LiFT: Unsupervised Reinforcement Learning with Foundation Models as Teachers

Taewook Nam[1*] **Juyong Lee**[1*] **Jesse Zhang**[2]
Sung Ju Hwang[1] **Joseph J. Lim**[1] **Karl Pertsch**[3,4]
[1]KAIST [2]University of Southern California
[3]University of California, Berkeley [4]Stanford University
{namsan,agi.is}@kaist.ac.kr

Abstract

We propose a framework that leverages foundation models as teachers, guiding a reinforcement learning agent to acquire semantically meaningful behavior without human feedback. In our framework, the agent receives task instructions grounded in a training environment from large language models. Then, a vision-language model guides the agent in learning the multi-task language-conditioned policy by providing reward feedback. We demonstrate that our method can learn semantically meaningful skills in a challenging open-ended MineDojo environment while prior unsupervised skill discovery methods struggle. Additionally, we discuss observed challenges of using off-the-shelf foundation models as teachers and our efforts to address them.

1 Introduction

A longstanding goal of robot learning and reinforcement learning (RL) is to train general agents that can perform a wide variety of helpful tasks in many different environments. Recently, researchers have used imitation learning and RL to train such general agents [1–7], but these methods generally require costly supervision in the form of large-scale human demonstration datasets, which is hard to scale. Unsupervised RL attempts to mitigate this issue by equipping agents with diverse skills through unsupervised learning objectives. Prior work generally teaches agents skills by optimizing information-theoretic diversity measures [8–13], but the learned skills are often meaningless for downstream tasks as diversity measures do not induce *meaningful* skills.

Imagine if instead, an agent deployed in a new environment had a human teacher, e.g., a robot dropped into a hospital with medical staff teaching them to perform important hospital duties. Not only would the teacher be able to guide the agent towards meaningful tasks to learn (e.g., finding certain medicines, checking on patients, etc.), but also the teacher could give feedback about how well the agent performed the task. This is a clear intuitive way of teaching agents, but the required supervision is too costly to scale. Can we teach RL agents in this manner but *without* human supervision?

Our key insight is to use foundation models (FMs) as teachers; by leveraging the common sense knowledge captured in FMs through web-scale pre-training, the agent can learn semantically meaningful behaviors, similar to how human teachers can steer the agent's learning process. Prior works have indeed demonstrated that the knowledge captured by FMs is useful for recognizing diverse objects [14], executable tasks [15–18] and task successes [19–22]. In this work, we leverage FMs to design an unsupervised RL approach to learn complex visuomotor tasks in lieu of *any* human supervision over what tasks to learn *or* how to reward the agent for those tasks.

[*]Equal contribution.

2nd Workshop on Agent Learning in Open-Endedness (ALOE) at NeurIPS 2023.

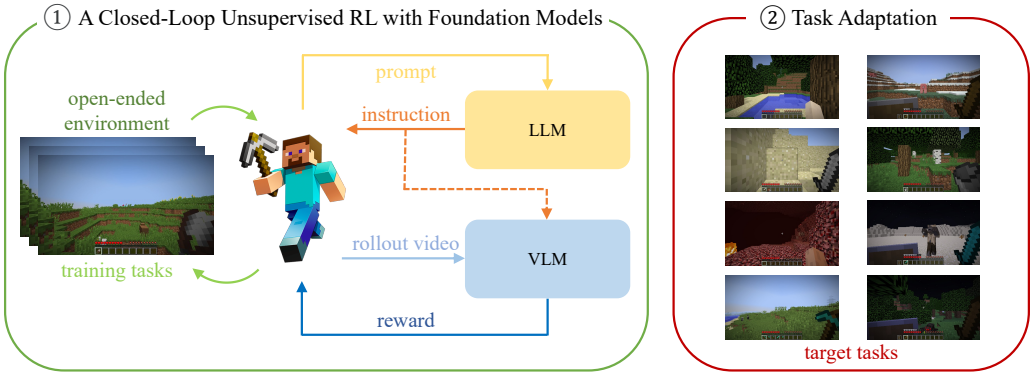


Figure 1: **LiFT overview.** Foundation models teach an agent in training environments by guiding them towards which tasks to learn (LLM) and rewarding them (VLM) for completing the tasks. The agent learns meaningful skills from the training environment and adapt to evaluation task instructions.

Specifically, we propose LiFT (unsupervised **L**earning wi**t**h **F**oundation models as **T**eachers), an unsupervised RL framework that can acquire meaningful behaviors in challenging open-ended environments *without* human supervision. Concretely, we leverage large language models (LLMs) to propose meaningful task instructions for a language-conditioned agent to attempt given a list of objects in the scene. After the agent attempts to solve the task instruction, we use a vision-language model (VLM) to give reward feedback to the agent for that task.

In our experiments, we provide two main empirical results. First, we evaluate our approach in the challenging MineDojo environment. Here, the agents need to acquire skills that are useful in the environment, such as milking a cow or shearing a sheep in a scene with a cow or sheep. We demonstrate that conventional unsupervised RL techniques fail since maximizing diversity in complex environments is not sufficient to direct exploration toward useful skills. In contrast, our approach acquires semantic guidance from foundation models and thus learns semantically meaningful skills. Second, our empirical analysis highlights the shortcomings of current VLMs for reward computation. We further discuss several strategies to improve reward quality, e.g., via policy initialization and reward post-processing. Our analysis results support that the quality of the VLM-based reward function is limiting, thereby compelling the use of pre-trained behavior priors or reward post-processing.

To summarize, our contributions are as follows:

- We present LiFT, unsupervised RL approach that leverages foundation models to guide agent training,
- We verify the efficacy of LiFT in a challenging open-ended environment and demonstrate that our approach outperforms prior unsupervised RL approaches.
- We perform extensive qualitative analysis, including learned behavior quality and limitations from foundation model components, to inform future research.

2 Problem Formulation

Our goal is to train a policy from a given environment without human-supervised reward signals, similar to the formulation of unsupervised RL [8–13]. We use foundation models (FMs), i.e., large language models (LLMs) and vision-language models (VLMs), to achieve this goal. While our FMs have received much human supervision through content on the internet created and annotated by humans, this content has been distilled into the weights of the FMs and therefore comes "for free" during the training of the RL agent.

Formally speaking, given an environment defined by reward-free Markovian decision process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability $\mathcal{P}$, and an initial state distribution ρ, we aim to pre-train a policy π, receiving the observation $o \in \mathcal{O}$ in a partially observable

MDP $\hat{\mathcal{M}} = (\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, \rho)$, which can quickly acquire target tasks representing a set of meaningful skills, sampled from unknown task distribution. In this work, we focus on zero-shot adaptation given a language instruction, thus each task also contains a language instruction δ describing the goal.

We assume that the agents have access to a text description of the interactable objects available in the environment $d(s_0)$ with $s_0 \sim \rho$. As our key idea is to leverage the semantic understanding of foundation models, the description is used to ground a text-based foundation model to the current state of the environment. We note that this assumption is more practical than the assumptions in prior works where language descriptions of the full agent state or trajectories were assumed [15, 23], as obtaining an object list is easily implementable in the real world with a pre-trained object detector [14].

3 LiFT Framework

We present LiFT (unsupervised **L**earning w**i**th **F**oundation models as **T**eachers), a closed-loop system for unsupervised RL with foundation model guidance. Our framework consists of two phases: **1) LLM task instruction proposal** which obtains a set of semantically meaningful instruction proposals based on the current training environment, and **2) VLM-guided policy learning** which trains a multi-task policy in the training environment using the obtained instructions, as illustrated in Figure 1. We describe how the agents acquire meaningful instruction proposals grounded in the current environment in Section 3.1, how we pre-train a policy based on the proposed instructions in Section 3.2, and the practical implementation of our framework in Section 3.3.

3.1 LLM Task Instruction Proposal

The first step of our approach is to generate a set of *imagined* task instructions that are useful to accomplish future target tasks. To achieve this without human supervision, the agent initialized in a training environment asks an off-the-shelf LLM to generate useful task instructions given the description of the available objects in the current environment $d(s_0)$. For example, by prompting the observation of a cow and a bucket in the current environment, the LLM suggests the instruction of "milk a cow." The initial state and the proposed task instructions are then used to train a multi-task language-conditioned policy, as described in the next section. For an example of the prompt we use, see Figure 2.

> Propose a skill for an Minecraft agent.
>
> You see blocks of grass, grass block.
> You face entities of cow.
> You have a bucket in your hand.
> Target Skill: milk a cow.
>
> You see blocks of [objects].
> You face entities of [entities].
> You have a [item] in your hand.
> Target Skill:

Figure 2: **An example LLM prompt.** See Appendix B for the full prompt and example results.

3.2 VLM-Guided Policy Learning

Given the proposed set of N-numbers of task instructions $\{\delta^{(i)}\}_{i=1}^{N}$ and their corresponding initial states, our goal is to train a multi-task policy $\pi(a|s, \delta)$ that follows the instructions. To accomplish this without expert demonstrations or human reward engineering, we leverage CLIP4Clip [24]-style VLMs, which are pre-trained to align videos and corresponding language labels on large video-language datasets, e.g., YouTube data, with contrastive learning. We use these VLMs to directly reward the RL agent based on the task instructions and videos of corresponding agent behavior.

Analogous to the training scheme in MineDojo [20], our policy is trained to follow the given instruction by maximizing the VLM "alignment score" between the current observation and the instruction as its reward. Specifically, our reward is defined by:

$$r_t = r(o_{t-H:t}, \delta) = \frac{\phi_V(o_{t-H:t})^\top \phi_T(\delta)}{|\phi_V(o_{t-H:t})| \cdot |\phi_T(\delta)|}, \tag{1}$$

where o_t is the visual observation of time step t with $o_{t-H:t}$ implying the sequence of observations with size of H, $|\cdot|$ refers to L2-norm of a vector, ϕ_T and ϕ_V are the text and video encoder of the VLM, δ is the language instruction, and H is the length of video that the VLM takes. This reward scheme works because video-language alignment VLMs are pre-trained to maximize the

cosine similarity of correctly paired videos and language instructions, which is directly represented in Equation 1.

Additionally, we empirically observe that the computed reward signal typically exhibits high variance, likely coming from noise induced by visual variations within the observed agent trajectories. Thus, we adapt the reward stabilization technique from DECKARD [25] to mitigate learning instability:

$$\tilde{r}_t = a \cdot \max(0, \text{avg}(r_{t-W:t}) - b)$$
$$\hat{r}_t = \begin{cases} \tilde{r}_t, & \text{if } \tilde{r}_t > \max(\tilde{r}_{:t-1}) \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $\text{avg}(\cdot)$ is the average function, a is a reward scaling factor, b is a bias used for reward shaping, and W is the size of a softening window length. The importance of this reward post-processing for learning success is discussed in Section 4.3.

Finally, we train a language-conditioned multi-task policy π by optimizing the following multi-task objective using any standard RL algorithm:

$$\sum_{i=1}^{N} \mathbb{E}_{o_t \sim \pi, \rho, \mathcal{P}} \left[\sum_t \hat{r}(o_{t-H:t}, \delta^{(i)}) \right]. \tag{3}$$

3.3 Practical Implementation

Foundation Models We use Vicuna-13B [26] for the task proposal and MineCLIP [20] for the reward-signal VLM without fine-tuning.

Task Instruction Proposal To amortize the querying cost of the LLM, we initialize the agent in a fixed set of starting states (i.e., individual Minecraft environments) and pre-compute possible task instructions using the LLM in these scenarios.

Policy and Value Network Our policy network is initialized with VPT [27], a pre-trained behavior cloning policy in the Minecraft domain. A randomly initialized policy is not able to learn meaningful behavior mainly due to the complexity of the environment and reward quality (see Section 4.3). To share the pre-trained state representation from VPT, our value function is implemented by adding a single fully-connected layer after the transformer layers.

To fine-tune the pre-trained policy and value function networks into multi-task language-conditioned networks, we add task-conditioning adapter layers [28] in the transformer layers of VPT. Each adapter layer concatenates an embedding of language instruction for a current task with an output from the previous layer. The concatenated vector is then used to compute the residual prediction of the previous output as proposed in [28]. We only update the parameters of the adapter layers and a value function head, while other parameters are frozen. See Appendix C for more details of the policy architecture.

RL Optimization Our policy is trained with the PPO algorithm [29]. A PPO iteration uses one rollout for each proposed task to optimize the multi-task objective in Equation (3). Following [27], the KL-divergence term is added to Equation (3) to prevent catastrophic forgetting. We provide hyperparameters for optimization in Appendix C.

4 Experiments

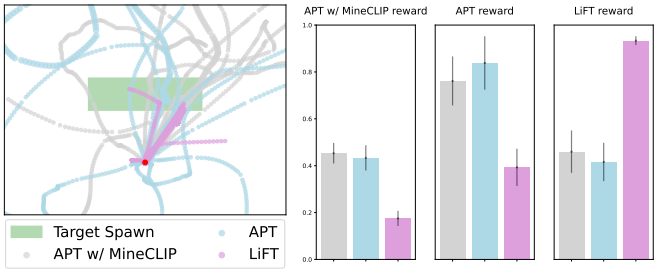
Our experiments aim to tackle two questions: (1) Can the agents acquire meaningful behaviors in a challenging, open-ended environment using the LiFT framework? and (2) Do current off-the-shelf foundation models demonstrate sufficient understanding of agent behaviors to enable scalable learning within the LiFT framework? Section 4.2 presents a comparison experiment against baselines answering the first question, and Section 4.3 highlights the reward quality from VLM with respect to the second question.

4.1 Experimental Setup

We evaluate our method with other baselines in MineDojo [20] environment. Minedojo is an open-ended environment with a variety of potential open-ended goals in simulated 3D game worlds. In our

	Success Rate
VPT	0.19 ± 0.01
APT	0.14 ± 0.02
APT w/ MineCLIP	0.16 ± 0.01
LiFT	**0.50** ± **0.03**
LiFT w/ Oracle Tasks	0.49 ± 0.02

Table 1: **Evaluation results.** The LiFT agent shows results that outperforms the unsupervised RL methods and is comparable to the baseline using the oracle task instructions annotated by humans.



(a) Trajectories in top view. (b) Normalized reward scores.

Figure 3: **Qualitative results.** The LiFT agent learns how to interact with the target in environment (green). The baselines optimize their own intrinsic reward, resulting in diverse but not semantically meaningful behaviors.

experiments, the agents perceive the world through RGB images and interact with the environment by choosing one of the discrete actions for each time step.

In particular, we choose 8 different environment setups from a task set proposed in [20]. Each environment setup is designed to evaluate a basic skill in Minecraft game, such as harvesting logs or combating monsters, thus necessary objects or animals are placed near the initial position or a necessary tool is provided to the agent.

For the task proposal, as described in Section 3.1, we prompt an LLM with the initial textual information of nearby objects, entities, and the current tool in the agent's hand retrieved from the game simulator state.

For the training, we use 5 different world seeds from each environment setup, resulting in 40 different initial states. The initial state is used for the task proposal, thus the number of LLM-generated task instructions is also 40. Note that the agent is not provided with any supervision about the evaluation tasks, such as the oracle task instruction or reward, for training.

After the training, we zero-shot evaluate the learned policy. We rollout the learned policy 20 times for each environment setup and measured the average success rates with the success criteria of each evaluation task. The language instruction for each evaluation task is provided to the policy if the policy is conditioned on language. For more details, see Appendix A.

We compare our method with below baselines:

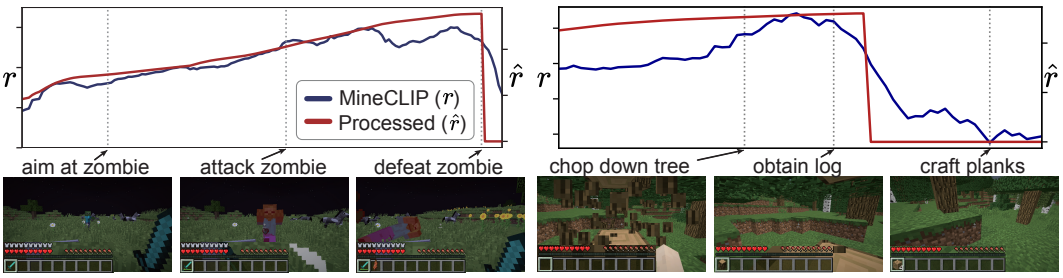
LiFT w/ Oracle Tasks is trained with the reward computed by the definition in Equation (1), similar to LiFT, but directly uses the oracle task instruction annotated by humans.

APT maximizes the state-entropy in visual embedding space by using variational autoencoder (VAE) [30]. The VAE is pre-trained with visual observations from VPT policy rollouts.

APT w/ MineCLIP is similar with **APT**, but maximizes the state-entropy in visual embedding space of MineCLIP, instead of VAE.

VPT is directly evaluated with the pretrained VPT [27] policy and is not finetuned. This baseline indicates the lower bound of our framework.

All methods are trained for 300 iterations of PPO update in the same training environment. For a fair comparison, all baselines except VPT use VPT-initialized policy and adapter layers for fine-tuning. The trajectories used for plotting and computing the reward scores are with 10 rollouts in a fixed evaluation task with the predefined spawn region of the target entity, and the scores are normalized with the maximum and minimum values of the whole collection. For ablation studies in Section 4.3, we train for 100 iterations of PPO updates. More details on the baseline implementations are in Appendix D.



(a) combat a zombie (b) craft planks

Figure 4: **VLM reward examples.** 4a and 4b plot computed rewards for each task instruction and its demonstration. While reward stabilization greatly reduces noise, the video-text alignment of VLM is not as precise as providing the highest reward for the most critical behavior in the demonstration.

4.2 Results

We present our quantitative comparison results in Table 1 and qualitative examples in Figure 3. Our approach outperforms the unsupervised RL baselines (APT and APT w/ MineCLIP) and achieves comparable results to the oracle approach of using human supervision for task proposal (LiFT w/ Oracle) with zero-shot evaluation.

During the unsupervised pre-training stage, our method succeeds in training a policy equipped with semantically meaningful behavior. For example, when the agent is placed in an initialization position near a sheep with a shear, our framework decides to learn how to shear a sheep. Figure 3a shows how our agents interact with the target entity given the task instruction.

On the other hand, the success rates of unsupervised RL baselines are even worse than the VPT baseline. Each agent of the baselines tries maximizing the diversity, as shown with the highest reward scores in Figure 3b. However, the learned behavior is not semantically meaningful in that the agents just wander around a wide range of regions, as shown in Figure 3a, or gaze at diverse entities without following the task description.

4.3 Analysis

We empirically observe that the main challenge of our framework for scalable learning is obtaining a decent quality of rewards. To highlight how different design choices influence reward function quality, we present an ablation experiment in Table 2 and a qualitative analysis of VLM rewards in Figure 4.

Reward Stabilization	Policy Initialization	Reward Type	Success Rate
✓	✓	**Cossim.**	**0.42** ± 0.01
	✓	Cossim.	0.33 ± 0.02
✓		Cossim.	0.00 ± 0.00
✓	✓	Softmax	0.35 ± 0.03

Table 2: **Ablation results.** The choice of rewarding scheme and VPT initialization are crucial for successful learning.

Reward Stabilization The reward post-processing in Equation (2) is crucial for successful learning of our framework, as shown with the comparison between the first row and the second row of Table 2. The examples in Figure 4 qualitatively show that post-processing greatly stabilizes the noisy VLM reward, justifying the performance improvement. While the rewards before processing contain fluctuation from visual changes in observation, the smoothing and clipping in the reward processing stably cancel out the task-irrelevant noise of the original rewards.

Policy Initialization The agents are unable to acquire meaningful behaviors in the absence of the VPT policy initialization, as demonstrated in the comparison between the first row and the third row of Table 2. We suspect that this failure primarily stems from the challenge of attempting to learn long-horizon and high-dimensional Minecraft skills entirely from scratch. However, the results also suggest that the VLM reward lacks the necessary density or precision for efficient training of agents from scratch, further discussed in the next paragraph.

VLM Reward Limitation We visualize examples of rewards computed by VLM for task accomplishing demonstrations in Figure 4. Both examples in Figure 4 illustrate that VLM rewards are generally high for behavior that is somewhat related to the task instructions but often fall short of providing the highest rewards for the most success-critical moment.

In particular, the agents struggle to learn the full task behavior from the rewards in Figure 4b, as they are dominantly rewarded only for a subtask of the target task. We suspect that this imperfection arises from the video-language alignment issue with the current off-the-shelf VLM. This work has attempted to alleviate the issues by using several strategies discussed above, yet we leave improving VLM reward for a more diverse set of tasks, including complex and delicate tasks, as future work.

For further analysis of the limitation of VLMs, we point to the limitations of the internet-scale dataset used for finetuning current off-the-shelf VLMs. Many of the YouTube videos used for training MineCLIP have misalignment issues between the speech caption used as the text label and the video itself. For example, the auditory explanation for certain actions comes far after what happens in the scene in many YouTube videos. We anticipate that this challenge can be mitigated through fine-tuning with in-domain expert data or data curation of videos with better alignment quality.

Another hypothesis we posit is that the contrastive learning objective without considering *precise success moment* might have not been appropriate for fine-grained feedback on motion. Similar observations have been reported with video-language alignment score of R3M on robot manipulation tasks [22]. We believe that a training objective more suitable for the fine-grained temporal understanding of video-language alignment is required.

Reward Type To examine whether a different formulation of reward can mitigate the aforementioned reward imperfectness, we explore a reward type of softmax, proposed in [20]. This exploration is aimed at assessing the potential of relative normalization through softmax rewards for reward stabilization. For each sample in training batch, we use its original task instruction as a positive instruction and consider the other instructions in the batch as negative instructions to compute the softmax reward. Given the comparison between the first row and the last row of Table 2, we conclude the our method does not significantly benefit from a naive application of softmax reward. We leave the careful design of negative instructions as future work.

5 Related Work

Unsupervised Reinforcement Learning Unsupervised RL aims to learn a policy, that is helpful to adapt downstream tasks, without human supervision. While diverse categories of intrinsic rewarding methods are proposed, including information gain [31, 32], competence [8, 10, 33–35], or state visitation diversity [9, 36], a core idea of these methods is maximizing behavior diversity in a learned representation space without semantic guidance. As the learned representation is not guaranteed to be well-aligned with human environment understanding, maximizing diversity in the representation spaces often lacks semantic meaningfulness. In this work, we overcome this limitation with guidance from foundation models.

Open-Ended Learning As an effort to develop generally capable agents, recent works have proposed training agents in an open-ended environment instead of a specified task scenario [37–39, 20]. MineDojo [20] cast Minecraft game as a developmental open-ended environment inspired by its high degree of freedom and further suggests to use of internet-scale knowledge about the game for open-ended learning. While MineDojo proposes a set of new tasks brainstormed with LLM and to reward agents using VLM trained on the YouTube knowledge base, how to integrate and ground them in an open-ended learning environment is yet explored. Voyager [16] introduces a lifelong learning framework that continually explores the world and extends the skill set in the MineDojo environment using LLM. Voyager uses LLM for code-level planning and refinement based on pre-defined basic skills, while our work aims to acquire visuo-motor policy without access to hand-engineered skills.

Foundation Models for RL The internet-scale knowledge capacity of recent FMs enables automating impractical human effort in RL framework [6, 25, 40–42, 22]. Existing works query pre-trained LLMs tasks to learn [15, 17, 20], language-level plans [43], and language labels [18, 5]; or use pre-trained VLMs to obtain visual feedback [19, 25, 44, 45]. Closest to ours, ELLM [15] uses LLMs to propose new tasks for agents to learn. However, their experiments are limited to simpler

environments which a captioning model can generate meaningful captions for, as they use generated caption alignment with proposed tasks to reward the agent. A line of work [25, 20, 16, 46] specifically focuses on using FMs for the Minecraft domain, while none of the works integrate pre-trained LLM and VLM for an unsupervised RL system. We focus on closing the loop for adaptation to open-ended environments.

6 Conclusion

We propose LiFT, an unsupervised RL framework for learning semantically meaningful behavior without human supervision. Our method uses two foundation models, LLM and VLM, to construct a closed-loop policy learning system. Our experimental results in the MineDojo environment demonstrate that agents with LiFT framework can acquire semantically meaningful behavior in a challenging open-ended environment. We demonstrate analysis for our strategies and the quality of the learned behaviors, showing the limitation of the prior approaches in unsupervised RL that we compare with. Our further analysis reveals imperfections in the quality of VLM rewards, we hope future work on enhancing VLM to provide higher-quality visual feedback.

References

[1] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. In *Conference on Robot Learning*, 2019.

[2] Karl Pertsch, Youngwoon Lee, Yue Wu, and Joseph J. Lim. Demonstration-guided reinforcement learning with learned skills. In *Conference on Robot Learning*, 2021.

[3] 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, 2021.

[4] Minho Heo, Youngwoon Lee, Doohyun Lee, and Joseph J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. In *Robotics: Science and Systems*, 2023.

[5] Jesse Zhang, Karl Pertsch, Jiahui Zhang, and Joseph J. Lim. Sprint: Scalable policy pre-training via language instruction relabeling, 2023.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alex Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *arXiv preprint arXiv:2307.15818*, 2023.

[7] Yevgen Chebotar, Quan Vuong, Karol Hausman, Fei Xia, Yao Lu, Alex Irpan, Aviral Kumar, Tianhe Yu, Alexander Herzog, Karl Pertsch, Keerthana Gopalakrishnan, Julian Ibarz, Ofir Nachum, Sumedh Anand Sontakke, Grecia Salazar, Huong T Tran, Jodilyn Peralta, Clayton Tan, Deeksha Manjunath, Jaspiar Singh, Brianna Zitkovich, Tomas Jackson, Kanishka Rao, Chelsea Finn, and Sergey Levine. Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=0I3su3mkuL.

[8] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *International Conference on Learning Representations*, 2018.

[9] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 2021.

[10] Michael Laskin, Hao Liu, Xue Bin Peng, Denis Yarats, Aravind Rajeswaran, and Pieter Abbeel. Cic: Contrastive intrinsic control for unsupervised skill discovery. *arXiv preprint arXiv:2202.00161*, 2022.

[11] Elad Hazan, Sham Kakade, Karan Singh, and Abby Van Soest. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pages 2681–2691. PMLR, 2019.

[12] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[13] Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2021.

[14] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary object detection via vision and language knowledge distillation, 2022.

[15] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. *International Conference on Machine Learning*, 2023.

[16] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[17] Jenny Zhang, Joel Lehman, Kenneth Stanley, and Jeff Clune. Omni: Open-endedness via models of human notions of interestingness. *arXiv preprint arXiv:2306.01711*, 2023.

[18] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J Lim. Bootstrap your own skills: Learning to solve new tasks with large language model guidance. In *7th Annual Conference on Robot Learning*, 2023. URL https://openreview.net/forum?id=a0mFRgadGO.

[19] Yuchen Cui, Scott Niekum, Abhinav Gupta, Vikash Kumar, and Aravind Rajeswaran. Can foundation models perform zero-shot task specification for robot manipulation? *Learning for Dynamics and Control Conference*, 2022.

[20] 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. *Advances in Neural Information Processing Systems*, 2022.

[21] Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning, 2023.

[22] Ademi Adeniji, Amber Xie, Carmelo Sferrazza, Younggyo Seo, Stephen James, and Pieter Abbeel. Language reward modulation for pretraining reinforcement learning. *arXiv preprint arXiv:2308.12270*, 2023.

[23] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

[24] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. Clip4clip: An empirical study of clip for end to end video clip retrieval and captioning. *Neurocomputing*, 508:293–304, 2022.

[25] Kolby Nottingham, Prithviraj Ammanabrolu, Alane Suhr, Yejin Choi, Hannaneh Hajishirzi, Sameer Singh, and Roy Fox. Do embodied agents dream of pixelated sheep?: Embodied decision making using language guided world modelling. *arXiv preprint arXiv:2301.12050*, 2023.

[26] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *https://lmsys.org/*, 2023.

[27] Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 2022.

[28] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

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

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[31] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[32] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[33] 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.

[34] Steven Hansen, Will Dabney, Andre Barreto, David Warde-Farley, Tom Van de Wiele, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BJeAHkrYDS.

[35] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *International Conference on Machine Learning*, pages 6736–6747. PMLR, 2021.

[36] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pages 11920–11931. PMLR, 2021.

[37] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021.

[38] Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

[39] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *arXiv preprint arXiv:1810.08272*, 2018.

[40] Amber Xie, Youngwoon Lee, Pieter Abbeel, and Stephen James. Language-conditioned path planning. *arXiv preprint arXiv:2308.16893*, 2023.

[41] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[42] Anurag Ajay, Seungwook Han, Yilun Du, Shaung Li, Abhi Gupta, Tommi Jaakkola, Josh Tenenbaum, Leslie Kaelbling, Akash Srivastava, and Pulkit Agrawal. Compositional foundation models for hierarchical planning. *arXiv preprint arXiv:2309.08587*, 2023.

[43] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

[44] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[45] Sumedh A Sontakke, Jesse Zhang, Sébastien M. R. Arnold, Karl Pertsch, Erdem Bıyık, Dorsa Sadigh, Chelsea Finn, and Laurent Itti. RoboCLIP: One demonstration is enough to learn robot policies. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=DVlawv2rSI.

[46] Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. *arXiv preprint arXiv:2306.00937*, 2023.

[47] William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019. URL http://minerl.io.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Supplementary Material

A Experiment Detail

Observation and Action space In our experiment, the agents observe 640×360 RGB rendered images of the Minecraft game. For action space, we follow the hierarchical action space of [20] to incorporate with a complex control interface of Minecraft game.

Training For training, the environment for training is in the total number of 40 with 3 different seeds. We select 8 different environment setups, sharing the biome, candidate spawning entity, and time. Among 8 environment setups, the biomes of 7 environment setups are chosen to be 'sunflower plains' where candidate target entities can be such as cow, chicken, sheep, horse, and so forth for the day and spider, zombie, and so on. For the other worlds, the biomes are chosen to be 'forest', where it is with more trees around with possible candidates similar to 'sunflower plains'. We select 6 environment setups to be in the day and 2 environment setups to be in the night.

Then, we set 5 different world seeds for each environment setup, so 40 different environments in total. The below Table 3 shows the 5 different world seeds used for each setup.

seed	world seeds
1	171529 167223 177048 159010 148055
2	189255 172172 195815 144565 131018
3	168267 143566 142612 145890 121242

Table 3: Table of values for world seeds.

For easier skill learning, we initialize a possible target entity in front of the agent for each environment. The spawn region is to be from $[-4, 1, 5]$ to $[4, 1, 8]$ in xyz-coordinate. The entities are randomly spawned in the spawn region but do not necessarily become the target of the task.

For training, we run 300 epochs of PPO step for each baseline. We set the time limit to 200 environment steps for each training epoch.

Evaluation For evaluation, the agents are asked to perform 8 tasks. We modify action space and environment, as explained in detail below. In the evaluation, the agents are asked to perform target tasks described in textual form. For one evaluation, we rollout for 20 times and compute the average success rate for the comparison results. The evaluation scores are measured when the training of 260, 280, and 300 epochs is over for each baseline. We report the average of three measures and the standard error of them with respect to three seeds.

We use random values for the world seeds to be different from the world seeds used in training. We exhibit the target tasks with human-annotated task descriptions as below.

	target entity	item	task description
1	cow	bucket	get milk from a cow
2	sheep	shear	shear a sheep and get some wool
3	chicken	diamond sword	hunt a chicken and get its meat
4	tree	(nothing)	collect logs
5	cow	diamond sword	kill a cow
6	sheep	diamond sword	kill a sheep
7	spider	diamond sword	kill a spider
8	zombie	diamond sword	kill a zombie

Table 4: Table of information on the target task environment.

The tasks are chosen to be diverse enough and meaningful for initial survival in the Minecraft environment from a task set proposed in [20].

For the qualitative analysis, we test the evaluation of the task 'shear a sheep and get some wool'. We sample 10 trajectories for each model we test.

B Full Prompt

The full prompt we use is as below. We use lidar rays in the MineDojo environment, with the range of from $[-45\pi/180, -45\pi/180, 20]$ to $[+45\pi/180, +45\pi/180, 20]$, for obtaining the information of initial surroundings. An agent gets the information of objects, and entities in front of it and an item in its hand. This information is parsed into the prompt.

Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.

Instruction:
You are proposing basic skills that a Minecraft agent can learn.
You just began Minecraft game.
Propose one of the most interesting tasks that you can learn from a given situation in the input as 'Target Task'.
Propose several not interesting tasks as 'Negative Tasks'.

Guidelines:
Prioritize using your item in your hand and target entities you face.
And list other possible tasks that are possible but less interesting tasks.
Each task should refer to one specific object in the input.
Make sure that the response is as concise as possible, like a pair of an action and a target object.
Answer with each task of less than 5 words.

Input:
You see blocks of grass, grass block, wheat, wood, flower.
You face entities of horse.
You have a diamond sword in your hand.
Response:
Target Task: combat a horse.
Negative Tasks: dig the ground, water the flower, grow a plant.

Input:
You see objects of {(', '.join(objects)) if len(objects) > 0 else 'nothing'}.
You face entities of {(', '.join(entities)) if len(entities) > 0 else 'nothing'}.
You have {item} in your hand.

Response:
Target Task:

We also provide a sample set of results, in the table below. We leave (nothing) when the agent does not observe anything or does not have anything in its hand. The observation heavily depends on stochasticity. Small entities like chicken, sometimes, do not get detected by the lidar, although it's observable in the screenshot. We observe that the LLM may propose (nothing) as a target task.

	objects	entities	item	task instruction proposal
1	flower grass grass block plant	cow sheep	bucket	milk entities
2	grass grass block plant	(nothing)	bucket	watering plants
3	dirt flower grass grass block water	cow	bucket	water the flower
4	dirt grass grass block plant	cow	bucket	milk a cow
5	flower grass grass block plant	cow sheep	bucket	water the cow/sheep
6	flower grass grass block plant water	chicken sheep	shears	Shear the sheep
7	grass grass block plant	sheep	shears	shear a sheep
8	grass grass block plant	(nothing)	shears	cut grass
9	dirt flower grass grass block plant	sheep	shears	shear a sheep
10	flower grass grass block plant	pig sheep	shears	shear animals
11	flower grass grass block plant water	(nothing)	(nothing)	drink water
12	grass grass block	(nothing)	(nothing)	observe
13	plant	(nothing)	(nothing)	observe plants
14	dirt grass grass block stone water	(nothing)	(nothing)	explore
15	flower grass grass block plant	sheep	(nothing)	shepherd sheep
16	grass grass block leaves wood	(nothing)	(nothing)	explore
17	grass grass block leaves wood	pig	(nothing)	chase a pig
18	grass grass block leaves wood	(nothing)	(nothing)	gather objects
19	dirt grass grass block leaves stone wood	(nothing)	(nothing)	collect dirt
20	dirt grass block leaves wood	(nothing)	(nothing)	walk towards nearest object
21	flower grass grass block plant water	chicken cow sheep	diamond sword	kill a chicken
22	grass grass block plant	cow	diamond sword	kill cow
23	plant	(nothing)	diamond sword	harvest plant
24	dirt grass grass block plant	cow	diamond sword	attack cow
25	grass grass block plant	cow pig sheep	diamond sword	hunt an animal
26	flower grass grass block plant water	chicken sheep	diamond sword	herd animals
27	grass grass block plant	sheep	diamond sword	shear sheep
28	dirt flower grass grass block plant water	(nothing)	diamond sword	kill a plant
29	dirt grass grass block plant	sheep	diamond sword	kill a sheep
30	grass grass block plant	sheep	diamond sword	shear a sheep
31	flower grass grass block plant water	sheep spider	diamond sword	kill a spider
32	grass grass block plant	spider	diamond sword	kill spider
33	grass grass block plant	(nothing)	diamond sword	(nothing)
34	dirt flower grass grass block plant	spider	diamond sword	kill spider
35	grass grass block plant	sheep spider	diamond sword	kill spider
36	flower grass grass block plant water	chicken zombie	diamond sword	combat a zombie
37	grass grass block plant	zombie	diamond sword	combat zombie
38	grass grass block plant	(nothing)	diamond sword	Harvest grass
39	dirt flower grass grass block plant	zombie	diamond sword	slay zombie
40	grass grass block plant	zombie	diamond sword	kill the zombie

Table 5: A result of task instruction proposal from LLM

C Training Detail

Network Architecture As described in Section 3.3, we added multi-task adapter layers in the transformer layers of VPT policy. Specifically, we added adapter layers after every feed-forward network in transformer layers, and the adapter output is computed by:

$$\mathbf{h}' = \mathbf{h} + g(\mathrm{ReLU}(f(\mathbf{h}||\phi_T(\delta)))), \tag{4}$$

where $||$ represents a vector concatenation, $\mathbf{h}$ is an output from the previous feed-forward network, δ is a task instruction of current observation, f is a down-projecting linear layer, and g is an up-projecting linear layer. We used publicly opened `bc-house-3x` VPT architecture and the pre-trained weights. We used MineCLIP as our VLM for the Minecraft domain and `mineclip_attn` variant if used for our experiments.

Hyperparameters We provide a hyperparameter set for our training:

name	value
PPO clip	0.2
PPO learning rate	0.0001
PPO max epoch	5
PPO batch size	80
PPO target KL	0.1
Value function loss scale	1
γ	0.999
GAE λ	0.95
GAE normalized	True
Adapter contraction factor	8
VPT KL loss scale	0.3
VPT KL loss scale decay	0.999
Eq.1 H	16
Eq.2 a	0.005
Eq.2 b	21
Eq.2 W	50

Table 6: **Hyperparameters for LiFT**

Action Space Mapping As VPT is pre-trained on [47] environment, the action space of Minedojo and VPT policy output has a discrepancy. We mapped the inventory opening action of VPT into the crafting action of Minecraft action space to enable our VPT-based policy to craft items. We introduced an additional policy head for a categorical distribution over a set of craftable items in the Minedojo environment, where we randomly initialized the new head.

D Baseline Implementation

We explain details of objectives and hyperparameters used during the experiment for the baselines.

LiFT w/ Oracle Tasks For 'LiFT w/ Oracle Tasks' baseline, we used human-annotated task description for training described in Table 4, without asking LLM for task instruction proposal.

APT, APT w/ MineCLIP For unsupervised RL baselines, we compute intrinsic rewards for particle-based entropy maximization in representation space following [9].

For 'APT' baseline, we first pre-trained a VAE on a set of VPT rollouts and used its encoder for the agent behavior learning. We used an image encoder of VAE as and representation encoder, instead of an encoder trained by contrastive loss, for simplicity. We also observed that continually fine-tuning the representation encoder during the agent behavior learning does not help to learn useful representation but rather introduces large training instability.

name	value
Buffer size	80000
Batch size	64
Number of particles	4096
k	96
Scale	0.01

Table 7: **Hyperparameters for APT**

For VAE architecture used in 'APT' baseline, we let a sequence of ResNet[48] blocks for both encoder and decoder. One ResNet block is composed of two ResNet layers with batch normalization after each layer. We define the reconstruction objective with mean-squared error loss. We resize the observation resolution to feed into the VAE encoder. We provide a hyperparameter set.

name	value
Input image	64x64x3
Latent dimension	512x4x4
Encoder blocks channels	(64, 128, 256, 512)
Decoder blocks channels	(512, 256, 128, 64)
Activation function	ELU
VAE learning rate	5e-4
VAE learning epoch	5

Table 8: **Hyperparameters for VAE**

For the 'APT w/ MineCLIP' baseline, We used MineCLIP visual encoder ϕ_T instead of the pre-trained VAE to compute the representations of the transitions. `mineclip_attn` variant if used for our experiments.

name	value
Buffer size	20000
Batch size	64
Number of particles	1024
k	12
Scale	0.01

Table 9: **Hyperparameters for APT w/ MineCLIP**

VPT For the 'VPT' baseline, We used publicly opened `bc-house-3x` VPT architecture and the pre-trained weights without finetuning during the evaluation.