SPRINT: SCALABLE SEMANTIC POLICY PRE-TRAINING VIA LANGUAGE INSTRUCTION RELABELING

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose SPRINT, an approach for scalable offline policy pre-training based on natural language instructions. SPRINT pre-trains an agent's policy to execute a diverse set of semantically meaningful skills that it can leverage to learn new tasks faster. Prior work on offline pre-training required tedious manual definition of pre-training tasks or learned semantically meaningless skills via random goal-reaching. Instead, our approach SPRINT (**S**calable **P**re-training via **R**elabeling Language **IN**s**T**ructions) leverages natural language instruction labels on offline agent experience, collected at scale (*e.g.*, via crowd-sourcing), to define a rich set of tasks with minimal human effort. Furthermore, by using natural language to define tasks, SPRINT can use pre-trained large language models to automatically expand the initial task set. By relabeling and aggregating task instructions, even *across* multiple training trajectories, we can learn a large set of new skills during pre-training. In experiments using a realistic household simulator, we show that agents pre-trained with SPRINT learn new long-horizon household tasks substantially faster than with previous pre-training approaches.

1 INTRODUCTION

When humans learn a new task, *e.g.*, how to cook a new dish, we rely on a large repertoire of previously learned *skills*, like "*chopping vegetables*" or "*boiling pasta*", that make learning more efficient. Improving learning efficiency is crucial for practical deployment of artificial agents; thus, many works in reinforcement learning (RL) aim to equip agents with a similar set of skills. To autonomously acquire such skills, recent works optimize for diverse agent behaviors (Eysenbach et al., 2019; Sharma et al., 2020; Mendonca et al., 2021), imitate short action sequences (Lynch et al., 2020; Pertsch et al., 2020), or reach randomly sampled goal states (Chebotar et al., 2021) from pre-collected experience. However, such objectives may result in the agent learning skills that are not semantically plausible in practice, *e.g.*, "*placing a knife in the microwave*" or "*half-closing the microwave door*." To focus pre-training on *plausible* skills, one could instead manually curate a set of pre-training tasks for the policy, but this requires tedious reward function design and does not scale well beyond a few dozen tasks (Yu et al., 2019). Yet, defining a large set of pre-training tasks is crucial: only a policy with a wide range of skills can accelerate learning on *many* downstream tasks. How can we define a *large* set of meaningful pre-training tasks in a scalable manner?

In this paper, we propose to leverage *natural language instructions* to define a large number of semantically meaningful tasks for policy pre-training. Natural language has recently been used to allow humans to effectively interact with agents (Lynch & Sermanet, 2021) or to generate long-horizon plans (Ahn et al., 2022). In the context of defining pre-training tasks, using natural language has two important benefits: (1) language is a natural and expressive interface for humans to specify tasks (in contrast to, *e.g.*, numerical reward functions) as it is the primary way to communicate tasks in our everyday lives. Thus, even non-experts can define tasks easily via language instructions. (2) By specifying pre-training tasks via natural language, we can leverage the knowledge captured in large language models to automatically generate more tasks through instruction relabeling.

To combine both benefits we introduce SPRINT (**S**calable **P**re-training via **R**elabeling Language **IN**s**T**ructions), a scalable pre-training approach that equips policies with a repertoire of seman-

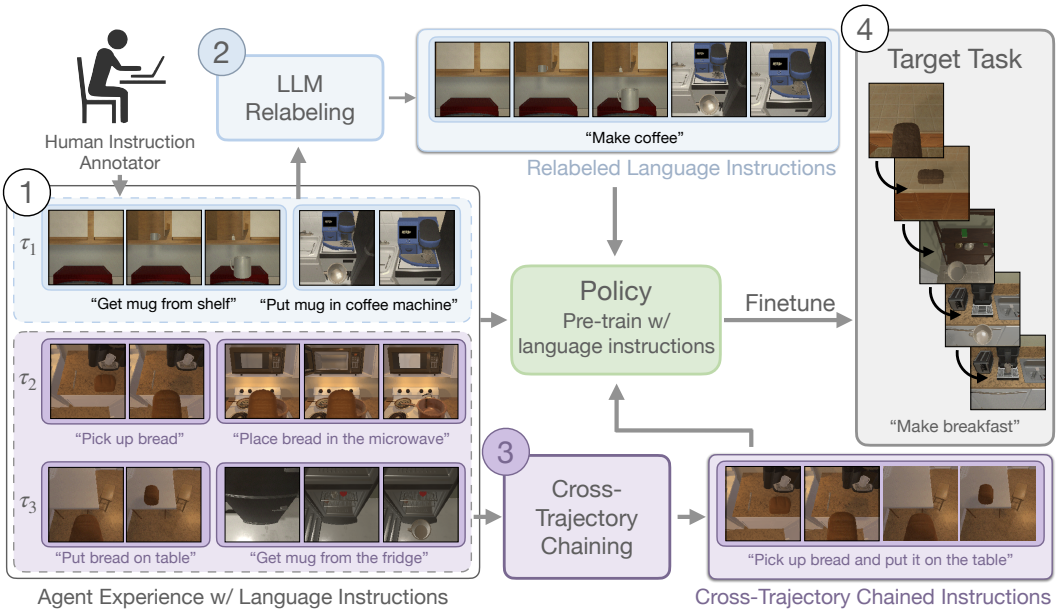

Figure 1: We propose SPRINT, a scalable approach for policy pre-training with semantic skills. We assume access to an offline dataset of agent experience with natural language instruction labels of the performed skills, *e.g.*, provided by human annotators (1). We use the instructions to pre-train a semantic skill policy via instruction-conditioned offline RL. To increase pre-training task diversity, we automatically generate new instructions via (2) language-model-based instruction relabeling, and (3) cross-trajectory skill chaining. We demonstrate that an agent pre-trained with SPRINT can leverage the diverse set of learned semantic skills to finetune efficiently on unseen target tasks (4).

tically meaningful skills (see Figure 1 for an illustration). SPRINT has three core components: (1) language-conditioned offline RL, (2) LLM-based skill aggregation and (3) cross-trajectory skill chaining. SPRINT assumes access to an offline dataset of state-action trajectories, each of which performs one or more *skills*. We assume that the data has corresponding natural language instruction labels for the performed skills, such as "*place mug in coffee machine*" or "*press brew button*". Such labels can be crowd-sourced from non-expert human annotators at scale. We use these annotations as task instructions and train a policy with language-conditioned offline RL to solve them. Crucially, SPRINT uses two techniques to expand this initial task set: firstly, we use a pre-trained large language model to relabel the language instructions, thereby creating new tasks. For example, the tasks "*place mug in coffee machine*" and "*press brew button*" can be combined into a new task: "*make coffee*." Secondly, we chain behaviors *across* multiple trajectories from the training data; starting with a skill like "*pick up bread*" from one trajectory and ending with "*place bread on table*" from another. This allows the policy to learn semantic skills completely unseen in the training data.

SPRINT trains a policy on the *combined* set of task instructions, thereby equipping the agent with a policy that can execute a wide range of semantically meaningful skills. Our experiments demonstrate that this allows for substantially more sample-efficient learning of new downstream tasks like "*prepare breakfast*" than prior pre-training approaches.

In summary, our contributions are threefold: (1) we propose SPRINT, which leverages natural language instructions for scalable policy pre-training via instruction-conditioned offline RL, (2) we expand the set of pre-training tasks via LLM-based skill relabeling and cross-trajectory chaining, (3) we demonstrate that SPRINT enables agents to more efficiently learn long-horizon household tasks in the ALFRED simulator (Shridhar et al., 2020) than prior pre-training approaches.

2 Related Work

Language in RL. There is a long-standing interest in leveraging natural language during behavior learning, *e.g.*, to structure agent's internal representations (Andreas et al., 2017b), to learn to interact

with text-based games (Narasimhan et al., 2015; Küttler et al., 2020), or guide long-horizon task learning via recipe-like plans (Branavan et al., 2009; Andreas et al., 2017a). The recent progress in training large, general-purpose language models has enabled approaches that directly generate and execute such plans solely from a high-level task description (Huang et al., 2022a; Ahn et al., 2022; Huang et al., 2022b). Others have shown how language in combination with vision inputs can be used to learn state representations (Nair et al., 2022) or reward functions (Fan et al., 2022). Finally, there is a rich body of work that explores using language as an intuitive agent interface for humans: given a free-form natural language command they aim to train language-conditioned policies that can execute the instruction in the environment (Lynch & Sermanet, 2021). In contrast, our work aims to leverage language for task-agnostic *agent pre-training*. Instead of training a policy to execute human language commands, we aim to equip an agent with a set of semantic skills that improves the efficiency of reinforcement learning on downstream tasks.

Pre-training Policies for RL. Developing policy pre-training approaches for faster downstream learning has been investigated for many years (Ijspeert et al., 2002; Theodorou et al., 2010; Hester et al., 2018). Recent advances in offline reinforcement learning (Levine et al., 2020) enabled approaches that can pre-train agents offline and effectively finetune them on online tasks (Peng et al., 2019; Singh et al., 2020; Nair et al., 2020; Kostrikov et al., 2022). However, these approaches require the pre-training data to have target-task reward annotations and the resulting pre-trained policies are trained only to solve the target task. In contrast, meta-RL approaches pre-train on a range of tasks and allow fast adaptation to *unseen* downstream tasks (Duan et al., 2016; Finn et al., 2017; Rakelly et al., 2019; Nam et al., 2022), yet require tedious manual definition of pre-training tasks by experts, making them less scalable. To avoid manual task design, other works have explored unsupervised pre-training approaches based on behavior diversification (Achiam et al., 2018; Eysenbach et al., 2019; Sharma et al., 2019) or extraction of behavior priors from offline agent experience (Pertsch et al., 2020; Ajay et al., 2020; Singh et al., 2021). Closest to ours is the approach of Chebotar et al. (2021), which performs unsupervised pre-training by learning to reach randomly sampled states from offline agent experience. Yet, unsupervised pre-training approaches learn semantically meaningless skills which, as we demonstrate in Section 5, lead to worse transfer to downstream tasks. In contrast, we introduce a *scalable* pre-training approach based on natural language instructions that equips agents with *semantically meaningful* skills and allows for efficient transfer to unseen tasks.

3 PRELIMINARIES

Offline RL. Offline reinforcement learning (RL) methods assume access to datasets of trajectories τ composed of ordered (s, a, s', r) tuples of current state, action, next state, and reward labels, from a Markov Decision Process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$. $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$ represents the transition probability distribution. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ denotes the reward function, and γ is the discount factor. These methods learn offline, with no environment interactions, and aim to train a policy $\pi(a|s)$ that maximizes the reward per episode. Temporal-difference (TD) learning methods also train a critic, $Q^\pi(s, a)$, representing the discounted future sum of rewards using policy π after taking action a from the state s (Sutton & Barto, 2018).

Goal-conditioned RL. Goal-conditioned RL algorithms condition policies and critics on a goal $g \in \mathcal{G}$, where $\mathcal{G}$ is the space of valid goals, typically the state space $\mathcal{S}$. The policy receives a reward upon reaching the goal, *e.g.*, a desired state. This formulation enables the design of *unsupervised* RL approaches, *i.e.*, methods that do not require manual task definition. These methods define tasks by sampling random goal states, *e.g.*, from an offline dataset of agent experience, and simply assign a sparse reward of 1 upon reaching the goal (Kaelbling, 1993; Schaul et al., 2015; Chebotar et al., 2021; Yang et al., 2022). In Section 4.1 we will show how to extend this formulation to language instruction goals to enable scalable pre-training on a wide range of semantic tasks.

4 SPRINT: SCALABLE SEMANTIC POLICY PRE-TRAINING WITH LANGUAGE INSTRUCTIONS

In this work, we propose SPRINT (**S**calable **P**re-training via **R**elabeling Language **IN**s**T**ructions), an approach for pre-training a policy to solve a wide range of semantic skills in order to enable efficient

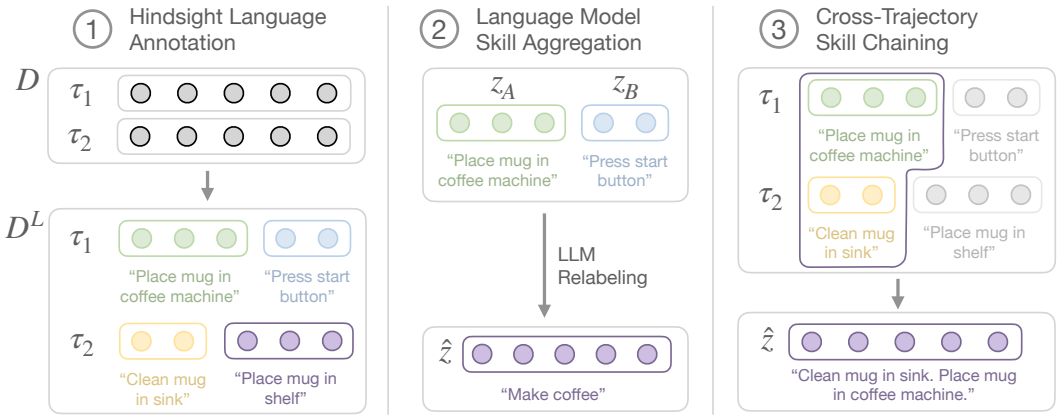

Figure 2: Overview of our approach SPRINT. **Left**: Hindsight annotation of agent experience with language instructions is a scalable approach for defining pre-training tasks. We pre-train via instruction-conditioned offline RL with a sparse goal-reaching reward (Section 4.1). **Middle**: We expand the set of pre-training tasks by aggregating language instructions with an LLM and adding the relabeled trajectories back into the pre-training dataset (Section 4.2). **Right**: We perform cross-trajectory chaining of skills to enable pre-training of skills that are unseen in the offline agent experience (Section 4.3).

finetuning on unseen tasks. SPRINT has three core technical components which we will describe in this section: (1) offline instruction-conditioned RL with crowd-sourced language instructions, (2) language-model-based skill aggregation and (3) cross-trajectory skill chaining. But first, we will detail the data we use for pre-training and give an intuitive overview of our approach.

Pre-training Data. Following prior work on agent pre-training, we assume access to a large offline dataset $\mathcal{D}$ of agent experience (Gupta et al., 2019; Lynch et al., 2020; Pertsch et al., 2020; Chebotar et al., 2021; Ebert et al., 2022; Pertsch et al., 2021). Such data can be collected at scale, *e.g.*, from prior RL runs, via teleoperation, through autonomous agent exploration or any combination thereof. We further assume that the data contains natural language descriptions of the skills performed by the agent, *e.g.*, "*put a mug in the coffee machine*" or "*push the brew button*." Such descriptions can be collected at scale by non-experts by annotating sequences from our offline dataset $\mathcal{D}$ *in hindsight*, *e.g.*, via crowd-sourcing through platforms like Amazon Mechanical Turk (Lynch & Sermanet, 2021; Shridhar et al., 2020). Given a randomly sampled sequence τ from the dataset $\mathcal{D}$, annotators can label sub-trajectories $\tau_1 = [s_0, a_0, s_1, \dots], \tau_2 = \dots$ with free-form language descriptions $z_1, z_2, \dots$ of the tasks performed in the respective sub-trajectories, for example:

$$\tau_{\bar{z}} = [\tau_{z_1}, \tau_{z_2}] = [\underbrace{[(s_0^{z_1}, a_0^{z_1}, z_1), ..., (s_T^{z_1}, a_T^{z_1}, z_1)]}_{\text{"Put a mug in the coffee machine"}}, \underbrace{[(s_0^{z_2}, a_0^{z_2}, z_2), ...]}_{\text{"Push the brew button"}}]. \tag{1}$$

This results in a dataset $\mathcal{D}^L = \{\tau_{\bar{z}_1}, \tau_{\bar{z}_2}, \dots\}$ of diverse trajectories with natural language descriptions (see Figure 2, left).

Approach Overview. Our approach leverages skill descriptions in $\mathcal{D}^L$ as instructions for the policy during pre-training: each description defines a task and the policy is rewarded for successfully executing the instruction. Intuitively, the more diverse the set of language instructions during pre-training, the more semantic skills the policy will learn and the more downstream tasks can it finetune on efficiently. Thus, SPRINT introduces two approaches for increasing the diversity of the pre-training instructions without additional human inputs. Firstly, SPRINT leverages pre-trained language models to aggregate the human annotated instructions (Figure 2, middle). Secondly, we propose an approach for cross-trajectory skill-chaining, that allows us to combine behaviors across multiple training trajectories to generate completely unseen instructions (Figure 2, right). Both approaches expand the set of pre-training tasks, leading to more effective pre-training.

Once the policy is trained, we can transfer it to a new task for finetuning, either by providing a language description z_{target} of the target task or by replacing the language instruction input with a vector

of trainable parameters and finetuning it jointly on the target task, similar to prior work (Chebotar et al., 2021; Laskin et al., 2021).

4.1 INSTRUCTION-CONDITIONED OFFLINE RL

To pre-train our policy with the natural language instruction dataset $\mathcal{D}^L$, we take inspiration from goal-conditioned RL approaches (introduced in Section 3): instead of conditioning the policy on goal states and rewarding it for reaching these states, we condition our policy $\pi(a|s, z)$ on *language instructions* z from $\mathcal{D}^L$ and provide a sparse reward to the agent for reaching the end-state s_T of the respective sub-trajectory. Formally, we define the reward as:

$$R(s, a, z) = \begin{cases} 1, & \text{for } s = s_T^z \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

We can optimize the policy $\pi(a|s, z)$ and critic $Q(s, a, z)$ on this reward function using any offline RL algorithm. In this work, we use Implicit Q-Learning (Kostrikov et al., 2022) since we found it to be performant and easy to tune.

4.2 LANGUAGE-MODEL-BASED INSTRUCTION AGGREGATION

LLM Prompt Example

Summarize the following steps.

1: Pick up the knife on the counter.
2: Slice the tomato with the knife.
Summary: Pick up a knife and slice the tomato.

1: [SKILL 1]
2: [SKILL 2]
...
Summary:

Figure 3: A shortened example of the LLM prompt. Full prompt in Section B.

Large language models (LLMs), trained on massive corpora of internet text data, have been shown to be effective at performing a variety of tasks – from question answering to program synthesis – when prompted with relevant text (Devlin et al., 2018; Reimers & Gurevych, 2019; Brown et al., 2020; Wang & Komatsuzaki, 2021; Rae et al., 2021; Hoffmann et al., 2022; Zhang et al., 2022; Chowdhery et al., 2022). Here we use LLMs to *aggregate*, *i.e.*, paraphrase, the existing language annotations in $\mathcal{D}^L$ (see Figure 2, middle, for an illustration). Given a trajectory that contains multiple sub-trajectories, we can aggregate adjacent sub-trajectories into a longer trajectory and relabel its natural language annotation with a summary of the individual instructions generated by the LLM, thereby generating a new *higher-level* pre-training task that encompasses instructions from multiple sub-trajectories.[1] We use a simple summarization prompt to instruct the language model (see Figure 3 for an example, Section B for the full prompt used). Specifically, we aggregate with OPT-13B (Zhang et al., 2022), an open-source 13 billion parameter large language model comparable with the second-largest GPT-3 text completion model, `text-curie-001` (Brown et al., 2020).[2] Like before, the reward for this new aggregated sub-trajectory is 1 at the last transition and 0 otherwise. Taking Eq. 1 as an example, we prompt the language model to summarize the two skills (z_1 : "*Put a mug in the coffee machine*," z_2 : "*Push the brew button*"), resulting in a new semantic annotation $\hat{z}_{1:2}$ describing both skills (*e.g.*, "*Make coffee*"). We then add the new trajectory back to our dataset $\mathcal{D}^L$. Using this technique, we generate new language annotations for all $\binom{N}{2}$ tuples of consecutive sub-trajectories in our dataset. In practice, this allows us to increase the number of pre-training task instructions by 2.5x without additional human effort.

4.3 CROSS-TRAJECTORY CHAINING

Agents trained with offline RL can combine behaviors from multiple trajectories via value propagation, *i.e.*, "stitch" them (Levine et al., 2020). For example, if trajectory (A) shows cleaning the mug in the sink and then placing it on the shelf, while trajectory (B) starts with placing the mug in

[1] Other relabeling operations, such as splitting an instruction into multiple lower-level instructions, can also be performed by the language model. However, such operations require grounding the language model to the agent's observations to determine sub-trajectory split points. We leave investigating this to future work.

[2] See appendix, Sections E.1 and E.2 for example aggregated skills and comparisons of different LLMs.

the coffee machine, offline RL algorithms are able to learn to clean the mug in the sink and then place it in the coffee machine (see Figure 2, right). In our case of *instruction-conditioned* offline RL, enabling such stitching behavior requires special care. Due to the different language instruction conditionings for the critic $Q(s, a, z_A)$ and $Q(s, a, z_B)$, values do not naturally propagate from trajectory (B) back to trajectory (A). Instead, we must actively add "chaining examples" to our training dataset (Chebotar et al., 2021). For this, we randomly sample a sub-trajectory $\tau_{z_A} = [s_{0:t}, a_{0:t}, z_A]$ from the training dataset $\mathcal{D}^L$, *e.g.*, part of trajectory (A), and *replace* its language instruction with the instruction z_B *from another trajectory* τ_{z_B}, *e.g.*, "*place mug in coffee machine*." Note that we do not need to sample the full trajectory (A). Instead, s_t can be any state from trajectory (A) from which we try to execute skill (B). Crucially, we *cannot* use the same reward function as before: since the last state s_t of the sampled sub-trajectory, *e.g.*, a mug in the sink, does not solve the instruction z_B, "*put mug in coffee machine*," we cannot set its reward to 1. Which reward should we use instead?

Let's recap that when using temporal-difference (TD) learning (Sutton & Barto, 2018), Q functions for the sparse reward definition from Equation 2 intuitively represent a value that is proportional to the probability of reaching the goal at time T (Eysenbach et al., 2022; Chebotar et al., 2021):

$$Q^\pi(s_t, a_t, z) = \mathbb{E}\left[\sum_{t'=t} \gamma^{t'} R(s_{t'}, a_{t'}, z)\right] = \mathbb{E}\left[\gamma^{T-t} \mathbb{1}\left[s_T = g_z\right]\right] \propto P^\pi(s_T = g_z | s_t, a_t). \quad (3)$$

Similarly, we want $Q(s, a, z_B)$ to represent the probability of reaching the goal g_{z_B}, the mug being in the coffee machine, from state s. Thus, we need to set the reward of the last state s_t in the sampled sub-trajectory to the probability of reaching the goal g_{z_B} from s_t, *i.e.*, $Q(s_t, a_t, z_B)$:

$$R(s, a, z_B) = \begin{cases} Q(s, a, z_B), & \text{for } s = s_t \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Finally, we can apply the skill aggregation approach from Section 4.2 in the cross-trajectory case. Instead of sampling two sub-trajectories from the same training trajectory as in skill aggregation, we can now sample τ_{z_A} and τ_{z_B} from *different* training trajectories and chain them together. The aggregate instruction $\hat{z}$ implies that the agent first finishes skill (A) and then finishes skill (B), *e.g.*, "*clean the coffee mug and place it in the coffee machine*." Thus, following the logic above, we need to set the reward for the final state in trajectory (B), s_{T_B}, to 1 since it solves the task $\hat{z}$, and the reward in the final state of trajectory (A), s_{T_A}, to the probability of solving skill (B) from there:

$$R(s, a, \hat{z}) = \begin{cases} 1, & \text{for } s = s_{T_B} \\ Q(s, a, z_B), & \text{for } s = s_{T_A} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

To generate the aggregate instruction $\hat{z}$ we can use the LLM summarization from Section 4.2. Yet, in practice we found the resulting summaries are often meaningless, since randomly paired instructions *from different trajectories* can rarely be summarized meaningfully. Instead, we saw better performance by simply concatenating the natural language instructions from τ_A and τ_B. Finally, since Q changes over the course of training, we compute the rewards in Eqs. 4 and 5 online during training.

5 Experiments

In our experiments, we investigate how well an agent pre-trained with SPRINT performs on unseen, semantically meaningful tasks. Specifically, we are interested in answering the following questions: (1) Does pre-training with SPRINT lead to more efficient finetuning on unseen target tasks than previous pre-training approaches? (2) Can agents pre-trained with SPRINT execute unseen language instructions zero-shot? (3) Does pre-training with *semantic* skills lead to better generalization to unseen environments than unsupervised pre-training approaches?

5.1 Experimental Setup

ALFRED Benchmark. We choose the ALFRED benchmark (Shridhar et al., 2020) for evaluation (see Figure 4), since it allows us to test our pre-trained agents on *semantically meaningful*, long-horizon tasks in a realistic household setting. Additionally, ALFRED provides a dataset of 6.6k

episodes of offline agent experience with crowd-sourced natural language instruction annotations (*e.g.*, "*pick up the dirty mug*", "*wash it in the sink*", ..., see Figure 4). The observation space in ALFRED consists of 300×300 egocentric RGB observations and the action space consists of 12 discrete action choices (e.g. *turn left*, *look up*, *pick up object*), along with 82 discrete object types (Pashevich et al., 2021). We leverage 73k natural language instructions and their associated observation-action trajectories from the dataset for pre-training. For more environment and dataset details, see Section D.

Evaluation Tasks. We create a set of 100 unseen long-horizon evaluation instructions (*EVAL_100*), by sampling sequences of 1 to 7 instructions from the ALFRED training dataset and asking human annotators to provide an instruction that summarizes the sequence (see appendix, Section D.2 for examples). Additionally, we create a set of 20 evaluation commands that test the agent's chaining capabilities (*EVAL_CHAIN*): we withhold trajectories and instruction summaries of sequences with 7 to 8 instructions from the ALFRED dataset, making sure that the respective skill chains are *not* present in any single training trajectory. Thus, the agent needs to learn to chain behaviors *across multiple trajectories* to solve these tasks. Finally, to test environment generalization capabilities, we also create a set of 10 tasks in household floor plans not seen in any trajectory in the training dataset (*EVAL_UNSEEN*), consisting in equal parts of 1 to 5-instruction sequences.

Figure 4: ALFRED. Figure drawn from Shridhar et al. (2020), with permission.

Comparisons. We compare SPRINT against the most common prior policy pre-training approaches: behavioral cloning and offline goal-conditioned RL. Specifically, we compare to the following prior works:

- **Language-conditioned BC (L-BC)** (Jang et al., 2021; Lynch & Sermanet, 2021): Behavior cloning (BC) conditioned on the ALFRED language instructions.
- **Actionable Models (AM)** (Chebotar et al., 2021): Goal-conditioned offline RL with randomly sampled goal observations from the ALFRED training set

We implement all methods with the same architecture and shared training parameters wherever possible; for more baseline and implementation details, see Section C. All results reported are means and standard deviations computed over 3 seeds.

5.1.1 SPRINT Solves Long-Horizon Tasks Zero-Shot

We first test the effectiveness of SPRINT's pre-training by analyzing zero-shot performance across the 100 tasks in the *EVAL_100* evaluation set. We report results in Figure 5 (left). Our approach, SPRINT, achieves 3x higher zero-shot task performance than prior pre-training approaches AM and L-BC. To better understand the differences between the methods, we report the breakdown of returns by length of the evaluation task in Figure 5, right. We find that L-BC achieves good performance on short-horizon tasks. However, on long-horizon tasks, SPRINT achieves much higher returns, since it can leverage the language-model to automatically generate longer-horizon pre-training tasks. In contrast, standard L-BC approaches train only on the human-provided, shorter-horizon annotations and thus cannot zero-shot perform long-horizon tasks. This trend holds even when evaluating with detailed step-by-step instructions (see appendix Section E.3). Similar to our approach, AM trains to reach long-horizon goals during pre-training but the results in Figure 5, right, show that AM's pre-training with goal-state conditioning is *less* effective than our language-conditioned pre-training.

To evaluate the pre-training approaches' ability to perform *unseen* skill sequences, we evaluate zero-shot performance on the *EVAL_CHAIN* task set in Figure 5. Unsurprisingly, both L-BC and AM largely fail to generate returns on these long-horizon tasks. In contrast, SPRINT can solve unseen long-horizon tasks with up to 8 subtasks (see Figure 7 for an example trajectory, appendix

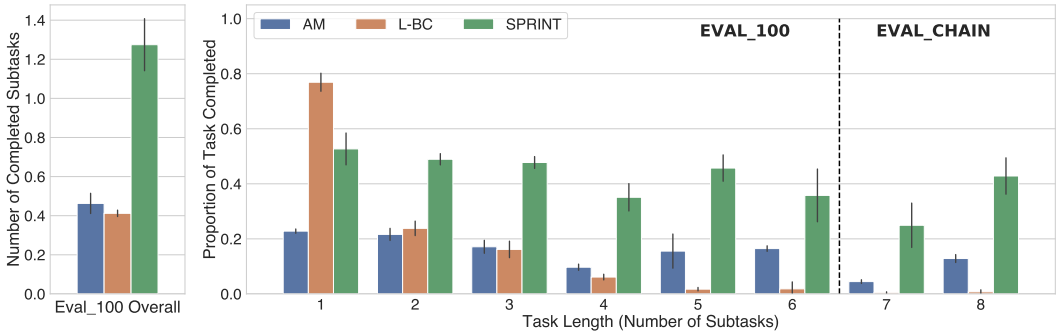

Figure 5: Zero shot evaluation performance on the *EVAL_100* and *EVAL_CHAIN* task sets. The left figure plots the overall number of completed subtasks on *EVAL_100* (*i.e.*, average return), with our method in green. The right figure plots the proportion of task completed (number of subtasks successfully executed divided by task length), split by task length, in both the task sets. See appendix, Table 3 for exact numerical results.

Section E.5 for more qualitative comparisons). This shows that, with the help of cross-trajectory chaining, SPRINT is able to go beyond the skills in the training data and execute unseen tasks.

5.1.2 SPRINT Agents Finetune Effectively in Unseen Environments

To test downstream task performance, we finetune the pre-trained agents on the *EVAL_UNSEEN* task set in unseen household floor plans. To implement finetuning for SPRINT and AM, we condition the policy on a language instruction or goal image from the target task respectively and then run IQL with online data collection. For L-BC, we first pre-train a language-conditioned Q-function with IQL on the pre-training dataset and then finetune policy and Q-function with online IQL.

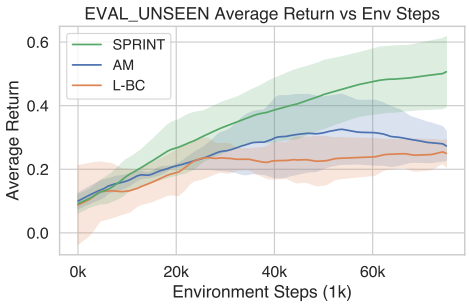

Figure 6: Finetuning results on floorplans not seen during pre-training.

We report finetuning results in Figure 6, with qualitative examples in appendix, Section E.5. Our approach, SPRINT, is able to achieve 2x higher downstream task return than the best prior work. Specifically, we find L-BC converges quickly to a low return as it is not pre-trained for longer horizon tasks. Meanwhile, AM struggles to transfer the learned skills to the new environment, possibly since the goal states from the new environment are unseen. In contrast, our method's pre-training with language conditioning allows for effective transfer even to unseen environments since the semantics of the tasks transfer well: the language description "*place cup in coffee machine*" transfers to many environments while the goal image for the same task might look very different. Thus, pre-training with semantic language instructions can enable better transfer for learning tasks in new environments than pre-training to reach goal states.

5.1.3 Ablation Studies

We verify the effectiveness of the components of our approach, with the following ablations:

- **SPRINT w/o chain**: removes cross-trajectory chaining (Section 4.3), instead trains only on within-trajectory human-provided and LLM-aggregated tasks
- **SPRINT w/o LLM-agg**: additionally removes the LLM aggregation (Section 4.2), thus trains offline RL agent only on the human-provided task annotations.

We report zero-shot evaluation results in Table 1. The results show the importance of the different components of our approach. Without cross-trajectory chaining, performance drops by ∼20-30%. Interestingly, we find that performance also degrades on the *EVAL_100* task set, indicating that cross-trajectory chaining can improve performance even on long-horizon tasks in the training set.

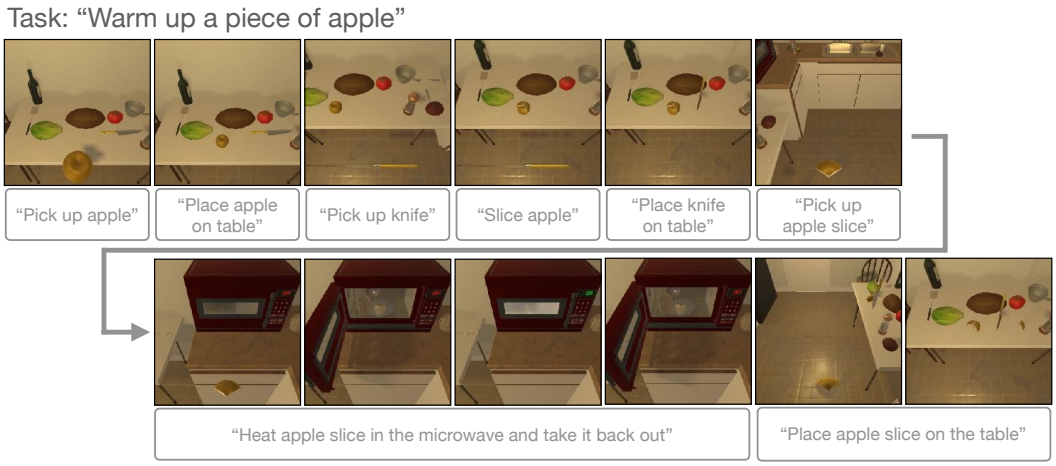

Figure 7: Example task execution of our pre-trained SPRINT agent for the task "*Warm up a piece of apple*". Successful execution of this task requires solving 8 subtasks in sequence and a total of 50 steps. This exact sequence of subtasks was never observed in the training data. SPRINT leverages cross-trajectory stitching and LLM aggregation to learn how to execute unseen tasks.

Table 1: *EVAL_100* and *EVAL_CHAIN* returns.

	EVAL_100	*EVAL_CHAIN*
SPRINT (ours)	**1.27** $\pm$ **0.13**	**2.59** $\pm$ **0.66**
SPRINT w/o chain	0.91 $\pm$ 0.03	2.04 $\pm$ 0.04
SPRINT w/o LLM-agg	0.38 $\pm$ 0.05	0.10 $\pm$ 0.04

We observe an even larger performance loss when removing the language model aggregation of pre-training tasks: the resulting agent is solely trained on the shorter horizon human task annotations, and thus struggles to solve long-horizon evaluation tasks, similar to the L-BC approach in Section 5.1.1. We provide the per-task-length breakdown of the ablation performances in appendix, Section E.4. This section also presents additional ablation experiments that use simple concatenation instead of LLM summarization for task aggregation, further supporting the importance of the usage of LLMs in our pre-training approach.

6 Discussion, Limitations, and Future Work

In this paper, we presented SPRINT, an approach for scalable agent pre-training with offline instruction-conditioned RL. We demonstrated how we can leverage easy-to-collect natural language instructions on offline agent experience for effective pre-training. Further, we introduced approaches for using pre-trained large language models and cross-trajectory skill chaining to automatically expand the set of pre-training tasks. In our experimental evaluations on the ALFRED household task benchmark (Shridhar et al., 2020) we demonstrated that SPRINT pre-training allows for substantially more efficient finetuning on downstream tasks in unseen environments than prior works. SPRINT represents a step towards scalable approaches for pre-training agents with a set of semantically meaningful skills.

Naturally, pre-training with semantic skills will be most effective on long-horizon, semantically meaningful tasks. Many of the common RL benchmarks like MetaWorld (Yu et al., 2019) and D4RL (Fu et al., 2020) do not evaluate agents on semantically meaningful or long-horizon tasks and thus we do not expect our method to show significant pre-training benefits there. However, with the recent trend towards evaluating agents on more realistic, semantically meaningful tasks, like BEHAVIOR (Srivastava et al., 2022) or MineDojo (Fan et al., 2022), we believe semantic pre-training approaches like SPRINT will be beneficial. Another limitation of our current approach is that SPRINT can *only* leverage language-annotated data. While we can collect such annotations at scale via crowd-sourcing, there will always be a lot of agent experience data *without* language annotations. Investigating approaches that can *combine* language-annotated and un-annotated data for effective pre-training is an interesting direction for future work.

REFERENCES

Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv*, 2018.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

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

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, 2017a.

Jacob Andreas, Dan Klein, and Sergey Levine. Learning with latent language. In *North American Chapter of the Association for Computational Linguistics*, 2017b.

Satchuthananthavale RK Branavan, Harr Chen, Luke Zettlemoyer, and Regina Barzilay. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 82–90, 2009.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

Yevgen Chebotar, Karol Hausman, Yao Lu, Ted Xiao, Dmitry Kalashnikov, Jacob Varley, Alex Irpan, Benjamin Eysenbach, Ryan C Julian, Chelsea Finn, and Sergey Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 1518–1528. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/chebotar21a.html`.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL `https://arxiv.org/abs/2204.02311`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL https://arxiv.org/abs/1810.04805.

Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL ^{2}: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *RSS*, 2022.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *ICLR*, 2019.

Benjamin Eysenbach, Tianjun Zhang, Ruslan Salakhutdinov, and Sergey Levine. Contrastive learning as goal-conditioned reinforcement learning, 2022. URL https://arxiv.org/abs/2206.07568.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *arXiv preprint arXiv: Arxiv-2206.08853*, 2022.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*, 2017.

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

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRL*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Association for the Advancement of Artificial Intelligence*, 2018.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022. URL https://arxiv.org/abs/2203.15556.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022b.

Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pp. 1398–1403. IEEE, 2002.

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.

Leslie Pack Kaelbling. Learning to achieve goals. In *IN PROC. OF IJCAI-93*, pp. 1094–1098. Morgan Kaufmann, 1993.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=68n2s9ZJWF8.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment, 2020. URL https://arxiv.org/abs/2006.13760.

Michael Laskin, Denis Yarats, Hao Liu, Kimin Lee, Albert Zhan, Kevin Lu, Catherine Cang, Lerrel Pinto, and Pieter Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *NeurIPS*, 2021.

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

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2020.

Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models, 2021.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation, 2022. URL https://arxiv.org/abs/2203.12601.

Taewook Nam, Shao-Hua Sun, Karl Pertsch, Sung Ju Hwang, and Joseph J. Lim. Skill-based meta-reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022.

Karthik Narasimhan, Tejas Kulkarni, and Regina Barzilay. Language understanding for text-based games using deep reinforcement learning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1–11, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1001. URL https://aclanthology.org/D15-1001.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. Episodic Transformer for Vision-and-Language Navigation. In *ICCV*, 2021.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019. URL https://arxiv.org/abs/1910.00177.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer. In *Association for the Advancement of Artificial Intelligence*, 2018.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *CoRL*, 2020.

Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J. Lim. Demonstration-guided reinforcement learning with learned skills. In *CoRL*, 2021.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu,

Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher, 2021. URL https://arxiv.org/abs/2112.11446.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *ICML*, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv*, abs/1907.01657, 2019.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *ICLR*, 2020.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL https://arxiv.org/abs/1912.01734.

Avi Singh, Albert Yu, Jonathan Yang, Jesse Zhang, Aviral Kumar, and Sergey Levine. Cog: Connecting new skills to past experience with offline reinforcement learning. *Conference on Robot Learning*, november 2020.

Avi Singh, Huihan Liu, Gaoyue Zhou, Albert Yu, Nicholas Rhinehart, and Sergey Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *ICLR*, 2021.

Sanjana Srivastava, Chengshu Li, Michael Lingelbach, Roberto Martín-Martín, Fei Xia, Kent Elliott Vainio, Zheng Lian, Cem Gokmen, Shyamal Buch, Karen Liu, et al. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Conference on Robot Learning*, pp. 477–490. PMLR, 2022.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *2010 IEEE International Conference on Robotics and Automation*, pp. 2397–2403. IEEE, 2010.

Ben Wang and Aran Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

Rui Yang, Yiming Lu, Wenzhe Li, Hao Sun, Meng Fang, Yali Du, Xiu Li, Lei Han, and Chongjie Zhang. Rethinking goal-conditioned supervised learning and its connection to offline RL. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=KJztlfGPdwW.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2019.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022. URL https://arxiv.org/abs/2205.01068.