RL has received a lot of attention: extrapolation to unseen rewards [32, 25], observations [24, 15, 10, 11, 12] and transition dynamics [33]. Each generalization scenario is best solved by their respective set of methods: sufficient exploration [25, 34], auxiliary learning signals [35, 36, 37] or data augmentation [33, 38]. Data augmentation is a promising technique, but typically relies on handcrafted domain information, which might not be available *a priori*. In fact, we will show in our experiments that generalization in the offline RL setting is poor even when using such handcrafted data augmentations, without additional representation learning mechanisms. In this work, we posit that representation learning should use instantaneous auxiliary signals in order to prevent overfitting onto a unique signal (e.g. reward across tasks) and improve generalization performance. Theoretical generalization guarantees have only been provided so far for limited scenarios, mostly for bandits [39], linear MDPs [40, 41, 42] and across reward functions [32, 43, 44].

Representation learning For simple POMDPs, near-optimal policies can be found by optimizing for the reward alone. However, more complex settings may require additional auxiliary signals in order to find state abstractions better suited for control. The problem of learning meaningful state representations (or abstractions) for planning and control has been extensively studied previously [45, 22], but saw real breakthroughs only recently, in particular due to advances in self-supervised learning (SSL). Outside of RL, SSL has achieved spectacular results by closing the gap between unsupervised and supervised learning on certain datasets [46, 47, 48, 49]. Representation learning, and specifically self-supervised learning, has also been used to achieve state-of-the-art generalization and sample efficiency results in RL on challenging control problems such as data efficient Atari [2, 50], DeepMind Control [11] and Procgen [36, 37, 15, 12]. Noteworthy instances of theoretically-motivated representation learning methods for RL include heuristic-guided learning [51], random Fourier features [42] and metric learning [52, 53].

Offline reinforcement learning When learning from a static dataset, agents should balance interpolation and extrapolation errors, while ensuring proper diversity of actions (i.e. prevent collapse to most frequent action in the data). Popular offline RL algorithms such as BCQ [54], MBS [55], and CQL [56] rely on a behavior regularization loss [57] as a tool to control the extrapolation error. Some methods, such as F-BRC [58] are defined only for continuous action spaces while others, such as MOReL [59] estimate a pessimistic transition model. The major issue with current offline RL algorithms such as CQL is that they are perhaps overly pessimistic for generalization purposes, i.e. CQL and MBS ensure that the policy improvement is well-supported by the batch of data.

3 Problem setting

3.1 Partially-observable Markov decision processes

A (infinite-horizon) partially-observable Markov decision process [POMDP, 19] M is defined by the tuple $M=\langle\mathcal{S},p_0,\mathcal{A},p_{\mathcal{S}},\mathcal{O},p_{\mathcal{O}},r,\gamma\rangle$, where $\mathcal{S}$ is a state space, $p_0=\mathbb{P}[s_0]$ is the starting state distribution, $\mathcal{A}$ is an action space, $p_{\mathcal{S}}=\mathbb{P}[\cdot|s_t,a_t]:\mathcal{S}\times\mathcal{A}\to\Delta(\mathcal{S})$ is a transition function, $\mathcal{O}$ is an observation space, $p_{\mathcal{O}}=\mathbb{P}[\cdot|s_t]:\mathcal{S}\to\Delta(\mathcal{O})$[2] is an observation function, $r:\mathcal{S}\times\mathcal{A}\to[r_{\text{min}},r_{\text{max}}]$ is a reward function and $\gamma\in[0,1)$ is a discount factor. The system starts in one of the initial states $s_0\sim p_0$ with observation $o_0\sim p_{\mathcal{O}}(\cdot|s_0)$. At every timestep $t=1,2,3,..$, the agent, parameterized by a policy $\pi:\mathcal{O}\to\Delta(\mathcal{A})$, samples an action $a_t\sim\pi(\cdot|o_t)$. The environment transitions into a next state $s_{t+1}\sim p_{\mathcal{S}}(\cdot|s_t,a_t)$ and emits a reward $r_t=r(s_t,a_t)$ along with a next observation $o_{t+1}\sim p_{\mathcal{O}}(\cdot|s_{t+1})$.

The goal of an RL agent is to maximize the cumulative discounted rewards $\sum_{t=0}^{\infty}\gamma^t r_t$ obtained over the entire episode. Value-based off-policy RL algorithms achieve this by estimating the state-action value function under a target policy π:

$$Q^{\pi}(s_t,a_t)=\mathbb{E}_{\mathbb{P}_t^{\pi}}[\sum_{k=1}^{\infty}\gamma^k r(s_{t+k},a_{t+k})|s_t,a_t], \tag{1}$$

for $s_t\in\mathcal{S},a_t\in\mathcal{A}$ and where $\mathbb{P}_t^{\pi}$ denotes the joint distribution of $\{s_{t+k},a_{t+k}\}_{k=1}^{\infty}$ obtained by executing π in the environment.

An important distinction from online RL is that, in the offline RL setting, we assume access to a historical dataset $\mathcal{D}^{\mu}$ (instead of a simulator) collected by logging experience of the policy, μ, in the form $\{o_{i,t},a_{i,t},r_{i,t}\}_{i=1,t=1}^{i=N,t=T}$ where, for practical purposes, the episode is truncated at T timesteps. Furthermore, we assume that the agent can only be trained on a limited collection of POMDPs $\mathcal{M}_{\text{train}}=\{M_i\}_{i=1}^{m}$, and its performance is evaluated on the set of test POMDPs $\mathcal{M}_{\text{test}}$. We assume that both $\mathcal{M}_{\text{train}}$ and $\mathcal{M}_{\text{test}}$ were sampled from a common task distribution and that every POMDP $M_i\in\mathcal{M}=\mathcal{M}_{\text{train}}\cup\mathcal{M}_{\text{test}}$ shares the same transition dynamics and reward function with $\mathcal{M}$ but has

---

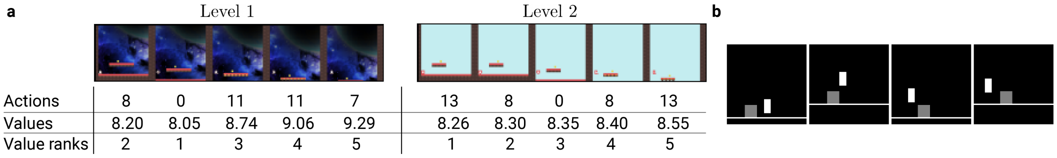
[2] $\Delta(\mathcal{X})$ denotes the entire set of distributions over the space $\mathcal{X}$.

a different observation function $p_{i,\mathcal{O}}$. Importantly, since we perform control from pixels, we are in the POMDP setting [see 60] and therefore emphasize the difference between observations o_t and corresponding states s_t throughout the paper.

3.2 Representation learning

Previous works in the RL literature have studied the use of auxiliary signals to improve generalization performance. Among others, [10, 11] define the similarity of two observations to depend on the distance between action sequences rolled out from that observation under their respective optimal policies. They achieve this by finding a latent space $\mathcal{Z} \subseteq \mathcal{S}$ in which the distance $d_{\mathcal{Z}}(z,z')$ for all $z,z' \in \mathcal{Z}$ is equivalent to distance between true latent states $d_{\mathcal{S}}(s,s')$ for all $s,s' \in \mathcal{S}$; the aforementionned works learn $\mathcal{Z}$ by optimizing action-based similarities between observations. In practice, latent space z is decoded from observation o using a latent state decoder $f:\mathcal{O}\to\mathcal{Z}$ from observation o_t. Throughout the paper, we assume that all value functions have a linear form in the latent decoded state, i.e. $Q_\theta(o,a)=\theta_a^\top f_\psi(o)=\theta_a^\top z_\psi$, which agrees with our practical implementation of all algorithms. Within this model family, the ability of an RL agent to correctly decode latent states from unseen observations directly affects its policy, and therefore, its generalization capabilities. In the next section, we discuss why representation learning is important for offline RL, and how existing action-based similarity metrics fail to recover the true latent states for important families of POMDPs.

4 Motivating example



	Level 1					Level 2				
Actions	8	0	11	11	7	13	8	0	8	13
Values	8.20	8.05	8.74	9.06	9.29	8.26	8.30	8.35	8.40	8.55
Value ranks	2	1	3	4	5	1	2	3	4	5

Figure 1: **(a)** Two levels of the Climber game from the Procgen benchmark [8] with *near-identical* true latent states and *near-identical* value functions but *drastically different* action sequences. **(b)** Four levels of the jumping task [61] where the constant reward signal makes policy similarity more informative than state value similarity.

Multiple recently proposed self-supervised objectives [10, 11] conjecture that observations $o_1 \in M_1, o_2 \in M_2$ that emit similar future action sequences under optimal policies π_1^*,π_2^* should be decoded into nearby latent states z_1,z_2. While this heuristic can correctly group observations with respect to their true latent state in simple action spaces, it fails to identify similar pairs of trajectories in POMDPs with multiple optimal policies[3]. For instance, two trajectories might visit an identical set of latent states, but have drastically different actions.

Fig. 1a shows one such example: two levels of the Climber game have a near-identical true latent state (see Appendix) and value function (average normalized mean-squared error of 0.0398 across episode), while having very different action sequences from a same PPO policy (average total variation distance of 0.4423 across episode). The problem is especially acute in Procgen, since the PPO policy is high-entropy for some environments (see Fig. 5), i.e. various levels can have multiple drastically different near-optimal policies, and hence fail to properly capture observation similarities.

In this scenario, assigning observations to a similar latent state by value function similarity would yield a better state representation than reasoning about action similarities. On the other hand, Fig. 1b shows a domain where grouping state representations by action sequences can be optimal. So how do we unify these similarity metrics under a single framework? In the next section, we use this insight to design a general way of improve representation learning through self-supervised learning of discounted future behavior.

---

[3] While optimal policies are not guaranteed to be unique, the optimal value function is unique.

5 Method

We propose measuring a generalized notion of future behavior similarity using generalized value functions, as defined by the corresponding cumulant function. The choice of cumulant determines which components of the future trajectory are most relevant for generalization.

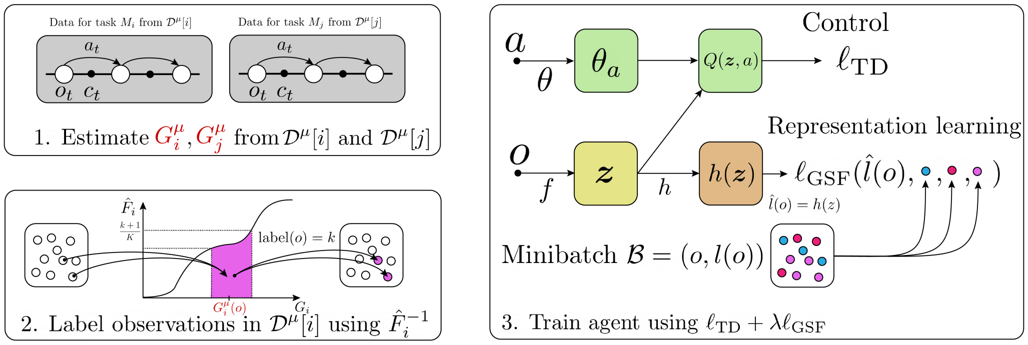

Data for task M_i from $\mathcal{D}^\mu[i]$
Data for task M_j from $\mathcal{D}^\mu[j]$
a_t
o_t c_t
1. Estimate G_i^μ, G_j^μ from $\mathcal{D}^\mu[i]$ and $\mathcal{D}^\mu[j]$
$\hat{F}_i$
$\frac{k+1}{K}$
$\frac{k}{K}$
label$(o) = k$
$G_i^\mu(o)$
G_i
2. Label observations in $\mathcal{D}^\mu[i]$ using $\hat{F}_i^{-1}$
a
θ
θ_a
$Q(z,a)$
Control
ℓ_{TD}
Representation learning
o
f
z
h
$h(z)$
$\ell_{\text{GSF}}(\hat{l}(o), \bullet, \bullet, \bullet)$
$\hat{l}(o) = h(z)$
Minibatch $\mathcal{B} = (o, l(o))$
3. Train agent using $\ell_{\text{TD}} + \lambda\ell_{\text{GSF}}$

Figure 2: Schematic view of GSF : the offline dataset $\mathcal{D}^\mu$ is used to estimate POMDP-specific GVFs wrt some cumulant function c, whose quantiles are then used to label each observation in the dataset.

5.1 Quantifying future behavior with GVFs

An RL agent's future discounted behavior can be quantified not only by its value function, but other auxiliary signals, for example, by its future observation occupancy measure, known as successor representation [62, 32]. The choice of the signal used during value iteration measures the properties the agent will exhibit in the future, such as accumulated returns, actions, or observation visitation density. See Thm. 1 in the Appendix for the connection between successor features and interpolation error in our method.

Following the work of [20], we can broaden the class of value functions to any kind of cumulative discounted signal, as defined by a bounded cumulant function $c : \mathcal{O} \times \mathcal{A} \to \mathbb{R}^d$, s.t. $|c(o,a)| \leq c_{\max}$ for $c_{\max} = \sup_{o,a \in \mathcal{O} \times \mathcal{A}} c(o,a)$. While typically cumulants are scalar-valued functions (e.g. reward), we also make use of the vector-valued case for learning the successor features [32], in which case the norm of $c(o,a)$ is bounded.

Definition 1 (Generalized value function) *Let c be any bounded function over* $\mathbb{R}^d$*, let* $\gamma \in [0,1]$ *and* μ *any policy. The generalized value function is defined, for any timestep* $t \geq 1$ *and* $o_t \in \mathcal{O}$*, as*

$$G^\mu(o_t) = \mathbb{E}_{\mathbb{P}_t^\mu}\Big[\sum_{k=1}^{\infty} \gamma^k c(o_{t+k}, a_{t+k}) | o_t\Big]. \tag{2}$$

Since, in our case, we can learn G^μ for each distinct POMDP M_i for the dataset $\mathcal{D}^\mu$, we index the GVF using the POMDP index, i.e. $G_i^\mu = \texttt{LearnGVF}(c, \mathcal{D}^\mu, i)$ (in practice, learning is parallelized).

Algorithm 1: $\texttt{LearnGVF}(c, \mathcal{D}^\mu, i, \theta^{(0)}, J, \alpha, \gamma)$: Offline estimation of GVF $\hat{G}_i^\mu$

```
Input :Cumulant function $c$, dataset $\mathcal{D}^\mu$, POMDP label $i$, initial parameters
       $\theta^{(0)}$, target parameters $\tilde{\theta}$, latent state decoder $f$, iterations $J$, learning rate $\alpha$, discount $\gamma$
for $j = 1,..,J$ do
    $o, a, o' \sim \mathcal{D}[i]$; // Sample transition from subset corresponding to POMDP $i$
    $c \leftarrow c(o,a)$;
    $o \leftarrow$ random crop$(o)$;
    $z, z' \leftarrow f(o), f(o')$;
    $\theta^{(j)} \leftarrow \theta^{(j-1)} - \alpha \nabla_{\theta^{(j-1)}} (G_{\theta^{(j-1)}}(z) - c - \gamma G_{\tilde{\theta}^{(j-1)}}(z'))^2$ ;
    Update target parameters $\tilde{\theta}$ with $\beta$ of online parameters $\theta$;
```

5.2 Measuring distances between GVFs of different POMDPs

Examining the difference between future behaviors of two observations quantifies the exact amount of expected behavior change between these two observations. Using the GVF framework, we could compute the distance between $o_1 \in M_1$ and $o_2 \in M_2$ by first estimating the latent state with $z = f(o)$ using a latent state decoder f, and then using the following distance as a measure of dissimilarity

$$d_\mu({o_1}^i, {o_2}^j) = ||G_i^\mu(f(o_1)) - G_j^\mu(f(o_2))||, i,j = 1,... \tag{3}$$

However, the distance between GVFs from two different POMDPs can have drastically different scales, i.e. $\sup_{o_1,o_2}|G_1^\mu(o_1) - G_2^\mu(o_2)| \leq \frac{c^\mu_{1,\max}+c^\mu_{2,\max}}{1-\gamma}$, thus making point-wise comparison meaningless. The issue is less acute for cumulants which are homogenous between different POMDPs (e.g. indicator functions for successor representation), and more problematic when the cumulant incorporates a more heterogeneous signal, such as the extrinsic reward function. To avoid this problem, we suggest performing a comparison based on order statistics.

Namely, a robust distance estimate between GVF signals across POMDPs can be obtained by looking at the cumulative distribution function of G_i denoted $F_i(g) = \mathbb{P}[G_i(o_t) \leq g]$ for all $o_t \in \mathcal{O}$. G_i is a deterministic GVF with the set of discontinuity points of measure 0, and as such F_i can be understood through the induced state distribution $\mathbb{P}_t^\mu$ (using continuous mapping theorem from [63]). It can be estimated from n independent and identically distributed samples of $\mathcal{D}^\mu$ as

$$\hat{F}_i(g) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{G_i<g}, G_i = \texttt{LearnGVF}(c,\mathcal{D}^\mu,i), g \in \big[-\frac{c_{i,\max}}{1-\gamma}, \frac{c_{i,\max}}{1-\gamma}\big] \tag{4}$$

and its inverse, the empirical quantile function [64]

$$\hat{F}_i^{-1}(p) = \inf\{g \in \left[-\frac{c^\mu_{i,\max}}{1-\gamma}, \frac{c^\mu_{i,\max}}{1-\gamma}\right] : p \leq F_i(g)\}, \tag{5}$$

for $p \in [0,1]$. We use the empirical quantile function to partition the range of all GVFs into K quantile bins, i.e. disjoint sets with identical size where the set corresponding to quantile k is defined as $I_i(k) = \{o \in M_i : F_i^{-1}(\frac{k}{K}) \leq G_i^\mu(o) \leq F_i^{-1}(\frac{k+1}{K})\}$ and its aggregated version as $I(k) = \cup_{i=1}^m I_i(k)$[4]. Importantly, we augment the dataset $\mathcal{D}^\mu$ with observation-specific labels, which correspond to the index of the quantile bin into which the GVF G of an observation $o \in M_i$ falls into $l_i(o) = \max_k \mathbb{1}_{o \in I_i(k)}$.

These self-supervised labels are then used in a multiclass InfoNCE loss [47], which is a variation of metric learning with respect to the quantile distance defined above [65, 66] and this forms the basis of our self-supervised learning objective.

5.3 Self-supervised learning of GSFs

After augmenting the offline dataset with observation labels as described above, we use a simple self-supervised learning procedure to minimize distance in the latent representation space between observations with identical labels.

First, the observation o is encoded using a non-linear encoder $f_\psi : \mathcal{O} \to \mathcal{Z}$ with parameters ψ into a latent state representation $z = f_\psi(o)$[5]. The representation z is then passed into two separate trunks: 1) a linear matrix θ_a which recovers the state-action value function $Q_\theta(o,a) = \theta_a^\top z$, and 2) a non-linear projection network $h_\theta : \mathcal{Z} \to \mathcal{Z}$ with parameters θ_h to obtain a new embedding, used for self-supervised learning. The projection $h_\theta(z)$ is then used in a multiclass InfoNCE loss [47, 66] where a linear classifier $\mathbf{W} \in \mathbb{R}^{|\mathcal{Z}| \times K}$ aims to correctly predict the observation labels (i.e. quantile bins $k = 1,2,..,K$) from $h_\theta(z)$, for temperature $\tau > 0$:

$$\ell_{\text{GSF}}(\theta,\psi,\mathbf{W}) = -\mathbb{E}_{o\sim\mathcal{D}^\mu}\left[\sum_{k=1}^{K} \mathbb{1}_{l(o)=k} \text{LogSoftmax}(\mathbf{W}^\top h_\theta(f_\psi(o))/\tau)_k\right]. \tag{6}$$

---

[4] A special case of quantile binning occurs when $K = n$, in which case the auxiliary task is to predict the rank of the GVF associated to a given observation in the current minibatch.

[5] This encoder is different from the one used to evaluate the GVFs.

5.4 Full Algorithm

Given m training tasks, GSF first learns GVF estimates $G_1^\mu, .., G_2^\mu$ by applying `LearnGVF` to task-specific data from the offline dataset $\mathcal{D}^\mu$. Each data point in $\mathcal{D}^\mu$ is then labeled with the quantile into which its GVF falls. These labels are then used to jointly optimize Eq.6 and a control loss with respect to the encoder parameters. To learn the value function Q_θ, we use CQL [56], which is trained using a linear combination of Q-learning [67, 16] and behavior-regularization:

$$\ell_{\text{CQL}}(\theta) = \mathbb{E}_{o,a,r,o' \sim \mathcal{D}^\mu}[(r + \gamma \max_{a' \in \mathcal{A}} Q_{\tilde{\theta}}(o',a') - Q_\theta(o,a))^2] + \lambda \mathbb{E}_{s \sim \mathcal{D}^\mu}[LSE(Q_\theta(o,a)) - \mathbb{E}_{a \sim \mu}[Q_\theta(o,a)]], \quad (7)$$

for $\lambda \geq 0$, $\tilde{\theta}$ target network parameters[6] and LSE being the log-sum-exp operator [7]. For domains with continuous actions, we also decode the Bolztmann policy π from Q_θ.

Fig. 2 provides a schematic view of the algorithm, while Alg. 2 in the Appendix presents the exact learning procedure for GSF as implemented on top of a CQL agent for a discrete action space.

Recovering existing self-supervised objectives The generality of our framework allows it to recover existing objectives such as CSSC and PSEs by carefully designing the cumulant function. Below, we highlight which existing algorithms can be recovered by GSF.

- **Cross-State Self-Constraint [CSSC, 10]**: In CSSC, observations o_1, o_2 are considered similar if they have identical future action sequences of length K under some fixed policy; a total of $|\mathcal{A}|^K$ distinct classes are possible. This approach can be approximated in our framework by picking $c(o_t, a_t) = \mathbb{1}_{a_t}(a), \forall a \in \mathcal{A}$. The problem reduces to a $|\mathcal{A}|^{T-t}$-way classification problem for observations of timestep t, which GSF approximates using K quantiles.
- **Policy similarity embedding [PSE, 11]**: PSEs balance the distance between local optimal behaviors and long-term dependencies in the transitions, notably using d_{TV}. If we consider the space of Boltzmann policies $\pi_{\text{Boltzmann}}$ with respect to a POMDP-specific value function Q, then choosing $c(o_t, a_t) = r(s_t, a_t)$ in GSF will effectively compute the distance between unnormalized policies.

The choice of K induces a bias-variance trade-off How should the number of quantiles K (read labels) be set, and what is the effect of smaller/ larger values of K on the learned representations? Thm. 1 highlights a trade-off when choosing the number of quantile bins empirically.

Theorem 1 *Let G_1, G_2 be generalized value functions with cumulants c_1, c_2 from respective POMDPs M_1, M_2, K be the number of quantile bins, n_1, n_2 the number of sample transitions from each POMDP. Suppose that $\mathbb{P}[\sup_{t=1,2,..} |c_1(o_{1,t}, \mu(o_{1,t})) - c_2(o_{2,t}, \mu(o_{2,t}))| > {}^{(1-\gamma)\varepsilon}/_{\gamma}] \leq \delta$. Then, for any $k = 1,2,..,K$ and $\varepsilon > 0$ the following holds without loss of generality:*

$$\mathbb{P}\left[\sup_{o_1,o_2 \in I(k)} |G_1(o_1) - G_2(o_2)| > 3\varepsilon\right] \leq 2e^{-2n_1\varepsilon^2/4} + \mathbb{P}\left[\sup_{k=1,2,..,K} \left|\hat{F}_1^{-1}\left({}^{k+1}/_{K}\right) - \hat{F}_1^{-1}\left({}^{k}/_{K}\right)\right| > \varepsilon\right] + p(n_1, K, \varepsilon) + \delta \quad (8)$$

The proof can be found in the Appendix Sec. A.5. For POMDP M_1, the error decreases monotonically with increasing bin number K (second term) but the variance of bin labels depends on the number of sample transitions n_1 (first term). The inter-POMDP error (third term) does not affect the bin assignment. Hence, choosing a large K will amount to pairing states by rankings, but results in high variance, as orderings are estimated from data and each bin will have $n = 1$. Setting K too small will group together unrelated observations, inducing high bias.

Limitations As is the case with all offline RL methods, GSF is limited by the compounding extrapolation error under low data coverage. We hypothesize that wise choices of K and c can mitigate the extrapolation error by learning observation groups with low intra-group variance, but, since they are environment-dependent, searching for an optimal (K, c) pair can be computationally expensive.

---

[6] A copy of θ updated solely using an exponential moving average (see Appendix).

[7] https://en.wikipedia.org/wiki/LogSumExp

6 Experiments

Unlike for single task offline RL [26], most prior work on zero-shot generalization from offline data either come up with an *ad hoc* solution suiting their needs, e.g. [33], or assess performance on benchmarks that do not evaluate generalization across observation functions [e.g., 68]. To accelerate progress in this field, we devised two benchmarks: offline Procgen (discrete actions) and offline Distracting Suite (continuous actions) - two offline RL datasets to directly test for generalization of RL agents across observation functions[8]. Moreover, for a standard comparison, we provide generalization results on the classical online Procgen simulator, comparing PPO to PPO with GSF and PPO with PSE, respectively.

GVF training In the offline setting, the training dataset is used to learn a set of task-specific GVFs in parallel as follows. First, we project each observation o into a latent representation $z = f_{\psi'}(o)$; we then pass z through a non-linear network $h_{\theta'} : \mathcal{Z} \to \mathbb{R}^{d_c \times m}$ where d_c is the dimensionality of the cumulant function's output. The output of $h_{\theta'}$ is then split into m disjoint chunks, which are in turn used in their respective temporal difference losses ℓ_{TD} in place of value functions. The procedure can be adapted to an online setting via a similar procedure, except that all GVF estimators are trained simultaneously and independently in separate simulators.

Offline Procgen benchmark We evaluate the proposed approach on an offline version of the Procgen benchmark [8], which is widely used to evaluate zero-shot generalization across complex visual perturbations. Given a random seed, Procgen supports sampling procedurally generated level configurations for 16 games under various complexity modes: “easy”, “hard” and “exploration”. More details can be found in Appendix.

Offline Procgen results We compare the zero-shot performance on the entire distribution of "easy" POMDPs for GSF against that of strong RL and representation learning baselines: behavioral cloning (BC) - to assess the quality of the PPO policy, CQL [56] - the current state-of-the-art on multiple offline benchmarks which balances RL and BC objectives, CURL [35], CTRL [12], DeepMDP [69] - which learns a metric closely related to bisimulation across the MDP, Value Prediction Network [VPN, 70] - which combines model-free and model-based learning of values, observations, next observations, rewards and discounts, Cross-State Self-Constraint [CSSC, 10] - which boosts similarity of observations with identical action sequences, as well as Policy Similarity Embeddings [34], which groups observation representations based on distance in optimal policy space.

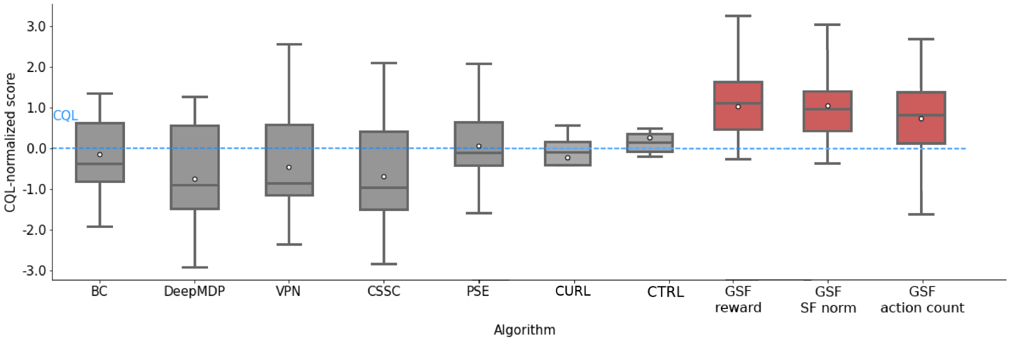


Figure 3: Returns on the offline Procgen benchmark [8] after 1M training steps. Boxplots are constructed over 5 random seeds and all 16 games; each method is normalized by the per-game median CQL performance. White dots represent average of distribution.

Fig. 3 shows the performance of all methods over 5 random seeds and all 16 games on the offline Procgen benchmark after 1 million gradient steps. Per-game average scores for all methods can be found in Tab. 2 (Appendix). The scores are standardized per-game using the downstream task's (offline RL) performance, in this case implemented by CQL. It can be seen that GSF performs better than other offline RL and representation learning baselines.

---

[8] Code can be found at `https://github.com/bmazoure/gsf_public`.

Using different cumulant functions can lead to different label assignments and hence different similarity groups. Fig. 3 examines the performance of GSF with respect to 3 cumulants: 1) $r(s_t,a_t)$, rewards s.t. GSF learns the policy's Q^μ-value, 2) $\mathbb{1}_{o_t}(o)$, the successor representation[9] [62, 32] s.t. GSF learns the distribution induced by μ over $\mathcal{D}^\mu$ [71] and 3) $\mathbb{1}_{a_t}(a)$, action counts, s.t. GSF learns discounted policy. While rewards and successor feature cumulant choices leads to similar performance, using action-based distance leads to larger variance.

Offline Distracting Control Suite results Following the same procedure as for offline Procgen results, we first formed an offline dataset from the challenging Distracting Control Suite [9] by saving the replay buffer of Soft Actor-Critic [72] trained for 1M frames on 4 environments with 2 different background perturbations. Next, we pre-trained G^μ_1, G^μ_2 with action and reward-based cumulants, which were then used in conjunction with CQL to learn a single multi-task policy. Fig 4 shows the online performance of GSF evaluated on 10 background perturbations not seen during training, normalized by per-environment median CQL scores.

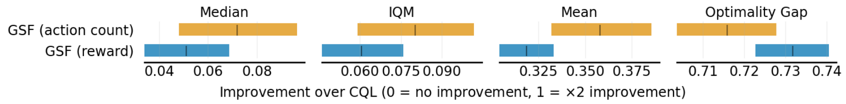


Figure 4: Improvement of GSF over CQL with action and reward-based similarity functions aggregated using performance metrics on 3 seeds and 4 environments of the Distracting Control Suite reported as suggested by [73] with 95% confidence intervals. We can see that using action counts results in higher mean, median and interquartile mean (IQM) statistics and lower optimality gap (i.e. fraction of scores falling under a certain threshold), than when using rewards, or when comparing with the performance of CQL.

The results are consistent with findings of [11], in that 1) learning policy-based similarity improves generalization capabilities of state representations, and 2) unlike in Procgen, policy-based similarity provides a better learning signal than value-based similarity.

Online Procgen results We additionally compare performance of GSF to that of PPO and PPO with PSEs in the classical online Procgen benchmark [5]. Figure 8 shows test returns for 20M frames obtained on the entire distribution of easy levels while training on 200 easy levels for PPO, PPO+PSEs [11] and our PPO+GSFs. PPO+GSF outperforms or matches both PPO and PPO+PSE on most environments, showing that GSF can be efficiently combined with both offline and online algorithms. See Appendix A.6.1 for detailed results.

7 Discussion

In this work we proposed Generalized Similarity Functions, a novel framework which combines reinforcement learning with representation learning to improve zero-shot generalization performance on challenging, pixel-based control tasks. GSF relies on computing the similarity between observation pairs with respect to any instantaneous accumulated signal, which leads to improved empirical performance on two newly introduced benchmarks, offline Procgen and offline Distracting Suite. Theoretical results suggest that GSF's hyperparameter choice depends on a bias-variance trade-off.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[2] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *International Conference on Learning Representations*, 2020.

---

[9] In the continuous observation space, we learn a d-dimensional successor feature vector z_ψ via TD and compute the quantiles over $||z_\psi||_1$.

[3] Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

[4] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289. PMLR, 2019.

[6] Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. *International Conference on Learning Representations*, 2019.

[7] R Devon Hjelm, Bogdan Mazoure, Florian Golemo, Felipe Frujeri, Mihai Jalobeanu, and Andrey Kolobov. The sandbox environment for generalizable agent research (segar). *arXiv preprint arXiv:2203.10351*, 2022.

[8] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[9] Austin Stone, Oscar Ramirez, Kurt Konolige, and Rico Jonschkowski. The distracting control suite–a challenging benchmark for reinforcement learning from pixels. *arXiv preprint arXiv:2101.02722*, 2021.

[10] Guan Ting Liu, Pu-Jen Cheng, and GuanYu Lin. Cross-state self-constraint for feature generalization in deep reinforcement learning. 2020.

[11] Rishabh Agarwal, Marlos C Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. *arXiv preprint arXiv:2101.05265*, 2021.

[12] Bogdan Mazoure, Ahmed M Ahmed, Patrick MacAlpine, R Devon Hjelm, and Andrey Kolobov. Cross-trajectory representation learning for zero-shot generalization in rl. *International Conference on Learning Representations*, 2022.

[13] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017.

[14] Zichuan Lin, Derek Yang, Li Zhao, Tao Qin, Guangwen Yang, and Tie-Yan Liu. Rd^2: Reward decomposition with representation decomposition. *Advances in Neural Information Processing Systems*, 33, 2020.

[15] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. *International Conference on Machine Learning*, 2021.

[16] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

[17] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012.

[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[19] Kevin P Murphy. A survey of pomdp solution techniques. *environment*, 2:X3, 2000.

[20] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768, 2011.

[21] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, volume 4, pages 162–169, 2004.

[22] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. *ISAIM*, 4:5, 2006.

[23] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10069–10076, 2020.

[24] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *International Conference on Learning Representations*, 2020.

[25] Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *International conference on machine learning*, pages 6961–6971. PMLR, 2020.

[26] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[27] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: Benchmarks for offline reinforcement learning. 2020.

[28] Peter L Bartlett. The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, 44(2):525–536, 1998.

[29] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *International Conference on Learning Representations*, 2019.

[30] Guillermo Valle-Pérez and Ard A Louis. Generalization bounds for deep learning. *arXiv preprint arXiv:2012.04115*, 2020.

[31] Lu Liu, William Hamilton, Guodong Long, Jing Jiang, and Hugo Larochelle. A universal representation transformer layer for few-shot image classification. *arXiv preprint arXiv:2006.11702*, 2020.

[32] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[33] Philip J Ball, Cong Lu, Jack Parker-Holder, and Stephen Roberts. Augmented world models facilitate zero-shot dynamics generalization from a single offline environment. *International Conference on Machine Learning*, 2021.

[34] Alekh Agarwal, Mikael Henaff, Sham Kakade, and Wen Sun. Pc-pg: Policy cover directed exploration for provable policy gradient learning. *Neural Information Processing Systems*, 2020.

[35] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *International Conference on Machine Learning*, 2020.

[36] Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *Neural Information Processing Systems*, 2020.

[37] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning. In *International Conference on Machine Learning*, pages 9870–9879. PMLR, 2021.

[38] Samarth Sinha and Animesh Garg. S4rl: Surprisingly simple self-supervision for offline reinforcement learning. *arXiv preprint arXiv:2103.06326*, 2021.

[39] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16(1):1731–1755, 2015.

[40] Justin Boyan and Andrew W Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, pages 369–376, 1995.

[41] Yuanhao Wang, Ruosong Wang, and Sham M Kakade. An exponential lower bound for linearly-realizable mdps with constant suboptimality gap. *arXiv preprint arXiv:2103.12690*, 2021.

[42] Ofir Nachum and Mengjiao Yang. Provable representation learning for imitation with contrastive fourier features. *Neural Information Processing Systems*, 2021.

[43] Ruosong Wang, Yifan Wu, Ruslan Salakhutdinov, and Sham M Kakade. Instabilities of offline rl with pre-trained neural representation. *arXiv preprint arXiv:2103.04947*, 2021.

[44] Ahmed Touati and Yann Ollivier. Learning one representation to optimize all rewards. *arXiv preprint arXiv:2103.07945*, 2021.

[45] Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757. Citeseer, 2005.

[46] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

[47] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[48] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

[49] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[50] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, Devon Hjelm, Philip Bachman, and Aaron Courville. Pretraining representations for data-efficient reinforcement learning. *arXiv preprint arXiv:2106.04799*, 2021.

[51] Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. Heuristic-guided reinforcement learning. *arXiv preprint arXiv:2106.02757*, 2021.

[52] Lanqing Li, Rui Yang, and Dijun Luo. Focal: Efficient fully-offline meta-reinforcement learning via distance metric learning and behavior regularization. *arXiv preprint arXiv:2010.01112*, 2020.

[53] Jiachen Li, Quan Vuong, Shuang Liu, Minghua Liu, Kamil Ciosek, Henrik Christensen, and Hao Su. Multi-task batch reinforcement learning with metric learning. *Advances in Neural Information Processing Systems*, 33:6197–6210, 2020.

[54] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019.

[55] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *NeurIPS*, 2020.

[56] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

[57] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.

[58] Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. *arXiv preprint arXiv:2103.08050*, 2021.

[59] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

[60] Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *arXiv preprint arXiv:1910.01741*, 2019.

[61] Remi Tachet, Philip Bachman, and Harm van Seijen. Learning invariances for policy generalization. *arXiv preprint arXiv:1809.02591*, 2018.

[62] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[63] Henry B Mann and Abraham Wald. On stochastic limit and order relationships. *The Annals of Mathematical Statistics*, 14(3):217–226, 1943.

[64] A. W. van der Vaart. Asymptotic statistics. cambridge series in statistical and probabilistic mathematics, 1998.

[65] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Neural Information Processing Systems*, 2020.

[66] Jiaming Song and Stefano Ermon. Multi-label contrastive predictive coding. *Neural Information Processing Systems*, 2020.

[67] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[68] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning*, pages 1094–1100. PMLR, 2020.

[69] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G Bellemare. Deepmdp: Learning continuous latent space models for representation learning. In *International Conference on Machine Learning*, pages 2170–2179. PMLR, 2019.

[70] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *arXiv preprint arXiv:1707.03497*, 2017.

[71] Marlos C Machado, Marc G Bellemare, and Michael Bowling. Count-based exploration with the successor representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5125–5133, 2020.

[72] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[73] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 34, 2021.

[74] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *International Conference on Learning Representations*, 2021.

[75] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with popart. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3796–3803, 2019.

[76] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pages 1407–1416. PMLR, 2018.

[77] Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Neal Wu, Efi Kokiopoulou, Luciano Sbaiz, Jamie Smith, Gábor Bartók, Jesse Berent, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018. [Online; accessed 4-October-2021].

[78] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

[79] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[80] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes] See end of Section 5
    (c) Did you discuss any potential negative societal impacts of your work? [Yes] We include a brief discussion in Appendix Section A.1

   - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   - (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   - (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix Section A.5
3. If you ran experiments...
   - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix Section A.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   - (a) If your work uses existing assets, did you cite the creators? [N/A]
   - (b) Did you mention the license of the assets? [N/A]
   - (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Due to the large size of the offline datasets, we include the code which can re-generate the data from scratch.
   - (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Broader Impact Statement

Learning generalizable state representations from limited interactions is of paramount importance for allowing for accessible and inclusive research, which does not rely on having access to large compute needs. In this work, we provide a first attempt to show that it is possible to learn generalizable state representations from an offline dataset containing high-dimensional observations. We hope that this work lays foundation for future research exploring data-efficient learning of generalizable and robust state representations from offline data.

However, zero-shot generalization is a double-edged sword: in order to have a good performance on unseen tasks, the agent has to extrapolate its set of inductive biases and knowledge collected on the training data onto unseen tasks. This process can be detrimental and heavily biased by the nature of the training set. For this reason, generally-capable agents, especially in real-world applications of sequential decision-making, have to be properly regularized, e.g. using safe policy iteration mechanisms.

A.2 Experimental details

Name	Description	Value
n_{MDPs}	Number of training POMDPs in the dataset	200
γ	Discount factor	0.99
Batch size	Batch size	1024
Hidden layers	Size of Q critic post-encoder layers	$256 \times 256 \times \lvert\mathcal{A}\rvert$
λ	CQL regularization coefficient	1
Data augmentation	Type of data augmentation use in all methods	Random crop
α	Target soft update rate	0.005
Hidden layers	Size of G network post-encoder layers	$256 \times 256 \times \lvert\mathcal{A}\rvert * n_{\text{MDPs}}$
G pretraining steps	Number of iterations to pretrain GVF estimator for	100,000
K	Number of quantiles	7
τ	Softmax temperature	0.5

Table 1: Experiments' parameters

Target updates: As is common in multiple deep RL algorithms [74, 58], the bootstrap target estimates are typically computed using a value network with parameters $\tilde{\theta}$, which are in turn periodically updated using an exponential moving average of historical values with the current online network parameters θ with update rate α:

$$\tilde{\theta} = \alpha\theta + (1-\alpha)\tilde{\theta} \tag{9}$$

A.2.1 Offline Procgen

For improved time efficiency in the offline experiments, we jointly train the estimators $G_1, G_2, .., G_m$ by first projecting each observation $o \in M_i$ into a latent representation $z = f_{\psi'}(o)$ and then passing the representation through a non-linear network $h' : \mathcal{Z} \to \mathbb{R}^{d_c \times m}$ where d_c is the dimensionality of the cumulant function's output. The output of h' is then split into m disjoint chunks, with chunk i used in the temporal difference loss ℓ_{TD} with the reward replaced by the cumulant function. However, due to different scales of cumulants across POMDPs, we use the PopArt re-normalization scheme [75] to prevent gradient instability. All GVFs are trained for $50k$ iterations using ℓ_{TD}.

Data augmentation was taken to be solely random crops, since this type of data augmentation was shown to be sufficient to improve performance of deep RL agents in pixel-based control tasks [74]. To do so, we first symmetrically padded the $64 \times 64 \times 3$ observation up to $68 \times 68 \times 3$, and applied a 64×64 random crop on the resulting tensor.

We used data augmentation on all methods in Fig. 3 to improve the robustness of the encoder and prevent premature convergence. Precisely, we applied random crops, since this type of data augmentation was shown to be sufficient to improve performance of deep RL agents in pixel-based control tasks [74]. To do so, we first symmetrically padded the $64\times 64\times 3$ observation up to $68\times 68\times 3$, and applied a 64×64 random crop on the resulting tensor. This allowed the information content of each observation to be preserved, while simultaneously preventing the encoder to overfit on static entities such as walls, food pellets, etc.

All baselines' code was taken from their respective repositories and adapted into our codebase (except DeepMDP which had to be implemented from third-party sources).

Hyperparameter search: We allowed each method to tune one hyperparameter due to computational budget constraints. For CQL, λ was tuned and found to be 1. For GSFs, we tuned K, the number of quantiles. For all other methods, we tuned the auxiliary loss coefficient(s). Performance of GSFs vs some hyperparameter choices can be seen in Fig. 7.

Dataset composition The dataset was obtained as follows: we first pre-trained a PPO [18] agent with the IMPALA architecture [76] for 25M timesteps on 200 levels of "easy" distribution for each environment[10] ("easy" mode is widely used to test generalization capabilities [8, 15, 12]). Then we conducted rollouts with the resulting policy under an ε_t-greedy strategy; ε_t was allowed to decay according to the rule $\varepsilon_t = 0.1 - 3.96\times 10^{-9} t$ which has two endpoints: $\varepsilon_0 = 0.1$ and $\varepsilon_{25M} = 0$. This was done in order to prevent collapse of the Q-values due to the log-sum-exp term in CQL to very low negative values (if action coverage is not sufficient in the dataset).

A.2.2 Offline Distracting Suite

Similarly to offline Procgen, we pre-trained action and reward-based generalized value functions on offline data collected by the SAC [72] replay buffer after 1M steps. These estimators were then used to label data in the dataset based and jointly optimize the GSF loss together with the actor loss of CQL. All scores are reported on 10 unseen tasks (different videos playing in background) after 100,000 gradient updates, and normalized by the average performance of CQL for each environment over 3 seeds. The CQL-normalized scores for all four environments (cartpole-swingup, cheetah-run, reacher-easy and walker-walk) were then run through the rlliable library [73] which computes aggregate performance statistics under a low number of random seeds. We used a Lagrangian coefficient for the GSF loss of 1.0 and $K = 10$ quantile bins for all environments in this experiment.

A.3 Detailed Algorithm

Alg.2 shows the learning procedure used by GSFs, on top of CQL. Note that, in our experiments, all methods (baselines and GSF) use random crops as data augmentation.

A.4 Compute resources

For all experiments, we used a mix of P100 and V100 GPUs on our internal cluster to conduct both hyperparameter search, as well as the final training of models reported in the figures and tables over multiple random seeds.

A.5 Proofs

Proof 1 (Thm. 1) *First, consider some arbitrary quantile bin $k = 1,2,..,K$.*

$$\begin{aligned}\sup_{o_1,o_2\in I(k)} |G_1^\mu(o_1) - G_2^\mu(o_2)| &= \sup_{o_1,o_2\in I(k)} |G_1^\mu(o_1) - G_1^\mu(o_2) + G_1^\mu(o_2) - G_2^\mu(o_2)| \\ &\leq \sup_{o_1,o_2\in I(k)} |G_1^\mu(o_1 - o_2)| + |G_1^\mu(o_2) - G_2^\mu(o_2)|\end{aligned} \tag{10}$$

---

[10] We use the TFAgents' implementation [77]

Algorithm 2: GSF : Offline RL with future behavior observation matching

```
Input :Dataset $\mathcal{D} \sim \mu$, initialized Q-function
       $Q_\theta$ with encoder $f_{\theta_f}$ and action weights $\theta_a$, per-POMDP set of GVFs $\mathcal{G} = \{G_i^\mu\}_{i\in}$,
       state projection network $h_\psi$, epoch number $J$, number of POMDPs $m$, number
       of quantiles $K$, temperature parameter $\tau$, exponential moving average parameter $\beta$
for epoch $j = 1,2,..,J$ do
    for minibatch $\mathcal{B} \sim \mathcal{D}$ do
        /* Data augmentation on observation                                  */
        $o \leftarrow$ random crop($o$) for all $o \in \mathcal{B}$;
        $z \leftarrow f_\psi(o)$ for all $o \in \mathcal{B}$ ;
        /* Update CQL agent                                                  */
        Update $\theta_a,\psi$ using $\nabla_{\theta_a,\psi}\ell_{\text{CQL}}(\theta)$ ;
        /* Compute $\mathcal{G}$ quantiles                                   */
        for POMDP $M_i = 1,2,..,m$ do
            Estimate $\hat{F}_i^{-1}$ of $G_i^\mu$ from $\mathcal{B}$ ;
            for observation $o \in \mathcal{B} \cap M_i$ do
                $l(o) \leftarrow k$ if $\hat{F}_i^{-1}(\frac{k}{K}) \leq G_i^\mu(o) \leq \hat{F}_i^{-1}(\frac{k+1}{K})$ ;
        /* Update encoder and projection network                             */
        Update $\theta_h,\psi,\mathbf{W}$ using $\nabla_{\theta_h,\psi,\mathbf{W}}\ell_{\text{GSF}}(\theta_h,\psi,\mathbf{W})$ computed with $z,l(o)$ and $\tau$ ;
        Update CQL agent's target network with $\beta$ of online parameters $\psi,\theta$;
```

Since the cumulant is, in practice, estimated from $\mathcal{D}^\mu$, it follows that $c(s,a) \in [-c^\mu_{i,max}, c^\mu_{i,max}] \subseteq [-c_{max},c_{max}]$ for all $s,a \in \mathcal{S},\mathcal{A}$. Since the disparity between cumulants for POMDP M_1,M_2 comes from the uneven coverage by μ, we can denote this as $\delta_{1,2}(t) = |c(o_{1,t},\mu(o_{1,t})) - c(o_{2,t},\mu(o_{2,t}))|$.

Suppose that $\mathbb{P}[\sup_{k=1,2,..}\delta_{1,2}(t+k) > \varepsilon] \leq p^\mu(c_1,c_2,\varepsilon)$. Then,

$$\begin{aligned}\mathbb{P}[|G_1^\mu(o_t) - G_2^\mu(o_t)| > \varepsilon] &= \mathbb{P}[\mathbb{E}_{\mathbb{P}_t^\mu}[\sum_{k=1}^{\infty}\gamma^k|c_i^\mu(o_{t+k},a_{t+k}) - c_j^\mu(o_{t+k},a_{t+k})||o_t] > \varepsilon] \\ &= \mathbb{P}[\mathbb{E}_{\mathbb{P}_t^\mu}[\sum_{k=1}^{\infty}\gamma^k\delta_{1,2}(t+k)|o_t] > \varepsilon] \\ &\leq \mathbb{P}[\sup_{k=1,2,..,K}\delta_{1,2}(t+k)\mathbb{E}_{\mathbb{P}_t^\mu}[\sum_{k=1}^{\infty}\gamma^k|o_t] > \varepsilon] \\ &\leq \mathbb{P}[\sup_{k=1,2,..,K}\delta_{1,2}(t+k) > \frac{(1-\gamma)\varepsilon}{\gamma}] \\ &\leq p^\mu(c_1,c_2,{}^{(1-\gamma)\varepsilon}/_{\gamma})\end{aligned} \tag{11}$$

since M_1,M_2 share the same induced distribution $\mathbb{P}_t^\mu$. Due to stationarity, we can drop the time index.

Now, the first term can be decomposed with the empirical distribution function $\hat{F}_i^{-1}$ estimated from n_i samples of POMDP M_i:

$$\begin{aligned}\left|F_1^{-1}\left(\frac{k+1}{K}\right) - F_1^{-1}\left(\frac{k}{K}\right)\right| &\leq \sup_{k'=1,2,..,K}\left|F_1^{-1}\left(\frac{k'+1}{K}\right) - F_1^{-1}\left(\frac{k'}{K}\right)\right| \\ &\leq \sup_{k'=1,2,..,K}\Delta(F_1,n_1,\frac{k'+1}{K}) + \Delta(F_1,n_1,\frac{k'}{K}) + \Delta(F_1,n_1,\frac{k'+1}{K},\frac{k'}{K}) \\ &\leq 2\sup_{k'=1,2,..,K}\Delta(F_1,n_1,\frac{k'}{K}) + \sup_{k'=1,2,..,K}\Delta(F_1,n_1,\frac{k'+1}{K},\frac{k'}{K})\end{aligned} \tag{12}$$

where

$$\Delta(F,n,p) = \left|F^{-1}(p) - \hat{F}^{-1}(p)\right|$$

and

$$\Delta(F,n,p_1,p_2)=\left|\hat{F}^{-1}(p_1)-\hat{F}^{-1}(p_2)\right|,$$

and dependence of F on n is implicit.

We now use the union bound to observe the fact that $\mathbb{P}[\sum_{i=1}^{n}|X_i|>n\varepsilon]\le\sum_{i=1}^{n}\mathbb{P}[|X_i|>\varepsilon]$ for $X_1,..,X_n$ real-valued random variables and $\varepsilon>0$.

Using this fact, and that events listed in Eq. 12 form an increasing sequence of supersets

$$\begin{aligned}\mathbb{P}\left[\left|F_1^{-1}\left(\frac{k+1}{K}\right)-F_1^{-1}\left(\frac{k}{K}\right)\right|>2\varepsilon\right]&\le\mathbb{P}\left[\sup_{k'\in[0,1]}\Delta(F_1,n_1,\frac{k'}{K})>\frac{\varepsilon}{2}\right]+\mathbb{P}\left[\sup_{k'=1,2,..,K}\Delta(F_1,n_1,\frac{k'+1}{K},\frac{k'}{K})>\varepsilon\right]\\&\le 2e^{-2n_1\varepsilon^2/4}+p(n_1,K,\varepsilon)\end{aligned}\tag{13}$$

Here, we used the known results of convergence of the empirical distribution function $\hat{F}$ to the true distribution function F as $n\to\infty$ [78]. Using the continuous mapping theorem [63] for a transformation with a set of discontinuities of measure 0, we transposed this result onto the empirical quantile function F^{-1}.

Since the error is monotonic in n, we symmetrize the bound by replacing n_1 by $\min(n_1,n_2)$, so that both M_1 and M_2 can be interchanged.

Proposition 1 *If $c(o_t,a_t)=\mathbb{1}_o(o_t)$ for all $o\in\mathcal{O}$, then $I(K)$ is the set which has the largest estimator variance and $I(1)$ has the smallest estimator variance.*

Proof 2 (Prop. 1) *For the specific choice of cumulant being the state indicator function, the following result is due to* **(author?)** *[71]:*

$$\frac{\gamma}{n(o)+1}-\frac{\gamma^2}{1-\gamma}\le(1+\gamma)-||G(o)||_1\le\frac{\gamma}{n(o)+1},\tag{14}$$

where $n(o)$ is the number of times observation o was visited by policy μ in POMDP M (here, G is a vector-valued function).

Rearranging, we obtain

$$\begin{aligned}\frac{\gamma}{n(o)+1}-\frac{\gamma^2}{1-\gamma}\le(1+\gamma)-||G(o)||_1\le\frac{\gamma}{n(o)+1}\\\frac{\gamma}{n(o)+1}-\frac{\gamma^2}{1-\gamma}-(1+\gamma)\le-||G(o)||_1\le\frac{\gamma}{n(o)+1}-(1+\gamma)\\(1+\gamma)-\frac{\gamma}{n(o)+1}\le||G(o)||_1\le\frac{\gamma^2}{1-\gamma}+(1+\gamma)-\frac{\gamma}{n(o)+1}\end{aligned}\tag{15}$$

The ordering of quantile bins of $\inf_{o\in I(k+1)}||G(o)||_1\ge\sup_{o\in I(k)}||G(o)||_1$ for $k=1,2,..,K-1$ implies the following:

$$\begin{aligned}\frac{\gamma^2}{1-\gamma}+1+\gamma-\inf_{o\in I(k+1)}\frac{\gamma}{n(o)+1}\ge\inf_{o\in I(k+1)}||G(o)||_1\\\sup_{o\in I(k)}||G(o)||_1\ge1+\gamma-\sup_{o\in I(k)}\frac{\gamma}{n(o)+1}\end{aligned}\tag{16}$$

Rearranging the previous inequality and using the approximation $||G(o)||_1\approx-n^{-1}(o)$, we see that ordering quantiles induces an ordering on the number of total samples contained in each bin:

$$\sup_{o\in I(K)}n(o)\le\inf_{o\in I(K-1)}n(o)\le\sup_{o\in I(K-1)}n(o)\le\ldots\le\inf_{o\in I(1)}n(o)\le\sup_{o\in I(1)}n(o)$$

Computing any statistic (e.g. $\hat{G}$) over the set $I(K)$ will yield at most as much samples as the least frequent pair of observations in the set $I(K-1)$, which in turn implies that the variance of the average statistic over $I(K)$ will be at least that of the variance of the average statistic over $I(K-1)$.

A.6 Additional results

True latent state similarity in Procgen The true latent state in Procgen consists of a byte vector describing the entire memory state of the environment and the agent. This vector can be extracted by using the command `list(env.callmethod("get_state"))` in Python. To assess the distance between latent state vectors, we computed the negative cosine similarity between them. Fig. 1 shows two pairs of trajectories, for which the average cosine similarity between true latent states across timesteps was 0.897.

Value-function similarity in Procgen Fig. 1 shows two pairs of trajectories with drastically dissimilar (discrete) action sequences. Computing the distance between them will result in a large quantity which doesn't necessarily decrease with latent state similarity. On the other hand, the values for the first sequence of states are 8.20, 8.05, 8.74, 9.06, 9.29, and 8.35, 8.30, 8.26, 8.55, 8.40 for the second sequence of states. We can see that values are much more similar in this coupling of tasks than actions (with respect to, e.g. ℓ_1) and, if we group the states by their corresponding value magnitude (or quantiles), the encoder will learn to assign all states which look like the sequences above to a neighboring latent representation, and hence, will allow better zero-shot generalization as it will learn to ignore backgrounds, platform length and small variations in agent's position.

		RL+Bisim.	RL+value pred.	RL+action dist.		RL+GVF dist.
Env	BC	DeepMDP [69]	VPN [70]	CSSC [10]	PSE [11]	GSF (reward)
bigfish	-0.443686	-0.296928	0.296928	0.116041	0.189761	0.153584
bossfight	0.319048	-0.945238	-0.757143	-0.878571	-0.355714	0.385714
caveflyer	-0.033058	-0.727273	-0.685950	-0.669421	-0.221488	0.123967
chaser	0.148368	-0.890208	-0.848994	-0.919057	-0.348961	-0.049456
climber	0.631579	-0.596491	-1.000000	-0.859649	-0.263158	1.894737
coinrun	0.070671	-0.742049	-0.597173	-0.597173	0.286926	0.466431
dodgeball	-0.132653	-0.244898	0.010204	-0.102041	-0.112245	-0.030612
fruitbot	-0.373832	-0.747664	-0.780374	-0.747664	-0.262617	0.238318
heist	-0.637681	-0.420290	-0.362319	-0.275362	0.034783	-0.159420
jumper	0.196078	-0.549020	-0.196078	-0.333333	0.035294	0.862745
leaper	-0.285714	0.074286	0.457143	0.497143	0.374857	-0.091429
maze	-0.368421	-0.254386	-0.245614	-0.228070	0.021053	0.122807
miner	-0.060606	-0.454546	-0.393940	-0.424243	0.127273	0.121212
ninja	-0.097015	-0.868159	-0.644279	-0.666667	-0.153731	0.044776
plunder	0.156863	-0.768627	-0.784314	-0.784314	-0.324706	-0.078431
starpilot	-0.022989	-0.772988	-0.750000	-0.767241	-0.220690	0.178161

Table 2: Average performance for 16 games of the Procgen benchmark [8] across 5 random seeds. All scores are standardized by the performance of CQL [56] on the downstream task (zero-shot generalization across the entire distribution of "easy" levels).

Entropy of PPO policies on Procgen While in some domains, policies can achieve optimality without much exploration, the Procgen benchmark requires PPO to have a non-zero entropy-boosting term (otherwise, results are suboptimal).

When the logging policies are high-entropy, many action sequences can possibly lead to high rewards. However, this does not imply that the observations in those sequences must have high similarity in latent space.

Choice of contrastive objective Here, we ablate the choice of the loss function used in GSF.

The multi-class InfoNCE objective is defined as follows. First, we pick the latent space distance to be the negative cosine similarity:

$$d(o_1,o_2)=-\frac{h_\theta(f_\psi(o_1))^\top h_\theta(f_\psi(o_2))}{||h_\theta(f_\psi(o_1))||_2||h_\theta(f_\psi(o_2))||_2} \tag{17}$$

The optimization objective is then taken to be the set-valued InfoNCE loss, defined for a set of positive samples $\mathcal{S}_P$, set of negative samples $\mathcal{S}_N$ and temperature parameter τ. Positive scores are computed

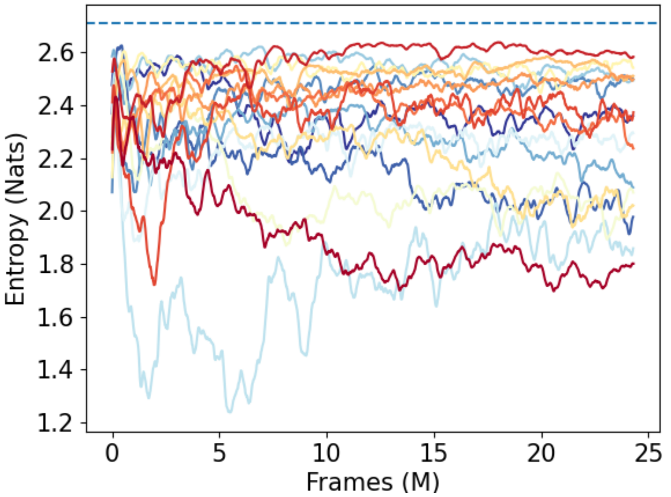


Figure 5: Average conditional entropy of online PPO policy during training phase for all 16 games of Procgen. Dotted line indicates theoretical maximum ($-\log 15$).

over randomly sampled positive pairs (i.e., elements of the same quantile), while negative scores are computed between representations belonging to different quantiles.

$$\ell_{\text{InfoNCE}}(\mathcal{O}_P,\mathcal{O}_N)=-\log\frac{\sum_{o_1,o_2\in\mathcal{O}_P}\exp\left(-d(o_1,o_2)\tau^{-1}\right)}{\sum_{o_1\in\mathcal{O}_P,o_2\in\mathcal{O}_N}\exp(-d(o_1,o_2)\tau^{-1})} \tag{18}$$

The batch version of the loss is defined as

$$\ell_{\text{InfoNCE}}(\theta,\psi)=\mathbb{E}_{\mathcal{O}_P,\mathcal{O}_N\sim\mathcal{B}}[\ell_{\text{InfoNCE}}(\mathcal{O}_P,\mathcal{O}_N)] \tag{19}$$

Fig. 6 compares the loss used by GSF with the alternative loss based on multi-class NCE. The loss based on categorical cross-entropy yields a more stable and lower error, as well as higher test returns. For this reason, we take $\ell_{\text{GSF}}=\ell_{\text{CCE}}$ in all experiments.

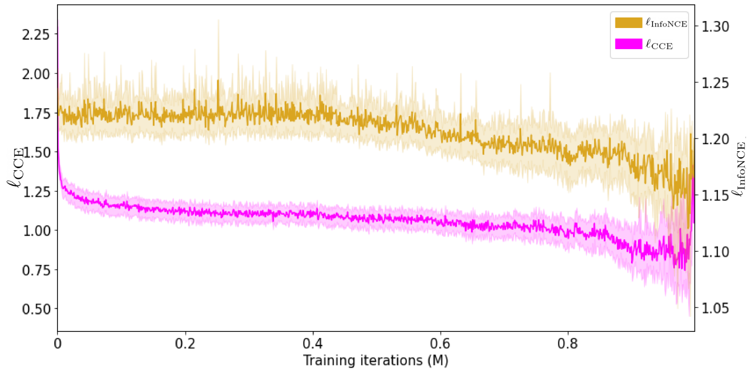


Figure 6: Comparison of two versions of our multi-class contrastive objective: (1) classification of labels via categorical cross entropy [79] and (2) pairwise InfoNCE [47]. Curves are averaged over 16 games and 5 random seeds.

Ablation on hyperparameters Fig. 7 shows performance of GSF for various combinations of hyperparameters, for the GVF being the Q-value of each POMDP. We picked the last combination of hyperparameters, as it has one of the highest inter-game median values, and lowest inter-quartile range (i.e. more stable performance across seeds).

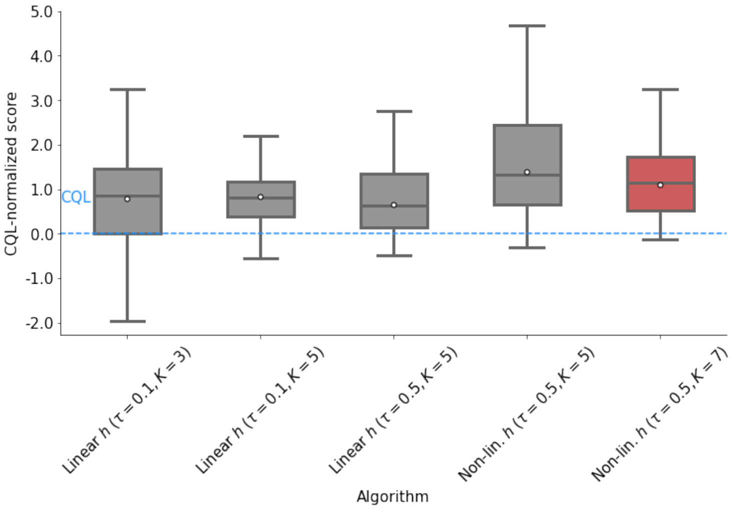


Figure 7: Ablation on GSF hyperparameters: 1) softmax temperature τ, 2) number of bins K and 3) structure of projection network h.

A.6.1 Online Procgen results

We compared PSEs and GSFs in the classical, online Procgen setting. Both PSE and GSF use 5 level-specific pre-trained value functions (PSE uses policies, GSF uses value functions), and all methods' performance is averaged over 5 seeds and tuned over the same parameter space.

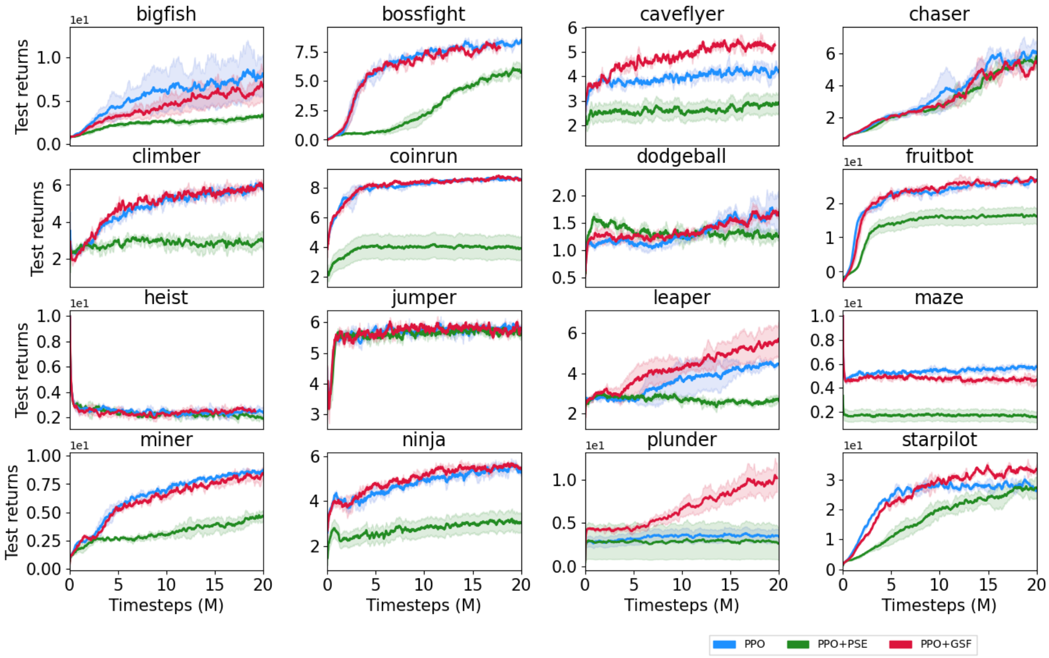


Figure 8: Test returns on the online Procgen benchmark [8] after 20M training frames. Curves are averaged over 5 random seeds.

GSFs can be seen to provide a performance boost in some challenging environments such as Plunder and Leaper, even though it was designed with offline RL generalization improvements in mind. Note that in the Climber game (example of PSE failure mode in Fig. 1), using a value-based similarity is much more beneficial than using action-based similarity.

In the online experiments, we do not use any data augmentation techniques, as all methods are allowed to query state-action pairs in the simulator, and hence no additional stochasticity in the inputs is required.

A.6.2 Qualitative UMAP results

In order to answer the question *"Does using GSFs induce an ordering on the state representations based on the choice of cumulant"*, we conducted the following simple experiment. First, we pre-trained a PPO agent and a PPO+GSF agent on the `Plunder` game in the online setting. Then, we generated 100 trajectories with each respective agent and performed dimensionality reduction of all state representations using UMAP [80]. We then visualized all resulting embeddings, by coloring each state with its value function.

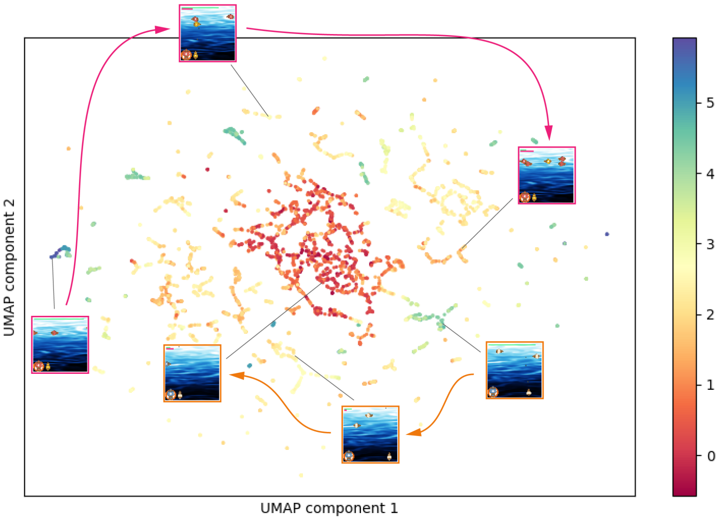


Figure 9: State representations learned with PPO and projected onto a 2-dimensional manifold using UMAP. Colors are proportional to the value function at each state. Some weak structure is visible, where a small subset of states is aggregated together due to having similar state values. States are not linearly separable by their value function.

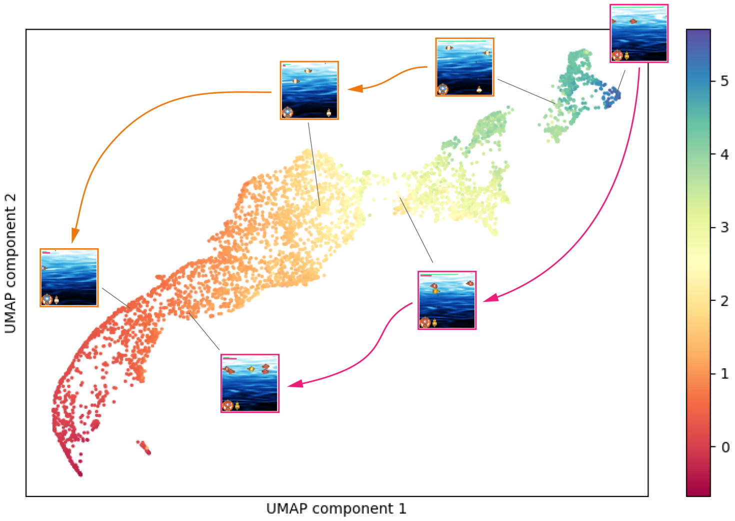


Figure 10: State representations learned with PPO+GSF and projected onto a 2-dimensional manifold using UMAP. Colors are proportional to the value function at each state. A clear separation between low-value and high-value states is visible. States *are* linearly separable by their value function.

Figure 9 shows how state embeddings learned with PPO exhibit a structure which is not entirely dependent on the value function. On the other hand, Figure 10 shows how introducing an auxiliary signal based on GSF aggregates state representations based on the magnitude of their value function.

We also overlaid three sample observations from two sample trajectories obtained on the test set of tasks, and show their approximate location in the space found by UMAP. We can immediately see that GSF representations are ordered not only according to their value function, but also temporally: these observations are mapped close to each other pairwise (e.g. higher value states are mapped close to each other for all tasks). This implies that the policy found by the GSF agent would behave similarly independently of the task-specific information, which implies better robustness to distracting features such as changing background images, color swaps, etc.

For thorough analysis, we also add a visualization of state representations learned in the offline setting by CQL (Figure 11) and CQL+GSF (Figure 12). While the latent structure learned by CQL+GSF is not as clear-cut as in the online setting due to out-of-distribution actions, it still exhibits a relative smoothness in terms of value function, i.e. states with similar value function values are placed close to each other in this space.

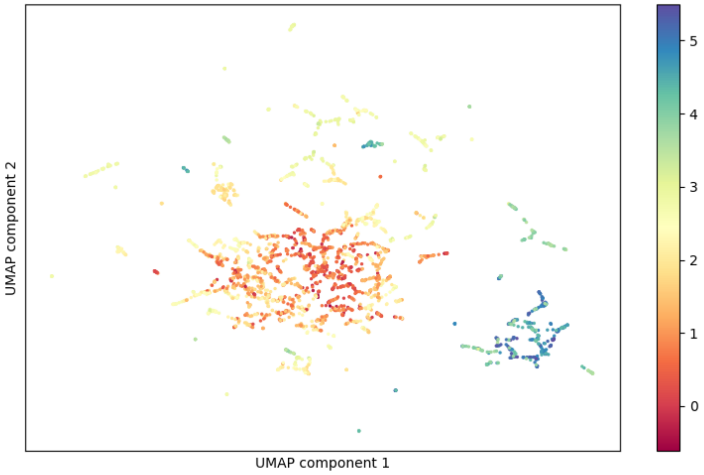


Figure 11: State representations learned with CQL and projected onto a 2-dimensional manifold using UMAP. Colors are proportional to the value function at each state. Some weak structure is visible, where a small subset of states is aggregated together due to having similar state values. States are not linearly separable by their value function.

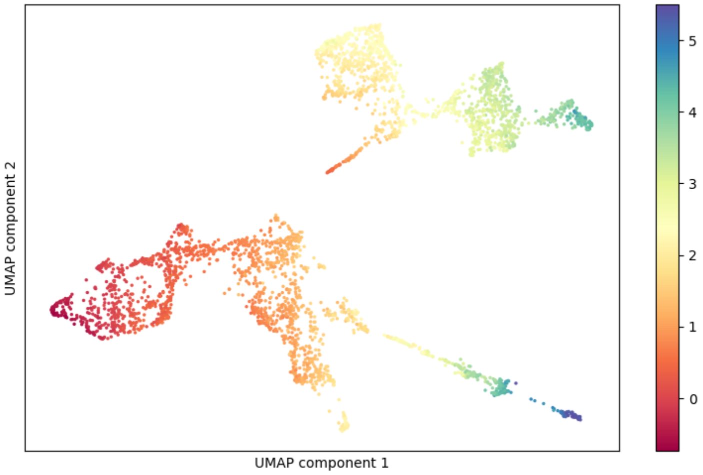


Figure 12: State representations learned with CQL+GSF in the offline setting and projected onto a 2-dimensional manifold using UMAP. Colors are proportional to the value function at each state. A clear separation between low-value and high-value states is visible. States *are* linearly separable by their value function.