VALUE FUNCTION SPACES: SKILL-CENTRIC STATE ABSTRACTIONS FOR LONG-HORIZON REASONING

Dhruv Shah[γβ], **Peng Xu**[γ], **Yao Lu**[γ], **Ted Xiao**[γ]
Alexander Toshev[γ], **Sergey Levine**[γβ], **Brian Ichter**[γ]

[γ]Google Research, Robotics @ Google
[β]Berkeley AI Research, UC Berkeley

ABSTRACT

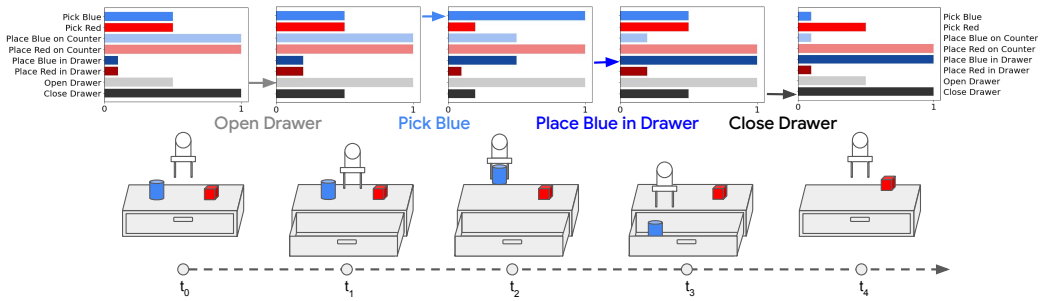
Reinforcement learning can train policies that effectively perform complex tasks. However for long-horizon tasks, the performance of these methods degrades with horizon, often necessitating reasoning over and composing lower-level skills. Hierarchical reinforcement learning aims to enable this by providing a bank of low-level skills as action abstractions. Hierarchies can further improve on this by abstracting the space states as well. We posit that a suitable state abstraction should depend on the capabilities of the available lower-level policies. We propose Value Function Spaces: a simple approach that produces such a representation by using the value functions corresponding to each lower-level skill. These value functions capture the affordances of the scene, thus forming a representation that compactly abstracts task relevant information and robustly ignores distractors. Empirical evaluations for maze-solving and robotic manipulation tasks demonstrate that our approach improves long-horizon performance and enables better zero-shot generalization than alternative model-free and model-based methods.

1 INTRODUCTION

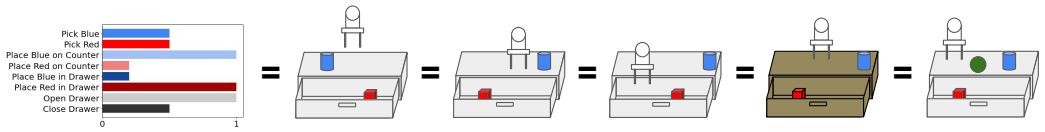
For an agent to perform complex tasks in realistic environments, it must be able to effectively reason over long horizons, and to parse high-dimensional observations to infer the contents of a scene and its affordances. It can do so by constructing a compact representation that is robust to distractors and suitable for planning and control. Consider, for instance, a robot rearranging objects on a desk. To succcessfully solve the task, the robot must learn to sequence a series of simple skills, such as picking and placing objects and opening drawers, and interpret its observations to determine which skills are most appropriate. This requires the ability to understand the capabilities of these simpler skills, as well as the ability to plan to execute them in the correct order.

Hierarchical reinforcement learning (HRL) aims to enable this by leveraging *abstraction*, which simplifies the higher-level control or planning problem. Typically, this is taken to mean abstraction of actions in the form of primitive skills (e.g., options (Sutton et al., 1999)). However, significantly simplifying the problem for the higher level requires abstraction of both states *and* actions. This is particularly important with rich sensory observations, where standard options frameworks provide a greatly abstracted state space, but do not simplify the perception or estimation problem. The nature of the ideal state abstraction in HRL is closely tied to the action abstraction, as the most suitable abstraction of state should depend on the kinds of decisions that the higher-level policy needs to make, which in turn depends on the actions (skills) available to it. This presents a challenge in designing HRL methods, because it is difficult to devise a state abstraction that is both highly abstracted (and therefore removes many distractors) and still sufficient to make decisions for long-horizon tasks. This challenge differs markedly from representation learning problems in other domains, like computer vision and unsupervised learning (Schroff et al., 2015; van den Oord et al., 2018; Chen et al., 2020), since it is intimately tied to the *capabilities* exposed to the agent via its skills.

We therefore posit that a suitable representation for a higher-level policy in HRL should depend on the capabilities of the skills available to it. If this representation is sufficient to determine the abilities and outcomes of these skills, then the high-level policy can select the skill most likely to perform the desired task. This concept is closely tied to the notion of affordances, which has long been studied



(a) A trajectory through the skill value function space for the task "Place blue block in drawer". VFS, visualized on top of corresponding scene, captures positional information about the contents of the scene, preconditions for interactions, and the effects of executing a feasible skill, making it suitable for high-level planning.



(b) VFS learns a skill-centric representation of the scene and ignores factors like arm pose, task-centric object positions, color of the desk, or additional objects, which do not affect the values of the low-level skills. All configurations shown above are *functionally equivalent* and hence, map to the same VFS representation.

Figure 1: Visualizing VFS embeddings in an example desk rearrangement task. VFS can capture the affordances of the low-level skills while ignoring exogenous distractors.

in cognitive science and psychology as an action-centric representation of state (Gibson, 1977), and has inspired techniques in robotics and RL (Zech et al., 2017; Xu et al., 2021; Mandikal & Grauman, 2021). Given a set of skills that span the possible interactions in an environment, we propose that the value functions corresponding to these policies can provide a representation that suitably captures the capabilities of the skills in the current state and thus can be used to form a compact embedding space for high-level planning. We call this embedding space Value Function Spaces (VFS). Note that we intend to use these skills as high-level actions without modifications – the important problems of skill discovery and test-time adaptation are beyond the scope of our work.

Figure 1a illustrates the state abstraction constructed by VFS for the desk rearrangement example discussed above: VFS captures the affordances of the skills and represents the state of the environment, along with preconditions for the low-level skills, forming a functional representation to plan over. Since VFS constructs a skill-centric representation of states using the value functions of the low-level skills, it captures functional equivalence of states in terms of their affordances: in Figure 1b, states with varying object or arm positions (i-iii), different background textures (iv), and distractor objects (v) are functionally equivalent for planning, and map to the same embedding in our representation. This significantly simplifies the high-level planning problem, making it easier for the higher level policy to generalize to novel environments to the limit of the skills themselves.

Statement of Contributions. The primary contribution of this work is VFS, a novel state representation derived from the value functions of low-level skills available to the agent. We show that VFS leverages the properties of value functions, notably their ability to model possible skills and completed skills, to form an effective representation for high-level planning, and is compatible with both model-free and model-based high-level policies. Empirical evaluations in maze-solving and robotic manipulation demonstrate that the skill-centric representation constructed by VFS outperforms representations learned using contrastive and information-theoretic objectives in long-horizon tasks. We also show that VFS can generalize to novel environments in a zero-shot manner.

2 Related Work

Hierarchical RL has been studied extensively in the literature, commonly interpreted as a *temporal abstraction* of the original MDP. Early works have interpreted the hierarchy introduced in this setting as an abstraction of state and action spaces (Sutton et al., 1999; Dietterich, 2000; Thomas & Barto, 2012). The popular options framework Sutton et al. (1999); Precup (2000) provides a natural way of incorporating temporally extended actions into RL systems. An agent that possesses the transition model and reward model for such a SMDP (known as an *option model*) is capable of sample-based

planning in discrete (Kocsis & Szepesvári, 2006) and continuous state spaces (Konidaris et al., 2014; Gopalan et al., 2017). However, doing so in environments with high-dimensional observations (such as images) is challenging. In this work, we explore the efficacy of learned skills operating on high-dimensional observations for long-horizon control in realistic environments.

To improve the quality of lower-level policies, recent work in HRL has studied various facets of the problem, including discovery of skills (Konidaris & Barto, 2009; Zhang et al., 2021b; Florensa et al., 2017; Warde-Farley et al., 2018), end-to-end training of both levels (Kulkarni et al., 2016; Tessler et al., 2017) and integrating goal-conditioned behaviors (Ghosh et al., 2019; Nachum et al., 2019). In this work, we assume that the low-level skills are given, and do not focus on discovering them. Instead, we focus on how skills can simplify the higher-level control problem by providing a representation that is suitable for either model-based or model-free control. The high-level policy does not rely on any hand-crafted rewards and only receives a sparse outcome reward.

Our method can also be interpreted as a representation learning approach. Representation learning techniques have been employed extensively in model-free RL by augmenting auxiliary tasks based on reconstruction losses (Lange et al., 2012; Higgins et al., 2017; Yarats et al., 2019) and predicting the future conditioned on past observations (Schmidhuber, 1990; Jaderberg et al., 2017; van den Oord et al., 2018; Shelhamer et al., 2016). Contrastive learning has also been used in recent works to discover a meaningful latent space and extract reward signals for RL (Sermanet et al., 2017; Warde-Farley et al., 2018; Dwibedi et al., 2018; Laskin et al., 2020). Unlike these prior works, our aim is specifically to learn a representation that is grounded in the capabilities of the low-level skills, which gives us a skill-centric abstraction of high-dimensional observations that can compactly represent affordances of the scene while being robust to functionally-irrelevant distractors.

Alongside developments in model-free RL, prior work has also sought to learn predictive models of the environment for sampling and planning, in a model-based RL framework. This has been demonstrated by learning dynamics using future predictions (Watter et al., 2015; Oh et al., 2017; Ebert et al., 2017; Banijamali et al., 2018; Ha & Schmidhuber; Hafner et al., 2018; Ichter & Pavone, 2019; Hafner et al., 2020; Zhang et al., 2019), learning belief representations (Gregor et al., 2019; Lee et al., 2019) and representing state similarity using the bisimulation metric (Castro, 2020; Zhang et al., 2021a; Agarwal et al., 2021). The combination of learned model-free policies with structures like graphs (Savinov et al., 2018; Eysenbach et al., 2019) and trees (Ichter et al., 2021) to plan over extended horizons has also been demonstrated to improve generalization and exploration. Although our method can utilize a model-based high-level controller to plan over temporally extended skills, our objective is not to develop better model-based RL algorithms. Rather, we focus on developing suitable state abstractions in a hierarchical RL framework, and then evaluate these representations by employing them in both model-based and model-free higher-level controllers.

3 Preliminaries

We assume that an agent has access to a finite set of temporally extended options O, or skills, which it can sequence to solve long-horizon tasks. These skills can be trained for a wide range of tasks by using manual reward specification (Huber & Grupen, 1998; Stulp & Schaal, 2011), using relabeling techniques (Andrychowicz et al., 2017), or via unsupervised skill discovery (Daniel et al., 2016; Fox et al., 2017; Warde-Farley et al., 2018; Sharma et al., 2020). In this work, the skills are learned via RL with sparse, hand-specified reward functions, but in general could come from any source. Since many prior works have focused on skill discovery, we do not explicitly address how these skills are produced, but assume that they are given. We assume that each skill has a maximum rollout length of τ time steps, after which it is terminated. We also assume that each skill $o_i \in O$ is accompanied by a critic, or value function, V_{o_i} denoting the expected cumulative skill reward executing skill o_i from current state s_t. This is readily available for policies trained with RL, but for an arbitrary policy, we can use policy evaluation to obtain a value function.

Our setting is closely related to the *options framework* of Sutton et al. (1999). Options are skills that consist of three components: a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, a termination condition $\beta : \mathcal{S}^+ \to [0, 1]$, and an initiation set $\mathcal{I} \subseteq \mathcal{S}$, where $\mathcal{S}, \mathcal{A}$ are the low-level state and action spaces in a fully observable decision process. An option $\langle \mathcal{I}, \pi, \beta \rangle$ is available at state s_t if and only if $s_t \in \mathcal{I}$. We do not assume we have an initiation set, and show that the value functions provide this information. We assume that the policies come with a termination condition, or alternatively, have a fixed horizon τ. Next, we describe a framework to formulate a decision problem that uses these options for planning.

We assume that the low-level observation and action space of the agent can be described as a fully observable *semi-Markov decision process* (SMDP) $\mathcal{M}$ described by a tuple $(S, O, R, P, \tau, \gamma)$, where $S \subseteq \mathbb{R}^n$ is the n-dimensional continuous state space; O is a finite set of temporally extended skills, or options, with a maximum horizon of τ time steps; $R(s'|s, o_i)$ is the task reward received after executing the skill $o_i \in O$ at state $s \in S$; $P(s'|s, o_i)$ is a PDF describing the probability of arriving in state $s' \in S$ after executing skill $o_i \in O$ at state $s \in S$; $\gamma \in (0, 1]$ is the discount factor. Given a finite set of options, our objective is to obtain a high-level decision-policy that selects *among* them, resulting in a sequence of options that reach the desired goal.

Note that generally, many image-based problems are partially observed so that the entire history of observation-action pairs may be required to describe the state. Explicitly addressing partial observability is outside the scope of our work, so we follow the convention in prior work by assuming full observability and refer to images as states (Lillicrap et al., 2016; Nair et al., 2018).

4 Skill Value Functions as State Spaces

The notion of value in RL is closely related to affordances, in that the value function predicts the capabilities of the skill being learned. The supervised learning problem of affordance prediction (Ugur et al., 2009) can in fact be cast as a special case of value prediction (Sutton & Barto, 2018; Graves et al., 2020). In this section, we construct a skill-centric representation of state derived from skill value functions that captures the affordances of the low-level skills, and empirically show that this representation is effective for high-level planning.

Given an SMDP $\mathcal{M}(S, O, R, P, \tau, \gamma)$ with a finite set of k skills $o_i \in O$ trained with sparse outcome rewards and their corresponding value functions V_{o_i}, we construct an embedding space Z by stacking these skill value functions. This gives us an abstract representation that maps a state s_t to a k-dimensional representation $Z(s_t) \coloneqq [V_{o_1}(s_t), V_{o_2}(s_t), ..., V_{o_k}(s_t)]$, which we call the Value Function Spaces, or VFS for short. This representation captures *functional information* about the exhaustive set of interactions that the agent can have with the environment by means of executing the skills, and is thus a suitable state abstraction for downstream tasks.

Figure 1(a) illustrates the proposed state abstraction for a conceptual desk rearrangement task rollout with eight low-level skills. A raw observation, such as an image of the robotic arm and desk, is abstracted by a 8-dimensional tuple of value functions of the available skills. This representation captures positional information about the contents of the scene (e.g., both blocks are on the counter and drawer is closed since the corresponding values are 1 at t_0), preconditions for interactions (e.g., both blocks can be lifted since the "pick" values are high at t_0), and the effects of executing a feasible skill (e.g., the value corresponding to the "Open Drawer" skill increases on executing it at t_1), making it suitable for high-level planning.

Since VFS learns a skill-centric representation of the scene, it is robust to exogenous factors of variation, such as background distractors and appearance of task-irrelevant components of the scene. This also enables VFS to generalize to novel environments with the same set of low-level skill, which we demonstrate empirically. Figure 1(b) illustrates this for the desk rearrangement task. All configurations shown are *functionally equivalent* – an open drawer with the red cube in it, blue cylinder on the countertop, empty robot arm – and can be interacted with identically. VFS maps these configurations to the same abstract state by ignoring factors like arm pose, task-centric position of objects, color of the desk, or additional objects, which do not affect the low-level skills.

5 Model-Free RL with Value Function Spaces

In this section, we instantiate a hierarchical model-free RL algorithm that uses VFS as the state abstraction and the skills as the low-level actions. We compare the long-horizon performance of VFS to alternate representations for HRL trained with constrastive and information theoretic objectives for the task of maze-solving and find that VFS outperforms the next best baseline by up to 54% on the most challenging tasks. Lastly, we compare the zero-shot generalization performance of these representations and empirically demonstrate that the skill-centric representation constructed by our method can successfully generalize to novel environments with the same set of low-level skills.

5.1 An Algorithm for Hierarchical RL

We instantiate a hierarchical RL algorithm that learns a Q-function $Q(Z, o)$ at the high-level using VFS as the "state" Z and the skills $o_i \in O$ as the temporally extended actions. Given such a Q-function, a greedy high-level policy can be obtained by $\pi_Q(Z) = \arg\max_{o_i \in O} Q(Z, o_i)$. We use DQN (Mnih et al., 2015), which uses mini-batches sampled from a replay buffer of transition tuples (Z_t, o_t, r_t, Z_{t+1}) to train the learned Q-function to satisfy the Bellman equation. This is done using gradient descent on the loss $\mathcal{L} = \mathbb{E}\left(Q(Z_t, o_t) - y_t\right)^2$, where $y_t = r_t + \gamma \max_{o_{t'} \in O} Q(Z_{t+1}, o_{t'})$.

In order to make the optimization problem more stable, the targets y_t are computed using a separate target network which is update at a slower pace than the main network. Q-learning also suffers from an overestimation bias, due to the maximization step above, and harms learning. Hasselt et al. (2016) address this overestimation by decoupling the selection of action from its evaluation. We use this variant, called DDQN, for subsequent evaluations in this work. Note that we are using the skills as given, without updating or finetuning them.

While there have been several improvements to DDQN (Hessel et al., 2018) and alternative algorithms for model-free RL in discrete action spaces (Schaul et al., 2015; Bellemare et al., 2017; Christodoulou, 2019), and these improvements will certainly improve the overall performance of our algorithm, our goal is to evaluate the efficacy of VFS as a state representation and we study it in the context of a simple DDQN pipeline.

5.2 Evaluating Long-Horizon Performance in Maze-Solving

To evaluate the long-horizon performance of VFS against commonly used representation learning methods, we use the versatile MiniGrid environment (Chevalier-Boisvert et al., 2018) in a fully observable setting, where the agent receives a top-down view of the environment. We consider two tasks: (i) *MultiRoom*, where the agent is tasked with reaching the goal by solving a variable-sized maze spanning up to 10 rooms. The agent must cross each room and open the door to access the following rooms; (ii) *KeyCorridor*, where the agent is tasked with reaching a goal up to 7 rooms away and may face locked doors. The agent must find the key corresponding to a color-coded door and open it to access subsequent rooms. Both these tasks have a sparse reward that is only provided for successfully reaching the goal. This presents a challenging domain for long-horizon sequential reasoning, where tasks may require over 200 time steps to succeed, making a great testbed for evaluating the ability of the state abstractions to capture relevant information for sequencing multiple skills. The agents have access to the following temporally extended skills – `GoToObject`, `PickupObject`, `DropObject` and `UnlockDoor` – where the first three skills are text-conditioned and `Object` may refer to a door, key, box or circle of any color. Since MiniGrid is easily reconfigurable, we also generate a set of holdout *MultiRoom* mazes with different grid layouts to evaluate the zero-shot generalization performance of the policies trained with these representations. Note that the grid layouts, as well as object positions, for these tasks are randomly generated for every experiment and are not static. Example grid layouts are shown in Figure 2. We provide further implementation details in Appendix A.1.

Baselines: We compare VFS extensively against a variety of competitive baselines for representation learning in RL using contrastive and information-theoretic objectives. We consider representations learned both offline and online (in loop with RL); note that VFS is constructed entirely from the values of the available skills and is not learned. Further, all baselines have access to these skills.

1. *Raw Observations:* We train a high-level policy operating on raw input observations.
2. *Autoencoder (AE):* We use an autoencoder to extract a compact latent space using a reconstruction loss on an offline dataset of trajectories, similar to Lange et al. (2012).
3. *Contrastive Predicting Coding (CPC):* We learn a representation by optimizing the InfoNCE loss over an offline dataset of trajectories (van den Oord et al., 2018).
4. *Online Variational Autoencoder (VAE):* We learn a VAE representation jointly (online) with the high-level policy operating on this representation (Yarats et al., 2019).
5. *Contrastive Unsupervised Representations for RL (CURL):* We learn a representation by optimizing a contrastive loss jointly (online) with the RL objective (Laskin et al., 2020).

Representation	MultiRoom				KeyCorridor	
	2	4	6	10	3	7
Raw Observations	0.64	0.46	0.42	0.29	0.47	0.32
AE (Lange et al., 2012)	0.70	0.64	0.51	0.34	0.59	0.33
CPC (van den Oord et al., 2018)	0.77	0.69	0.55	0.37	0.63	0.35
VAE† (Yarats et al., 2019)	0.79	0.74	0.58	0.49	0.79	0.50
CURL† (Laskin et al., 2020)	0.82	0.76	0.63	0.43	**0.83**	0.54
VFS (Ours)	**0.98**	**0.92**	**0.83**	**0.77**	0.82	**0.68**

Table 1: Success rates of different representations for model-free RL across varying levels of difficulty and time horizons (second row denotes complexity in terms of number of rooms). VFS explicitly captures the capabilities of the low-level skills, greatly simplifying the control problem for HRL, and outperforms all baselines. Online methods (VAE and CURL; denoted by †) learned jointly with the RL objective outperform their offline counterparts (AE and CPC), but their performance degrades for longer-horizon tasks.

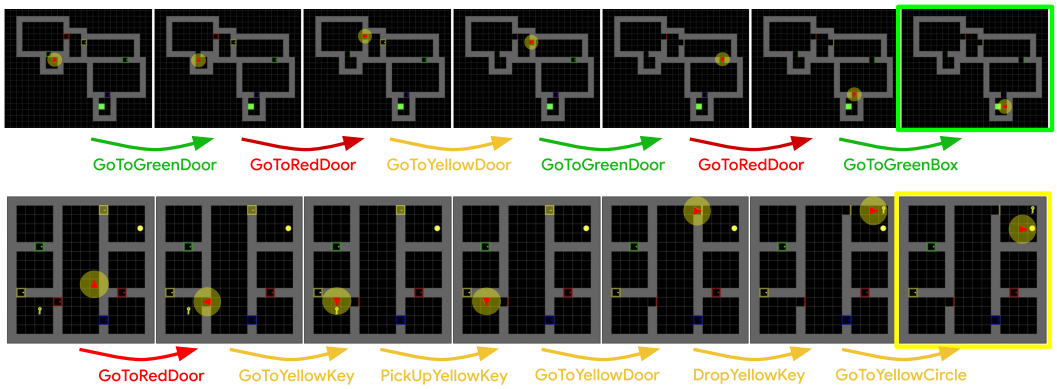


Figure 2: Successful rollouts of HRL using VFS to solve long-horizon tasks in *MultiRoom-6* **(top)** and *KeyCorridor-7* **(bottom)** by sequentially executing multiple low-level skills (labeled under the arrows).

Evaluation: We run experiments for the two tasks with varying number of rooms, to study the performance of the algorithms with increasing time horizon, and report the success rates in Table 1. HRL from raw observations demonstrates a success rate of 64% in the two-room environment, which can be attributed to the powerful set of skills available to the high-level policy, but this quickly drops to 29% in the largest environment. The offline baselines (AE and CPC) construct a compact state abstraction, which makes the learning problem easier for DDQN, and show significant improvements in smaller environments (MultiRoom-2,4 and KeyCorridor-3). However, they are unable to improve the performance in larger environments. We hypothesize that this is due to the inability of the representations to capture information necessary for the high-level policy, since they are learned independent of the controller. The performance of representations learned online (VAE and CURL), which are implemented analogous to their offline counterparts (AE and CPC, respectively) support this hypothesis and improves the performance across all tasks by learning representations jointly with the controller, and scoring up to 54% in the most challenging environment. We hypothesize that the limited performance of these methods is due to the lack of direct influence of the downstream task on the representation. Despite being constructed offline, VFS explicitly captures the capabilities of the low-level skills and provides an action-centric representation for the high-level policy, greatly simplifying the control problem. This is reflected in the performance of VFS across all tasks – it outperforms all baselines, scoring 98% on the simplest task and up to 68% on the most challenging task, beating the next best method by over 25%. Figure 2 shows sample rollouts of the high-level policy using VFS as the state representation in the two tasks discussed.

5.3 Zero-Shot Generalization

We evaluate the generalization abilities of these representations by training them for the *KeyCorridor* task (as above) and evaluating on *MultiRoom*. Since the skills available in *MultiRoom* are a subset of

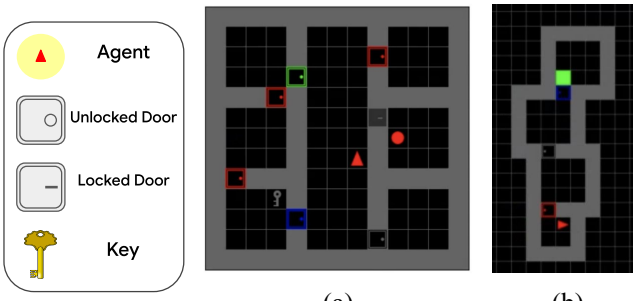


Figure 3: We study the zero-shot generalization by using policies trained in `KeyCorridor` **(a)** and deploying in `MultiRoom` **(b)**.

Representation	MR4	MR10
Raw	0.15	0.03
AE	0.41	0.23
CPC	0.47	0.20
VAE†	0.25	0.03
CURL†	0.27	0.07
VFS (Ours)	**0.87**	**0.67**
HRL-Target	0.92	0.77

Table 2: Success rates for zero-shot generalization. VFS learns a skill-centric representation that can generalize to novel environments without collecting new data.

those in *KeyCorridor*, we ignore any invalid actions executed by the agent. Note that the high-level agent has not been trained in *MultiRoom* and the policies are not updated in the target environments.

Table 2 shows the success rates of the representations in the *MultiRoom* tasks with 4 (*MR4*) and 10 rooms (*MR10*). Unsurprisingly, HRL with raw observations fails to generalize, because the maze layout differs significantly (e.g. see Figure 3). AE and CPC learn a compact representation from high-dimensional observations and allow the high-level policy to generalize to simpler tasks like *MR4* and achieve up to 47% success rate. Interestingly, their online counterparts (VAE and CURL) also fail to generalize and perform poorly, likely because representations learned jointly with the RL policy can overfit to the source environment. VFS learns a skill-centric representation that can generalize zero-shot to tasks that use the same skills, and thus outperforms the next best baseline by up to 180%, closely matching the performance of an HRL policy trained from scratch in the target environment with online interaction (*HRL-Target*).

6 Model-Based Planning with Value Function Spaces

In this section, we introduce a simple model-based RL algorithm that uses VFS as the "state" for planning, which we term VFS-MB. We study the performance of VFS in the context of a robotic manipulation task, and compare it to alternate representations for model-based RL. We find that the performance of VFS as an abstract representation for raw image observations outperforms all baselines and closely matches that of a pipeline with access to oracular state of the simulator, showing the efficacy of our method in long-horizon planning.

6.1 A Simple Algorithm for Model-Based RL

We use a simple model-based planner that learns a *one-step* predictive model, using VFS as the "state." Specifically, this model learns the transition dynamics $Z_{t+1} = \hat{f}(Z_t, o_t) \,\forall\, o_t \in O$ via supervised learning using a dataset of prior interactions in the environment. Note that the predictive model (and the subscript of Z) is over high-level policy steps, which may be up to τ steps of the low-level skills. This dataset can be collected simply by executing the available skills in the environment for τ steps, where τ is the maximum horizon of the SMDP $\mathcal{M}$, or until termination.

In order to use the learned model $\hat{f}(Z_t, o_t)$, a goal latent state Z_g, and a scoring function ϵ (e.g. mean squared error) for the high-level task, we need to solve the following optimization problem for the optimal sequence of skills $(o_t, \ldots, o_{t+H-1})$ to reach the goal:

$$(o_t, \ldots, o_{t+H-1}) = \underset{o_t,\ldots,o_{t+H-1}}{\arg\min}\ \epsilon(\hat{Z}_{t+H}, Z_g) : \hat{Z}_t = Z_t,\ \hat{Z}_{t'+1} = \hat{f}(\hat{Z}_{t'}, o_{t'}) \tag{1}$$

We use a sampling-based method to find solutions to the above equation. We use random shooting (Rao, 2009) to randomly generate K candidate option sequences, predict the corresponding Z sequences using the learned model $\hat{f}$, compute the rewards for all sequences, and pick the candidate action sequence leading to a latent state closest to the goal, according to Equation 1. We execute the policy using model-predictive control: the policy executes only the first skill o_t, receives the updated

Z_{t+1} from the environment, and recalculates the optimal action sequence iteratively. Figure 4 shows an overview of the algorithm. We provide further implementation details in Appendix A.2

Note that our goal is to study the behavior of VFS as a state representation in existing RL pipelines, rather than developing a better model-based algorithm. Using sophisticated methods that adjust the sampling distribution, as in the cross-entropy method (Finn & Levine, 2017; Hafner et al., 2018; Chua et al., 2018) or path integral optimal control (Williams et al., 2015; Lowrey et al., 2019), as well as more sophisticated models and planning or control methods would likely improve overall performance further, but is outside the scope of this work.

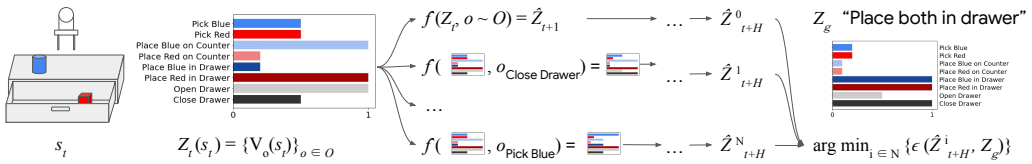


Figure 4: Overview of model-based planning with VFS. We learn a one-step predictive model for the embedding Z_t and use random shooting with a scoring function ϵ to find the best action to the encoded goal Z_g.

6.2 Application: Robotic Manipulation

We evaluate the performance of VFS-MB in a complex image-based task using a simulated manipulation environment with an 8-DoF robotic arm, similar to the setup used by Jang et al. (2021). The robot only has access to high-dimensional egocentric visual observations. We consider the task of semantic rearrangement, which requires rearranging objects into semantically meaningful positions – e.g., grouped into (multiple) clusters, or moved to a corner (Figure 5). Efficiently solving this task requires the robot to plan over a series of manipulation maneuvers, interacting with up to 10 distinct objects. This requires long-horizon planning, and presents a major challenge for RL methods. Each of the methods we compare have access to a library of `MoveANearB` skills, trained as a multi-task policy with MT-Opt (Kalashnikov et al., 2021). There are 10 objects, and 9 possible destinations near which they can be moved, resulting in a total of 90 skills, which then comprise the action space for planning. The skills control the robot at a frequency of 0.5 Hz, taking an average of 14 time steps to execute with a success rate of 94%. We provide further details about these skills in Appendix A.2.

We consider two versions of the task, which arrange either 5 or 10 objects to semantic positions. The *O5* environment uses a random subset of the 10 objects in every experiment; lesser objects allow a smaller planning horizon, making planning problem simpler than in the *O10* environment, which has all 10 objects. We randomize the object positions on the table and command the algorithms to reach the same goal with a planning horizon $H = 7$ steps for the smaller environment and $H = 15$ steps for the larger environment, reporting the task success rate averaged over 20 experimental runs.

Baselines: To evaluate the efficacy of the proposed representation for high-level model-based planning, we compare it against four alternative representations used in conjunction with the algorithm described in Section 6.1. All methods have access to the skills and use them as the low-level action space. *Raw Image* learns a policy on raw visual observations from the robot's onboard camera. *VAE* uses a variational autoencoder to project the observations to a $100-$dimensional learned embedding space, similar to Corneil et al. (2018). We train this VAE offline from trajectories collected by rolling out the models. *CPC* uses contrastive predictive coding (van den Oord et al., 2018) to learn a similar representation from offline trajectories. We also compare against an *Oracle State* baseline that has access to privileged information – the simulator state – and learns the model on this . This gives us an upper bound for the performance of the baselines.

Evaluation: Table 3 shows the success rates on the two versions of the task for each prior method. The model trained using image observations fares poorly and fails at all but the simplest starting configurations. It succeeds in 2/20 experiments in *O10*, largely due to poor model predictions. The VAE and CPC representations, which learn a more compact embedding space to plan over, succeed in up to 65% of the tasks in the simpler *O5* environment. However, their performance falls sharply in the *O10* environment, where the longer horizons read to greater error accumulation. VFS constructs an effective representation that captures the state of the scene as well as the affordances of the skills,

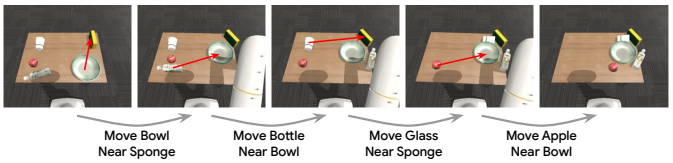


Figure 5: Example rollout of model-based RL with VFS as the state representation for robotic manipulation. The robot plans over multiple low-level skills to achieve the semantic goal "move all to top-right corner". Red arrows specify the next skill planned by the model, and is overlaid for visualization purposes only.

Representation	O5	O10
Raw Image	0.25	0.1
VAE	0.6	0.3
CPC	0.65	0.4
VFS (Ours)	**1**	**0.8**
Oracle State	1	0.85

Table 3: Success rates for robotic manipulation. VFS outperforms all baselines and closely matches oracular performance.

and succeeds in all tasks in the *O5* environment, matching the oracle performance. It also succeeds in 80% of the tasks in *O10*, closely matching the oracle's performance of 85%.

To understand the factors captured by VFS, we sample encoded observations from a large number of independent trajectories and visualize their 2D t-SNE embeddings (van der Maaten & Hinton, 2008). Figure 6 shows that VFS can successfully capture information about objects in the scene and affordances (e.g. which object is in the robot's arm and can be manipulated), while ignoring distractors like the poses of the objects on the table and the arm.

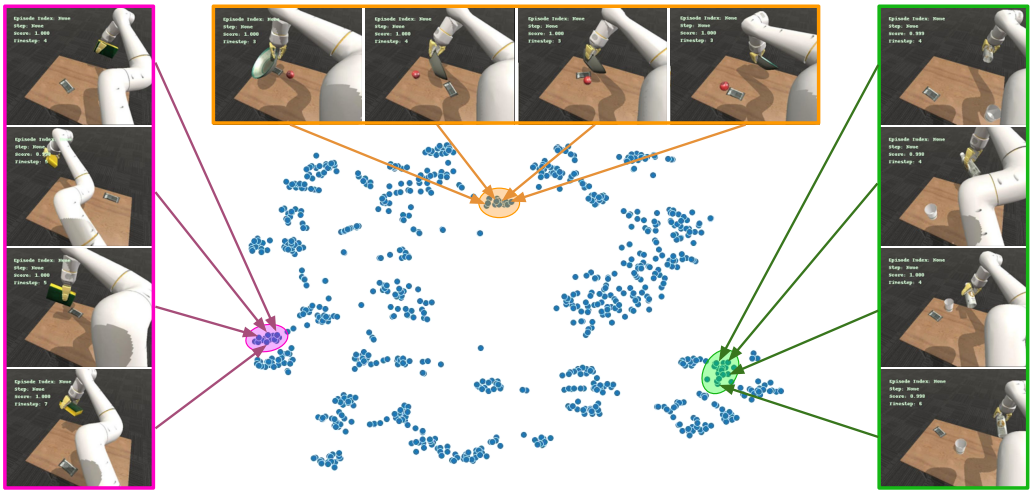


Figure 6: A t-SNE diagram of observations encoded by VFS, showing functionally equivalent observations mapped to the same representation. VFS discovers clusters with **(top)** the arm grasping the bowl with apple and chocolate on the table, **(right)** bottle in arm with glass and chocolate on table; **(left)** sponge in arm with chocolate on the table. Note that these observations occur across independent trajectories and are unlabeled.

7 Discussion

We proposed Value Function Spaces as state abstractions: a novel skill-centric representation that captures the *affordances* of the low-level skills. States are encoded into representations that are *invariant* to exogenous factors that do not affect values of these skills. We show that this representation is compatible with both model-free and model-based policies for hierarchical control, and demonstrate significantly improved performance both in terms of successfully performing long-horizon tasks and in terms of zero-shot generalization to novel environments, which leverages the invariances that are baked into our representation.

The focus of our work is entirely on *utilizing* a pre-specified set of skills, and we do not address how such skills are learned. Improving the low-level skills jointly with the high-level policy could lead to even better performance on particularly complex tasks, and would be an exciting direction for future work. More broadly, since our method connects skills directly to state representations, it could be used to turn unsupervised skill discovery methods directly into unsupervised representation learning methods, which could be an exciting path toward more general approaches that retain the invariances and generalization benefits of our method.

REPRODUCIBILITY STATEMENT

The primary contribution of our work, VFS, is a skill-centric representation designed to work with existing model-based and model-free pipelines. The implementation simply involves concatenating the value functions corresponding to the available skills into an embedding vector that can be used for HRL or planning. We provide all necessary information for setting up VFS to work with a reader's RL algorithm of choice in Sections 5.1, 6.1 and Appendix A. Our model-free experiments are conducted on an open-source gym environment and we provide configuration details for setting up our quantitative experiments in Appendix A.1. We hope that this encourages the community to utilize and build upon the ideas presented in the paper. We plan to release the code along with information about the proprietary environments used in a public release of this article upon publication.

REFERENCES

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=qda7-sVg84.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/453fadbd8a1a3af50a9df4df899537b5-Paper.pdf.

Ershad Banijamali, Rui Shu, Mohammad Ghavamzadeh, Hung Hai Bui, and Ali Ghodsi. Robust locally-linear controllable embedding. In *AISTATS*, 2018.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic markov decision processes. In *AAAI*, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. *CoRR*, 2020. URL https://arxiv.org/abs/2002.05709.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym, 2018. URL https://github.com/maximecb/gym-minigrid.

Petros Christodoulou. Soft actor-critic for discrete action settings. *CoRR*, 2019. URL http://arxiv.org/abs/1910.07207.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

Dane Corneil, Wulfram Gerstner, and Johanni Brea. Efficient model-based deep reinforcement learning with variational state tabulation. In *Proceedings of the 35th International Conference on Machine Learning*, 2018. URL https://proceedings.mlr.press/v80/corneil18a.html.

Christian Daniel, Gerhard Neumann, Oliver Kroemer, and Jan Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 2016. URL http://jmlr.org/papers/v17/15-188.html.

Thomas G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Int. Res.*, 2000.

Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *Annual Conference on Robot Learning*, Proceedings of Machine Learning Research, 2017.

Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/5c48ff18e0a47baaf81d8b8ea51eec92-Paper.pdf.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1oK8aoxe.

Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. Multi-level discovery of deep options. *CoRR*, 2017. URL http://arxiv.org/abs/1703.08294.

Dibya Ghosh, Abhishek Gupta, and Sergey Levine. Learning actionable representations with goal conditioned policies. In *ICLR*, 2019. URL https://openreview.net/forum?id=Hye9lnCct7.

James J. Gibson. The theory of affordances. In *Perceiving, acting, and knowing: toward an ecological psychology*. 1977. URL https://hal.archives-ouvertes.fr/hal-00692033.

Nakul Gopalan, Marie desJardins, Michael L. Littman, James MacGlashan, S. Squire, Stefanie Tellex, John Winder, and Lawson L. S. Wong. Planning with abstract markov decision processes. In *ICAPS*, 2017.

Daniel Graves, Johannes Günther, and Jun Luo. Affordance as general value function: A computational model. *CoRR*, 2020. URL https://arxiv.org/abs/2010.14289.

Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aäron van den Oord. *Shaping Belief States with Generative Environment Models for RL*. 2019.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*. URL https://papers.nips.cc/paper/7512-recurrent-world-models-facilitate-policy-evolution.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), 2018. URL https://ojs.aaai.org/index.php/AAAI/article/view/11796.

Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher P. Burgess, Alexander Pritzel, Matthew M. Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *ArXiv*, abs/1707.08475, 2017.

Manfred Huber and Roderic A. Grupen. Learning robot control - using control policies as abstract actions. In *In NIPS'98 Workshop: Abstraction and Hierarchy in Reinforcement Learning*, 1998.

Brian Ichter and Marco Pavone. Robot motion planning in learned latent spaces. *IEEE Robotics and Automation Letters*, 2019.

Brian Ichter, Pierre Sermanet, and Corey Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. In *Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=yhy25u-DrjR.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJ6yPD5xg.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv.

Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *CoRR*, 2021. URL https://arxiv.org/abs/2104.08212.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, 2006.

George Konidaris and Andrew Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems*, 2009.

George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Constructing symbolic representations for high-level planning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014. URL https://ojs.aaai.org/index.php/AAAI/article/view/9004.

Tejas D. Kulkarni, Karthik R. Narasimhan, Ardavan Saeedi, and Joshua B. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

S. Lange, Martin A. Riedmiller, and Arne Voigtländer. Autonomous reinforcement learning on raw visual input data in a real world application. *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 2012.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *Advances in Neural Information Processing Systems*, 2019. URL http://arxiv.org/abs/1907.00953.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Kendall Lowrey, Aravind Rajeswaran, Sham Kakade, Emanuel Todorov, and Igor Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byey7n05FQ.

Priyanka Mandikal and Kristen Grauman. Learning dexterous grasping with object-centric visual affordances, 2021.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015. URL http://dx.doi.org/10.1038/nature14236.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016. URL https://proceedings.mlr.press/v48/mniha16.html.

Ofir Nachum, Haoran Tang, Xingyu Lu, Shixiang Gu, Honglak Lee, and Sergey Levine. Why does hierarchy (sometimes) work so well in reinforcement learning?, 2019.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 2018.

Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/ffbd6cbb019a1413183c8d08f2929307-Paper.pdf.

Doina Precup. Temporal abstraction in reinforcement learning, 2000.

Anil V. Rao. A survey of numerical methods for optimal control. 2009.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. Semi-parametric topological memory for navigation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SygwwGbRW.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. URL https://proceedings.mlr.press/v37/schaul15.html.

Jürgen Schmidhuber. Making the world differentiable: On using self-supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical report, 1990.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

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

Freek Stulp and Stefan Schaal. Hierarchical reinforcement learning with movement primitives. In *2011 11th IEEE-RAS International Conference on Humanoid Robots*, 2011.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 1999. URL https://www.sciencedirect.com/science/article/pii/S0004370299000521.

Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J. Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Philip S. Thomas and Andrew G. Barto. Motor primitive discovery. In *2012 IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, 2012.

Emre Ugur, Erol Şahin, and Ethan Oztop. Predicting future object states using learned affordances. 2009. URL https://hdl.handle.net/11511/55960.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, 2018. URL http://arxiv.org/abs/1807.03748.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

David Warde-Farley, Tom Van de Wiele, Tejas D. Kulkarni, Catalin Ionescu, Steven Hansen, and Volodymyr Mnih. Unsupervised control through non-parametric discriminative rewards. *CoRR*, 2018. URL http://arxiv.org/abs/1811.11359.

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems*, 2015.

Grady Williams, Andrew Aldrich, and Evangelos A. Theodorou. Model predictive path integral control using covariance variable importance sampling. *CoRR*, 2015. URL http://arxiv.org/abs/1509.01149.

Danfei Xu, Ajay Mandlekar, Roberto Martín-Martín, Yuke Zhu, Silvio Savarese, and Li Fei-Fei. Deep affordance foresight: Planning through what can be done in the future, 2021.

Denis Yarats, Amy Zhang, Ilya Kostrikov, Brandon Amos, Joelle Pineau, and Rob Fergus. Improving sample efficiency in model-free reinforcement learning from images. *CoRR*, abs/1910.01741, 2019. URL http://arxiv.org/abs/1910.01741.

Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 2017. URL https://doi.org/10.1177/1059712317726357.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=-2FCwDKRREu.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=r-gPPHEjpmw.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. SOLAR: Deep structured representations for model-based reinforcement learning. In *ICML*, 2019. URL https://proceedings.mlr.press/v97/zhang19m.html.

A Implementation Details

A.1 Model-Free RL

Maze-solving environment. We run our model-free experiments using MiniGrid, a suite of open-source gym environments[1] developed by Chevalier-Boisvert et al. (2018). Specifically, we use the following registered environments:

1. *MultiRoom*
   (a) MiniGrid-MultiRoom-N2-S4-v0
   (b) MiniGrid-MultiRoom-N4-S5-v0
   (c) MiniGrid-MultiRoom-N6-v0
2. *KeyCorridor*
   (a) MiniGrid-KeyCorridorS3R3-v0
   (b) MiniGrid-KeyCorridorS6R3-v0

Additionally, we also create a modification of 1-(a) above that spans 10 rooms. We modify the base class of MiniGrid to remove partial observability, and hence, the agents receive a fully observable top-down view of the grid at all times.

Skills for Maze-Solving. We assume that the algorithms have access to the following temporally extended skills – `GoToObject`, `PickupObject`, `DropObject` and `UnlockDoor` – where the first three skills are text-conditioned and `Object` may refer to a door, key, box or circle. The skills corresponding to the key and unlocking are not valid in *MultiRoom*, but we use the same action specification for the agents; if an agent executes an invalid action, the state of the world does not change. We train the `GoTo` skills individually using the A2C (Mnih et al., 2016) implementation recommended by the developer[2]. Each of these skills are trained with a sparse outcome reward (+1 if a trajectory is successful, 0 otherwise), without any reward shaping or relabeling. The remaining skills are oracular and given to us by the MiniGrid environment. We choose $\tau = 50$ to avoid the `GoTo` action from taking too long; if it fails to reach the goal in 50 steps, the skill is considered unsuccessful. This results in a set of 13 skills of varying lengths – from 1 to 50 time steps.

Baselines. Since we have 13 skills, the proposed VFS representation is a 13-dimensional vector. For consistency across methods, we enforce all baselines to learn a 13-dimensional embedding as a state representation. We use a standard implementation of the AE and VAE baselines. For CPC, we use our own PyTorch implementation based on this widely used repository[3]. For CURL, we use the authors' official implementation[4]. The AE and CPC representations are trained offline a dataset of 10^5 observations collected by executing the skills in an open-loop manner. The HRL policy is trained to convergence for all baselines.

Hierarchical RL. As discussed in Section 5.1, we use Q-learning to approximate the Q-function of the high-level (semi) MDP and infer the policy indirectly. Our focus is not to identify the best algorithm for learning a high-level policy, and one could run this analysis with any choice of RL algorithm – actor-critic, Q-learning, policy gradient, etc. For the sake of this work, we present results using the simple DQN framework. For the results in the paper, we use a Double-DQN to avoid overestimation bias. For the experiment with raw observations, we use the convolutional network used widely in the original DQN and DDQN papers – 3 convolutional layers followed by two fully-connected layers. For the other baselines, with a 13-dimensional representation, we use four fully-connected layers. All layers use ReLu activations and we optimize the objective using RMSProp with momentum parameter 0.95.

[1] github.com/maximecb/gym-minigrid
[2] github.com/lcswillems/rl-starter-files
[3] github.com/davidtellez/contrastive-predictive-coding
[4] github.com/MishaLaskin/curl

HRL Evaluation. Note that the grid layouts, as well as object positions, in MiniGrid are randomly generated for every experiment and are not static. For quantitative evaluation of the representations (see Table 1 for results), we initialize 20 distinct grid layouts for each task and environment size. We run all baselines with 5 random seeds and report the mean success rate over 100 experimental runs per baseline per task (a total of 3600 experiments). The generalization results (see Table 2) were also conducted under the same setting.

A.2 Model-Based RL

Dataset of interactions. To learn the dynamics function $\hat{f}$ in a supervised learning manner, we collect a dataset of trajectories in the manipulation simulator. We initialize the simulator randomly and execute the available skills in an open-loop fashion to collect tuples of transitions (s, o, s'), where executing a skill o at state s terminates in state s' at the end of the skill (up to τ time steps). This gives us a dataset of interactions which we use to train a one-step predictive model. Note that this model predicts the future states for the high-level policy. We collect a dataset of one million tuples which is used to train the model for *all* baselines.

Learned Skills. The individual skill value functions are distilled from a multitask RL policy trained with MT-Opt Kalashnikov et al. (2021). The value functions $Q_\theta(s, a)$ operate on a state space of (472, 472, 3) RGB images that are center cropped from egocentric (512, 640, 3) egocentric visual observations and an a 8D action space of end-effector translation (3D), end-effector orientation (4D), and gripper closedness (1D). The reward functions for each task are engineered ground-truth success detectors that are used as sparse rewards.

Learning the Dynamics Function. We use a neural network to learn a one-step predictive model $Z_{t+1} = \hat{f}(Z_t, o_t) \,\forall\, o_t \in O$ that can approximate the transition dynamics of the skills. For all non-image baselines, we use a fully-connected network with 3 hidden layers, each of dimension 500. For the raw image baseline, we use a MobileNetv2 encoder (Sandler et al., 2018) followed by two fully-connected layers. All layers use ReLu activations and we optimize the objective using the Adam optimizer. For random shooting, we sample $K = 50$ actions and use controller horizons as described in Section 6.2.

Evaluation. We evaluate the representations on a set of 20 randomly initialized start-goal configurations in the robotic manipulation task. Common semantic goals for this task include "cluster all objects around the bowl", "divide the objects into two factions", "move objects to the top-left corner" and so on. A trajectory is successful if all semantic/proximity relations are valid; two objects are considered "close" if they are less than 5cm apart.