Value Explicit Pretraining for Learning Transferable Representations

Kiran Lekkala, Henghui Bao, Sumedh A. Sontakke, Erdem Bıyık, Laurent Itti

Abstract— **Understanding visual inputs for a given task amidst varied changes is a key challenge posed by visual reinforcement learning agents. We propose *Value Explicit Pretraining* (VEP), a method that learns generalizable representations for transfer reinforcement learning. VEP enables efficient learning of new tasks that share similar objectives as previously learned tasks, by learning an encoder that trains representations to be invariant to changes in environment dynamics and appearance. To pretrain the encoder with *suboptimal unlabeled demonstration data* (sequence of observations and sparse reward signals), we use a self-supervised contrastive loss that enables the model to relate states across different tasks based on the Monte Carlo value estimate that is reflective of task progress, resulting in temporally smooth representations that capture the objective of the task. A major difference between our method and the existing approaches is the use of suboptimal unlabeled data that do not always solve the task. Experiments on Ant locomotion, a realistic navigation simulator and the Atari benchmark show that VEP outperforms current SoTA pretraining methods on the ability to generalize to unseen tasks. VEP achieves up to $2\times$ improvement in rewards, and up to $3\times$ improvement in sample efficiency. For videos of VEP policies, visit our website.**

I. Introduction

While performing everyday tasks, humans have an innate ability to appropriately extract information from what they perceive. This is often regardless of various changes related to the appearance and the dynamics of the tasks. This ability stems from understanding the objective of the tasks. While this ability is natural to humans, we need to equip robots with generalizable representations of their visual observations to achieve the same advantages.

Unfortunately, learning generalizable representations for control is still an open problem in visual sequential decision-making. Typically in such representation learning works, an encoder f_ϕ is learned using a large offline dataset via a predetermined objective function. Subsequently, f_ϕ is used for control by mapping high-dimensional visual observations from the environment $\mathbf{o}_{:t}$ into a lower-dimensional latent representation $\mathbf{z}_t$. The representation $\mathbf{z}_t$ is fed into a policy $\pi(\cdot \mid \mathbf{z}_t)$ to generate an action $\mathbf{a}_t$ to solve a task. The key question in visual representation learning is: *what should the learned f_ϕ be?*

The challenge in learning f_ϕ mainly lies in discovering the correct *inductive biases* that yield representations that can be used to learn a variety of downstream tasks in a sample efficient manner. It is unclear, however, what such useful inductive biases are. Initial approaches [1, 2, 3, 4] to this problem included simply reusing pretrained vision models trained to solve computer vision tasks like image recognition, zero-shot for control. Works like R3M [5] and VIP [6] tried to utilize temporal consistency, enforcing images that are temporally close in a video demonstration are embedded close to each other. Other works like Voltron [7] and masked visual pretraining [8, 9, 10, 11] attempt to use image reconstruction as one such inductive bias.

While biases induced by pretraining objectives like image reconstruction and temporal consistency have been shown to greatly improve downstream policy performance, these pretraining objectives used to learn f_ϕ are *distinct* from the downstream usage of f_ϕ, e.g., the task of image reconstruction is very different from that of action prediction. There exists an unmet need for representation learning approaches that *explicitly* encode information directly useful for downstream control during the process of learning f_ϕ.

This is, of course, challenging—how do we encode control-specific information without actually training online on a control task? Our crucial insight is that encoding control-specific information in the representations generated by f_ϕ is possible by harnessing the power of Monte Carlo estimates of control heuristics computed offline using the suboptimal, unlabeled demonstration datasets.

Our key contribution is *Value Explicit Pretraining (VEP)*, a contrastive learning approach that utilizes offline suboptimal, unlabeled demonstrations (without any action labels) to learn a representation for visual observations. Our method utilizes the insight that Monte Carlo value estimates across multiple tasks share a similar propensity of success, and, in tasks with related goals, also share a similar optimal policy. For example, in shooter games on Atari, despite differences in the visual appearances of adversaries, the strategy to effectively shoot them is similar. Our approach thus focuses on the similarity of progress towards the objective, as opposed to visual similarity.

VEP utilizes this intuition to learn an encoder using a contrastive loss which embeds observations with similar value function estimates across a set of training tasks near each other. We investigate the performance gains obtained by utilizing the VEP representation for policy learning, both on the training set of tasks and on visually distinct yet related held-out tasks. We experiment on the Atari benchmark and on a visual navigation benchmark comparing VEP to state-of-the-art methods like VIP [6] and SOM [12]. We find up to a $2\times$ improvement in the rewards obtained on both benchmarks and $3\times$ improvement in sample efficiency of online RL algorithms trained on VEP.

This work was supported by the National Eye Institute (grant R61EY037527), the National Science Foundation (award 2318101), and a research grant from Sandia National Laboratories. The authors affirm that the views expressed herein are solely their own, and do not represent the views of the United States government or any agency thereof.

All authors are with Thomas Lord Department of Computer Science at the University of Southern California. Correspondence: kiran.klekk@gmail.com.

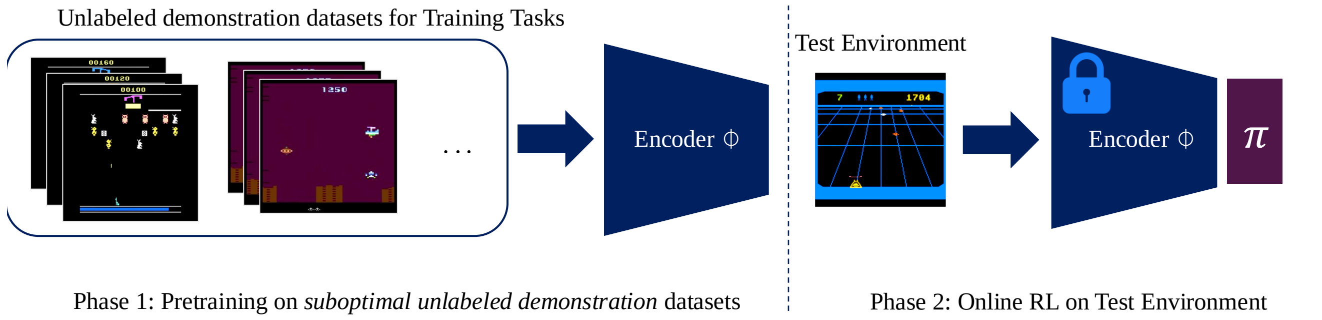
Unlabeled demonstration datasets for Training Tasks
Encoder Φ
Test Environment
Encoder Φ
π
Phase 1: Pretraining on *suboptimal unlabeled demonstration* datasets
Phase 2: Online RL on Test Environment

Fig. 1. **High-level overview of our setting.** The encoder f_ϕ is pretrained using suboptimal, unlabeled demonstration data from a set of train tasks, that is then reused for an unseen task. We evaluate pretrained encoders produced by our method and the baselines on the Atari and Navigation benchmarks.

II. Related Work

Representation Learning for Robotics. In addition to the general idea that representations have the role of encoding the essential information for a given task while discarding irrelevant aspects of the original data, typical state representation learning methods attempt to embed an observation into a latent representation that could be utilized by the downstream task [13]. It is also important that these methods produce a low dimensional representation that allows the control policy to *efficiently* learn the downstream task. Traditionally, unsupervised methods like variational autoencoders [14] learn disentangled representations that are used to correlate with underlying factors that cause variation in observation data [15] for policy learning [16]. However, in many environments, these representations prove difficult to learn an optimal policy since the temporal structure is missing in these representations. Anand et al. [17] explore this direction and learn representations by enforcing temporal structure through contrastive loss. However, they do not consider the generalization of the learned representations to unseen tasks.
National Eye

Pretraining for RL. Pretraining for representation learning, in the context of RL, involves learning transferable knowledge that helps the agent utilize its observations better [18]. Compared to traditional unsupervised methods for pretraining, the objective of self-supervised pretraining for RL is to learn representations by exploiting the underlying structure within the data distribution. Majority of the earlier *online pretraining* works learn representations that model the task dynamics that is learned through expert videos during the RL procedure [19]. More recent *offline pretraining* methods like [20] build on the prior work [17] by pretraining an encoder using unlabeled data and then finetune on a small amount of task-specific data. In comparison with these approaches, our method focuses on learning representations that not only aid in solving in-distribution tasks but also generalize to the out-of-distribution by learning representations that relate to general objectives and do not overfit to individual task-specific attributes.

Transfer after Pretraining. Transferring knowledge or skills learned from a given set of tasks to an unseen set of tasks is an active research area. Early works like progressive networks [21] attempt to solve it by reusing features learned from source tasks through adapters. Gamrian and Goldberg [22] perform image-to-image translation using GANs. However these methods are limited to predefined source or target domains. More recent works focus on the more challenging problem of using only expert videos for offline pretraining that could later be transferred to solve a novel downstream task. These methods have gained popularity in RL for their use of self-supervised pretraining based on *contrastive learning* [23]. Compared to these methods, our method only requires unlabeled data that consists of episodes that are not always successful in achieving the task objective.

Meta-learning for Transfer. Several meta-learning approaches in robotics and RL share the paradigm of pre-training on training tasks and evaluating on test tasks. Meta-reinforcement learning operates through a two-phase structure where agents are meta-trained on a distribution of related tasks to learn a meta-policy that captures shared patterns, then evaluated on unseen test tasks with minimal additional training [24, 25]. Information-theoretic task selection methods [26] have also been developed to optimize which training tasks are used during meta-training to improve performance on test tasks. However, all these methods are limited due to their requirement of having labeled/demonstration datasets. Although value function based pretraining was studied in [27, 28], they also require multi-task demonstration data to learn representations unlike our method that relies only on unlabeled non-expert data.

Baselines for VEP. *Value Implicit Pretraining (VIP)* [6] encodes the goal (positive) and the start (anchor) images close and the middle images (negatives) further away in the embedding space. By training on this objective through sampling multiple sub-episodes, the encoder recursively learns temporally smooth and continuous embeddings in a trajectory. *Time Contrastive Learning (TCN)* involves sampling the positive within a certain margin distance d_{thresh} from the anchor and a negative anywhere from the positive to the end of the trajectory [23]. If the anchor is sampled at time instant t_a, positive is sampled at t_p and the negative sampled at t_n, then $|t_n - t_a| > |t_p - t_a|$. It then uses the standard triplet loss for optimization, although other contrastive losses could also be used. Unlike TCN, Eysenbach et al. [12] sample the positives from *State Occupancy Measure (SOM)* that could be embedded close to the anchor. The negative, on the other hand, is sampled anywhere from the other episodes of the same task or from the other tasks. State occupancy measure at a specific instant t is a truncated geometric distribution

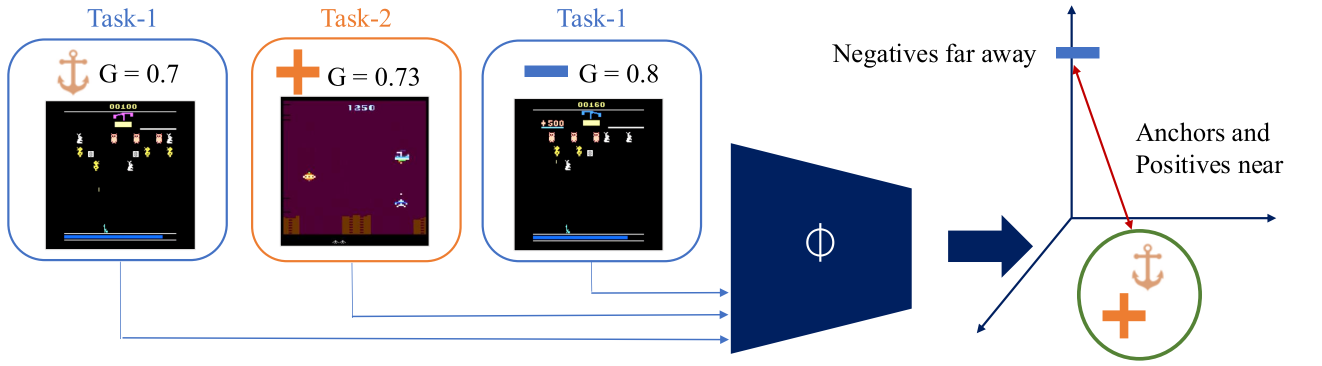

Fig. 2. **Description of our method (VEP).** We compute Monte Carlo value estimates, as denoted by G, for each frame. We then use a contrastive learning-based pretraining method that learns task-agnostic representations based on G. The figure is a pictorial representation of a training scenario where we only have two tasks resulting in the **negative** getting sampled from the same task as the **anchor**. This may not be the case if we have more training tasks.

$\text{Geo}_t^H(1-\gamma)$ with probability mass re-distributed over the interval $[t, H]$, where H is the horizon. [29].

III. Problem Setting and Preliminaries

Let $\mathcal{T}_{train} = \{\mathcal{T}_1, \mathcal{T}_2, ...\mathcal{T}_m\}$ be the set of training tasks with the associated suboptimal, unlabeled datasets that consist of a stream of images and sparse reward signals, denoted by $\mathcal{D}_{train}$. During pretraining, we assume that the encoder model f_ϕ parameterized by ϕ has access to $\mathcal{D}_{train}$. Data in $\mathcal{D}_{train}$ corresponding to $\mathcal{T}_i$ consists of a sequence of frames $\{o_t^i\}$ and sparse reward values $\{r_t^i\}$. The encoder f_ϕ learns to encode images/observations o_t into an embedding z_t, which is taken as an input by the policy π to perform a test-task. The set of test-tasks are denoted by $\mathcal{T}_{test} = \{\mathcal{T}_{m+1}, \mathcal{T}_{m+2}, ...\mathcal{T}_n\}$. Note that although $\mathcal{T}_{train} \cap \mathcal{T}_{test} = \varnothing$, all the tasks in $\mathcal{T}_{train} \cup \mathcal{T}_{test}$ share a semantically similar objective.

For evaluation, we selected tasks that have semantically similar objectives, in three settings or benchmarks: 1) In Atari games, we obtain several shooter games that all contain the “FIRE” action in the action space and whose objective semantically relates to “shoot up the enemy”. 2) For urban visual-based navigation, every task corresponds to navigating to the same goal destination with respect to the start location but in different cities; we use several cities and photographs taken along the available streets for the agent to navigate. 3) More challenging Ant locomotion benchmark [30] consists of a quadruped Ant controlling its limbs to reach a goal location. The agent has access to different observations pertaining to a specific maze environment.

For both our method and the baselines, the encoder f_ϕ is trained only using the unlabeled data $\mathcal{D}_{train}$ without any fine-tuning. The objective of our method is to efficiently learn the encoder using suboptimal, unlabeled data consisting of a sequence of observations and sparse reward signals, $\mathcal{D}_{train}$, from the source tasks $\mathcal{T}_{train}$, such that the embeddings from f_ϕ could be zero-shot transferred to unseen test tasks $\mathcal{T}_{test}$.

A. Contrastive Representation Learning

Typically, contrastive representation learning methods for RL utilize offline video demonstration datasets. These methods typically input a batch of anchors $\mathbf{o}_{an}$, positives $\mathbf{o}_{ps}$, and negatives $\mathbf{o}_{ng}$ and minimize a predetermined similarity metric that enables an encoder model to learn consistent and meaningful representations that can be used for downstream tasks. The earliest known formulation by Schroff, Kalenichenko, and Philbin [31] uses Euclidean distance to embed the positives and the anchor close to each other and the negatives far away from the anchor.

$$\mathcal{L}_{\text{triplet}} = \sum_{\mathbf{z}\in\mathcal{X}} \mathbf{max}\Big[\mathbf{0}, ||\mathbf{z}_{an} - \mathbf{z}_{ps}||_2^2 - ||\mathbf{z}_{an} - \mathbf{z}_{ng}||_2^2 + \epsilon\Big] \quad (1)$$

In the above equation $\mathbf{z}_{an}$, $\mathbf{z}_{ps}$ and $\mathbf{z}_{ng}$ represent the embeddings that are obtained after passing observations $\mathbf{o}_{an}$, $\mathbf{o}_{ps}$ and $\mathbf{o}_{ng}$ (anchors, positives and negatives) through the encoder network f_ϕ. Other metrics like cosine similarity could also be used instead of Euclidean distance, to compute the similarity between embeddings. This loss is used by VEP and all our baselines except for VIP.

Similar to recent methods like [6], the InfoNCE [32] objective can also be used to optimize the encoder parameters. Unlike the triplet loss from Eq. (1), InfoNCE permits utilizing *multiple negative examples* for calculating the loss (via the expectation term in the denominator of Eq. (2)). As depicted below, InfoNCE aims to maximize mutual information of the anchors and positives. This loss is used by VIP:

$$\mathcal{L}_{\text{InfoNCE}} = \mathbb{E}_{\mathbf{z}_{ps}}\left[-\log \frac{\mathcal{S}_\phi\left(\mathbf{z}_{an}, \mathbf{z}_{ps}\right)}{\mathbb{E}_{\mathbf{z}_{ng}}\mathcal{S}_\phi\left(\mathbf{z}_{an}, \mathbf{z}_{ng}\right)}\right] \quad (2)$$

where $\mathcal{S}_\phi$ computes the similarity between two embeddings in the ϕ-representation space. In our experiments that use InfoNCE, we use cosine similarity for $\mathcal{S}_\phi$.

B. Discounted Returns and Value Functions

We consider a POMDP (Partially Observable Markov Decision Process) denoted by the tuple ($\mathcal{O}$, $\mathcal{S}$, $\mathcal{A}$, p, θ, r, T, γ) representing an observation space $\mathcal{O}$, state space $\mathcal{S}$, action space $\mathcal{A}$, transition distribution p, emission function θ, reward function r, time horizon T, and discount factor γ. An agent in state $\mathbf{s}_t$ takes an action $\mathbf{a}_t$ and consequently causes a transition in the environment through $p(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$. The agent receives the next observation $\mathbf{o}_{t+1}$ and reward r_t that is calculated using the state $\mathbf{s}_t$ and action $\mathbf{a}_t$. The objective for the agent is to learn a policy π which maximizes the expected discounted sum of rewards. The discounted sum of

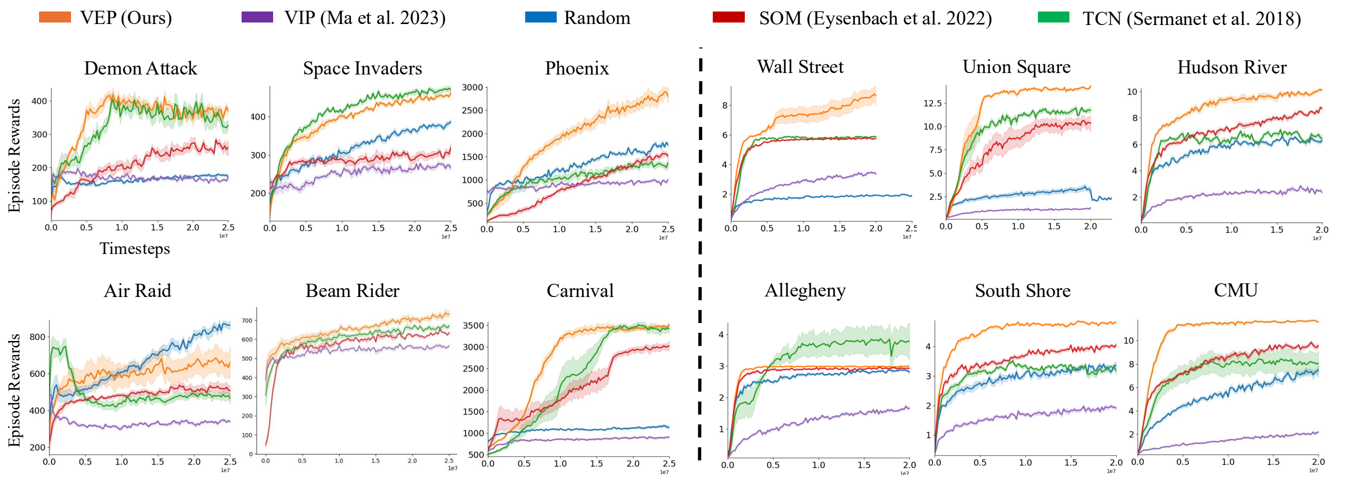

Fig. 3. (a). **Pretraining results on Atari (Left).** Performance of different pretraining methods on the respective games as mentioned above on the left. The encoder is pretrained only on the first two games (`Demon-Attack` and `Space-Invaders`) and is evaluated on the other out-of-distribution games. **Pretraining results on Navigation (Right).** Performance of different pretraining methods on the respective cities as mentioned above right. Similar to the Atari experiments, for all the baselines, suboptimal, unlabeled data from the first two tasks (*Wall Street* and *Union Square*) were used for pretraining. VEP representations improve PPO policy performance by up to 2×.

rewards at a state $\mathbf{s}_t$ in a trajectory τ is given by $\mathcal{G}$:

$$\mathcal{G}(\mathbf{s}_t, \tau) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=t}^{T} \gamma^{(k-t)} r_k \quad (3)$$

The expected return of state $\mathbf{s}_t$ under policy π is called the value of $\mathbf{s}_t$ and denoted by $\mathcal{V}^{\pi}(\mathbf{s}_t)$.

IV. Method

The true value $\mathcal{V}^{\pi}(\mathbf{s}_t)$ of a state $\mathbf{s}_t$ under a policy π intuitively defines the propensity for the success of solving a task by following policy π. If two states have similar value estimates, they likely have a similar expected return under π.

With this in mind, we now motivate VEP with an example. Consider the task of shooting an adversary in the Atari game of `Space-Invaders`. Assume that there exists an optimal policy for this task denoted by $\pi^*(\cdot \mid \mathbf{o}_t)$ which operates on image observations and the associated optimal value function $\mathcal{V}^{\pi^*}(\cdot)$. Now, consider a slightly perturbed version of this game in which all the adversaries are colored orange. If policy π^* must solve this perturbed task, it must be invariant to the color of the adversary. One way to achieve this invariance is to enforce that the value estimates of states with similar propensities for success are similar, e.g., the value estimate of a state containing a bullet very close to an adversary should be the same regardless of whether the adversary is yellow or orange. VEP utilizes this very intuition by learning representations that induce such an invariance. We assume access to unlabeled data from suboptimal agents doing the task (playing the game), consisting of only observations and reward values (obtained sparsely) for the set of training tasks $\mathcal{T}_{train}$: This kind of data can be obtained from various online sources of gameplay, and does not contain any action labels. Further, it is assumed to be generated by a suboptimal agent that contains at least a few positive reward signals during gameplay. These suboptimal, unlabeled datasets consist of demonstration data that are *not always guaranteed to succeed* in the task. If a sequence ends up with no reward at the terminal state, the last sub-sequence starting with the last non-zero reward and leading to the terminal state is rejected since all the frames in that sub-sequence end up with value estimates of 0. Note that since the task objectives allow for sparse rewards during the task, we would still be using parts of the episode (sequence), although the episode might not achieve the final task completion.

We also do not have access to the true reward function, but operate under a sparse reward setting, assuming that a reward of 1 at a few timesteps in the suboptimal, unlabeled data and 0 everywhere else. We now compute a value estimate to each observation using Eq. (3). Ideally, this value estimate would be computed using $\mathcal{V}^{\pi^*}(\cdot)$, but since we do not have access to the true value function of the optimal policy, we utilize a Monte Carlo estimate of this using Eq. (3). The computation of value estimates is fully algorithmic and requires no human effort.

Having obtained several suboptimal, unlabeled datasets, for tasks in $\mathcal{T}_{train}$, and computed value estimates at each frame with $\mathcal{G}(\cdot)$ from Eq. (3), we now train the encoder ϕ using a contrastive learning objective. This procedure first involves sampling a scalar value estimate g between 0 and 1 and then further sample multiple observations from $\mathcal{D}_{train}$ values within v_{thresh} of g. Subsequently, an encoder ϕ is learned which embeds these observations close to each other. Consequently, observations with a similar propensity for success have similar embeddings.

A. Implementation

To make the training computationally efficient, we preprocess $\mathcal{D}_{train}$ and save a dictionary that maps sorted Monte Carlo value estimates $\mathcal{G}(\cdot)$ to the indices of corresponding observations with the same Monte Carlo value estimate. This speeds-up the value lookup subroutines through binary search.

We first sample a batch of value estimates from the dataset determined by training batch size b_G. Next, we sample 3 training tasks ($\mathcal{T}_i$, $\mathcal{T}_j$ and $\mathcal{T}_k$) from $\mathcal{T}_{train}$ that we use to sample anchor, positive and negative respectively. Subsequently, the pretraining objective becomes:

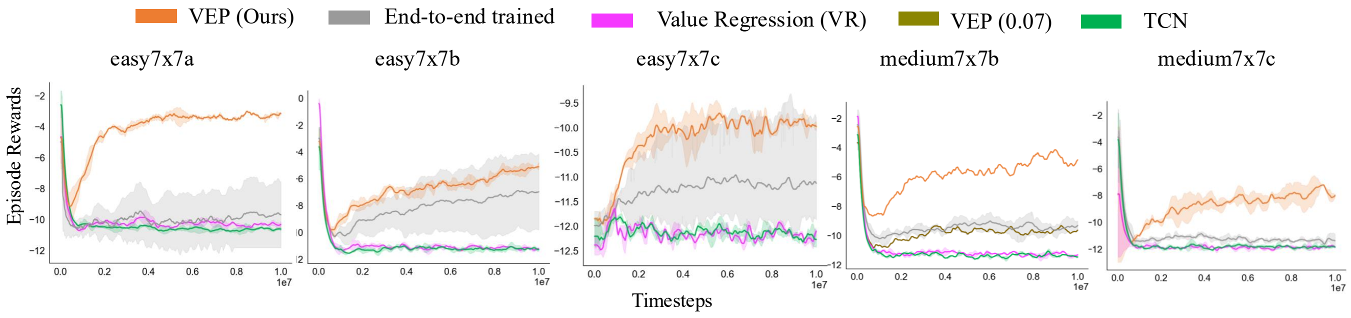

Fig. 4. **Pretraining results on Ant locomotion.** Performance of different pretraining methods on the respective mazes. The encoder is pretrained only on two mazes (`empty7x7` and `medium7x7a`) and is evaluated on the other out-of-distribution mazes that are shown above. The in-domain results are shown in Figure 6. We also performed ablation studies by varying the v_{thresh} parameter for VEP. Note that $v_{thresh} = 0.03$ for the optimal VEP baseline. We added another VEP baselines for `maze7x7b`, by varying the v_{thresh} parameter, that are shown in the legend above as VEP (v_{thresh}). As the value threshold increases, the performance of our method degrades.

$$\max_{\phi} \sum_{\mathcal{T}_i} \sum_{\mathcal{T}_j} \sum_{\mathcal{T}_k} \mathop{\mathbb{E}}_{g \sim \text{Unif}(\mathcal{G})} [\mathcal{S}_\phi(\mathbf{z}_{an}^{\mathcal{T}_i}, \mathbf{z}_{ps}^{\mathcal{T}_j}) - \mathcal{S}_\phi(\mathbf{z}_{an}^{\mathcal{T}_i}, \mathbf{z}_{ng}^{\mathcal{T}_k})]$$

where $\mathcal{G} \subset (0, 1]$ is the set of Monte Carlo value estimates of all observations in $\mathcal{D}_{train}$. As mentioned before, $\mathcal{S}_\phi$ computes the similarity between 2 embeddings that are obtained from an encoder parameterized by weights ϕ. $\mathbf{z}_{an}^{\mathcal{T}_i}$ corresponds to the embedding of the anchor, i.e., observation sampled from task $\mathcal{T}_i$ with a value estimate within an v_{thresh} of g, and $\mathbf{z}_{ps}^{\mathcal{T}_j}$ corresponds to a positive, that is sampled from task $\mathcal{T}_j$, and has a value estimate within an v_{thresh} of g and the negative $\mathbf{z}_{ng}^{\mathcal{T}_k}$ is sampled from task $\mathcal{T}_k$ that has a value estimate beyond v_{thresh} from g.

Intuitively, this objective encourages the positives and anchors from all the sampled tasks to embed near each other, using a value function estimate to organize the latent space of the learned encoder ϕ. For full implementation details like batch sizes etc., please refer to the website.

Algorithm 1 Value Explicit Pretraining

```
Require: $\mathcal{D}_{train}$, the set of suboptimal, unlabeled demonstrations collected
    from tasks $\{\mathcal{T}_i\}_{i=1}^{i=m}$
Require: Encoder $f_\phi$ parameterized by $\phi$
Require: $b_G$, as the batch size
Require: $v_{thresh}$ as the value thresholds
Require: $N$ as the number of iterations
 1: Randomly initialize $\phi$
 2: Compute value estimates $\mathcal{G}(.)$ for every frame $\mathbf{o_t}$ in $\mathcal{D}_{train}$
 3: Remove all frames in the $\mathcal{D}_{train}$ having value estimate of 0
 4: For every task $\mathcal{T}_i$, create a dictionary $\mathbf{V}^i$ mapping sorted value estimates
    as keys to list of frame indices in $\mathcal{D}_{train}$
 5: while iterations until $N$ do
 6:     Sample a $b_G$ sized batch of values $g \sim (0, 1]$
 7:     For each sampled value $g$, sample a frame $o_{an} \sim \mathcal{T}_i$ that has a value
        estimate of within $v_{thresh}$ from $g$
 8:     Sample a positive $\mathbf{o}_{ps} \sim \mathcal{T}_j$ from $\mathcal{D}_{train}$ within $v_{thresh}$
 9:     Find negatives $\mathbf{o}_{ng} \sim \mathcal{T}_k$ that are beyond $v_{thresh}$ from $\mathbf{o}_{an}$
10:     Estimate embeddings $\mathbf{z}_{an}$, $\mathbf{z}_{ps}$, $\mathbf{z}_{ng}$ for a batch of $\mathbf{o}_{an}$, $\mathbf{o}_{ps}$, $\mathbf{o}_{ng}$
        by propagating through $f_\phi$
11:     Compute contrastive loss using $\mathbf{z}_{an}$, $\mathbf{z}_{ps}$ and $\mathbf{z}_{ng}$
12:     Optimize $\phi$
13: end while
```

V. Experimental Setup

We study whether utilizing VEP as a pretraining objective to learn an encoder improves **(1)** policy learning on in-distribution tasks, i.e., those tasks for which data was available to pre-train the encoder and **(2)** whether the learned encoder aids transfer learning of new tasks. For additional details, please visit our website.

A. Environments

Atari. We used six Atari games with “FIRE” in their action set, which all are *Shoot’em up* games similar in spirit to Space Invaders. Although all the games share a common objective of shooting enemies that spawn from above, there are significant differences in appearances and dynamics across games. We then split these games into $\mathcal{T}_{\text{train}}$ and $\mathcal{T}_{\text{test}}$. For pretraining the encoder, we use suboptimal, unlabeled data, without action labels from the D4RL datasets [33]. The value estimates of each frame at timestep t in a sequence are then computed using (3) with T being the closest frame in the episode that obtains a reward.

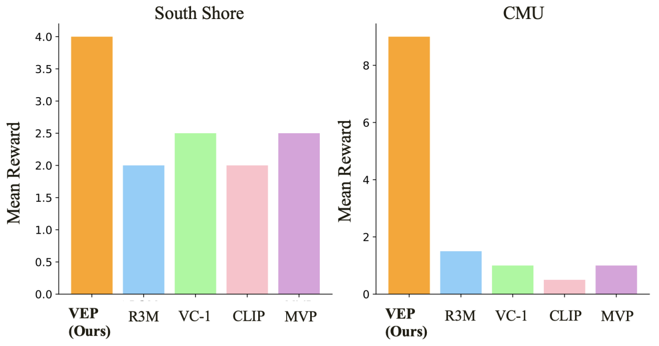

Fig. 5. **Comparison with other existing pretrained models.** We show the bar plot that compares VEP with other existing pretrained models using the mean cumulative reward of the policy on the out-of-distribution task.

Navigation. We build an engine that loads the StreetLearn dataset [34] to perform visual navigation, based on gym [35]. In a typical navigation task, the agent is designed to randomly respawn within a radius r of a predetermined location (src_x, src_y), with the objective reaching a goal location that is sampled within a radius r of a location $(d_x + src_x, d_y + src_y)$. *Sparse reward* acquisition is structured through a linear distribution of L reward points (including the reward obtained upon reaching the target) uniformly spanning the starting point to the goal. The agent only earns *sparse rewards* as it moves closer to the goal with each milestone radius from the goal destination. We have six cities for this benchmark, and we established consistent horizontal and vertical displacements (d_x, d_y) between the starting and target points across all cities, avoiding the need for any explicit goal information (details are provided in the supplementary material). The agent is then expected to transfer to an unseen test city after learning from the suboptimal, unlabeled data obtained from a set of cities. Note that each task corresponding to a specific city is non-trivial since the agent needs to navigate in an unknown city with a different map and appearance. Tasks across all the cities are all solvable within a predefined horizon. To ensure that the dataset had at least a few sparse rewards, the start and the end locations of the path was chosen to be between the actual source and destination locations.

We use the same encoder architecture for both Atari and Navigation to embed pixel observation into vector space. To enable to temporal understanding of the state, all the embeddings in the past four timesteps are concatenated together and passed onto the policy. For Navigation, apart from the image embeddings, we also obtain odometry information $(odom_x, odom_y)$, of the agent, that is concatenated with the image embedding and passed into a linear layer. This enables the agent to understand its ego-centric pose with respect to the source location, which is crucial for understanding the objective and navigating to the goal.

Ant locomotion in a maze. To assess the efficacy of VEP on a more challenging benchmark, we use recently proposed Mujoco Ant based Visual Maze environment [30]. This benchmark contains 7 different mazes and pictorially shown in Figure 4. `empty7x7`, `easy7x7` and `medium7x7` correspond to mazes having no obstacles, single obstacle and multiple obstacles respectively, and the alphabet at the end of each name corresponds to a specific maze. For pretraining, we used the suboptimal expert data from `empty7x7` and `medium7x7a`. The pretrain data for each maze consists of various visual variations of the maze, during evaluation there is a new variation that is not present in the offline pretrain dataset. This is important to note only for the in-domain evaluation (`empty7x7` and `medium7x7a`). For other out-of-domain mazes, we have both visual variations and structure variations pertaining to each maze. The pretrained data consists of the agent reaching a randomly sampled destination location from a randomly sampled source location. Unlike the data from the previous 2 benchmarks, there are no reward values associated with the pretrained data, and so we associated the reward 1 at the last frame it reaches the destination, otherwise 0 everywhere else.

We randomly sampled a source and destination position in each maze and made it fixed during the entire training process. The observation space is a 9 channel stacked image, i.e., RGB images corresponding to top, egocentric and following (visually captures ant joints) cameras. These are further stacked across three timesteps (t, $t-1$ and $t-2$). The agent obtains a 27 channel image and needs to output an 8 dimensional continuous control vector as an action. At every timestep, the agent obtains a reward equal to the change of distance to the goal. This is slightly different from the original reward function of providing the normalized negative distance to the goal at every timestep. The original reward function is primarily used as an evaluation metric for models trained using behavior cloning and did not show promising results when we used it to train our RL agent. We borrowed the architecture of the encoder from the original paper.

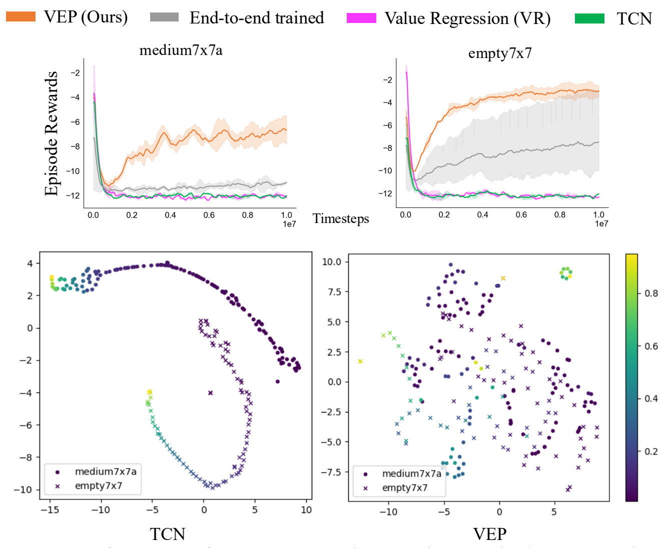

Fig. 6. **Performance of our method on in-domain tasks in Ant locomotion benchmark (Top).** Top left and right corresponds to `medium7x7a` and `empty7x7` mazes. **TSNE embeddings of our method compared to TCN (Bottom).** Color scale on the right corresponds to the value estimate. Notice how, even though the objective across both mazes is the same, TCN clusters the embeddings corresponding to the observations in the episodes for both the mazes separately, suggesting overfitting to the appearances of the maze. For the same episodes, our method learns task-agnostic representations that help in efficient downstream policy learning.

We first pretrain the encoder f_ϕ using the method described in the previous section and visually shown in Fig. 2 by using a sequence of unlabeled trajectories from both games. Once we obtain the pretrained encoder, we use an online RL algorithm, in our case PPO [36], to train a policy.

B. Results

Online RL experiments on Atari. For experiments involving Atari games, we trained the policy by freezing the pretrained encoder, without any additional fine-tuning. The encoder is pretrained using offline suboptimal, unlabeled data from `Demon-Attack` and `Space-Invaders`, and evaluated on a set of in-distribution and out-of-distribution environments. We find that the pretrained encoder is able to outperform baselines on the in-distribution by nearly 25%. This margin is larger in the transfer experiments, most notably on `Phoenix`, with nearly $2\times$ improvement over baselines.

Online RL experiments on Navigation. VEP outperforms all of our baselines by a larger margin in the navigation set as seen in Fig. 3. VEP also outperformed the end-to-end trained baseline by achieving the same performance $2.1\times$ faster (results shown on the website). In addition, we evaluate our method on out-of-distribution tasks along with existing state-of-the-art *Vision-language pretrained models*. Specifically, we compared our method (VEP) with CLIP [37], MVP [8], R3M [5] and VC-1 [38] and the results are shown in Figure 5. We hypothesize that the better performance of our method in the Navigation tasks was due to a more similar distribution of value estimates across the cities in the Navigation task, than the Atari games. Detailed specifications of the value estimates for all the Atari games and the cities in Navigation are described in the website.

Online RL experiments on Ant locomotion. Unlike single layer policies used in Atari and Navigation, we had to use a 2 layer policy with 256 hidden units that takes a 64 dimensional embedding and outputs an 8 dimensional action. The horizon of an episode was set to 1200. Since continuous control tasks are more challenging, we had to increase the number of epochs per each update iteration from 10 to 20 compared to the discrete control tasks. As presented in Figure 4, VEP outperforms all the baselines and achieves state of the art performance in both in-domain and out-of-domain mazes. For more details regarding the experiments on Ant locomotion, please visit our website.

Ablation studies on VEP. To understand the thresholding of the value estimates during sampling, we pretrained 3 different encoders using VEP with thresholds, $0.03, 0.07$ and 0.15. We were unable to go lower than 0.03, due to some samples not being able to qualify for any positive sample because of how long the episode was. We show the results in Fig. 4. Though we show the results only for `medium7x7c` due to space constraints, we observed similar trends across all OOD mazes (additional plots available on our website).

Comparison with value regression baseline. Our method might seem similar to an objective that would learn representations by simply regressing the MC value estimate. We compared our method with an encoder pretrained by regressing the ground-truth value estimate which we call *Value Regression (VR)*. We use a linear layer on top of the embedding dimension and train the entire model to predict the value estimate. After pretraining, we freeze the encoder and remove the linear layer that exposes the penultimate layer for Online RL training.

Based on the results comparing Value Regression baseline and the ablation studies performed, we can conclude that the strength of our method comes from large number of sampling the triplets (anchor, positive and negative) based on the value estimate. The contrastive learning loss ensures that the encoder learns representations useful for the downstream task. The value estimate is merely used as a tool for sampling the right triplets, and simply by regressing the encoder to predict the right value estimate does not help in learning useful representations compared to VEP, as evident in Figure. 4. Repeated sampling from a diverse set of tasks coupled with having anchor and positive within a range of value estimates helps the encoder learn task-agnostic representations.

TSNE visualizations of embeddings. For a randomly sampled episodes, each in the two in-domain tasks (`medium7x7a` and `empty7x7`), we plotted the TSNE visualizations of the embeddings obtained by passing the observations from these episodes through the pretrained encoders (Figure. 6). Although embeddings from TCN are more temporally smooth, they nevertheless get clustered separately suggesting overfitting to the appearance of the environment. We argue that although temporal smoothness in the embedding space approximates an implicit value function [6] and is crucial for learning meaningful representations, this kind of task-specific clustering (Figure. 6), assuming that these tasks share the same objective/goal, is detrimental for generalization for unseen tasks. Our method on the other hand learns both temporally consistant embeddings and task-agnostic representations.

Quality of suboptimal, unlabeled data. We also evaluated our method by using different amounts of diversity and optimality in our dataset. Specifically, we compared with the datasets with episodes of length less than 400, 500-800, 1000-1400. All episodes in the respective datasets have a cumulative reward between 12-15. We also used suboptimal, unlabeled data that consist of episodes that complete 10% of the actual task. Further, we also included episodes that consist of suboptimal data from 3 and 4 cities. As shown in Figure 7, we observe that our method is not affected by the suboptimality of the dataset. In other words, VEP uses non-zero value estimates during pretraining, and the suboptimality of the dataset has a negligible effect on the quality of representations learnt. Additionally, using multi-city datasets did help in learning better representations, except `Wall Street` and `Union Square`. In regards to those two cities, the original evaluations (red and orange bars) were obtained using a pretrained encoder trained on data from those very cities (In-domain), and hence performed the best.

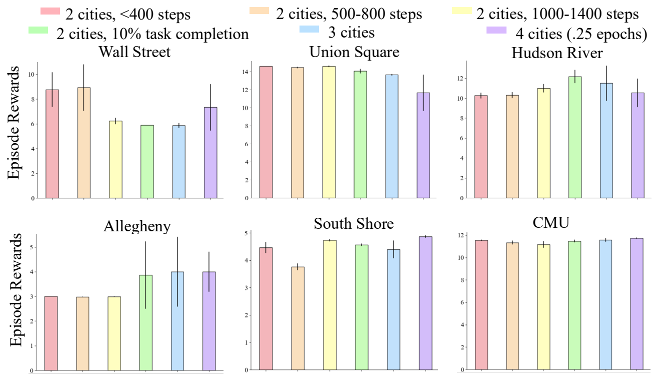

Fig. 7. **Performance comparison on the quality of suboptimal, unlabeled data** Each of the above bar plots corresponds to the evaluation of the encoder in a different city. Each coloured bar corresponds to a specific suboptimal, unlabeled dataset used for pretraining. We also provide 95% confidence intervals along with the mean cumulative reward.

VI. Conclusion

We formulated a method to learn representations of states from different tasks solely based on the temporal distance to the goal frame. This way, the skills learned from the training tasks could be transferred to novel but related tasks. We showed the efficacy of our method by performing comprehensive evaluations on discrete and continuous control benchmarks like Atari, Visual Navigation and Ant locomotion.

For all the curent experiments, we used PPO for training the policy parameters. Future work would involve using more sample efficient algorithms like SAC [39]. We also plan on experimenting with behavior cloning/few-shot imitation learning on manipulation benchmarks. Although our work shows the potential of intra-task pretraining, it is currently limited to the tasks that have similar objectives but still have different dynamics and appearances. We plan to study large-scale pretraining for more complex tasks in the future.

References

[1] Rutav M Shah and Vikash Kumar. "RRL: Resnet as representation for Reinforcement Learning". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9465–9476.

[2] Zhecheng Yuan et al. "Pre-trained image encoder for generalizable visual reinforcement learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 13022–13037.

[3] Simone Parisi et al. "The unsurprising effectiveness of pre-trained vision models for control". In: *international conference on machine learning*. PMLR. 2022, pp. 17359–17371.

[4] Litian Gong et al. "AutoFocus-IL: VLM-based Saliency Maps for Data-Efficient Visual Imitation Learning without Extra Human Annotations". In: *arXiv preprint arXiv:2511.18617* (2025).

[5] Suraj Nair et al. "R3M: A Universal Visual Representation for Robot Manipulation". In: *Conference on Robot Learning*. PMLR. 2023, pp. 892–909.

[6] Yecheng Jason Ma et al. "VIP: Towards Universal Visual Reward and Representation via Value-Implicit Pre-Training". In: *The Eleventh International Conference on Learning Representations*. 2023.

[7] Siddharth Karamcheti et al. "Language-Driven Representation Learning for Robotics". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2023.

[8] Ilija Radosavovic et al. "Real-world robot learning with masked visual pre-training". In: *Conference on Robot Learning*. PMLR. 2023, pp. 416–426.

[9] Tete Xiao et al. "Masked visual pre-training for motor control". In: *arXiv preprint arXiv:2203.06173* (2022).

[10] Younggyo Seo et al. "Masked world models for visual control". In: *Conference on Robot Learning*. PMLR. 2023, pp. 1332–1344.

[11] Anthony Liang, Jesse Thomason, and Erdem Bıyık. "Visarl: Visual reinforcement learning guided by human saliency". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2024, pp. 2907–2912.

[12] Benjamin Eysenbach et al. "Contrastive learning as goal-conditioned reinforcement learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 35603–35620.

[13] Timothée Lesort et al. "State representation learning for control: An overview". In: *Neural Networks* 108 (2018), pp. 379–392.

[14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations (ICLR)*. 2014.

[15] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *International conference on learning representations*. 2017.

[16] David Ha and Jürgen Schmidhuber. "Recurrent world models facilitate policy evolution". In: *Advances in neural information processing systems* 31 (2018).

[17] Ankesh Anand et al. "Unsupervised state representation learning in atari". In: *Advances in neural information processing systems* 32 (2019).

[18] Zhihui Xie et al. "Pretraining in deep reinforcement learning: A survey". In: *arXiv preprint arXiv:2211.03959* (2022).

[19] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. "Self-supervised exploration via disagreement". In: *International conference on machine learning*. PMLR. 2019, pp. 5062–5071.

[20] Max Schwarzer et al. "Pretraining representations for data-efficient reinforcement learning". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12686–12699.

[21] Andrei A Rusu et al. "Progressive neural networks". In: *arXiv preprint arXiv:1606.04671* (2016).

[22] Shani Gamrian and Yoav Goldberg. "Transfer learning for related reinforcement learning tasks via image-to-image translation". In: *International conference on machine learning*. PMLR. 2019, pp. 2063–2072.

[23] Pierre Sermanet et al. "Time-contrastive networks: Self-supervised learning from video". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2018, pp. 1134–1141.

[24] Yan Duan et al. "Rl^2: Fast reinforcement learning via slow reinforcement learning". In: *arXiv preprint arXiv:1611.02779* (2016).

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

[26] Ricardo Luna Gutierrez and Matteo Leonetti. "Information-theoretic task selection for meta-reinforcement learning". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 20532–20542.

[27] Chethan Anand Bhateja et al. "Robotic Offline RL from Internet Videos via Value-Function Pre-Training". In: *NeurIPS 2023 Foundation Models for Decision Making Workshop*.

[28] Jiahui Zhang et al. "ReWiND: Language-Guided Rewards Teach Robot Policies without New Demonstrations". In: *Conference on Robot Learning (CoRL)*. 2025.

[29] Bogdan Mazoure et al. "Accelerating exploration and representation learning with offline pre-training". In: *arXiv preprint arXiv:2304.00046* (2023).

[30] Joseph Ortiz et al. "DMC-VB: A Benchmark for Representation Learning for Control with Visual Distractors". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 6574–6602.

[31] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823.

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

[33] Justin Fu et al. "D4rl: Datasets for deep data-driven reinforcement learning". In: *arXiv preprint arXiv:2004.07219* (2020).

[34] Piotr Mirowski et al. "The streetlearn environment and dataset". In: *arXiv preprint arXiv:1903.01292* (2019).

[35] Greg Brockman et al. "Openai gym". In: *arXiv preprint arXiv:1606.01540* (2016).

[36] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[37] Alec Radford et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PmLR. 2021, pp. 8748–8763.

[38] Arjun Majumdar et al. "Where are we in the search for an artificial visual cortex for embodied intelligence?" In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 655–677.

[39] Tuomas Haarnoja et al. "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. Pmlr. 2018, pp. 1861–1870.