REPRESENTATION MATTERS: OFFLINE PRETRAINING FOR SEQUENTIAL DECISION MAKING

Mengjiao Yang
Google Research
sherryy@google.com

Ofir Nachum
Google Research
ofirnachum@google.com

ABSTRACT

The recent success of supervised learning methods on ever larger offline datasets has spurred interest in the reinforcement learning (RL) field to investigate whether the same paradigms can be translated to RL algorithms. This research area, known as *offline RL*, has largely focused on offline policy optimization, aiming to find a return-maximizing policy exclusively from offline data. In this paper, we consider a slightly different approach to incorporating offline data into sequential decision-making. We aim to answer the question, what unsupervised objectives applied to offline datasets are able to learn state representations which elevate performance on downstream tasks, whether those downstream tasks be online RL, imitation learning from expert demonstrations, or even offline policy optimization based on the same offline dataset? Through a variety of experiments utilizing standard offline RL datasets, we find that the use of pretraining with unsupervised learning objectives can dramatically improve the performance of policy learning algorithms that otherwise yield mediocre performance on their own. Extensive ablations further provide insights into what components of these unsupervised objectives – e.g., reward prediction, continuous or discrete representations, pretraining or finetuning – are most important and in which settings[1].

1 INTRODUCTION

Within the reinforcement learning (RL) research field, *offline RL* has recently gained a significant amount of interest (Levine et al., 2020; Lange et al., 2012). Offline RL considers the problem of performing reinforcement learning – i.e., learning a policy to solve a sequential decision-making task – exclusively from a static, offline dataset of experience. The recent interest in offline RL is partly motivated by the success of *data-driven* methods in the supervised learning literature. Indeed, the last decade has witnessed ever more impressive models learned from ever larger static datasets (Halevy et al., 2009; Krizhevsky et al., 2012; Brown et al., 2020; Dosovitskiy et al., 2020). Solving offline RL is therefore seen as a stepping stone towards developing scalable, data-driven methods for policy learning (Fu et al., 2020). Accordingly, much of the recent offline RL research focuses on proposing new policy optimization algorithms amenable to learning from offline datasets (e.g., Fujimoto et al. (2019); Wu et al. (2019); Agarwal et al. (2020); Kumar et al. (2020); Yu et al. (2020); Matsushima et al. (2020)).

In this paper, we consider a slightly different approach to incorporating offline data into sequential decision-making. We are inspired by recent successes in semi-supervised learning (Mikolov et al., 2013; Devlin et al., 2018; Chen et al., 2020), in which large and potentially unlabelled offline datasets are used to learn *representations* of the data – i.e., a mapping of input to a fixed-length vector embedding – and these representations are then used to accelerate learning on a downstream supervised learning task. We therefore consider whether the same paradigm can apply to RL. Can offline experience datasets be used to learn representations of the data that accelerate learning on a downstream task?

This broad and general question has been partially answered by previous works (Ajay et al., 2020; Singh et al., 2020). These works focus on using offline datasets to learn representations of *behaviors*, or actions. More specifically, these works learn a spectrum of behavior policies, conditioned on a latent z, through supervised action-prediction on the offline dataset. The latent z then effectively provides an abstract action space for learning a hierarchical policy on a downstream task, and this straightforward paradigm is able to accelerate learning in a variety of sequential decision-making

[1]Code available at https://github.com/google-research/google-research/tree/master/rl_repr.

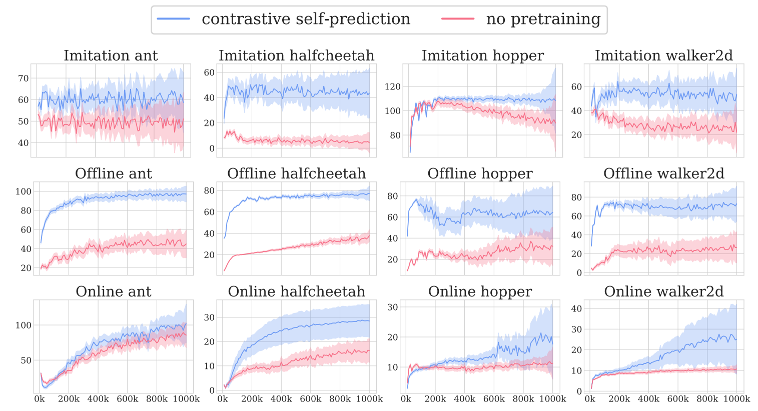

Figure 1: A summary of the advantages of representation learning via contrastive self-prediction, across a variety of settings: imitation learning, offline RL, and online RL. Each subplot shows the aggregated mean reward and standard error during training, with aggregation over offline datasets of different behavior (e.g., expert, medium, etc.), with five seeds per dataset (see Section 3). Representation learning yields significant performance gains in all domains and tasks.

settings. Inspired by these promising results and to differentiate our own work, we focus our efforts on the question of representation learning for *observations*, or states, as opposed to learning representations of behaviors or actions. That is, we aim to answer the question, can offline experience datasets be used to learn representations *of state observations* such that learning policies from these pretrained representations, as opposed to the raw state observations, improves performance on a downstream task?[2]

To approach this question, we devise a variety of offline datasets and corresponding downstream tasks. For offline datasets, we leverage the Gym-MuJoCo datasets from D4RL (Fu et al., 2020), which provide a diverse set of datasets from continuous control simulated robotic environments. For downstream tasks, we consider three main categories: (1) low-data imitation learning, in which we aim to learn a task-solving policy from a small number of expert trajectories; (2) offline RL, in which we aim to learn a task-solving policy from the same offline dataset used for representation learning; and (3) online RL, in which we aim to learn a task-solving policy using online access to the environment.

Once these settings are established, we then continue to evaluate the ability of state representation learning on the offline dataset to accelerate learning on the downstream task. Our experiments are separated into two parts, *breadth* and *depth*. First for breadth, we consider a diverse variety of representation learning objectives taken from the RL and supervised learning literature. The results of these experiments show that, while several of these objectives perform poorly, a few yield promising results. This promising set essentially comprises of objectives which we call *contrastive self-prediction*; these objectives take sub-trajectories of experience and then use some components of the sub-trajectory to predict other components, with a contrastive loss when predicting states – e.g., using a contrastive loss on the affinity between a sequence of states and actions and the next state, akin to popular methods in the supervised learning literature (Mikolov et al., 2013; Devlin et al., 2018).

These initial findings guide our second set of experiments. Aiming for depth, we devise an extensive ablation based on contrastive self-prediction to investigate what components of the objective are most important and in which settings. For example, whether it is important to include reward as part of the sub-trajectory, or whether discrete representations are better than continuous, whether pre-training and fixing the representations is better than finetuning, etc. In short, we find that state representation learning can yield a dramatic improvement in downstream learning. Compared to performing policy learning from raw observations, we show that relatively simple representation learning objectives on offline datasets can enable better and faster learning on imitation learning, offline RL, and online RL (see Figure 1). We believe these results are especially compelling for

[2]Whether the two aspects of representation learning – action representations and state representations – can be combined is an intriguing question. However, to avoid an overly broad paper, we focus only on state representation learning, and leave the question of combining this with action representation learning to future work.

the imitation learning setting – where even a pretraining dataset that is far from expert behavior yields dramatic improvement in downstream learning – and in the offline RL setting – where we show the benefits of representation learning are significant even when the pretraining dataset *is the same as* the downstream task dataset. We hope that these impressive results guide and encourage future researchers to develop even better ways to incorporate representation learning into sequential decision-making.

2 Background and Related Work

Representation learning for RL has a rich and diverse existing literature, and we briefly review these relevant works.

Abstraction and Bisimulation Traditionally, representation learning has been framed as learning or identifying *abstractions* of the state or action space of an environment (Andre & Russell, 2002; Mannor et al., 2004; Dearden & Boutilier, 1997; Abel et al., 2018). These methods aim to reduce the original environment state and action spaces to more compact spaces by clustering those states and actions which yield similar rewards and dynamics. Motivated by similar intuitions, research into *bisimulation* metrics has aimed to devise or learn similarity functions between states (Ferns et al., 2004; Castro & Precup, 2010). While these methods originally required explicit knowledge of the reward and dynamics functions of the environment, a number of recent works have translated these ideas to stochastic representation learning objectives using deep neural networks (Gelada et al., 2019; Zhang et al., 2020; Agarwal et al., 2021). Many of these modern approaches effectively learn reward and transition functions in the learned embedding space, and training of these models is used to inform the learned state representations.

Representations in Model-Based Learning The idea of learning latent state representations via learning reward and dynamics models leads us to related work in the model-based RL literature. Several recent model-based RL methods use latent state representations as a way to simplify the model learning and policy rollout elements of model-based policy optimization (Oh et al., 2017; Silver et al., 2018; Hafner et al., 2020), with the rollout in latent space sometimes referred to as 'imagination' (Racanière et al., 2017; Hafner et al., 2019). Similar ideas have also appeared under the label of 'embed to control' (Watter et al., 2015; Levine et al., 2019). Other than learning representations through forward models, there are also works which propose to learn *inverse* models, in which an action is predicted based on the representations of its preceding state and subsequent state (Pathak et al., 2017; Shelhamer et al., 2016).

Contrastive Objectives Beyond model-based representations, many previous works propose the use of contrastive losses as a way of learning useful state representations (Wu et al., 2018; Nachum et al., 2018; Srinivas et al., 2020; Stooke et al., 2020). These works effectively define some notion of similarity between states and use a contrastive loss to encourage similar states to have similar representations. The similarity is usually based on either temporal vicinity (pairs of states which appear in the same sub-trajectory) or user-specified augmentations, such as random shifts of image observations (Srinivas et al., 2020). Previous work has established connections between the use of contrastive loss and mutual information maximization (van den Oord et al., 2019) and energy-based models (LeCun & Huang, 2005).

State Representation Learning in Offline RL The existing works mentioned above almost exclusively focus on online settings, often learning the representations on a continuously evolving dataset and in tandem with online policy learning. In contrast, our work focuses on representation learning on offline datasets and separated from downstream task learning. This serves two purposes: First, using static offline datasets makes comparisons between different methods easier, avoiding confounding factors associated with issues of exploration or nonstationary datasets. Second, the offline setting is arguably more practical; in practice, static offline datasets are more common than cheap online access to an environment (Levine et al., 2020). Previous work in a similar vein to ours includes Stooke et al. (2020) and Shelhamer et al. (2016), which propose to use unsupervised pretraining, typically only on expert demonstrations, as a way of initializing an image encoder for downstream online RL. Our own work complements these existing studies, by presenting extensive comparisons of a variety of representation learning objectives in several distinct settings. Moreover, our work is unique for showing benefits of representation learning on non-image tasks, thus avoiding the use of any explicit or implicit prior knowledge that is typically exploited for images (e.g., using image-based augmentations or using a convolutional network architecture).

3 Task Setups

We now continue to our own contributions, starting by elaborating on the experimental protocol we design to evaluate representation learning in the context of low-data imitation learning, offline RL (specifically, offline policy optimization), and online RL in partially observable environments. This protocol is summarized in Table 1.

Table 1: A summary of our experimental setups. In total, there are 16 choices of offline data and downstream task combinations each for imitation learning, offline RL, and online RL. Given that we run each setting with five random seeds, this leads to a total of 240 training runs for every representation learning objective we consider.

		Imitation	
Choose domain $\in$ {halfcheetah, hopper, walker2d, ant}		Offline dataset:	{domain}-{data}-v0
Choose data $\in$ {medium, medium-replay}	$\rightarrow$	Downstream task:	Behavioral cloning (BC) on first N
Choose $N \in \{10000, 25000\}$			transitions from {domain}-expert-v0
		Offline RL	
Choose domain $\in$ {halfcheetah, hopper, walker2d, ant}		Offline dataset:	{domain}-{data}-v0
Choose data $\in$ {expert, medium-expert, medium,	$\rightarrow$	Downstream task:	Behavior regularized actor critic (BRAC)
medium-replay}			on data from {domain}-{data}-v0
		Online RL	
Choose domain $\in$ {halfcheetah, hopper, walker2d, ant}		Offline dataset:	{domain}-{data}-v0 with random masking
Choose data $\in$ {expert, medium-expert, medium}	$\rightarrow$	Downstream task:	Soft actor critic (SAC) on randomly
medium-replay}			masked version of {domain}

3.1 Datasets

We leverage the Gym-MuJoCo datasets from D4RL (Fu et al., 2020). These datasets are generated from running policies on the well-known MuJoCo benchmarks of simulated locomotive agents: halfcheetah, hopper, walker2d, and ant. Each of these four domains is associated with four datasets – expert, medium-expert, medium, and medium-replay – corresponding to the quality of the policies used to collect that data. Each dataset is composed of a number of trajectories $\tau := (s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_T)$. For example, the dataset ant-expert-v0 is a dataset of trajectories generated by expert task-solving policies on the ant domain, while the dataset halfcheetah-medium-v0 is generated by mediocre, far from task-solving, policies.

Notably, although the underyling MuJoCo environments are Markovian, the datasets are not necessarily Markovian, as they may be generated by multiple distinct policies.

3.2 Imitation Learning in Low-Data Regime

Imitation learning (Hussein et al., 2017) seeks to match the behavior of an agent with that of an expert. While expert demonstrations are often limited and expensive to obtain in practice, non-expert experience data (e.g., generated from a mediocre agent randomly interacting with an environment) can be much more easily accessible.

To mimic this practical scenario, we consider an experimental protocol in which the downstream task is behavioral cloning (Pomerleau, 1991) on a small set of expert trajectories – selected by taking either the first 10k or 25k transitions from an expert dataset in D4RL, corresponding to about 10 and 25 expert trajectories, respectively. We then consider either the medium or medium-replay datasets from the same domain for representation learning.[3] Thus, this set of experiments aims to determine whether representations learned from large datasets of mediocre behavior can help elevate the performance of behavioral cloning on a much smaller expert dataset.

3.3 Offline RL with Behavior Regularization

One of the main motivations for the introduction of the D4RL datasets was to encourage research into fully offline reinforcement learning; i.e., whether it is possible to learn return-maximizing policies exclusively from a static offline dataset. Many algorithms for this setting have recently been proposed, commonly employing some sort of *behavior regularization* (Kumar et al., 2019; Jaques et al., 2019; Wu et al., 2019). In its simplest form, behavior regularization augments a vanilla actor-critic algorithm with a divergence penalty measuring the divergence of the learned policy from the offline data, thus compelling the learned policy to choose the same actions appearing in the dataset.

[3]To avoid issues of extrapolation when transferring learned representations to the expert dataset, we include the small number of expert demonstrations in the offline dataset during pretraining.

While the actor and critic are typically trained with the raw observations as input, with this next set of experiments, we aim to determine whether representation learning can help in this regime as well. In this setting, the pretraining and downstream datasets are the same, determined by a single choice of domain (halfcheetah, hopper, walker2d, or ant) and data (expert, medium-expert, medium, or medium-replay). For the downstream algorithm, we use behavior regularized actor-critic (BRAC) (Wu et al., 2019), which is a simple behavior regularized method employing a KL divergence penalty. Notably, although the original BRAC paper uses different regularization strengths and policy learning rates for different domains, we fix these to values which we found to generally perform best (regularization strength of 1.0 and policy learning rate of 0.00003).

Thus, this set of experiments aims to determine whether learning BRAC from learned state representations is better (in terms of performance and less dependence on hyperparameters) than learning BRAC from the raw states, even when the state representations are learned using the same offline dataset.

3.4 Online RL in Partially Observable Environments

In this set of experiments, we aim to determine whether representations learned from offline datasets can improve or accelerate learning in an online domain. One of the most popular online RL algorithms is soft actor critic (SAC) (Haarnoja et al., 2019). SAC is a well-performing algorithm on its own, and so to increase the difficulty of the downstream task, we consider a simple modification to make our domains partially observable: zero-masking out a random dimension of the state observation. This modification also brings our domains closer to practice, where partial observability due to flaky sensor readings is common (Dulac-Arnold et al., 2019).

Accordingly, the offline dataset is determined by a choice of domain (halfcheetah, hopper, walker2d, or ant) and data (expert, medium-expert, medium, or medium-replay), with the same masking applied to this dataset. Representations learned on this dataset are then applied downstream, where SAC is trained on the online domain, with the representation module providing an embedding of the masked observations of the environment within a learned embedding space.

3.5 Evaluation

Each representation learning variant we evaluate is run with five seeds on each of the experimental setups described above. Unless otherwise noted, a single seed corresponds to an initial pretraining phase of 200k steps, in which a representation learning objective is optimized using batches of 256 sub-trajectories randomly sampled from the offline dataset. After pretraining, the learned representation is fixed and applied to the downstream task, which performs the appropriate training (BC, BRAC, or SAC) for 1M steps. In this downstream phase, every 10k steps, we evaluate the learned policy on the downstream domain environment by running it for 10 episodes and computing the average total return. We normalize this total return according to the normalization proposed in Fu et al. (2020), such that a score of 0 roughly corresponds to a random agent and a score of 100 to an expert agent. We average the last 10 evaluations within the 1M downstream training, and this determines the final score for the run. To aggregate over multiple seeds and task setups, we simply compute the average and standard error of this final score.

4 Experiments: Breadth Study

We begin our empirical study with an initial assessment into the performance of a broad set of representation learning ideas from the existing literature.

4.1 Representation Learning Objectives

We describe the algorithms we consider below. While it is infeasible for us to extensively evaluate all previously proposed representation learning objectives, our choice of objectives here aims to cover a diverse set of recurring themes and ideas from previous work (see Section 2).

We use the notation

$$\tau_{t:t+k} := (s_t, a_t, r_t, \dots, s_{t+k-1}, a_{t+k-1}, r_{t+k-1}, s_{t+k})$$

to denote a length-$(k+1)$ sub-trajectory of state observations, actions, and rewards; we use $s_{t:t+k}, a_{t:t+k}, r_{t:t+k}$ to denote a subselection of this trajectory based on states, actions, and rewards, respectively. We use ϕ to denote the representation function; i.e., $\phi(s)$ is the representation associated with state observation s, and $\phi(s_{t:t+k}) := (\phi(s_t), \dots, \phi(s_{t+k}))$. All learned functions, including ϕ, are parameterized by neural networks. Unless otherwise noted, ϕ is parameterized as a two-hidden layer fully-connected network with 256 units per layer and output of dimension 256 (see further details in Appendix A).

Inverse model Given a sub-trajectory $\tau_{t:t+1}$, use $\phi(s_{t:t+1})$ to predict a_t. That is, we train an auxiliary f such that $f(\phi(s_{t:t+1}))$ is a distribution over actions, and the learning objective is $-\log P(a_t|f(\phi(s_{t:t+1})))$. This objective may be generalized to sequences longer than $k+1=2$ as $-\log P(a_{t+k-1}|f(\phi(s_{t:t+k}), a_{t:t+k-1}))$.

Forward raw model Given a sub-trajectory $\tau_{t:t+1}$, use $\phi(s_t), a_t$ to predict r_t, s_{t+1}. That is, we train an auxiliary f, g such that $f(\phi(s_t), a_t)$ is a distribution over next states and $g(\phi(s_t), a_t)$ is a scalar reward prediction. The learning objective is $||r_t - g(\phi(s_t), a_t)||^2 - \log P(s_{t+1}|f(\phi(s_t), a_t))$. This objective may be generalized to sequences longer than $k+1=2$ as $||r_t - g(\phi(s_{t:t+k-1}), a_{t:t+k-1})||^2 - \log P(s_{t+1}|f(\phi(s_{t:t+k-1}), a_{t:t+k}))$.

Forward latent model; a.k.a., DeepMDP (Gelada et al., 2019) This is the same as the forward raw model, only that f now describes a distribution over next state representations. Thus, the log-probability with respect to f becomes $-\log P(\phi(s_{t+1})|f(\phi(s_t), a_t))$.

Forward energy model This is the same as the forward raw model, only that f is no longer a distribution over raw states. Rather, f maps $\phi(s_t), a_t$ to the same embedding space as ϕ and the probability $P(s_{t+1}|f(\phi(s_t), a_t))$ is defined in an energy-based way:

$$\frac{\rho(s_{t+1})\exp\{\phi(s_{t+1})^\top W f(\phi(s_t), a_t)\}}{\mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W f(\phi(s_t), a_t)\}]}, \tag{1}$$

where W is a trainable matrix and ρ is a non-trainable prior distribution (we set ρ to be the distribution of states in the offline dataset).

(Momentum) temporal contrastive learning (TCL) Given a sub-trajectory $\tau_{t:t+1}$, we apply a contrastive loss between $\phi(s_t), \phi(s_{t+1})$. The objective is

$$-\phi(s_{t+1})^\top W \phi(s_t) + \log \mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W \phi(s_t)\}], \tag{2}$$

where W and ρ are as in the forward energy model above. This objective may be generalized to sequences longer than $k+1=2$ by having multiple terms in the loss for $i=1,\ldots,k$:

$$-\phi(s_{t+i})^\top W_i \phi(s_t) + \log \mathbb{E}_\rho[\exp\{\phi(\tilde{s})^\top W_i \phi(s_t)\}]. \tag{3}$$

If momentum is used, we apply the contrastive loss between $f(\phi(s_t))$ and $\phi_{target}(s_{t+i})$, where f is a learned function and ϕ_{target} denotes a non-trainable version of ϕ, with weights corresponding to a slowly moving average of the weights of ϕ, as in Stooke et al. (2020); He et al. (2020).

Attentive Contrastive Learning (ACL) Following the theme of contrastive losses and inspired by a number of works in the RL (van den Oord et al., 2019) and NLP (Mikolov et al., 2013) literature which apply such losses between tokens and contexts using an attention mechanism, we devise a similar objective for our settings. Implementation-wise, we borrow ideas from BERT (Devlin et al., 2018), namely we (1) take a sub-trajectory $s_{t:t+k}, a_{t:t+k}, r_{t:t+k}$, (2) randomly mask a subset of these, (3) pass the masked sequence into a transformer, and then (4) for each masked input state, apply a contrastive loss between its representation $\phi(s)$ and the transformer output at its sequential position. We use $k+1=8$ in our implementation. Figure 3 provides a diagram of ACL.

Value prediction network (VPN) Taken from Oh et al. (2017), this objective uses an RNN starting at $\phi(s_t)$ and inputting $a_{t:t+k}$ for k steps to predict the k-step future rewards and value functions. While the original VPN paper defines the $(k+1)$-th value function in terms of a max over actions, we avoid this potential extrapolation issue and simply use the $(k+1)$-th action provided in the offline data. As we will elaborate on later, VPN bears similarities to ACL in that it uses certain components of the input sequence (states and actions) to predict other components (values).

Deep bisimulation for control This objective is taken from Zhang et al. (2020), where the representation function ϕ is learned to respect an L1 distance based on a bisimulation similarity deduced from Bellman backups.

4.2 RESULTS

The results of these representation learning objectives are presented in Figure 2. Representation learning, even before the extensive ablations we will embark on in Section 5, on average improves downstream imitation learning, offline RL, and online RL tasks by 1.5x, 2.5x, and 15% respectively. The objectives that appear to work best – ACL, (Momentum) TCL, VPN – fall under a class of objectives we term *contrastive self-prediction*, where *self-prediction* refers to the idea that certain components of a sub-trajectory are predicted based on other components of the same sub-trajectory,

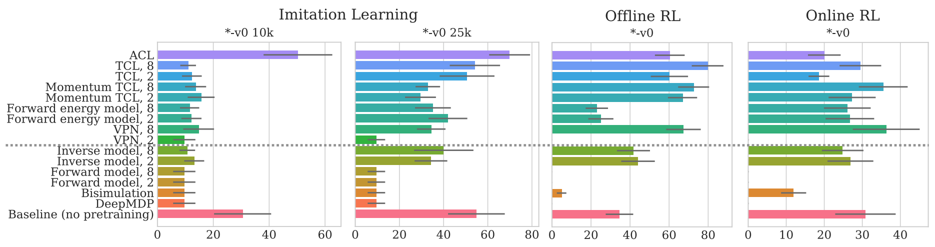

Figure 2: Performance of downstream imitation learning, offline RL, and online RL tasks under a variety of representation learning objectives. x-axis shows aggregated average rewards (over five seeds) across the domains and datasets described in Section 3. Methods that failed to converge are eliminated from the results (see Appendix A). ACL is set to the default configuration that favors imitation learning (see Section 5). When applicable, we also label variants with $k + 1 \in \{2, 8\}$. Methods above the dotted line are variants of contrastive self-prediction. ACL performs well on imitation learning. VPN and (momentum) TCL perform well on offline and online RL.

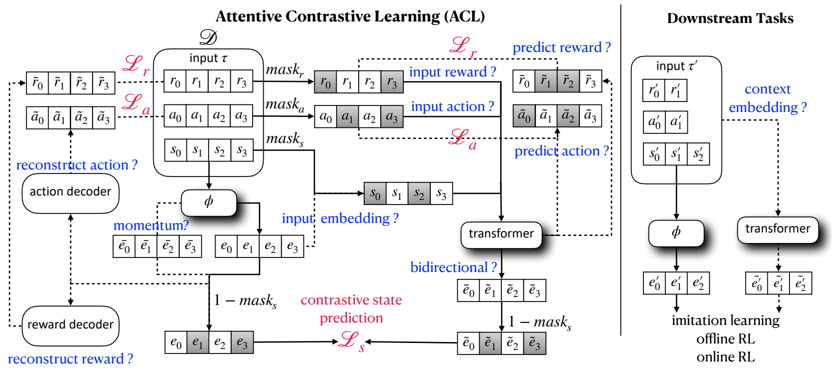

Figure 3: A pictoral representation of our depth study based on contrastive self-prediction. We use the transformer-based architecture of attentive contrastive learning (ACL) as a skeleton for ablations with respect to various representation learning details. Solid arrows correspond to the configuration of ACL. Dotted arrows and blue text are factors considered in the ablation study. Gray blocks are masked state/action/reward entries. After the pretraining phase, the representation network ϕ is reused for downstream tasks, unless 'context embedding' is true, in which case the transformer is used.

while *contrastive* refers to the fact that this prediction should be performed via a contrastive energy-based loss when the predicted component is a state observation.

We also find that a longer sub-trajectory $k + 1 = 8$ is generally better than a short one $k + 1 = 2$. The advantage here is presumably due to the non-Markovian nature of the dataset. Even if the environment is Markovian, the use of potentially distinct policies for data collection can lead to non-Markovian data.

Despite these promising successes, there are a number of objectives which perform poorly. Raw predictions of states (forward model) yields disappointing results in these settings. Forward models of future representations – DeepMDP, Bisimulation – also exhibit poor performance. This latter finding was initially surprising to us, as many theoretical notions of state abstractions are based on the principle of predictability of future state representations. Nevertheless, even after extensive tuning of these objectives and attempts at similar objectives (e.g., we briefly investigated incorporating ideas from Hafner et al. (2020)), we were not able to achieve any better results. Even if it is possible to find better architectures or hyperparameters, we believe the difficulty in tuning these baselines makes them unattractive in comparison to the simpler and better performing alternatives.

5 Experiments: Depth Study

The favorable results of objectives based on the idea of contrastive self-prediction is compelling, but the small number of objectives evaluated leaves many questions unanswered. For example, when generating the context embedding for a specific prediction, should one use past states (as in TCL and Momentum TCL) or also include actions and/or rewards (as in ACL and VPN)? Should this

Table 2: Factors of contrastive self-prediction considered in our ablation study and summaries of their effects. Input action and input reward default to true. The remaining factors default to false. For each effect entry, $\downarrow$ means decreased performance, $\uparrow$ means improved performance, and $=$ means no significant effect.

Factor	**Description**	**Imitation**	**Offline**	**Online**
reconstruct action	Add action prediction loss based on $\phi(s)$.	$\downarrow$	$\uparrow$	$\uparrow$
reconstruct reward	Add a reward prediction loss based on $\phi(s)$.	$\downarrow$	$\uparrow$	$\uparrow$
predict action	Add an action prediction loss based on transformer outputs. Whenever this is true, we also set 'input embed' to true.	$\downarrow$	$\uparrow$	$\uparrow$
predict reward	Add a reward prediction loss based on transformer outputs. Whenever this is true, we also set 'input embed' to true.	$\downarrow$	$\uparrow$	$\uparrow$
input action	Include actions in the input sequence to transformer.	$\downarrow$	$\uparrow$	$\uparrow$
input reward	Include rewards in the input sequence to transformer.	$\downarrow$	$\uparrow$	$\uparrow$
input embed	Use representations $\phi(s)$ as input to transformer, as opposed to raw observations.	$\downarrow$	$=$	$\uparrow$
bidirectional	To generate sequence output at position i, use full input sequence as opposed to only inputs at position $\leq i$.	$\downarrow$	$=$	$\uparrow$
finetune	Pass gradients into ϕ during learning on downstream tasks.	$\downarrow$	$\downarrow$	$\uparrow$
auxiliary loss	Use representation learning objective as an auxiliary loss during downstream learning, as opposed to pretraining.	$\downarrow$	$\downarrow$	$\uparrow$
momentum	Adopt an additional momentum representation network. Whenever this is true, we also set 'input embed' to true.	$\downarrow$	$\downarrow$	$\uparrow$
discrete embedding	Learn discrete representations. Following Hafner et al. (2020), we treat the 256-dim output of ϕ as logits to sample 16 categorical distributions of dimension 16 each and use straight-through gradients.	$\downarrow$	$\downarrow$	$\downarrow$
context embedding	Following Devlin et al. (2018), use transformer output as representations for downstream tasks. Whenever this is true, we also set 'input embed' to true.	$\downarrow$	$\downarrow$	$\downarrow$

context use the same representation network ϕ (as in TCL and VPN), a momentum version of it (as in Momentum TCL), or a completely separate network (as in ACL)?

We use this section to study these and other important questions by conducting a series of ablations on the factors which compose a specific contrastive self-prediction objective and how it is applied to downstream learning. We describe all these factors in Table 2, as well as a high-level summary of their effects. Further anecdotal observations found during our research are summarized in Appendix C.

We choose the transformer-based implementation of ACL to serve as the skeleton for all these ablations (see Figure 3), due to its general favorable empirical performance in the previous section, as well as its ease of modification. For each downstream task below, we present the ablations with respect to the *default* configuration of the factors in Table 2 that corresponds to the original ACL introduced in Section 4, and change one factor at a time to observe its effect on downstream task performance.

5.1 RESULTS

The results of our ablation studies are presented in Figure 4, and we highlight some of the main findings below. We also take the best performing ablation from each row (imitation, offline RL, and online RL) and plot the performance during training in Figure 1.

Let us first consider the effects of inclusion or prediction of actions and rewards. We notice some interesting behavior across the different downstream modes. Namely, it appears that imitation learning is best served by focusing only on state contrastive learning and not including or predicting actions and rewards, whereas the offline and online RL settings appear to benefit from these. Due to the mixed results we initially observed from including or predicting actions and rewards, we also introduce the idea of *reconstructing* actions and rewards based on $\phi(s)$, and we found this to have much more consistent benefit in the RL settings, although it still degrades imitation learning performance. This disconnect between objectives which are good for imitation learning vs. RL, first seen in Section 4, thus continues to be present in these ablations as well, and we find that no single objective dominates in all settings.

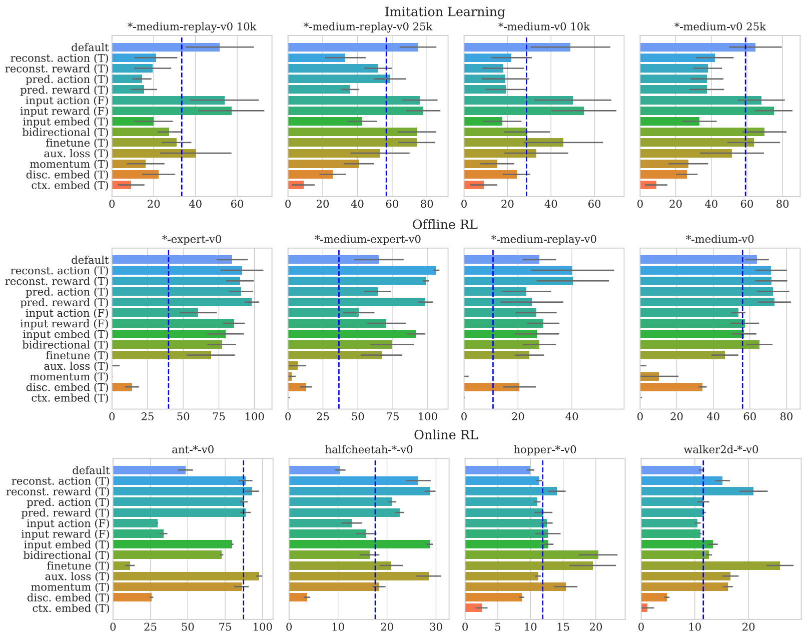

Figure 4: Ablation results on imitation learning, offline RL, and online RL. x-axis shows average rewards and standard error aggregated over either different Gym-MuJoCo datasets (imitation and offline RL) or domains (online RL). Blue dotted lines show average rewards without pretraining. (T) and (F) mean setting each factor to true or false (opposite from the default configuration). Reconstructing, predicting, or inputting action or reward (row 2-7) impairs imitation performance but are important for offline and online RL. Bidirectional transformer hurts imitation learning when downstream sample size is small. Finetuning and auxiliary loss can help online RL. Additional results are presented in Appendix B.

We also evaluate a number of representation learning paradigms popular in the NLP literature (Devlin et al., 2018), namely using bidirectional transformers, finetuning, and context embedding. Although these techniques are ubiquitous in the NLP literature, we find mixed results in RL settings. Context embedding consistently hurts performance. Bidirectional transformer hurts imitation learning but helps online RL. Finetuning leads to a modest degredation in performace in imitation and offline RL but can improve online RL depending on the domain being evaluated.

We additionally considered using the representation learning objective as an auxiliary training loss, which is popular in the online RL literature (Shelhamer et al., 2016; Stooke et al., 2020). And indeed, we find that it can dramatically improve representation learning in online RL, but at the same time, dramatically degrade performance in the offline settings (imitation learning or offline RL).

6 Conclusion

Overall, our results show that relatively simple representation learning objectives can dramatically improve downstream imitation learning, offline RL, and online RL (Figure 1). Interestingly, our results suggest that the ideal representation learning objective may depend on the nature of the downstream task, and no single objective appears to dominate generally. Our extensive ablations also provide a number of intriguing insights, showing that representational paradigms which are popular in NLP or online RL may not translate to good performance in offline settings.

Even with this multitude of fresh insight into the question of representation learning in RL, our study is limited in a number of aspects, and these aspects can serve as a starting point for future work. For example, one may consider additional downstream tasks such as multi-task, transfer, or exploration. Alternatively, one can extend our ablations to real-world domains like robot learning. Or, one may consider ablating over different network architectures.

Despite these limitations, we hope our current work proves useful to RL researchers, and serves as a guide for developing even better and more general representation learning objectives.

REFERENCES

David Abel, Dilip Arumugam, Lucas Lehnert, and Michael Littman. State abstractions for lifelong reinforcement learning. In *International Conference on Machine Learning*, pp. 10–19. PMLR, 2018.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning, 2020.

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.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020.

David Andre and Stuart J Russell. State abstraction for programmable reinforcement learning agents. In *Aaai/iaai*, pp. 119–125, 2002.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Pablo Castro and Doina Precup. Using bisimulation for policy transfer in mdps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Richard Dearden and Craig Boutilier. Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, 89(1-2):219–283, 1997.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019.

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

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

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

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*, pp. 2170–2179. PMLR, 2019.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

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*, pp. 9729–9738, 2020.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Aviral Kumar, Justin Fu, Matthew Soh, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, 2019.

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

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

Yann LeCun and Fu Jie Huang. Loss functions for discriminative training of energy-based models. In *AIStats*, volume 6, pp. 34. Citeseer, 2005.

Nir Levine, Yinlam Chow, Rui Shu, Ang Li, Mohammad Ghavamzadeh, and Hung Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. *arXiv preprint arXiv:1909.01506*, 2019.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 71, 2004.

Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. *arXiv preprint arXiv:2006.03647*, 2020.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Near-optimal representation learning for hierarchical reinforcement learning. *arXiv preprint arXiv:1810.01257*, 2018.

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

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning*, pp. 2778–2787. PMLR, 2017.

Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1):88–97, 1991.

Sébastien Racanière, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5694–5705, 2017.

Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017.

Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *CoRR*, abs/1612.07307, 2016. URL http://arxiv.org/abs/1612.07307.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020.

Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning, 2020.

Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. Decoupling representation learning from reinforcement learning, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. *arXiv preprint arXiv:1506.07365*, 2015.

Yifan Wu, George Tucker, and Ofir Nachum. The laplacian in rl: Learning representations with efficient approximations. *arXiv preprint arXiv:1810.04586*, 2018.

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

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Appendix

A Experimental Details

A.1 Representation Network

We parametrize the representation function ϕ as a two-hidden layer fully connected neural network with 256 units per layer and output dimension 256. A Swish (Ramachandran et al., 2017) activation function is applied to the output of each hidden layer. We experimented with representation dimension sizes $16, 64, 256$, and 512, and found 256 and 512 to generally work the best (see Figure 7 in Appendix B.3).

A.2 Transformer Network

The BERT-style transformer used in attentive contrastive learning (ACL) consists of one preprocessing layer of 256 units and ReLU activaiton, followed by a multi-headed attention layer (4 heads with 128 units each), followed by a fully connected feed forward layer with hidden dimension 256 and ReLU activation, finally followed by an output layer of 256 units (the same as ϕ's output). We experimented with different number of attention blocks and number of heads in each block, but did not observe significant difference in performance.

When masking input (sequences of state, actions, or rewards), we randomly choose to 'drop' each item with probability 0.3, 'switch' with probability 0.15, and 'keep' with probability 0.15. 'Drop' refers to replacing the item with a trainable variable of the same dimension. 'Switch' refers to replacing the item with a randomly sampled item from the same batch. 'Keep' refers to leaving the item unchanged. These probability rates where chosen arbitrarily and not tuned.

A.3 Action Prediction and Reconstruction

Whenever a loss includes action prediction or reconstruction, we follow Haarnoja et al. (2019), and (1) utilize an output distribution given by a tanh-squashed Gaussian and (2) apply an additive adaptive entropy regularizer to the action prediction loss.

A.4 Other Networks

With few exceptions, all other functions f, g mentioned in Section 4 are two-hidden layer fully connected neural networks with 256 units per layer and using a Swish (Ramachandran et al., 2017) activation.

The only exception is Momentum TCL, where f is the same structure but using a residual connection on the output.

A.5 Training

During pretraining, we use the Adam optimizer with learning rate 0.0001, except for the TCL variants, for which we found 0.0003 to work better. For Momentum TCL, we use a moving average with rate 0.05.

A.6 Convergence Failures

Representations learned under objectives including forward-raw model, VPN (with $k + 1 = 2$), and DeepMDP consistently diverge and output NaNs on offline and online RL, and are therefore removed from the results in Figure 2. The bisimulation objective on offline and online RL fails to converge in some runs but occasionally succeeds, therefore the means of succeeded runs are computed and shown in Figure 2.

B Additional Experimental Results

B.1 Fully-Observable Online Environments

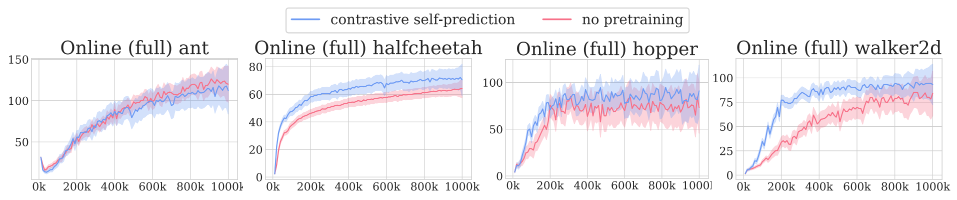

Figure 5: Average reward of best ACL ablation on fully-observable online RL compared to the baseline without pretraining.

B.2 Additional Contrastive Learning Results

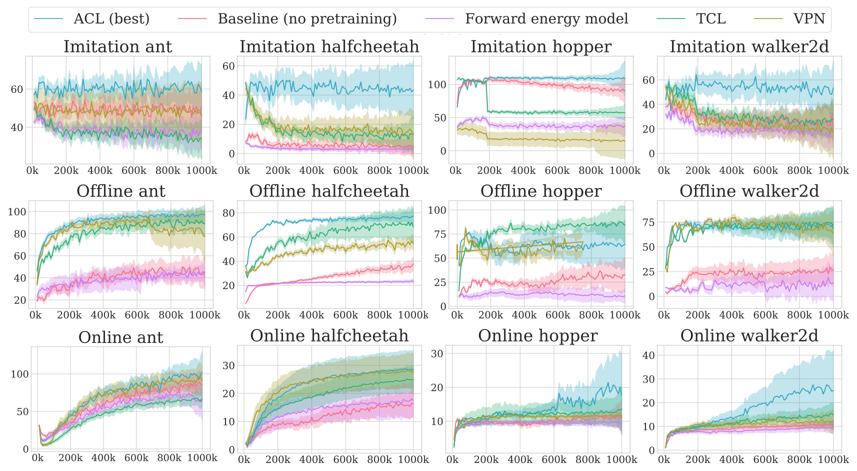

Figure 6: Additional training curves of contrastive learning objectives aggregated over different offline datasets in the same domain. Both in this figure and in Figure 1, we plot the best variant of ACL according to the ablation study, namely we set "input reward" to false in imitation learning, "reconstruct action" to true in offline RL, and "auxiliary loss" (in ant and halfcheetah) or "finetuning" (in hopper and walker2d) to true in online RL. The best variant of ACL generally performs the best compared to other contrastive learning objectives, although TCL's performance is competitive in offline RL.

B.3 Ablation over Representation Size

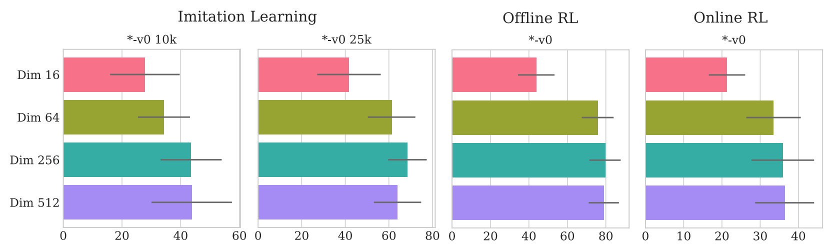

Figure 7: Average reward across domains and datasets with different representation dimensions. 256 and 512 work the best (this ablation is conducted with “reconstruct action” and “reconstruct reward” set to true).

B.4 Ablation over Pretraining Window Size

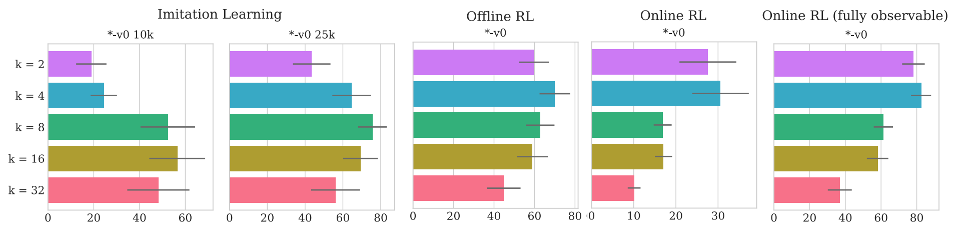

Figure 8: Average reward across domains and datasets with different pretraining window k in imitation learning, offline RL, and partially/fully observable online RL.

B.5 Ablation over Prediction Direction

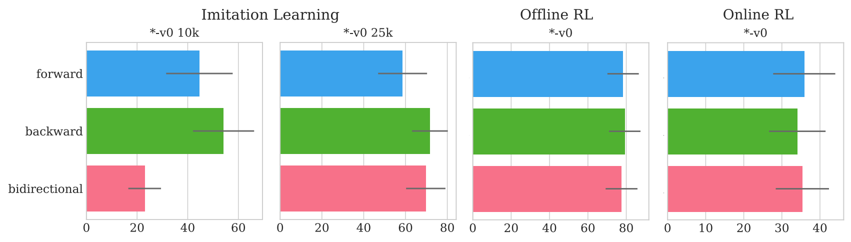

Figure 9: Average reward across domains and datasets with different prediction direction during embedding pretraining.

B.6 Ablation over Compounding Factors and on Sparse Reward

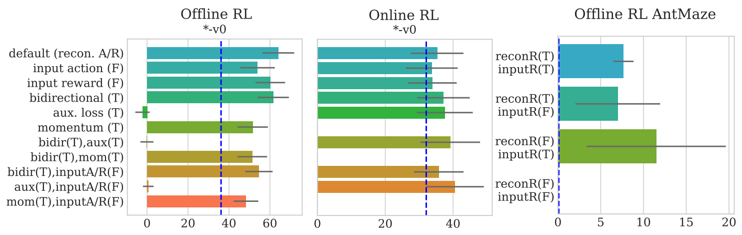

Figure 10: Left: Ablation (with compounding factors) with reconstructing action/reward as default. Right: reward ablation on antmaze-umaze with sparse reward.

B.7 Ablation Results for Individual Domains and Datasets

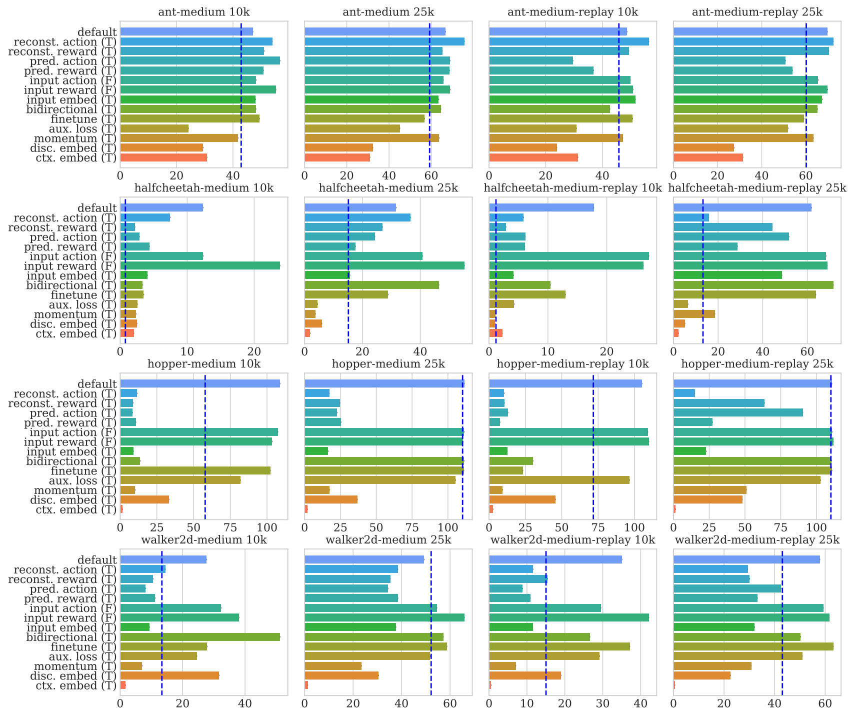

Figure 11: Imitation learning ablation on individual domains and datasets. The negative impact of inputting action and reward to pretraining is more evident in halfcheetah and walker2d. Reconstructing/predicting action/reward is especially harmful in halfcheetah, hopper, and walker2d. There always exists some variant of ACL that is better than without representation learning (blue lines) in all domain-dataset combinations.

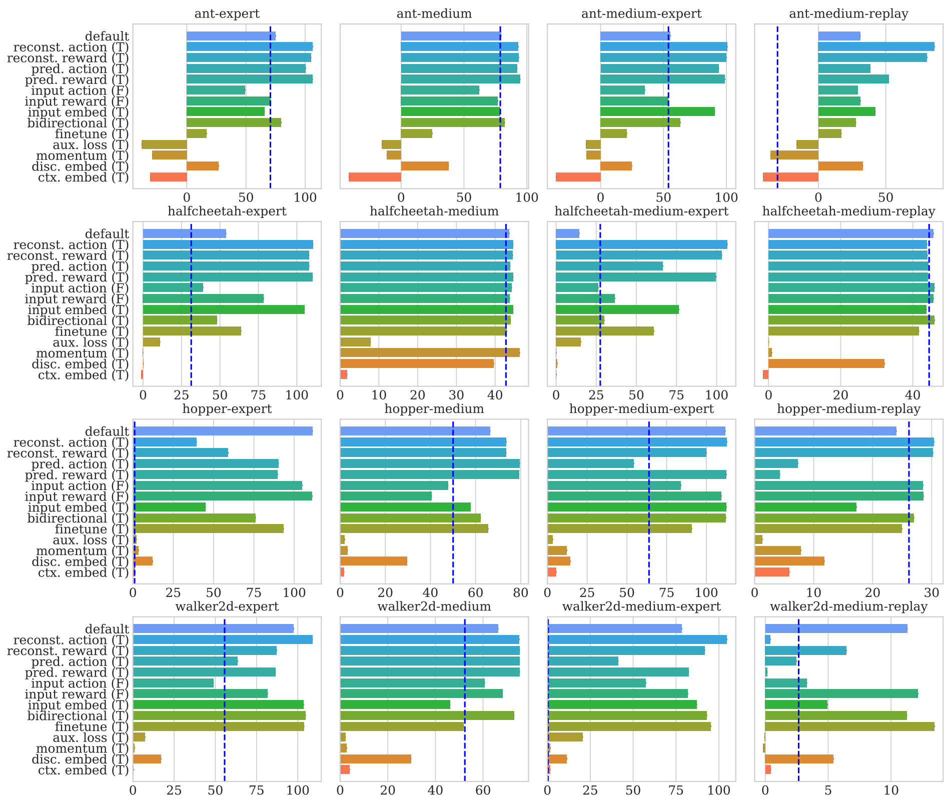

Figure 12: Offline RL ablation on individual domains and datasets. The benefit of representation learning is more evident when expert trajectories are present (e.g., expert and medium-expert) than when they are absent (medium and medum-replay). Reconstructing action and reward is more important in ant and halfcheetah than in hopper and walker2d.

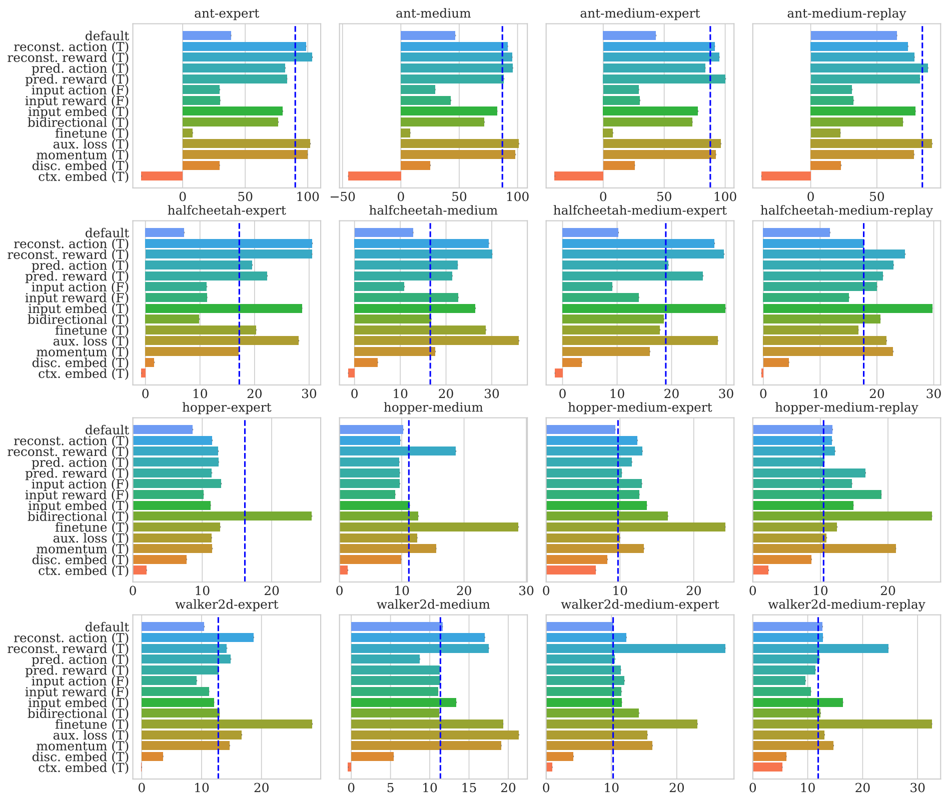

Figure 13: Online RL ablation on individual domains and datasets. Auxiliary loss generally improves performance in all domains and datasets. Finetuning improves halfcheetah, hopper, and walker2d but significantly impairs ant.

C Additional Anecdotal Conclusions

1. **More ablations.** Although we present our ablations as only changing one factor at a time, we also experimented with changing multiple factors at a time. We did not find any of these additional ablations to change the overall conclusions.
2. **Reconstruct action.** One ablation that did work surprisingly well was to only reconstruct the action (with no other loss). This appeared to perform poorly on imitation learning, but well on other settings.
3. **More transformers.** We experimented with a different application of transformers than ACL. Namely, we attempted to treat each dimension of the state as a token in a sequence (as opposed to using the whole state observation as the token). We found this to provide promising results, although it did not convincingly improve upon the configuration of ACL. Still, it may merit further investigation by future work.
4. **Transformer architecture.** We experimented with a different number of attention blocks or number of heads in each block, but did not observe significant differences in performance.
5. **Normalized or regularized representations.** We experimented with applying an explicit normalization layer on the output of ϕ and found no benefits. We also experimented with a stochastic representation along with a KL-divergence regularizer to the standard normal distribution, and again found no benefits.

D Additional Interpretations of Results

While we wanted to avoid making claims without exhaustive proof regarding why certain design choices are better than others, we believe many of our findings are interpretable, and here is a selection of our hypotheses:

1. Reward prediction helps in offline/online RL because rewards are critical to the downstream task.
2. Bisimulation-style approaches perform poorly because they are too eager to reduce the latent space (e.g., in the absence of reward, ϕ would be constant).
3. The poor performance of auxiliary training in offline RL may reflect the fact that offline RL is generally more liable to divergence in training, which would dominate gradients from any auxiliary objective and render the representation learning objective useless.
4. The poor performance of context embeddings in all setups may be explained as a consequence of an overly-rich representation – i.e., using context embedding means in the downstream task the same state may appear multiple times as different representations (since it is in a different context), and this can complicate learning in near-Markovian environments, unlike NLP.