OPTIMIZING LATENT GOAL BY LEARNING FROM TRAJECTORY PREFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

A glowing body of work has emerged focusing on instruction-following policies for open-world agents, aiming to better align the agent's behavior with human intentions. However, the performance of these policies is highly susceptible to the initial prompt, which leads to extra efforts in selecting the best instructions. We propose a framework named ***P**reference **G**oal **T**uning* (PGT). PGT allows policies to interact with the environment to collect several trajectories, which will be categorized into positive and negative examples based on preference. A preference optimization algorithm is used to fine-tune the initial goal latent representation using the collected trajectories while keeping the policy backbone frozen. The experiment result shows that with minimal data and training, PGT achieves an average relative improvement of 72.0% and 81.6% over 17 tasks in 2 different foundation policies respectively, and outperforms the best human-selected instructions. Moreover, PGT surpasses full fine-tuning in the out-of-distribution (OOD) task-execution environments by 13.4%, indicating that our approach retains strong generalization capabilities. Since our approach stores a single latent representation for each task independently, it can be viewed as an efficient method for Continual Learning, without the risk of catastrophic forgetting or task interference. In short, PGT enhances the performance of agents across nearly all tasks in the Minecraft Skillforge benchmark and demonstrates robustness to the execution environment.

1 INTRODUCTION

Recently, pre-training foundation policies in open-world environments with web-scale unlabeled datasets have become an increasingly popular trend in the domain of sequential control(Baker et al., 2022; Zhang et al., 2022; Collaboration et al., 2024; Brohan et al., 2023a; Yang et al., 2023). These foundation policies possess broad world knowledge, which can be transferred to downstream tasks. In the realm of foundation policies, there exists a category known as goal-conditioned policies, which are capable of processing input goals (instructions) and executing the corresponding tasks (Ding et al., 2019; Chane-Sane et al., 2021). The goal can be in different modalities, such as text instructions (Lifshitz et al., 2023), video demonstrations (Cai et al., 2023), or multi-model instructions (Cai et al., 2024; Brohan et al., 2023b;a)).

However, much like large language models, these instruction-following policies are highly susceptible to the selection of "prompts"(Lifshitz et al., 2023; Wang et al., 2023; Kim et al., 2024). Researchers rely on trial and error to find the optimal prompt manually, and sometimes the quality of prompts doesn't align with human judgment. For instance, OpenVLA (Kim et al., 2024) shows much stronger sensitivity to the phrase "Pepsi can" than "Pepsi"; for the same meaning of collecting wood logs, GROOT has a large fluctuation in performance for different reference videos. Moreover, it is unclear whether an agent's failure to complete a task is due to the foundation policy's inherent limitations or the lack of a suitable prompt.

A common viewpoint from the LLM community thinks that most of the abilities are learned from the pre-training phase (Ouyang et al., 2022; Zhao et al., 2023), while post-training is a method to elicit these abilities for solving tasks with rather small compute (Ziegler et al., 2020; Touvron et al., 2023). In this paper, we follow the roadmap of LLMs to consider post-training for the goal-conditioned foundation policies, hoping to improve the downstream task performance efficiently and effectively. On top of that, we identify several desiderata for the post-training for this type of policy:

- **Improved Elicitation of Pre-trained Abilities.** This refers to (1) leveraging a broader range of abilities and (2) making it easier to harness these abilities, which leads to better performance on downstream tasks without the need for labor-intensive prompts.
- **Task Environment Generalization.** In open-world settings, a single task may be executed in vastly different contexts, making the policy's ability to generalize across environments crucial.
- **Efficient data exploitation.** As it's usually hard or expensive to collect training trajectory data for open-world foundation policy (Villalobos et al., 2024), the post-training is expected to be data-efficient. Meanwhile, it's also important to avoid over-fitting on the small amount of data.
- **Continued adaptation of tasks.** The ability to continually learn from experiences in open-world environments is crucial for generalist AI systems, and thus we expect the open-world foundation policy can continually learn more skills without degrading general ability.

To achieve these desiderata, we propose a framework named ***P**reference **G**oal-**T**uning* (PGT). Firstly, an initial prompt is provided by humans, which may be suboptimal or not carefully refined. This task prompt is embedded into a *goal latent representation*, which is typically a high-dimensional vector. Next, PGT allows the foundation policy to interact with the environment under the guidance of the *goal latent representation*, for a small number of episodes ($\sim 10^2$ of trajectories in practice). These trajectories are then categorized into positive and negative samples based on designed rewards or human preferences. To elicit the ability from the pre-trained foundation policy, the backbone is fixed and a preference learning algorithm (Rafailov et al., 2024a; Azar et al., 2024; Christiano et al., 2017; Hong et al., 2024) is applied to fine-tune the *goal latent representation* via collected trajectories. This training process can be iterative, as the fine-tuned latent goal representation can be used to recollect data once again.

We validate PGT in the open-ended Minecraft video game environment Johnson et al. (2016), with 2 foundation policies and 17 tasks, in both in-distribution and out-of-distribution environments, showing that this framework can enhance performance for foundation policies across almost all tasks. For in distribution settings, we achieved an average improvement of 72.0% and 81.6% in two different policies: GROOT (Cai et al., 2023) and STEVE-1 Lifshitz et al. (2023). For out-of-distribution settings, the figures are 73.8% and 36.9%. We conduct extensive studies on different initial prompts and discover that PGT surpasses all human-selected prompts.cipt Finally, we explore the potential of our method as an efficient approach to Continue Learning (CL). Since we only need to store a latent goal representation for each task in CL, our method is computationally light, storage-tight, with no fear of catastrophic forgetting or task interference in sight.

2 Preliminary

2.1 Sequential Control

In sequential control settings, the environment is defined as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, d_0 \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the transition dynamics, and d_0 is the initial state distribution. A policy $\pi(a|s)$ interacts with the environment starting from $s_0 \sim d_9$. At each timestep $t \geq 0$, an action $a_t \sim \pi(a|s_t)$ is sampled and applied to the environment, after that, the environment transitions to $s_{t+1} \sim \mathcal{P}(s_t, a_t)$ and return reward $r_0 \sim \mathcal{R}(s_t, a_t)$. The goal of a policy is to maximize the expected cumulative reward $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$, where $\gamma \in (0, 1]$ is a discount factor.

A goal-conditioned policy can be formulated as $\pi(a|s, g)$, where $g \in \mathcal{G}$ is a goal from goal space $\mathcal{G}$. The target of a goal-conditioned policy is to maximize the expected return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t^g]$, where r_t^g is the goal-specific reward achieved at time step t.

2.2 GOAL-CONDITIONED POLICIES

GROOT GROOT (Cai et al., 2023) is a goal-conditioned foundation policy trained on video data through self-supervised learning with a C-VAE(Sohn et al., 2015) framework. GROOT can follow video instructions in open-world environments. The instruction is encoded into a latent representation by the non-causal encoder, and the policy is a decoder module implemented by a causal transformer, which decodes the goal information in the latent space and translates it into a sequence of actions in the given environment states in an auto-regressive manner.

STEVE-1 STEVE-1 (Lifshitz et al., 2023) is also a goal-conditioned policy on Minecraft environment. STEVE-1 utilizes the goal latent representation of MineCLIP(Fan et al., 2022) to embed the future result video clip in dataset Andrychowicz et al. (2017), and finetunes a VPT model (Baker et al., 2022) as the policy network under the guidance of the MineCLIP embedding. As a C-VAE (Sohn et al., 2015) model is trained to predict "future video embedding" from text, STEVE-1 supports both text and video as instructions.

2.3 PREFERENCE LEARNING

While a large self-supervised learning model possesses a broad world knowledge, it is very hard to align their behavior with human preference. Many efforts are being made to solve this problem. Direct Preference Optimization(DPO) (Rafailov et al., 2024b), as one method, serves as a way to directly optimize the model's outputs based on human preferences. Assuming we have a foundation model π_{ref}. Then pairs of answers are produced given prompt x: $(y_1, y_2) \sim \pi_{\text{ref}}(y \mid x)$, and human labelers express their preference, denoted as $y_w \succ y_l \mid x$. We can obtain a dataset of preference $\mathcal{D} = \{x^{(i)}, y_w^{(i)}, y_l^{(i)}\}_{i=1}^{N}$ DPO derives the optimization objective as:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right]. \quad (1)$$

Details of the mathematical derivation of DPO loss can be found in Appendix A.1

3 METHODOLOGY

3.1 PREFERENCE GOAL TUNING

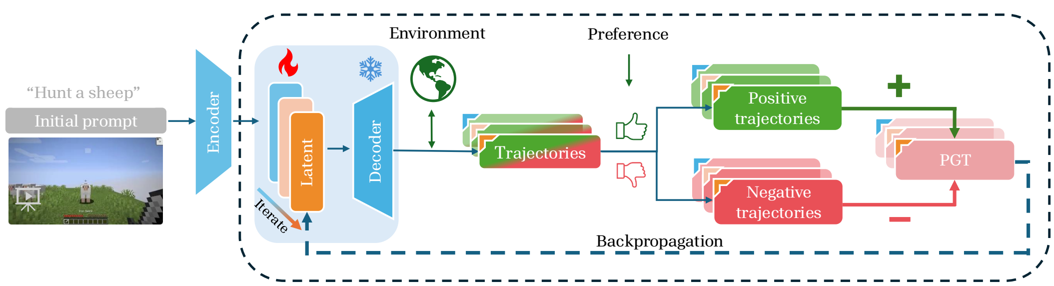

Figure 1: Pipeline of our Preference Goal Tuning (PGT). The process begins by selecting an initial prompt (can be video or text), encoding it into a latent representation, and allowing the policy to interact with the environment multiple times to collect trajectories. These trajectories are then classified as positive or negative based on human preferences or rewards. Then, the model is fine-tuned using the collected data, with only the latent goal embedding being trainable. Iterative training is supported.

In this section, we propose a novel policy post-training framework named ***Preference Goal-Tuning*** (PGT). This approach achieves significant performance improvements for foundation policies with minimal data and computational resources. Our method consists of two stages: the first stage is the data collection phase, and the second stage is the training phase. An illustration of our method is in Figure 1. The details are as follows:

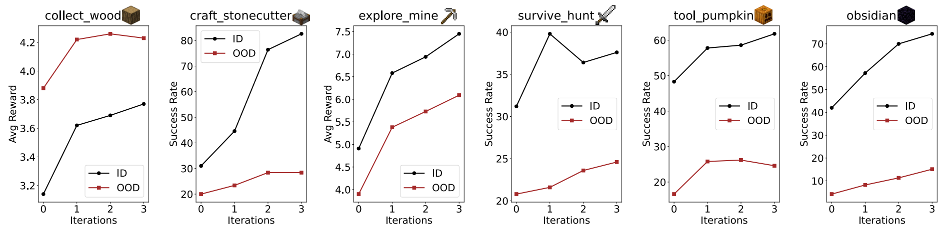

Figure 2: Improvements with training iterations of our methods.

Data Collection Phase We first select an initial prompt, which may be suboptimal or not carefully refined. This initial prompt is embedded into a high-dimensional vector by the encoder of the goal-conditioned policy. We allow the foundation policy to interact with the environment several times, collect the synthetic trajectories, and divide them into positive trajectories and negative trajectories based on *human preference* or *reward from environment*.

When utilizing *human preference*, human annotators are required to label each trajectory, identifying those preferred and not preferred by humans as positive and negative samples, respectively. Since less than 100 samples need to be annotated, the human labor cost remains manageable.

On the other hand, we utilize *reward from environment* for tasks like `collect wood`(), `collect obsidian`() and `explore chest`(). we can obtain reward information from the Minecraft simulator, so we can use rewards as a supervisory signal for preference learning, selecting the top and bottom trajectories based on rewards as positive and negative samples for training.

Training Phase During the training phase, we adopted a learning approach to obtain an optimal *goal latent representation*. During fine-tuning, only the *goal latent representation* is trainable. Initially, we only leverage positive examples with traditional behavior cloning (BC) loss, but it does not yield the expected results. We conduct preliminary experiments on four tasks and find that using BC loss does not lead to significant performance improvements; in fact, performance even declines on 3 out of 4 tasks. Discussion of this choice is in Section 3.2.

Recent studies have emphasized the importance of negative samples (Tajwar et al., 2024a), prompting us to incorporate them into the training data. Following the derivation approach of Direct Preference Optimization (DPO), we obtained desired loss for PGT (2):

$$\mathcal{L}_{\text{PGT}}(g, g_{\text{ref}}) = -\mathbb{E}_{(\tau^{(w)},\tau^{(l)})\sim\mathcal{D}}\left[\log\sigma\left(\beta\sum_{t=1}^{T}\log\frac{\pi(a_t^{(w)}\mid s_t^{(w)};g)}{\pi(a_t^{(w)}\mid s_t^{(w)};g_{\text{ref}})}-\log\frac{\pi(a_t^{(l)}\mid s_t^{(l)};g)}{\pi(a_t^{(l)}\mid s_t^{(l)};g_{\text{ref}})}\right)\right]. \tag{2}$$

Details of derivation lies in Appendix A.2. Given the small amount of data and the limited number of trainable parameters, the training converges quickly.

Iterative Training Our method supports iterative training. During the first training loop, the initial prompt is encoded into a *goal latent representation*, which we denote as g_0. According to 2, we set g_{ref} as g_0 and initialize g as g_0, then fine-tuning g to g_1. We then use g_1 to recollect trajectories and repeat the training loop. Our experiments demonstrate that iterative training continues to improve performance for up to three rounds. See Figure 2 for iterative training details.

3.2 Design Choices

In this section, we address the key design choices of our method and provide a comparative analysis of relevant baselines to justify why we use negative examples and parameter-efficient fine-tuning.

Utilizing negative samples A straightforward approach is to utilize only self-generated positive samples for behavior cloning (BC), and some studies have proved filtering and cloning is enough in many settings (Oh et al., 2018; Gulcehre et al., 2023). However, this approach does not explicitly indicate "which behaviors should be avoided", which is conducive to policy optimization (Tajwar

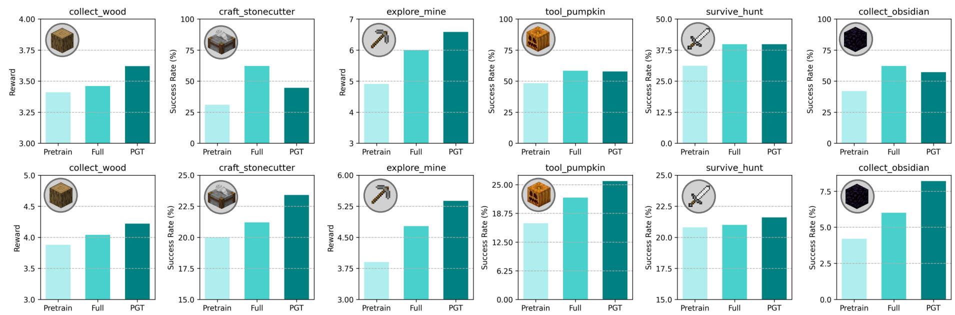

Figure 3: Comparison between full finetuning and PGT. Upper: In Distribution(ID). Lower: Out of Distribution(OOD).

et al., 2024b). Incorporating negative data in our training process allows us to fully leverage the available reward signals, helping the policy distinguish between desirable and undesirable trajectories. As a comparison, we trained a version of the BC algorithm (with double data size of the positive samples to control the amount of data) and conducted experiments with both soft prompt fine-tuning and full fine-tuning, and the results are listed in Table 1.

Table 1: Performance improvements of the PGT-Loss over BC-Loss.

Task	Soft Prompt			Full Fine-Tuning		
	Pretained	BC	PGT	Pretained	BC	PGT
collect_wood	3.14	3.28	**3.62**	3.14	3.26	**3.46**
obsidian	42.0	18.2	**57.2**	42.0	15.0	**62.2**
explore_mine	4.91	4.76	**6.58**	4.91	4.80	**6.00**
tool_pumpkin	48.3	45.4	**57.8**	48.3	48.6	**58.4**

Tuning Latent Goal Only We compare the results of fine-tuning latent goal only and its counterpart that fine-tuning the entire policy model. There are two main reasons why we only fine-tune goal latent instead of fine-tuning the full policy. First, the correspondence between the location of fine-tuned parameters and the feature extraction level within the model matters (Wang et al., 2022), and thus fine-tuning the goal latent offers strong interpretability. For a goal-conditioned foundation policy trained through supervised learning with large datasets, the latent goal space usually holds high-level semantic meanings. However, since the human intention of the instruction and the embedding in the goal space may not always align, the instructions selected by humans might not map well to an optimal latent representation in the goal space. Our method aims to obtain this representation in goal space through a small amount of training. Second, as only a small amount of data is used, full-parameter fine-tuning is highly prone to over-fitting in open-world settings. For example, in Minecraft, task `collect wood`() requires the agent to collect logs from trees, regardless of the biome, seed, and initial location. We believe that with a small amount of training data, full-parameter fine-tuning is likely to memorize environment-specific information to reduce the loss, which could lead to reduced generalization ability.

The experimental results were consistent with our expectations. We found that in settings identical to the data collection phase (in-distribution environments, ID), tuning soft prompts achieved comparable results to full fine-tuning. However, when rolling out in another setting for the same task (out-of-distribution environments, OOD), soft prompt method outperformed the counterpart across all tasks. Detailed results are in Fig 3, and detailed numerical results provided in Appendix C.2. For details of our OOD settings, see Appendix B.4. It is important to note that the absolute performance in the OOD setting is not directly comparable to the ID baseline, as the tasks may become either easier or harder in the OOD environment.

Table 2: Success rates for different methods on tasks in *Minecraft SkillForge*. Δ represents the relative improvements of success rate between policy before and after post-training. GRO and STE represent the base policy of GROOT and STEVE-1 respectively. For tasks evaluated by success rate, the percentage sign (%) is omitted; the same applies to other parts of this paper.

Task	In Distribution						Out of Distribution					
	GRO	GRO+	Δ	STE	STE+	Δ	GRO	GRO+	Δ	STE	STE+	Δ
wood	3.14	**3.62**	15.3%	3.73	**3.90**	4.6%	3.88	**4.22**	8.8%	4.22	**4.29**	1.7%
dirt	27.0	**62.8**	132.6%	16.3	**36.4**	123.3%	15.4	**54.6**	254.5%	30.4	**48.0**	57.9%
wool	30.4	**40.8**	34.2%	43.3	**56.6**	30.7%	34.0	**41.6**	22.4%	45.6	**60.2**	32.0%
seagrass	20.2	**20.8**	3.0%	4.2	**21.8**	419.0%	7.8	**9.4**	20.5%	41.4	**49.0**	18.4%
stonecutter	31.0	**44.6**	43.9%	14.1	**19.0**	34.8%	20.0	**23.4**	17.0%	36.2	**48.4**	33.7%
ladder	5.4	**10.4**	92.6%	30.9	**40.2**	30.1%	4.4	**9.6**	118.2%	29.6	**41.2**	39.2%
enchant	15.0	**18.4**	22.7%	0	0	-	19.4	**21.8**	12.4%	0	0	-
crafting_table	5.4	**14.6**	170.4%	4.0	**9.6**	140.0%	6.0	**18.4**	206.7%	2.0	**6.4**	220.0%
mine	4.91	**6.58**	34.0%	6.46	**7.32**	13.3%	3.9	**5.38**	37.9%	3.49	**5.37**	53.9%
chest	15.7	**21.2**	35.0%	3.4	**4.2**	23.5%	**38.4**	38.2	-0.5%	0.5	**0.6**	20.0%
hunt	31.2	**39.8**	27.6%	**2.9**	1.0	-65.5%	20.8	**21.6**	3.8%	**1**	0.2	-80.0%
combat	31.7	**36.6**	15.5%	0	0	-	83.4	**85.6**	2.6%	0	0	-
plant	2.71	**3.09**	14.0%	1.74	**1.81**	4.0%	2.85	**3.11**	9.1%	1.79	**1.94**	8.4%
pumpkin	48.3	**57.8**	19.7%	1.3	**6.2**	376.9%	16.6	**25.8**	55.4%	7.6	**14.0**	84.2%
bow	77.4	**85.8**	10.9%	88.9	**97.8**	10.0%	77.4	**90.6**	17.1%	65.2	**88.0**	35.0%
flint	1.2	**7.4**	516.7%	73.6	**76.6**	4.1%	1.2	**5.8**	383.3%	48.0	**52.0**	8.3%
obsidian	42.0	**57.2**	36.2%	0.4	**0.7**	75.0%	4.2	**8.2**	95.2%	0	0	-

4 Unlocking Post-training for Open-world Foundation Policy

To demonstrate the effectiveness of our approach, we conducted several experiments on Minecraft SkillForge (Cai et al., 2023) benchmark. This benchmark covers over 30 diverse and representative tasks from 6 major categories. We put the details of this benchmark in Appendix B.2 . Through our experiments, the following contributions of our method are verified:

- PGT remarkably improves the performance of two foundation policies, surpassing the best human-selected prompt.
- PGT serves as an efficient continue learning method.
- PGT improves long-horizon tasks performance with a combination of planner and controller.
- PGT elicits skills that was not achievable with traditional prompts.

4.1 Boosting performance over prompt engineering.

Our approach significantly improves the instruction-following capability of the model. By fine-tuning specific aspects of the model's behavior, we achieve greater task performance compared to traditional prompt engineering techniques, which rely on manually crafted inputs.We discarded tasks in SkillForge that were either too difficult (with zero success rate) or too easy (with a 100 percent success rate), selecting 17 tasks for evaluation without cherry-picking. Tasks selection details are in Appendix B.3.

We experimented with two Foundation policies, GROOT and STEVE-1, in both in-distribution (ID) and out-of-distribution (OOD) settings. The changes made to the OOD settings relative to ID are detailed in Appendix B.4. For in distribution settings, we achieved an average improvement of 72.0% and 81.6% in GROOT and STEVE-1 respectively. For out-of-distribution settings, the growths are 73.8% and 36.9%. Results showed improvements for both models in both two settings across nearly all 17 tasks, with a particularly significant improvement in tasks like `collect dirt`(), `craft crafting table`(), `tool-use flint` (). Detailed results can be found in Table 2.

Different Initial Prompts To validate the robustness of our method, we chose a representative task `collect wood`(), and selected 5 different initial prompts and performed iterative training

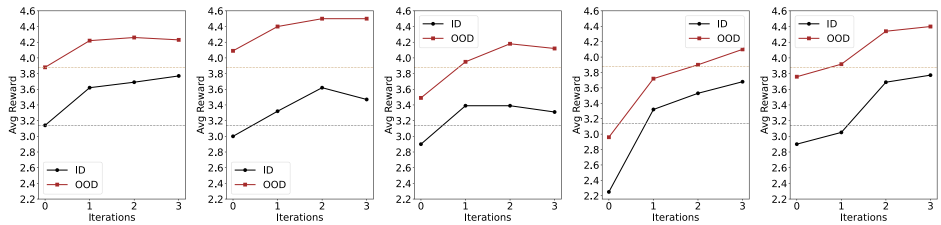

Figure 4: Different initial prompt results. Each line graph represents a different prompt, and the horizontal line represents the performance of best human-selected prompt.

Table 3: Multitask learning on Minecraft different tasks.

Task	In Distribution(ID)				Out of Distribution(OOD)			
	pretrained	ensemble	MTL	Ours	pretrained	ensemble	MTL	Ours
collect_wood	3.14	3.46	**3.64**	3.62	3.88	4.04	**4.30**	4.22
craft_stonecutter	31.0	62.2	**66.8**	44.6	20.0	21.2	18.6	**23.4**
explore_mine	4.91	6.00	5.98	**6.58**	3.90	4.77	4.70	**5.38**
survive_hunt	31.2	39.8	**44.2**	39.8	20.8	21.0	**31.4**	21.6
tool_pumpkin	48.3	58.4	**61.4**	57.8	16.6	22.2	22.8	**25.8**
collect_obsidian	42.0	**62.2**	53.2	57.2	4.2	6.0	**10.2**	8.2

on each. We found that, for any initial prompts, the results after iterative training consistently outperformed the best human-selected reference video. This implies that for nearly any initial prompt, our method consistently surpasses even a carefully selected initial prompt by a human. We present the result in Figure 4.

4.2 Efficient Continual Learning.

Our method is an efficient approach to Continual Learning, as it requires only minimal training for each task, followed by storing a high-dimensional latent (typically consisting of a few hundred floating-point values) as a task representation. As a result, our method avoids issues like catastrophic forgetting and task interference.

We compare PGT with multiple Continue Learning Baselines: Multi-Task Learning (MTL), Naive Continual Learning(NCL), Knowledge Distillation (KD) (Hinton et al., 2015), Experience Replay (ER) (Lopez-Paz & Ranzato, 2022), Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017). It's worth mentioning that every continual learning baseline is conducted under full-parameter fine-tuning, which has a parameter size one million times larger than ours. We first implemented the multi-task learning (MTL) baselines on six representative tasks, with the results presented in the table 3. We find that, similar to the results of full-parameter fine-tuning, our method achieved comparable performance to MTL in in-distribution(ID) settings, while surpassing MTL in Out of Distribution(OOD) settings.

We experiment in the following order: `collect obsidian`() to `tool pumpkin`() to `craft craftingtable`() to `explore climb`(). The result after continue learning 4 tasks is in Table 4, and we place the detailed result of continue leaning after each task in Appendix C.4. We conduct experiment of Naive Continual Learning (NCL) (Table 8), Knowledge Distillation (KD)(Table 9), Experience Replay (ER)(Table 10), and Elastic Weight Consolidation (EWC) (Table 11).

Experiment results show that in addition to being more efficient in terms of computational resources and storage, our method excels in handling diverse tasks, demonstrating superior generalization capabilities. In out-of-distribution settings, we outperform the ensemble in each of the 6 tracks, and we achieve comparable results to Multi-Task Learning.

Table 4: Different continue learning baselines.

Task	In Distribution(ID)					Out of Distribution(OOD)				
	ER	EWC	KD	NCL	PGT	ER	EWC	KD	NCL	PGT
collect_obsidian	60.2	64.6	**66.8**	61.2	57.2	6.0	5.4	5.4	6.8	**8.2**
tool_pumpkin	**65.4**	60.0	60.8	61.4	57.8	25.0	23.8	20.6	20.4	**25.8**
craft_table	8.6	6.8	6.8	7.2	**14.6**	9.0	7.4	5.8	7.0	**18.4**

Table 5: Long Horizon Task: Craft object from scratch. The numbers represent success rate (%)

Task	Wooden Stick	Wooden Sword	Oak Boat	Oak Wood	Large Chest
Pretrain	99.5	94.0	80.7	60.8	37.8
PGT	**100**	**100**	**89.5**	**80.7**	**64.9**

4.3 Solving long-horizon challenges with high-level planner.

It is a common approach to combine a high-level planner and a low-level controller in the pursuit of functional and general agent. We link the GROOT agent with JARVIS-1 planner (Wang et al., 2023), to generate items from scratch in Minecraft. We selected five representative items and observed improvements in long-horizon task performance compared to the baseline. The result can be seen in Table 5.

4.4 Eliciting new skills.

For task `use trident()` , given standard gameplay videos, the agent was unable to complete the task. As a result, standard PGT pipeline cannot collect positive data. Instead, we recorded 20 trajectories by human as positive examples and trained using PGT. Even though the success rate was still low, we found several success examples, meaning that the agent acquired the ability to complete the task. This implies that during the pretraining phase, the agent already possessed the ability to complete the task, but lacked the appropriate prompt to elicit this ability. Our method, through minimal training on the soft prompt, successfully activated this capability.

4.5 Ablation Study

We compare our method with other parameter efficient method: LoRA (Hu et al., 2021), BitFit (Zaken et al., 2022) and VeRA (Kopiczko et al., 2024). We still utilize positive and negative examples and PGT-Loss for all parameter efficient methods. We found that our method performed well among the four types of parameter-efficient fine-tuning. Moreover, in task `expore mine()` and `collect obsidian()`, LoRA fine-tuning also demonstrated promising results. The result is in Figure 5, and numerical result is in Appendix C.3.

5 Related Work

5.1 Foundation Models for Decision-making.

Foundation models has gained huge success in the field of NLP (Brown et al., 2020; OpenAI, 2024) and CV (Kirillov et al., 2023), and an increasing number of studies are exploring the potential of foundation models in sequential control (Yang et al., 2023): VPT (Baker et al., 2022) is a foundation policy trained by video data, which is capable of mining diamond in Minecraft, GROOT (Cai et al., 2023) is an instruction-following policy trained on video data, and is able to solve a variety of tasks on open-world environment. STEVE-1(Lifshitz et al., 2023) fine-tuned the vpt model to follow goal latent representation in MineCLIP (Fan et al., 2022). In robotics, In the field of robotics, there are also many foundation policies like RT-1 (Brohan et al., 2023b), RT-2 (Brohan et al., 2023a), BC-Z (Jang et al., 2022), GTAO (Reed et al., 2022).

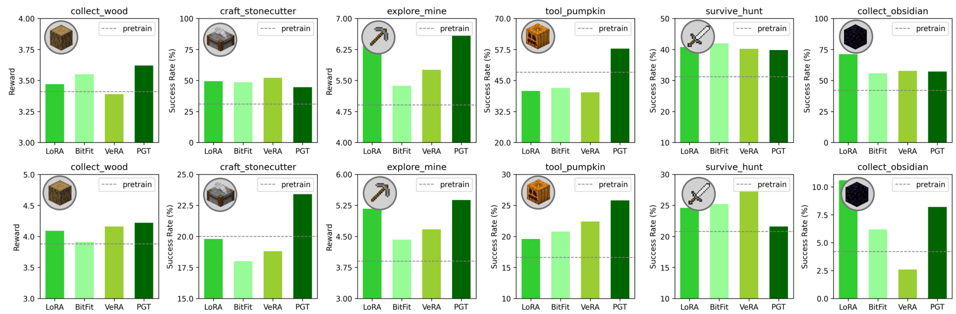

Figure 5: Result of different parameter efficient method. Horizontal line indicates pretraining performance. Upper: In Distribution(ID). Lower: Out of Distribution(OOD).

5.2 Preference Learning

Directly obtaining high-quality human annotations, such as expert numerical ratings (Akrour et al., 2014; Fürnkranz et al., 2012), or expert demonstrations (Silver et al., 2016), is often extremely time-consuming, labor-intensive, and brain-consuming to annotators (Knox & Stone, 2009). Fortunately, the cost is greatly reduced by letting them label pairs or groups of data with simply their preferences Christiano et al. (2017). As a fruitful method to leverage more low-annotation-difficulty data, preference learning has been studied extensively in recent years. Christiano et al. (2017); Ziegler et al. (2020); Ouyang et al. (2022) utilized preference data to teach a reward model, and conducted reinforcement learning on sequential decision-making games or language modeling, demonstrating the efficiency and wide application of preference learning. These methods rely on another model for simulating the reward function and on-policy data. Therefore, some simpler alternatives that do not require reinforcement learning soon emerged (Rafailov et al., 2024a; Azar et al., 2024; Meng et al., 2024) or even without reference model for regularization (Hong et al., 2024). Even though these methods do not strictly demand on-policy data, researchers (Tajwar et al., 2024a) found that preference pairs generated by the current policy can improve fine-tuning efficiency.

6 Limitations and Future Work

PGT has shown remarkable capability in improving task performance. However, it still has some limitations and untapped potential awaiting further exploration.

Limitations PGT requires multiple interactions with the environment to obtain positive and negative samples. While this is feasible in simulated environments like Minecraft, in other domains, such as robotics, the cost of interacting with the environment can be very high, or opportunities for interaction may be limited (due to the risk of damage to the robots). In such cases, PGT may not be suitable.

Potentials Our method holds significant potential. First, all of our experiments were conducted in the Minecraft environment, but there are many instruction-following policies in the robotics domain as well. We believe that PGT could also achieve promising results in robotics. Second, the current experiments only cover several simple long-horizon task, like building a large chest from scratch. We are thrilled to explore how PGT can help solving longer and more complex tasks in Minecraft, like the ultimate goal: Killing the Ender Dragon.

7 Conclusion

We have introduced a framework named ***P**reference **G**oal-**T**uning* (PGT), which is an efficient post training method for foundation policies. It utilize a small amount of human preference data to fine-tuning *goal latent* in goal-conditioned policies. PGT significantly enhances the capability of the foundation policy with minimal data and training, easily surpass best human-selected instructions. Our method also demonstrates potential for acquiring new skills and serving as an efficient method for Continual Learning.

REFERENCES

Riad Akrour, Marc Schoenauer, Michèle Sebag, and Jean-Christophe Souplet. Programming by feedback. In *International Conference on Machine Learning*, volume 32, pp. 1503–1511. JMLR. org, 2014.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Bowen Baker, Ilge Akkaya, Peter Zhokhov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos, 2022. URL `https://arxiv.org/abs/2206.11795`.

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, Alexander 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, 2023a. URL `https://arxiv.org/abs/2307.15818`.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale, 2023b. URL `https://arxiv.org/abs/2212.06817`.

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`.

Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. Groot: Learning to follow instructions by watching gameplay videos, 2023.

Shaofei Cai, Bowei Zhang, Zihao Wang, Xiaojian Ma, Anji Liu, and Yitao Liang. GROOT-1.5: Learning to follow multi-modal instructions from weak supervision. In *Multi-modal Foundation Model meets Embodied AI Workshop @ ICML2024*, 2024. URL `https://openreview.net/forum?id=zxdi4Kdfjq`.

Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In *International conference on machine learning*, pp. 1430–1440. PMLR, 2021.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Embodiment Collaboration et al. Open x-embodiment: Robotic learning datasets and rt-x models, 2024. URL https://arxiv.org/abs/2310.08864.

Yiming Ding, Carlos Florensa, Pieter Abbeel, and Mariano Phielipp. Goal-conditioned imitation learning. *Advances in neural information processing systems*, 32, 2019.

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. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.

Johannes Fürnkranz, Eyke Hüllermeier, Weiwei Cheng, and Sang-Hyeun Park. Preference-based reinforcement learning: a formal framework and a policy iteration algorithm. *Machine learning*, 89:123–156, 2012.

Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, et al. Reinforced self-training (rest) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

William H. Guss, Brandon Houghton, Nicholay Topin, Phillip Wang, Cayden Codel, Manuela Veloso, and Ruslan Salakhutdinov. Minerl: A large-scale dataset of minecraft demonstrations, 2019. URL https://arxiv.org/abs/1907.13440.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Jiwoo Hong, Noah Lee, and James Thorne. Reference-free monolithic preference optimization with odds ratio. *arXiv preprint arXiv:2403.07691*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

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, 2022. URL https://arxiv.org/abs/2202.02005.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *Ijcai*, volume 16, pp. 4246–4247, 2016.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. URL https://arxiv.org/abs/2304.02643.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526, March 2017. ISSN 1091-6490. doi: 10.1073/pnas.1611835114. URL http://dx.doi.org/10.1073/pnas.1611835114.

W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, 2009.

Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024. URL https://arxiv.org/abs/2310.11454.

Shalev Lifshitz, Keiran Paster, Harris Chan, Jimmy Ba, and Sheila McIlraith. Steve-1: A generative model for text-to-behavior in minecraft. 2023.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning, 2022. URL https://arxiv.org/abs/1706.08840.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024a. URL https://arxiv.org/abs/2305.18290.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024b. URL https://arxiv.org/abs/2305.18290.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022. URL https://arxiv.org/abs/2205.06175.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *arXiv preprint arXiv:2404.14367*, 2024a.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. Preference fine-tuning of llms should leverage suboptimal, on-policy data. *arXiv preprint arXiv:2404.14367*, 2024b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Will we run out of data? limits of llm scaling based on human-generated data, 2024. URL https://arxiv.org/abs/2211.04325.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648. Springer, 2022.

Zihao Wang, Shaofei Cai, Anji Liu, Yonggang Jin, Jinbing Hou, Bowei Zhang, Haowei Lin, Zhaofeng He, Zilong Zheng, Yaodong Yang, Xiaojian Ma, and Yitao Liang. Jarvis-1: Open-world multi-task agents with memory-augmented multimodal language models. *arXiv preprint arXiv: 2311.05997*, 2023.

Sherry Yang, Ofir Nachum, Yilun Du, Jason Wei, Pieter Abbeel, and Dale Schuurmans. Foundation models for decision making: Problems, methods, and opportunities, 2023. URL `https://arxiv.org/abs/2303.04129`.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models, 2022. URL `https://arxiv.org/abs/2106.10199`.

Qihang Zhang, Zhenghao Peng, and Bolei Zhou. Learning to drive by watching youtube videos: Action-conditioned contrastive policy pretraining, 2022. URL `https://arxiv.org/abs/2204.02393`.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL `https://arxiv.org/abs/1909.08593`.

A MATHEMATICAL DERIVATION

A.1 DPO DERIVATION

Assume we have a foundation model π_{ref}, a prompt x, and a pair of responses y_w, y_l, and a human labeler prefers y_w to y_l for the prompt x, denote as $y_w \succ y_l \mid x$. Following this process, we can obtain a dataset of preference $\mathcal{D} = \left\{x^{(i)}, y_w^{(i)}, y_l^{(i)}\right\}_{i=1}^{N}$ Starting from original RL-Based optimization objective:

$$\max_{\pi_\theta} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi_\theta(y|x)}\big[r_\phi(x,y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi_\theta(y \mid x) \;||\; \pi_{\text{ref}}(y \mid x)\big], \tag{3}$$

Under any reward function $r(x,y)$, reference model π_{ref} and a general non-parametric policy class. We have:

$$\begin{aligned}
&\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}, y\sim\pi}\big[r(x,y)\big] - \beta\mathbb{D}_{\text{KL}}\big[\pi(y|x) \;||\; \pi_{\text{ref}}(y|x)\big] \\
=&\max_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[r(x,y) - \beta\log\frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right] \\
=&\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\exp(\frac{r(x,y)}{\beta})\pi(y|x)}{\pi_{\text{ref}}(y|x)}\right] \\
=&\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right)} - \log Z(x)\right]
\end{aligned} \tag{4}$$

where we have partition function:

$$Z(x) = \sum_{y}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right).$$

Note that the partition function is a function of only x and the reference policy π_{ref}, but does not depend on the policy π. We can now define

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right),$$

which is a valid probability distribution as $\pi^*(y|x) \geq 0$ for all y and $\sum_y \pi^*(y|x) = 1$. Since $Z(x)$ is not a function of y, we can then re-organize the final objective in Eq 4 as:

$$\min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{E}_{y\sim\pi(y|x)}\left[\log\frac{\pi(y|x)}{\pi^*(y|x)}\right] - \log Z(x)\right] \tag{5}$$

$$= \min_{\pi} \mathbb{E}_{x\sim\mathcal{D}}\left[\mathbb{D}_{\text{KL}}(\pi(y|x) \mid\mid \pi^*(y|x)) - \log Z(x)\right] \tag{6}$$

Now, since $Z(x)$ does not depend on π, the minimum is achieved by the policy that minimizes the first KL term. Gibbs' inequality tells us that the KL-divergence is minimized at 0 if and only if the two distributions are identical. Hence we have the optimal solution:

$$\pi^*(y|x) = \frac{1}{Z(x)}\pi_{\text{ref}}(y|x)\exp\left(\frac{1}{\beta}r(x,y)\right) \tag{7}$$

for all $x \in \mathcal{D}$. This completes the derivation.

We choose Bradly-Terry(BT) to model preference as:

$$p(y_1 \succ y_2 \mid x) = \frac{\exp\left(r(x,y_1)\right)}{\exp\left(r(x,y_1)\right) + \exp\left(r(x,y_2)\right)}. \tag{8}$$

So the oracle reward function of human preference r can be represented as:

$$r(x,y) = \beta\log\frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta\log Z(x) \tag{9}$$

Substituting Eq. 9 into Eq. 8 we obtain:

$$\begin{aligned} p(y_1 \succ y_2|x) &= \frac{\exp\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta\log Z(x)\right)}{\exp\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} + \beta\log Z(x)\right) + \exp\left(\beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} + \beta\log Z(x)\right)} \\ &= \frac{1}{1 + \exp\left(\beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)} - \beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)}\right)} \\ &= \sigma\left(\beta\log\frac{\pi^*(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta\log\frac{\pi^*(y_2|x)}{\pi_{\text{ref}}(y_2|x)}\right). \end{aligned}$$

Through derivation, the optimization objective becomes:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta;\pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta\log\frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right)\right]. \tag{10}$$

A.2 PGT LOSS

Our PGT method is based on modeling DPO with a sequential decision-making process. We define a N step trajectory as $\tau = (s_t, a_t)_{t=1}^N, s_t \in \mathcal{S}, a_t \in \mathcal{A}$. In the context of a given task, given a trajectory τ, we assume that there exists a Bradley-Terry-Model-liked oracle reward function $r(\tau)$ evaluating the merits of this trajectory. The task can be subjective or objective. We have the following objective:

$$\max_{g} \mathbb{E}_{\tau\sim\pi(\tau;g)}\left[r(\tau)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(\tau;g) \mid\mid \pi(\tau;g_{\text{ref}})\right] \tag{11}$$

By applying the same derivation method as DPO, we have:

$$\min_{g} \mathbb{E}_{\tau\sim\pi(\tau;g)}\left[r(\tau)\right] - \beta\mathbb{D}_{\text{KL}}\left[\pi(\tau;g) \mid\mid \pi(\tau;g_{\text{ref}})\right] \tag{12}$$

$$= \min_{g} \beta\mathbb{E}_{\tau\sim\pi(\tau;g)}\left[\log\frac{\exp(\frac{r(\tau)}{\beta})\pi(\tau;g_{\text{ref}})}{\pi(\tau;g)}\right] \tag{13}$$

which has the closed form of the optimal g^* that meets

$$\pi(\tau; g^*) = \frac{\exp(\frac{r(\tau)}{\beta})\pi(\tau; g_{\text{ref}})}{Z} \tag{14}$$

where $Z = \sum_\tau \pi(\tau; g_{\text{ref}}) \exp\left(\frac{1}{\beta} r(\tau)\right)$. So we can obtain

$$r(\tau) = \beta \log \frac{\pi(\tau; g^*)}{\pi(\tau; g_{\text{ref}})} + \beta \log Z \tag{15}$$

Consider the Bradly-Terry(BT) model:

$$p(\tau_1 \succ \tau_2) = \frac{\exp(r(\tau_1))}{\exp(r(\tau_1)) + \exp(r(\tau_2))}. \tag{16}$$

fill Eq. 15 into Eq. 16, we have:

$$p(\tau_1 \succ \tau_2) = \sigma\left(\beta \log \frac{\pi(\tau_1; g^*)}{\pi(\tau_1; g_{\text{ref}})} - \beta \log \frac{\pi(\tau_2; g^*)}{\pi(\tau_2; g_{\text{ref}})}\right). \tag{17}$$

Decompose τ into factors, $p(\tau) = p(s_1)\left[\Pi_{t=1}^{N-1} \pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)\right] \pi(a_N \mid s_N)$, we can get:

$$\log p(\tau^{(w)} \succ \tau^{(l)}) \tag{18}$$

$$= \log \sigma\left(\beta \sum_{t=1}^{T} \log \frac{\pi(a_t^{(w)} \mid s_t^{(w)}; g^*)}{\pi(a_t^{(w)} \mid s_t^{(w)}; g_{\text{ref}})} - \log \frac{\pi(a_t^{(l)} \mid s_t^{(l)}; g^*)}{\pi(a_t^{(l)} \mid s_t^{(l)}; g_{\text{ref}})}\right). \tag{19}$$

Finally, our optimization objective becomes:

$$\mathcal{L}_{\text{PGT}}(g, g_{\text{ref}}) = -\mathbb{E}_{(\tau^{(w)}, \tau^{(l)}) \sim \mathcal{D}}\left[\log \sigma\left(\beta \sum_{t=1}^{T} \log \frac{\pi(a_t^{(w)} \mid s_t^{(w)}; g)}{\pi(a_t^{(w)} \mid s_t^{(w)}; g_{\text{ref}})} - \log \frac{\pi(a_t^{(l)} \mid s_t^{(l)}; g)}{\pi(a_t^{(l)} \mid s_t^{(l)}; g_{\text{ref}})}\right)\right]. \tag{20}$$

B Experiment Details

B.1 Minecraft

Minecraft is a popular sandbox game that allows players to freely create and explore their world. Since Minecraft is an open-world environment, many recent works have designed agents and conducted explorations within Minecraft (Johnson et al., 2016). In this work, we conduct experiments on 1.16.5 version MineRL (Guss et al., 2019) and MCP-Reborn.

B.2 Minecraft SkillForge Benchmark

Minecraft SkillForge Benchmark is a comprehensive task suite that covers various types of tasks in Minecraft. All tasks are categorized into six major groups:

- collect task: These tasks are designed to evaluate an AI agent's capability in resource acquisition proficiency and spatial awareness.
- craft task: These tasks are designed to shed light on an AI agent's prowess in item utilization, the intricacies of Minecraft crafting mechanics, and the nuances of various game mechanic interactions.
- explore task: These tasks are designed to evaluate an AI agent's navigation proficiency, understanding of diverse environments, and intrinsic motivation for exploration.
- survive task: These tasks are designed to analyze an AI agent's ability to ensure its survival, adeptness in combat scenarios, and capability to interact with the environment to meet basic needs.

- tool task: These tasks are designed to deeply investigate an AI agent's capabilities in tool utilization, precision in tool handling, and contextual application of various tools to carry out specific tasks.
- build task: These tasks are devised to evaluate an AI agent's aptitude in structural reasoning, spatial organization, and its capability to interact with and manipulate the environment to create specific structures or outcomes.

B.3 Task Metrics and Selection

For most tasks, the environment logs the rewards when the corresponding objectives are achieved. We define tasks with a reward function greater than 0 as successful, and the frequency of successfully completing a task is referred to as the success rate. However, tasks like "collect_wood" "explore_mine" and "survive_plant" have a success rate of over 95% across different agents, and the specific values of the reward function are meaningful, reflecting the agents' capabilities in these tasks, so we use the detailed reward value as the metric.

We removed the tasks that are too easy that agents can perform a success rate of 100% while the specific value of the reward is either high enough (e.g. collect_grass) or not meaningful (e.g. survive_sleep). Also, to simplify the experiment, We removed the tasks for which the reward function cannot be directly obtained from the game, including subjective tasks (e.g. building tasks) and objective tasks where the environment does not log explicit rewards (e.g. craft_smelt). Moreover, mining obsidian is a high requirement for the agent's sensitivity to the objectives, and the agent needs to stay focused on the same goal over extended time steps to perform useful actions; therefore, we consider this task to be quite important and add it to the testing tasks apart from *Minecraft SkillForge*.

B.4 Out of Distribution settings

We designed the Out of Distribution (OOD) setting with the goal of preventing the policy from overfitting to the environment and relying on it to dictate behavior. Thus, without altering the core meaning of the tasks, we made the following modifications to create the OOD setting:

- **seed** We change the seed of the Minecraft world.
- **agent location** We change the initial location of agent to perform a task.
- **biome** We change the biome of agent while keeping the task solvable. For example, change biome to savanna of task `collet wood`.
- **tool** We modified the auxiliary tools while ensuring the tasks remained solvable. For example, in the `hunt sheep`, we replaced the sword with an axe.
- **object location** We change the location of object that the agent need to interact. For example, we changed the position of the stonecutter from being held in the hand to being placed in front of the agent.

For each task, we applied one or more of the aforementioned OOD modifications. It is important to note that the absolute performance in the OOD setting is not directly comparable to the baseline, as the tasks may become either easier or harder in the OOD environment.

C Experiment Results

C.1 Behaviour Cloning Results

This baseline employs behavior cloning, trained exclusively on positive samples, without the inclusion of negative data or preference learning. We present results for both tuning soft prompt and the full parameters (Table 1).

C.2 Full Fine-tuning Results

We compare the results of our method with full fine-tuning. The latter involves $\sim$100M parameters, while the former only has 512 parameters, which is merely one in hundreds of thousands of the

Table 6: Comparisons between tuning soft prompt and full fine-tuning. The soft prompt method can bring better improvements than the counterpart, especially on OOD settings.

Task	In Distribution (ID)			Out of Distribution (OOD)		
	Pretrained	Full	Soft prompt	Pretrained	Full	Soft prompt
collect_wood	3.14	3.46	**3.62**	3.88	4.04	**4.22**
craft_stonecutter	31.0	**62.2**	44.6	20.0	21.2	**23.4**
explore_mine	4.91	6.00	**6.58**	3.90	4.77	**5.38**
tool_pumpkin	48.3	**58.4**	57.8	16.6	22.2	**25.8**
survive_hunt	31.2	**39.8**	**39.8**	20.8	21.0	**21.6**
obsidian	42.0	**62.2**	57.2	4.2	6.0	**8.2**

Table 7: Parameter efficient fine-tuning result.

Task	In Distribution(ID)				Out of Distribution(OOD)			
	LoRA	BitFit	VeRA	PGT	Lora	BitFit	VeRA	PGT
collect_wood	3.47	3.55	3.39	**3.62**	4.09	3.91	4.16	**4.22**
craft_stonecutter	49.4	48.6	**52.2**	44.6	19.8	18.0	18.8	**23.4**
explore_mine	6.52	5.37	5.76	**6.58**	5.17	4.42	4.67	**5.38**
survive_hunt	39.8	40.8	**42.0**	39.8	24.6	25.2	**27.4**	21.6
tool_pumpkin	50.4	56.2	52.8	**57.8**	19.6	20.8	22.4	**25.8**
collect_obsidian	**71.2**	55.8	57.8	57.2	**10.6**	6.2	2.6	8.2

other. We found that in in-distribution settings, the soft prompt method achieves results comparable to those of full fine-tuning. However, in out-of-distribution (OOD) environments, soft prompt tuning outperformed across all tasks. Result can be found in table 6.

C.3 Parameter Efficient Fine-tuning Results

We conduct parameter efficient fine-tuning on Lora (Hu et al., 2021), BitFit (Zaken et al., 2022), VeRA (Kopiczko et al., 2024), and the result is in Table 7.

C.4 Continue Learning Results

All of our continual learning baselines are based on fine-tuning the entire policy model, and the order of tasks for continual learning is as follows: `collect obsidian`() to `tool pumpkin`() to `craft craftingtable`() to `explore climb`(). We implemented Multi-Task Learning (MTL) (Table 3), Naive Continual Learning (NCL) (Table 8), Knowledge Distillation (KD)(Table 9), Experience Replay (ER)(Table 10), and Elastic Weight Consolidation (EWC)(Table 11).

Table 8: Continue Learning: Naive Continue Learning

Task	collect obsidian	tool pumpkin	craft table	expolore climb	pretrain	PGT
collect obsidian	6.0	4.6	7.0	6.8	4.2	**8.2**
tool pumpkin		23.6	24.2	20.4	16.6	**25.8**
craft table			5.2	7.0	6.0	**18.4**

Table 9: Continue Learning: Knowledge Distillation

Task	collect obsidian	tool pumpkin	craft table	expolore climb	pretrain	PGT
collect obsidian	6.0	5.2	6.6	5.4	4.2	**8.2**
tool pumpkin		24.6	23.4	20.6	16.6	**25.8**
craft table			7.6	5.8	6.0	**18.4**

Table 10: Continue Learning: Experience Replay

Task	collect obsidian	tool pumpkin	craft table	expolore climb	pretrain	PGT
collect obsidian	6.0	6.6	5.0	6.0	4.2	**8.2**
tool pumpkin		22.8	21.8	25.0	16.6	**25.8**
craft table			5.2	9.0	6.0	**18.4**

Table 11: Continue Learning: Elastic Weight Consolidation

Task	collect obsidian	tool pumpkin	craft table	expolore climb	pretrain	PGT
collect obsidian	6.0	8.2	5.4	5.4	4.2	**8.2**
tool pumpkin		23.6	24.0	23.8	16.6	**25.8**
craft table			5.0	7.4	6.0	**18.4**