Guiding Skill Discovery with Foundation Models

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning diverse skills without hand-crafted reward functions could accelerate reinforcement learning in downstream tasks. However, existing skill discovery methods focus solely on maximizing the diversity of skills without considering human preferences, which leads to undesirable behaviors and possibly dangerous skills. For instance, a cheetah robot trained using previous methods learns to roll in all directions to maximize skill diversity, whereas we would prefer it to run without flipping or entering hazardous areas. In this work, we propose a **Fo**undation model **G**uided (FoG) skill discovery method, which incorporates human intentions into skill discovery through foundation models. Specifically, FoG extracts a score function from foundation models to evaluate states based on human intentions, assigning higher values to desirable states and lower to undesirable ones. These scores are then used to re-weight the rewards of skill discovery algorithms. By optimizing the re-weighted skill discovery rewards, FoG successfully learns to eliminate undesirable behaviors, such as flipping or rolling, and to avoid hazardous areas in both state-based and pixel-based tasks. Interestingly, we show that FoG can discover skills involving behaviors that are difficult to define. Interactive visualisations are available from `https://sites.google.com/view/submission-fog`.

1 Introduction

Reinforcement learning (RL) has shown promising results in robotics [40, 46] and games [44, 48]. Typically, RL requires carefully designed reward functions, which demand significant expert efforts [36, 39]. In contrast, Unsupervised RL [16, 33] aims to eliminate task-specific reward functions and train agents in a self-supervised manner. One key direction in unsupervised RL is pre-training agents to acquire diverse skills that can potentially be useful in downstream tasks [5, 30], termed unsupervised skill discovery. Most prior methods in unsupervised skill discovery focus on maximizing skill diversity, encouraging agents to achieve diversity in both low-level behaviors and high-level policies. For instance, a cheetah robot trained using previous methods [27, 30] learns to flip or roll (low-level behavior) in all directions (high-level policy). However, wide motions like flipping or rolling could damage the robot, and entering restricted areas might pose safety risks. Ideally, we want agents to learn skills that are not only diverse, but also aligned with specific intentions, such as eliminating undesirable behaviors or avoiding certain areas.

To integrate human intentions into skill discovery, we introduce a **Fo**undation model **G**uided (FoG) method. More specifically, FoG (see Figure 1) utilizes foundation models [32, 24, 3] to assign higher scores for desirable behaviors and lower for undesirable ones. These scores are then used to re-weight the rewards of unsupervised skill discovery algorithms. By optimizing these re-weighted rewards, FoG learns diverse skills while aligning with given human intentions. FoG stands out from previous methods by being more autonomous, as it does not rely on costly expert demonstrations like [11],

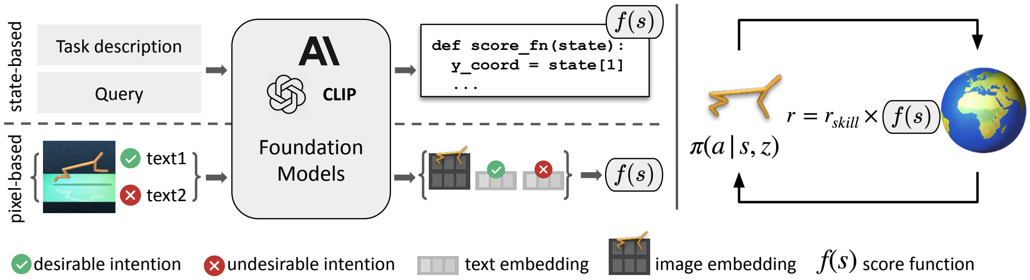

Figure 1: FoG leverages foundation models (such as ChatGPT, Claude and CLIP) to score states in relation to given commands during training. These scores are used to re-weight the rewards of the underlying skill discovery algorithm. **Left**: In state-based tasks (top row), task descriptions are provided to foundation models, which are queried to generate a score function $f(s)$ based on our requirements. In pixel-based tasks (bottom row), the current visual state, textual descriptions of desirable and undesirable intentions are input to foundation models to obtain embeddings. These embeddings are then used to form the score function $f(s)$, see Equation (8). **Right**: During training, rewards of the underlying skill discovery method (r_{skill}) are re-weighted using the score function. Re-weighting r_{skill} (we use METRA [30]) by the score function is equivalent with using the score function as the distance metric in the DSD objective.

and more versatile, as it works with both visual inputs and compact state information, unlike [34], which requires precise ground-truth states.

Our main contributions are threefold: 1) We introduce a novel foundation model guided unsupervised skill discovery method (FoG), which learns diverse and desirable skills. 2) We evaluate FoG alongside six state-of-the-art baselines on both state-based (i.e. structured, low-dimensional representations) and pixel-based tasks. FoG outperforms baselines in both scenarios, showcasing superior input-agnostic generalization capabilities. 3) We show FoG can learn behaviors that are challenging to define, such as being ‘twisted’ and ‘stretched’ on a humanoid robot, suggesting its potential for more complex applications. The FoG codebase can be found in the supplemental materials.

2 Preliminaries and Problem Setting

We consider a reward-free Markov Decision Process defined as $\mathcal{M} = (\mathcal{A}, \mathcal{S}, p)$. $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space and p is the transition dynamics function. A latent vector $z \in \mathcal{Z}$ (also called ‘skill’) is sampled during training and its conditioned policy $\pi(\cdot|s,z)$ is executed to get a skill trajectory $\tau = (s_0, s_1, ..., s_T)$ following the process: $p(\tau|z) = p(s_0)\prod_{t=0}^{T-1}\pi(a_t|s_t,z)p(s_{t+1}|s_t,a_t)$. $\pi(\cdot|s,z)$ can be learned by optimizing unsupervised exploration objectives we discuss below (distance-maximization) or in Section 5 (mutual information).

FoG utilizes the Distance-maximizing Skill Discovery (DSD) [29] objective. Unlike mutual information based methods [5], DSD aims to maximize the Wasserstein dependency measure (WDM) [25] defined as:

$$I_{\mathcal{W}}(S;Z) = \mathcal{W}(p(s,z), p(s)p(z)), \tag{1}$$

where $\mathcal{W}$ is the 1-Wasserstein distance on the metric space $(S \times Z, d)$ for distance metric d. By maximizing the objective in Equation (1), the agent will not only maximize the diversity of skills, but also maximize the distance metric d [30]. Under some simplifying assumptions [25, 45], maximization of Equation (1) can then be rewritten as:

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)}\left[\sum_{t=0}^{T-1}(\phi(s')-\phi(s))^{\top} z\right] \quad \text{s.t.} \;\; \|\phi(x)-\phi(y)\|_2 \le d(x,y), \;\; \forall (x,y) \in S, \tag{2}$$

where ϕ is a function that maps states to a D-dimensional space, which is the same as the skill space Z. Intuitively, Equation (2) aims to align the direction of z and $\phi(s') - \phi(s)$ (to learn distinguishable and diverse skills), while maximizing the length of $||\phi(s') - \phi(s)||$, which leads to an increase in the distance between states based on the given distance metric d due to the Lipschitz constraint [29]. In principle, $d(x,y)$ in Equation (2) can be replaced by any of the distance metrics in Table 1, resulting in different unsupervised skill discovery methods. Equation (2) can be optimized with dual gradient descent, incorporating a Lagrange multiplier λ and a small slack variable $\epsilon > 0$:

$$\text{Update } \phi \text{ to maximize:} \quad \mathbb{E}[(\phi(s') - \phi(s))^\top z] + \lambda \cdot \min(\epsilon, d(s, s') - \|\phi(s) - \phi(s')\|) \tag{3}$$

$$\text{Update } \lambda \text{ to minimize:} \quad \lambda \cdot \mathbb{E}[\min(\epsilon, d(s, s') - \|\phi(s) - \phi(s')\|)] \tag{4}$$

$$\text{Update } \pi \text{ with reward:} \quad (\phi(s') - \phi(s))^\top z \tag{5}$$

For derivation of these equations we refer to [27, 29, 30].

3 Foundation Model Guided Skill Discovery

FoG extracts a score function from foundation models based on human intentions to re-weight skill discovery rewards, illustrated in Figure 1. For state-based tasks, the foundation model is queried to output a score function aligned with our intentions. In pixel-based tasks, the score function is formed using state and intentional text embeddings from the foundation models. The skill-conditioned policy is then trained to maximize these re-weighted rewards during unsupervised skill discovery.

3.1 Score Function

We extract a score function from foundation models that can assign higher values for desirable states and lower values for undesirable states with respect to the given intentions. This score function is then used to re-weight rewards of the underlying skill discovery method. We define the score function $f : S \to [0, 1]$ which takes a state as input and outputs a value between 0 and 1, indicating the desirability of the given state. This score function is then used to reweight the skill discovery rewards. The skill discovery reward r_{skill} of Equation (5) therefore becomes:

$$r = f(s') \times r_{skill} = f(s')(\phi(s') - \phi(s))^\top z, \tag{6}$$

where we care about the states s' the agent reaches instead of the state s the agent comes from. Thus, the score function f takes s' as the input. Since we use METRA [30] as the underlying skill discovery algorithm, and use the score function to re-weight the METRA rewards, this is equivalent to using it as the distance metric in the DSD objective:

$$\sup_{\pi, \phi} \mathbb{E}_{p(\tau, z)} \left[\sum_{t=0}^{T-1} (\phi(s_{t+1}) - \phi(s_t))^\top z \right] \quad \text{s.t.} \quad \|\phi(s) - \phi(s')\|_2 \leq f(s'), \quad \forall (s, s') \in S_{adj}, \tag{7}$$

where S_{adj} represents the set of adjacent state pairs. The derivation of Equation (7) can be found in Appendix A. By using the score function as the distance metric in the DSD objective, FoG not only maximizes the diversity of skills, but also maximizes the output of the score function, leading to skills that are more aligned with our intentions.

In practice, we find that a binary score function works well, i.e. outputting 1 if the state is desirable and α if it is not, where $0 \leq \alpha < 1$. We examine different values of α and a non-binary score function in Section 4.2.4.

3.2 Implementation Details

Our work builds on top of METRA [30], which is the state-of-the-art unsupervised skill discovery method that works for both state-based and pixel-based input. FoG re-weights the skill discovery reward of METRA by the score function that is extracted from foundation models. For state-based tasks, we ask foundation models to generate the score function directly. For pixel-based tasks, we use foundation models to output embeddings to form the score function. All code is available through the supplemental materials.

State-based: We ask ChatGPT or Claude to generate a score function $f(s)$ that equals 1 if the state satisfies our intentions, and α otherwise. Unlike Eureka [21], which queries foundation models to generate a reward function for training agents from scratch, FoG instead asks for a score function to modulate skill discovery. Prompt details for state-based tasks and examples of resulting output score functions are provided in Appendix F.7.1.

Pixel-based: We use CLIP [32], a vision-language model that is trained to align images and text, to first generate embeddings for images (pixel-based states) and texts (textual descriptions of our intentions). Then, the score function is formed by computing the Cos similarity between the image and text embedding. If the current state is more similar to the description of the desirable intention, the output is 1. Conversely, if it is more similar to the undesirable one, the output is α. The score function can be expressed as Equation (8).

$$f(s) = \begin{cases} 1, & \text{if } Cos(E_s, E_{t1}) > Cos(E_s, E_{t2}). \\ \alpha, & \text{otherwise.} \end{cases} \tag{8}$$

where E_s is the embedding of the current pixel-based state, E_{t1} and E_{t2} are the embeddings of the textual descriptions of desirable and undesirable intentions, respectively. Setting $\alpha = 0$ attempts to not learn undesirable behaviors at all (since $\alpha \times r_{skill} = 0$) while setting $\alpha = 1$ reduces FoG to the underlying skill discovery algorithm METRA. We examine different values of α in Section 4.2. Details of textual descriptions of desirable and undesirable intentions can be found in Appendix F.7.2.

4 Experiments

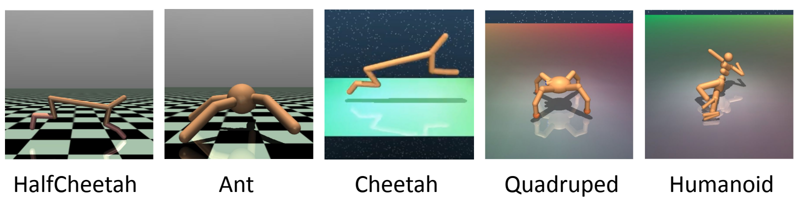

Figure 2: Environments used in our work. HalfCheetah and Ant are state-based while the other three are pixel-based.

Our experiments aim to address the following questions:

- How does FoG perform in state-based tasks where more context and informative features are provided? (Section 4.1)
- In pixel-based tasks, where only visual information is provided, can FoG guide agents to learn diverse and desirable behaviors and skills? (Section 4.2)

We use common environments in unsupervised skill discovery literature, see Figure 2, including two state-based tasks and three pixel-based tasks: HalfCheetah and Ant are state-based tasks from OpenAI gym [4], Cheetah, Quadruped and Humanoid are pixel-based tasks from DMC [42].

We have six baselines for FoG to compare against:

- METRA [30], the state-of-the-art unsupervised skill discovery method.
- METRA+, which integrates human intentions through hand-coded reward functions, and was also used as a baseline in DoDont [11].
- LSD [27], an unsupervised skill discovery method that maximizes DSD objective with Euclidean distance as the distance metric.
- DoDont [11], a demonstration-guided unsupervised skill discovery method, learns diverse and desirable behaviors shown in the demonstrations. In some cases, it needs additional state-based inputs alongside with pixel-based input to work properly, more details can be found in Appendix D.
- DoDont+, a variant of DoDont that replaces expert demonstrations with demonstrations annotated using foundation models.
- FR-SAC, a SAC [9] agent rewarded using scores from foundation models (**F**oundation **R**ewards) using Equation (13).

All agents in the same task are trained with the same number of environment steps and all experiments are performed three times with different independent seeds, and average results with error bars are reported. For simplicity, we set $\alpha = 0$ for all experiments. Details about environments and baseline implementations can be found in Appendix F. See website[1] for videos of the learned behaviors and skills.

4.1 State-based Tasks

To test whether FoG can work in state-based tasks, we train FoG in HalfCheetah and Ant. Following the details in Section 3.2, we input the description of the tasks, information about state space and action space to foundation models as context, then ask foundation models to generate a score function that returns 1 when the requirement in the query is satisfied otherwise α. In HalfCheetah, we train FoG to eliminate dangerous behaviors (flipping over). In Ant, we train FoG to avoid a specific area, in this case to not go south.

Results of these experiments are visualized in Figure 3, with generated score functions for both tasks at the right. We first of all see that foundation models can recognize feature dimensions of the state that are important for meeting our requirements. For example, in HalfCheetah the second dimension of the state is the angle of Cheetah's front tip, which is important for determining if the agent flips over or not. In Ant, the first dimension of the state is the y-coordinate of Ant, which can be used to locate the agent in a south-north position. We see foundation models clearly set the right threshold and implement the logic to fulfil the intention we asked for, i.e., if the angle of the Cheetah's front tip is larger than 1.57 in radians (90 degrees) it flips over, and if the y-coordinate of Ant is larger than 0 it is in the north part of the plane. By re-weighting the skill discovery rewards using the generated score function from foundation models, FoG learns to not roll in HalfCheetah while METRA flips a lot (left sub-figure of Figure 3). In Ant, FoG learns to always move to north and METRA learns to go in every direction (mid-left part in Figure 3).

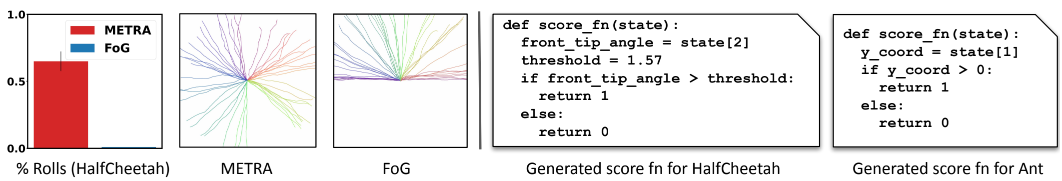

Figure 3: Comparison between METRA and FoG on state-based HalfCheetah and Ant. In both tasks, foundation models successfully capture the relevant state dimension and set threshold for it. **Left**: FoG learns not to roll in HalfCheetah, while METRA rolls over 50% of the time, violating our intention. **Right**: FoG learns to not move to south in Ant, and METRA learns to move in all directions.

4.2 Pixel-based Tasks

We now conduct experiments in pixel-based tasks, where only visual information is available. Unlike in state-based tasks, where we ask foundation models to directly generate a score function, in pixel-based tasks we leverage foundation models to output embeddings of 1) the visual state and 2) textual descriptions of our desirable and undesirable intentions. The score function is then computed from Equation (8). We examine FoG in four aspects:

- Can FoG learn diverse skills while eliminating undesirable behaviors? (Section 4.2.1)
- Can FoG learn diverse skills without entering certain areas? (Section 4.2.2)
- Can FoG learn complex behaviors that are difficult to clearly define? (Section 4.2.3)
- What are the most critical design choices of FoG? (Section 4.2.4)

4.2.1 Learn to eliminate undesirable behaviors

We first focus on guiding the agent to learn desirable low-level behaviors (e.g., standing normal) while eliminating undesirable ones (e.g., flipping over) that could potentially damage the robot. In

[1] `https://sites.google.com/view/submission-fog`

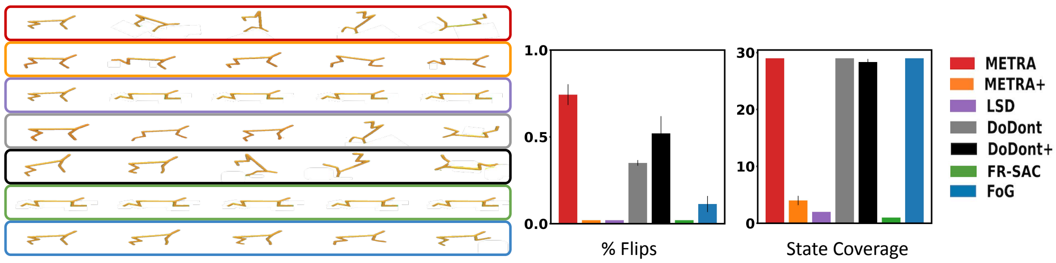

Figure 4: **Left**: Executions of example skills from different agents in pixel-based environment, Cheetah. From top to bottom: METRA, METRA+, LSD, DoDont, DoDont+, FR-SAC, FoG. **Right**: Percentage of flips (which should be prevented based on the guidance) and state coverage for different agents. METRA, METRA+, DoDont, and DoDont+ discover diverse states but often flip. LSD and FR-SAC fail to learn diverse skills. FoG excels with high state coverage and minimal flipping.

pixel-based Cheetah, we use `‘agent flips over’` and `‘agent stands normally’` as textual descriptions to express our intentions.

As shown in the left part of Figure 4, FoG (bottom) consistently learns to run without flipping, demonstrating the lowest percentage of flips during evaluation. In contrast, other methods struggle to prevent flipping effectively. METRA flips in over 70% of episodes, DoDont in more than 35%, and DoDont+ in 50% of the episodes. LSD, FR-SAC and METRA+ struggle to learn to move in different directions, discovering static behaviors and rarely flipping. Although METRA, DoDont, DoDont+ and FoG achieve similar state coverage, FoG effectively prevents flipping.

The poor performance of METRA+ suggests that defining a proper score function manually is not trivial (we follow the definition in [11] and use $r_{run} - r_{flip}$ as the score function). The poor performance of DoDont stems from the inaccurate classifier, which exploits the color of the ground to distinguish different states (normal and flipping postures), outputting high scores for unseen undesirable behaviors. A more in-depth analysis on the failure of DoDont can be found in Appendix D. FR-SAC fails to learn meaningful behaviors, suggesting only using foundation model scores to train RL agents is insufficient (see more analysis in Section 4.2.4). To evaluate how these learned skills perform in downstream tasks, we train a controller to select from the learned set of skills. This controller trained using FoG skills shows quick adaptations in the downstream tasks, as shown in Appendix C.

4.2.2 Learn to avoid hazardous areas

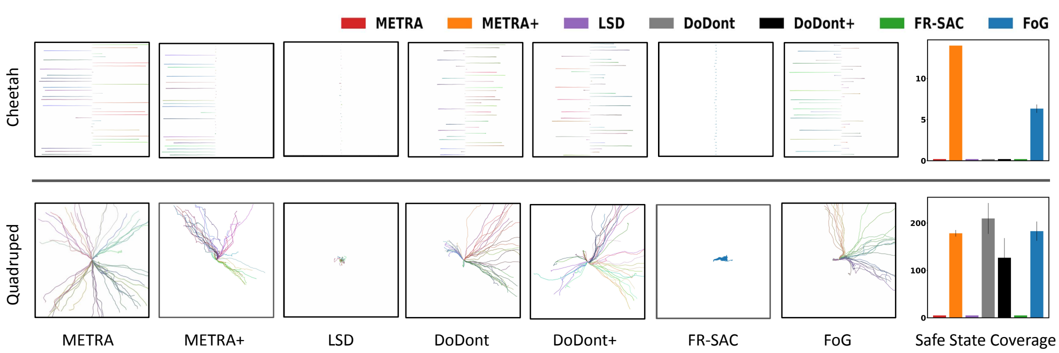

Figure 5: **Top**: Results on the pixel-based environment Cheetah, with learned skills shown in x-coordinates. METRA+ learns to perfectly avoid the undesirable area and FoG has a strong preference to go to the desirable area, as also clearly visible from the Safe State Coverage on the right. Other agents fail. **Bottom**: Results on the pixel-based environment Quadruped, with learned skills shown as xy-coordinates. Similar conclusions can be drawn regarding most of agents. Unlike in Cheetah, DoDont successfully learns to avoid the bottom-left areas.

Previous methods focus solely on maximizing skill diversity, often leading agents to explore in all possible directions. In practice, however, we want agents to avoid certain areas when they are hazardous. For instance, a robot operating in a factory should be able to avoid prohibited areas. To test whether FoG can learn to avoid certain areas (high-level policies, as opposed to low-level behaviors in Section 4.2.1), we train FoG in the pixel-based versions of Cheetah and Quadruped. We designate the right area in Cheetah and the bottom-left area in Quadruped as hazardous and train agents to avoid them. Since there are no explicit indicators of directions in these two tasks, we express our intentions through colors. For example, in Cheetah, we use descriptions like `‘ground is blue’` and `‘ground is orange’` to signal whether the agent is on the left or right part and then form the score function following Equation (8). Figure 5 illustrates the learned skills and ‘Safe State Coverage’ (the coverage of safe areas minus that of hazardous areas) of different agents. FoG clearly biases movement toward the safe areas. In Cheetah it prefers to go to the left part and in Quadruped it avoids the bottom-left area, resulting in higher safe state coverage than the baselines. In contrast, METRA explores all directions indiscriminately, LSD and FR-SAC fail to move, leading to the lowest safe state coverage. DoDont performs well in Quadruped but not in Cheetah (the classifier are unsure about initial states thus harm the exploration). The slightly worse performance of DoDont+ (compared to DoDont) in Quadruped stems from its inaccurate demonstrations annotated by foundation models. METRA+ performs the best, likely because that defining a score function in these tasks is straightforward (assigning 1 to states in safe regions and 0 for ones in hazardous regions [11]). The results suggest that with expert-level demonstrations and ‘perfect’ hand-crafted score function, DoDont and METRA+ could potentially outperform FoG. However, the strength of FoG shines in scenarios where obtaining expert-level demonstrations or crafting a perfect score function is challenging, which is generally the case.

Non-expert demonstrations (like ones annotated by foundation models, which are used in DoDont+) introduce inaccuracies to the classifier, with annotation accuracy around 70%. This leads to an inaccurate classifier that consistently generates unreliable signals, ultimately resulting in poor performance. In contrast, FoG leverages CLIP on-the-fly. Although CLIP does not achieve perfect accuracy, it still allows the agent to learn effectively. As shown in Section 4.2.4, the more accurate the scoring function, the better the performance of FoG.

4.2.3 Learning in Humanoid

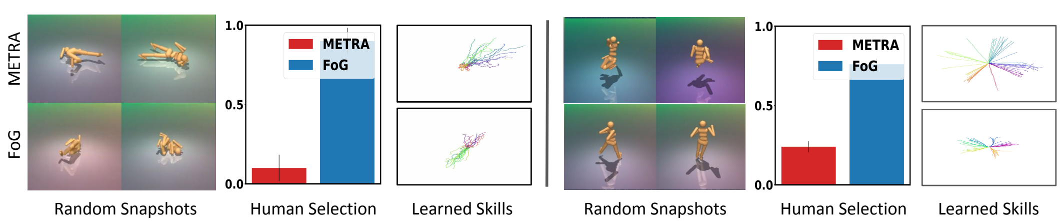

Figure 6: Learning results of METRA and FoG on Humanoid (**Left**) and Puppet (**Right**). Humans participants pick FoG to be more desirable 90% and 70% of the time in two tasks. Learned skills (shown in xy-coordinates) of different agents.

Humanoid is a challenging high-dimensional control task with a 21-D action space. Defining postures of this humanoid robot could be both hard and subjective, e.g. when it is “twisted" or “stretched", “running" or “walking", etc. This also makes it hard to design a reward function that can guide the agent to learn such behaviors. Since FoG uses foundation models, it overcomes this problem by directly evaluating whether a given frame or state is desirable—assigning higher scores to configurations like “twisted,” which we want to encourage. This allows FoG to recognize and reward subtle behaviors that are otherwise hard to specify explicitly. We could not compare FoG with DoDont [11] as the original paper does not include results on Humanoid, probably because demonstrations of a humanoid robot are challenging to obtain (an issue we also encountered).

First, we train FoG in the Humanoid task using intention descriptions `‘agent is stretched’` and `‘agent is twisted’`. To quantitatively assess whether the agent has successfully learned to twist, we create a questionnaire and ask ten human participants to evaluate videos of different agents, selecting the ones they perceive as more “twisted". Videos and the questionnaire can be found on the project website and details of the experimental setup can be found in Appendix F.5.

In the left part of Figure 6, it is clear that FoG learns to exhibits more “twisted" postures while METRA tends to appear more “stretched". The ‘Human Selection’ shows how participants perceive the trained skills, with 90% of the time participants selecting FoG as more “twisted", further validating the observed outcomes. Both FoG and METRA successfully learn to move in different directions, highlighting the diversity of the learned skills. FoG’s ability to move in different directions with “twisted" postures suggests its potential to guide agents in discovering skills involving behaviors with subjective definitions.

To further analyze FoG, we modify the ‘Humanoid’ task to a ‘Puppet’ variant, where the humanoid is pulled by a string above the head, i.e. the humanoid always keeps upright. The details of Puppet environment can be found in Appendix F.1. Besides learning diverse skills, we also ask the puppet to show running postures. See results in the right part of Figure 6. METRA learns to wriggle to all different directions with squat postures, whereas FoG learns to show more natural postures while moving in all directions. Similar to the Humanoid experiment, 70% of participants judged FoG to exhibit a more ’running’ posture. See the website for videos.

4.2.4 Ablation Study

FoG introduces two hyperparameters. The first, α in the binary score function of Equation (8), controls the re-weighting of skill discovery rewards for undesirable states. Higher values make rewards for undesirable and desirable states less distinguishable, increasing the likelihood of agents learning undesirable behaviors. We evaluate three values, $\alpha = 0, 0.5, 0.8$. As shown in the left part of Figure 7, higher α leads to more undesirable behaviors (e.g., increased flipping in the Cheetah task). Directly using similarity of visual states and textual intentions (**sim**, calculated with Equation (13)) to re-weight rewards yields poor performance. While $\alpha = 0$ works well across experiments, it may overly constrain exploration in some cases (see Appendix E.1).

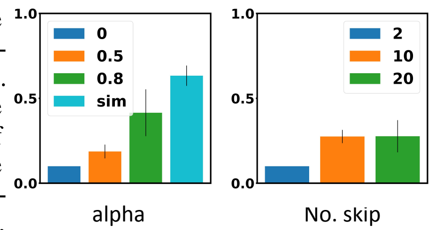

Figure 7: Percentages of flips that different FoG shows on the Cheetah environment. Smaller α and N return better performance.

In pixel-based tasks, obtaining embeddings for every pixel state is computationally expensive. Instead, embeddings are computed every Nth state, with the score applied to the following $(N-1)$ states. Smaller N values improve accuracy but increase costs. As shown in the right part of Figure 7, smaller N leads to fewer flips (better performance), but there is no significant difference between $N = 10$ and $N = 20$, suggesting behaviors in Cheetah are quite smooth thus skipping 10 or 20 states leads to similar results.

Using scores as step-wise reward signals: FoG uses foundation model scores to re-weight the unsupervised skill discovery rewards, learning diverse and desirable behaviors. However, directly optimizing these scores is not ideal. In Figure 8, scores for pre-collected episodes aligned with human intentions (‘Yes’) and misaligned ones (‘No’) reveal significant noise despite correct overall trends (we use the same textual intentions from previous experiments, i.e. Cheetah in Section 4.2.1 and Quadruped in Section 4.2.2). For example, in Cheetah, after flipping upside down at step 50, the agent consistently receives low scores. In Quadruped, scores either remain high or gradually decrease as the agent moves diagonally. This noise makes direct score optimization unreliable. As can be seen in Section 4.2.1 and Section 4.2.2, the agent trained solely with such noisy reward signals (FR-SAC) learns only static postures, resulting in low (safe) state coverages, suggesting that directly optimizing these scores is insufficient.

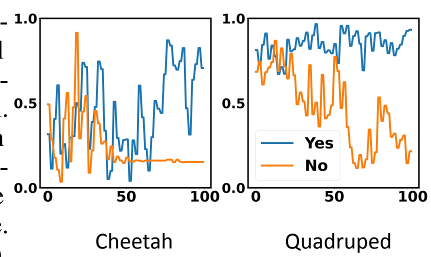

Figure 8: Scores outputted by foundation models on pre-collected episodes that are (not) aligned with human intentions.

Sensitivity to score function noise: Although FoG’s CLIP-based score function is not perfectly accurate, it still enables the agent to learn effective behaviors. To assess how performance depends on the score accuracy, we inject noise by flipping the score output $(0 \leftrightarrow 1)$ with probability b during training. As the score function becomes noisier, the percentage of flips in Cheetah increases (see Figure 9) while the state coverage remains mostly constant (all 29, except 26.7 ± 0.67 for

$b = 0.5$). These findings indicate that while FoG is robust to some noise, improved scoring enhances performance.

5 Related Work

Mutual information (MI) based unsupervised skill discovery aims to maximize MI between latent skill variables and visited states to learn diverse and distinguishable skills [5, 37, 15]. However, these methods do not always encourage the agent to discover distant states, as the MI objective can be satisfied by learning simple and static skills [30, 27]. To address this limitation, [29] introduces a Distance-maximizing Skill Discovery (DSD) framework that learns diverse skills while maximizing the traveled distance under the given distance metric d. LSD [27] uses Euclidean distance between states as the distance metric to encourage agents to visit states that are as far apart as possible. CSD [29] employs a density function over visited states as the distance metric, to encourage agents to visit less frequently visited states. However, LSD and CSD only work with state-based inputs and fail in pixel-based tasks. METRA [30] instead uses a temporal distance function that is applicable in visual tasks as well, as the distance metric to push the agent to discover states that are temporally far apart. LGSD [34] utilizes foundation models to first convert state-based inputs to text descriptions, then uses embedding distance between text descriptions as the distance metric to encourage agents to learn semantic diverse skills. DoDont [11] employs demonstrations to guide agents in learning desirable behaviors. Specifically, it trains a classifier over the demonstrations of what the agent should and should not do, and uses it as a distance metric in DSD, encouraging agents to learn to maximize intentions of the given demonstrations. Some distance metrics used by different methods are summarized in Table 1. Note that FoG can be interpreted as using a score function extracted from foundation models as the distance metric in DSD. We refer to Section 3 for further details.

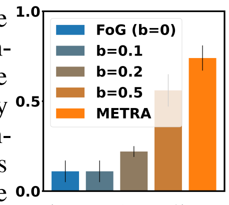

Figure 9: Flips of FoG with different level of inaccuracy injected.

FoG is most closely related to DoDont and LGSD, as both these methods aim to incorporate human preferences into skill discovery. However, DoDont relies on expert demonstrations, which can be costly [7, 31] or infeasible for tasks where human performance is limited (e.g., defining "stretched" or "twisted" posture for a humanoid robot). Additionally, DoDont's classifiers require ground-truth state-based inputs to avoid being misled by unrelated information when learning behavioral intentions (see Appendix D). LGSD leverages language models [1] but is limited to state-based tasks, as language models cannot process visual inputs. Furthermore, querying them in a step-wise, chat-style manner is computationally expensive. In contrast, FoG utilizes vision-language models and extracts a score function, applied either once (state-based tasks) or via batch processing (pixel-based tasks), to re-weight the underlying skill discovery rewards. It therefore has a fast response time and works well in both state-based and pixel-based tasks. See Appendix B for an extended discussion of related work.

6 Conclusion and Future Work

We propose a novel unsupervised skill discovery method, FoG, guided by foundation models to incorporate human intentions. FoG first extracts a score function from foundation models based on input intentions, assigning higher preference to desirable states and lower preference to undesirable ones. This score function is then used to re-weight the underlying skill discovery rewards. By optimizing re-weighted rewards, FoG discovers not only diverse but also desirable skills. In addition, we also show FoG can learn skills involving behaviors that are complex and subjectively defined.

Although FoG performs well, it is not without limitations. First, there is no guarantee that score functions generated by foundation models are always appropriate. Additionally, since the score function is defined based on individual states, FoG may struggle to capture process-based alignment. This limitation could be addressed by defining the score function over a sequence of states [38]. Furthermore, we believe FoG could benefit from more advanced and task-specific foundation models [18, 47, 26, 43]. One could also explore the performance of FoG with more complex intentions and more challenging tasks. Some preliminary results can be found in Appendices E.2 and E.3. We hope FoG inspires future efforts in incorporating human intentions in unsupervised skill discovery.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] A. Adeniji, A. Xie, C. Sferrazza, Y. Seo, S. James, and P. Abbeel. Language reward modulation for pretraining reinforcement learning. *arXiv preprint arXiv:2308.12270*, 2023.

[3] F. Bordes, R. Y. Pang, A. Ajay, A. C. Li, A. Bardes, S. Petryk, O. Mañas, Z. Lin, A. Mahmoud, B. Jayaraman, et al. An introduction to vision-language modeling. *arXiv preprint arXiv:2405.17247*, 2024.

[4] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[5] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[6] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

[7] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *Conference on Robot Learning (CoRL)*, 2024.

[8] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[9] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[10] M. Hussonnois, T. G. Karimpanal, and S. Rana. Controlled diversity with preference: Towards learning a diverse set of desired skills. *arXiv preprint arXiv:2303.04592*, 2023.

[11] H. Kim, B. Lee, H. Lee, D. Hwang, D. Kim, and J. Choo. Do's and don'ts: Learning desirable skills with instruction videos. *arXiv preprint arXiv:2406.00324*, 2024.

[12] H. Kim, B. K. Lee, H. Lee, D. Hwang, S. Park, K. Min, and J. Choo. Learning to discover skills through guidance. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] S. Kim, J. Kwon, T. Lee, Y. Park, and J. Perez. Safety-aware unsupervised skill discovery. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 894–900. IEEE, 2023.

[14] M. Klissarov, P. D'Oro, S. Sodhani, R. Raileanu, P.-L. Bacon, P. Vincent, A. Zhang, and M. Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. *arXiv preprint arXiv:2310.00166*, 2023.

[15] M. Laskin, H. Liu, X. B. Peng, D. Yarats, A. Rajeswaran, and P. Abbeel. Unsupervised reinforcement learning with contrastive intrinsic control. *Advances in Neural Information Processing Systems*, 35:34478–34491, 2022.

[16] M. Laskin, D. Yarats, H. Liu, K. Lee, A. Zhan, K. Lu, C. Cang, L. Pinto, and P. Abbeel. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021.

[17] K. Lee, L. Smith, and P. Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

[18] H. Liu, C. Li, Y. Li, and Y. J. Lee. Improved baselines with visual instruction tuning, 2023.

[19] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[20] C. Lu, S. Hu, and J. Clune. Intelligent go-explore: Standing on the shoulders of giant foundation models. *arXiv preprint arXiv:2405.15143*, 2024.

[21] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[22] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo. Embodiedgpt: Vision-language pre-training via embodied chain of thought. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] T. Nam, J. Lee, J. Zhang, S. J. Hwang, J. J. Lim, and K. Pertsch. Lift: Unsupervised reinforcement learning with foundation models as teachers. *arXiv preprint arXiv:2312.08958*, 2023.

[24] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[25] S. Ozair, C. Lynch, Y. Bengio, A. Van den Oord, S. Levine, and P. Sermanet. Wasserstein dependency measure for representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[26] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[27] S. Park, J. Choi, J. Kim, H. Lee, and G. Kim. Lipschitz-constrained unsupervised skill discovery. In *International Conference on Learning Representations*, 2022.

[28] S. Park, D. Ghosh, B. Eysenbach, and S. Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

[29] S. Park, K. Lee, Y. Lee, and P. Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

[30] S. Park, O. Rybkin, and S. Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. *arXiv preprint arXiv:2310.08887*, 2023.

[31] K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Guided reinforcement learning with learned skills. *arXiv preprint arXiv:2107.10253*, 2021.

[32] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[33] S. Rajeswar, P. Mazzaglia, T. Verbelen, A. Piché, B. Dhoedt, A. Courville, and A. Lacoste. Mastering the unsupervised reinforcement learning benchmark from pixels. In *International Conference on Machine Learning*, pages 28598–28617. PMLR, 2023.

[34] S. Rho, L. Smith, T. Li, S. Levine, X. B. Peng, and S. Ha. Language guided skill discovery. *arXiv preprint arXiv:2406.06615*, 2024.

[35] J. Rocamonde, V. Montesinos, E. Nava, E. Perez, and D. Lindner. Vision-language models are zero-shot reward models for reinforcement learning. *arXiv preprint arXiv:2310.12921*, 2023.

[36] C. Schenck and D. Fox. Perceiving and reasoning about liquids using fully convolutional networks. *The International Journal of Robotics Research*, 37(4-5):452–471, 2018.

[37] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[38] S. Sontakke, J. Zhang, S. Arnold, K. Pertsch, E. Bıyık, D. Sadigh, C. Finn, and L. Itti. Roboclip: One demonstration is enough to learn robot policies. *Advances in Neural Information Processing Systems*, 36, 2024.

[39] H. Sowerby, Z. Zhou, and M. L. Littman. Designing rewards for fast learning. *arXiv preprint arXiv:2205.15400*, 2022.

[40] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone. Deep reinforcement learning for robotics: A survey of real-world successes. *arXiv preprint arXiv:2408.03539*, 2024.

[41] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[42] S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020.

[43] D. Valevski, Y. Leviathan, M. Arar, and S. Fruchter. Diffusion models are real-time game engines, 2024.

[44] M. Vasco, T. Seno, K. Kawamoto, K. Subramanian, P. R. Wurman, and P. Stone. A super-human vision-based reinforcement learning agent for autonomous racing in gran turismo. *arXiv preprint arXiv:2406.12563*, 2024.

[45] C. Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009.

[46] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *Conference on robot learning*, pages 2226–2240. PMLR, 2023.

[47] Y. Yao, T. Yu, A. Zhang, C. Wang, J. Cui, H. Zhu, T. Cai, H. Li, W. Zhao, Z. He, Q. Chen, H. Zhou, Z. Zou, H. Zhang, S. Hu, Z. Zheng, J. Zhou, J. Cai, X. Han, G. Zeng, D. Li, Z. Liu, and M. Sun. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint 2408.01800*, 2024.

[48] C. Zhang, Q. He, Z. Yuan, E. S. Liu, H. Wang, J. Zhao, and Y. Wang. Advancing drl agents in commercial fighting games: Training, integration, and agent-human alignment. *arXiv preprint arXiv:2406.01103*, 2024.

Appendix

Table of Contents

A Derivation of Equation (7) **13**

B Extended Related Work **14**

C Downstream Tasks **14**

D Analysis of DoDont **15**

E Additional Experiments **15**
- E.1 Quadruped Learns to Not Flip . . . 15
- E.2 Results on Franka Kitchen . . . 15
- E.3 Results on Multiple Intentions . . . 16

F Experiment Details **16**
- F.1 Environment Details . . . 16
- F.2 Baseline Details . . . 17
- F.3 Non-binary Score Function . . . 17
- F.4 Computation Usage . . . 18
- F.5 Experimental Setup for Human Judge . . . 18
- F.6 Foundation Models . . . 18
- F.7 Prompts Used . . . 19

G Impact Statement **21**

A Derivation of Equation (7)

The original DSD objective is shown in Equation (2). It is crucial to define a appropriate distance metric to encourage agents to not only learn diverse skills but also maximize the given distance metric. [30] uses the temporal distance as the distance metric for the DSD objective in METRA, shown in Equation (9).

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right] \quad \text{s.t.} \;\; \|\phi(s) - \phi(s')\|_2 \leq 1, \;\; \forall (s, s') \in S_{adj}. \tag{9}$$

Now, we use the score function $f(s')$ to re-weight the METRA rewards to get the objective of FoG. The new objective (FoG) now becomes Equation (10):

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} f(s') \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right] \quad \text{s.t.} \;\; \|\phi(s) - \phi(s')\|_2 \leq 1, \;\; \forall (s, s') \in S_{adj}. \tag{10}$$

Following [12], let scaled state function $\tilde{\phi}(s) = \phi(s) f(s)$. By replacing $\phi(s)$ with $\tilde{\phi}(s)/f(s)$ and transforming the constraint in Equation (10) (since $f(s) \geq 0$), we derive Equation (11) (Equation (7)), which is using the score function as the distance metric in the DSD objective.

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\tilde{\phi}(s_{t+1}) - \tilde{\phi}(s_t)\right)^\top z \right] \quad \text{s.t.} \;\; \|\tilde{\phi}(s) - \tilde{\phi}(s')\|_2 \leq f(s'), \;\; \forall (s, s') \in S_{adj}. \tag{11}$$

Hereby, we show that using the score function to re-weight the METRA rewards is equivalent as using it as the distance metric in the DSD objective.

B Extended Related Work

Mutual Information Based Unsupervised Skill Discovery: FoG builds on top of unsupervised skill discovery methods, allowing agents to learn diverse skills without the use of hand-crafted reward functions. One line of research in unsupervised skill discovery focuses on maximizing mutual information (MI) $I(\cdot;\cdot)$ between skills Z and states S, i.e., $I(S;Z) = H(S) - H(S|Z) = H(Z) - H(Z|S)$, where $H(\cdot)$ denotes entropy. By associating states $s \in S$ with different latent skill vectors $z \in Z$, these methods learns diverse skills that are mutually distinct [5, 37, 15]. SASD [13] and EDL [10] integrate preference into MI methods by a pre-defined function and human feedback, and they operate only with state-based input. In contrast, FoG eliminates the need for human involvement and supports both state and pixel-based input.

Table 1: Distance metrics used by different methods in the distance-maximizing skill discovery objective. q_θ is a density function parameterized by θ. Temporal distance is defined as the minimum number of environmental steps needed for the agent to go from one state to another state. s_{lang} is the textual description of the state s. p_φ is a classifier parameterized by φ.

LSD	CSD	METRA	LSGD	DoDont	Ours
$\|\|s' - s\|\|$	$-\log q_\theta(s'\|s)$	temporal dis	dis(s'_{lang}, s_{lang})	$p_\varphi(s', s)$	score fn

Foundation Models in Reinforcement Learning: FoG leverages foundation models to guide unsupervised skill discovery in learning desirable behaviors. Thanks to success of foundation models [41, 19] they can now be used to provide information for RL agents. Motif [14] and IGE [20] employs large language models to generate exploration bonuses. Eureka [21] uses large language models to generate reward functions for state-based robotic tasks, outperforming human designed reward functions across multiple tasks. Lift [23] uses LLM and VLM to guide learning in MineDoji [6]. LAMP [2] and [35] utilize the similarity between pixel embedding and text-commands embedding, as output by a vision-language model, as the step-wise reward in visual robotic tasks. Results show that such step-wise signals alone barely work (matching the results we had in Section 4.2.4), and require either fine-tuning or special task modifications to perform well. Task-specific foundation models generally can achieve better performance on specific tasks, such as Minedojo [6] in Minecraft and EmbodiedGPT [22] in robotics. Despite this, FoG demonstrates that pre-trained foundation models, even without fine-tuning or any modifications of tasks, can be used to guide RL agents to discover diverse and desirable skills. In state-based tasks, FoG uses foundation models to generate a score function aligned with human intentions. Unlike Eureka [21], FoG: 1) avoids iterative feedback loops with the environment, as Eureka requires multiple rounds of feedback to refine the reward function, and 2) uses the score function to re-weight skill discovery rewards, whereas Eureka directly trains agents with the generated reward function.

C Downstream Tasks

After obtaining skills, we can train a controller to select these (frozen) learned skills to achieve given downstream goals. We follow the implementation of [30], and set $g \sim [-10, 10]$ as the goal. During training, the agent receives a reward of 10 if the goal is reached. We train a controller to select a skill z every $K = 50$ steps, and the learned policy $\pi(\cdot|s, z)$ is executed for K steps. We use SAC [9] for training the controller and all hyperparameters are kept the same as the METRA codebase. Results are shown in Figure 10. The controller that is trained using frozen skills learned by FoG shows better performance at the beginning and converges faster than the baselines, indicating that FoG effectively learns meaningful skills that can be quickly adapted to downstream tasks. LSD does not learn useful skills thus the trained controller performs poorly. METRA slightly lags behind of DoDont.

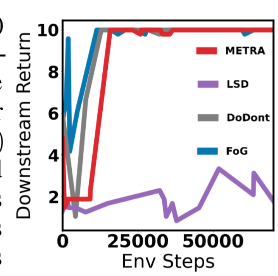

Figure 10: Downstream task performance.

D Analysis of DoDont

The performance of DoDont in our paper is quite different to the one from the original paper due to different experimental setup. Here, we provide a more in-depth analysis of why DoDont fails in our experiments.

Failure in Section 4.2.1: To keep a fair comparison, we use pixel inputs for both the classifier and the RL part in DoDont (since FoG does not require ground-truth state information and works with only pixel inputs), which differs from the original experiments in the DoDont paper. In the Appendix D.1 of DoDont paper, authors mentioned that DoDont uses state information as input for the RL agent (both the policy and the critic). The classifier might exploit the background color as a shortcut to distinguish between different states rather than observing the agent's embodiment, thus DoDont instead uses a non-colored ground (see in Figure 11). However, the backbone of DoDont, METRA, cannot learn diverse skills without the colored ground (since there will be no indication of directions). Thus, DoDont uses state-information for the RL part. During the training, image states are first input to the classifier to get rewards, then the corresponding compact ground-truth states are used to train the RL agents along with the rewards obtained from the classifier. Our experiments show that indeed, if pixel inputs are used for both the RL part and the classifier, DoDont fails (see results in Figure 4). The classifier indeed exploits the background color as a shortcut to distinguish between different states rather than observing the agent's embodiment, classifies unseen 'Dont' states as 'Do'. See videos on `https://sites.google.com/view/submission-fog`.

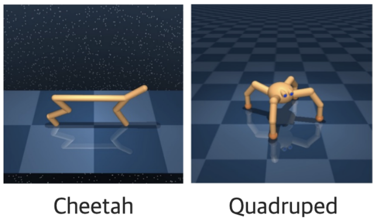

Figure 11: Tasks with non-colored ground that DoDont uses.

Failure in Section 4.2.2: In Figure 5, DoDont successes in Quadruped while fails in Cheetah. The performance of the classifier shows that it is able to accurately classifying "going left" and "going right", but unsure about states at the beginning. Our intuition is that such uncertainty hurts the exploration at the beginning, resulting in poor performance later on. See videos on `https://sites.google.com/view/submission-fog`.

E Additional Experiments

E.1 Quadruped Learns to Not Flip

Although we found that setting $\alpha = 0$ works well in experiments presented in Section 4, sometimes it might hurt the exploration. Similar with experiments performed in Section 4.2.1, here, we train FoG to not flip in Quadruped. We see in Figure 12, FoG learns to not flip most of time (less than 20%) when setting $\alpha = 0$, but it almost always stays near the starting point and does not explore, resulting in low state coverage. After loosing α a bit and set it to 0.1, FoG learns to eliminate all flips and has a significant higher state coverage.

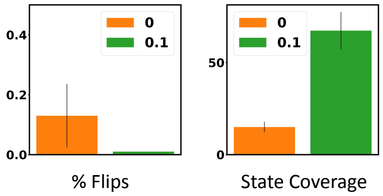

Figure 12: Results on the Quadruped task. Setting $\alpha = 0$ explores less (lower state coverage) thus results in worse performance (more flips).

E.2 Results on Franka Kitchen

To examine FoG in more complicated tasks, we train FoG in Franka Kitchen (introduced by [8]) with different textual descriptions of intentions, such as '`robotic arm is stretched`', '`robotic arm is twisted`' and '`robotic arm is on the right of the scene`'. Results can be seen in Figure 13. By using different intentions, we see robotic arms clearly bias the movements to different areas. However, we did not find a way to use these skills to better solve the downstream tasks yet. We hope this could inspire future efforts in investigating FoG in more complex tasks.

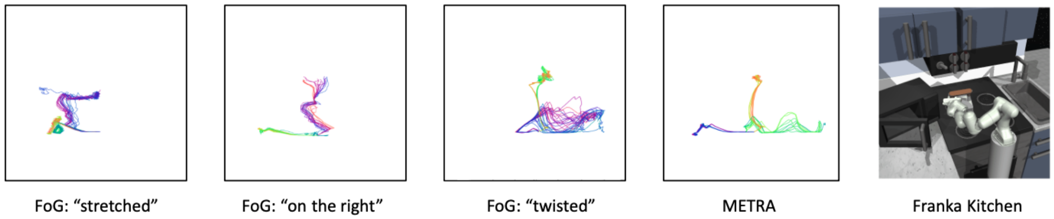

Figure 13: In Franka Kitchen, different skills FoG learned with different textual descriptions of intentions. Skills are displayed with x-y coordinates of the robotic arm.

E.3 Results on Multiple Intentions

In Section 4, only one intention is used in FoG. In principle, multiple intentions could be used simultaneously to form the score function. Then, Equation (8) becomes:

$$f(s) = \begin{cases} 1, & \text{if } Cosine(E_s, E_{t1}^1) > Cosine(E_s, E_{t2}^1) \text{ and} \\ & \quad Cosine(E_s, E_{t1}^2) > Cosine(E_s, E_{t2}^2) \text{ and} \\ & \quad ... \\ & \quad Cosine(E_s, E_{t1}^n) > Cosine(E_s, E_{t2}^n) \\ \alpha, & \text{otherwise.} \end{cases} \tag{12}$$

where E_{t1}^n and E_{t2}^n are the nth textual descriptions of our intentions. Now, the score function $f(s)$ only assigns higher values to desirable states when all provided intentions are satisfied. For example, we could ask FoG to not only learns to not flip, but also to avoid the right area. The textual descriptions we should use are: 1) `‘agent flips over’`, `‘agent stands normally’`; 2) `‘ground is Yellow-Orange’`, `‘ground is Green-Blue’`. See the result in Figure 14, the agent does not learn to avoid the right part at all but it does learn to eliminate flips (not shown in the figure). We found that using multiple intentions restricts the exploration too much so that the agent might just learn to fulfill one intention and ignore others or ignore all of them and learns to not move at all. Using multiple intentions in FoG still needs more investigations and we hope the preliminary results and ideas presented in this section could inspire future efforts.

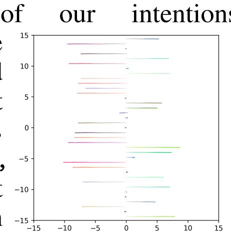

Figure 14: Skills learned by FoG with two intentions.

F Experiment Details

F.1 Environment Details

State-based: HalfCheetah and Ant are from OpenAI Gym [4]. The state space of HalfCheetah is 18-dimensional and the one of Ant is 29-dimensional. HalfCheetah has a 6-dimensional action space while Ant has a 8-dimensional action space.

Pixel-based: Cheetah, Quadruped and Humanoid are from DeepMind Control Suite [42]. Following previous work [17, 28, 30], pixel-based DMC tasks are all with gradient-colored floors to indicate different directions. The size of visual observations is $64 \times 64 \times 3$. The dimension of action space for Cheetah, Quadruped and Humanoid are 6, 12 and 21, respectively. The episode length is 200 for Ant, HalfCheetah and Cheetah, 400 for Quadruped and Humanoid.

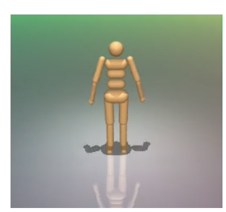

Figure 15: The Puppet environment.

Modified Humanoid: Since none of existing unsupervised skill discovery methods can train the visual Humanoid agent to stand up, limiting FoG to showcase more interesting behaviors, such as running, etc. We created a ‘Puppet’ task based on the DMC Humanoid environment, see Figure 15. The humanoid robot is pulled by a puppet anchor on the top of its head. Thus, the humanoid robot keeps standing by default and never falls down. The anchor also moves with the humanoid.

F.2 Baseline Details

METRA: We take the official codebase[2] from [30] and use default hyperparameters for all experiments performed in this paper.

METRA+: We follow the implementation of METRA+ in the DoDont paper. For experiments in Section 4.2.1, we use $r_{run} - r_{flip}$ as the reward. For experiments in Section 4.2.2, we assign +1 for the safe region and 0 for the hazardous region.

LSD: We take the codebase of METRA, by setting correct arguments (turning off the dual regularization and turning on the spectral normalization), to run LSD. Detailed instructions can be found in the METRA codebase.

DoDont: We take the official codebase from [11] and implement the training of the instruction net ourselves. We use eight demonstrations for each task, so four for “dos" and four for “donts". Demonstrations are obtained from trained FoG agents and can be found on our project website: `https://sites.google.com/view/submission-fog`. We stop the training of the classifier after it has more than 97% of accuracy.

DoDont+: A variant of DoDont, instead of using expert-level demonstrations, it uses demonstrations annotated by foundation models. In our case, we use CLIP to score frames (follow Equation (8)) in demonstrations that are used to train DoDont, and assign frames with score of 0 in the “dos" demonstration to “donts" demonstrations, and vice versa. Since CLIP cannot perfectly score frames, some states from “dos" demonstration are moved to “donts" demonstrations, and some states from “donts" demonstration are moved to “dos" demonstration. After training, the classifier of DoDont+ has about 70% of accuracy.

FR-SAC: A soft actor-critic RL agent with using the score function as the reward function. We reuse the FoG codebase and set the number of skills to 1. Then, we train the skill-conditioned policy with the scores obtained from the foundation model (i.e. using the score function as the reward function), reducing to a normal RL agent.

F.2.1 Hyperparameters Details

We use $\alpha = 0$ and $N = 2$ for all our experiments, unless otherwise mentioned. We train all agents in the same task with the same number of epochs and the performance at the end of training is reported. Details can be seen in Table 2. The same number of episodes is executed in each epoch, and within each episode the same number of environment steps is taken. We train continuous skills and the number of dimensions we used to train all agents in each task can be found in Table 2. We refer readers to read [30] for details of all used hyperparameters.

Table 2: Number of epochs and dimensions of skills we used for training agents in different environments.

HalfCheetah	Ant	Cheetah	Quadruped	Humanoid
9000	9000	2000	3000	4000
4D	2D	4D	4D	2D

F.3 Non-binary Score Function

Instead of using a binary score function in Equation (8), we can also form a non-binary score function.

$$f(s) = \frac{e^{Cosine(E_s, E_{text1})}}{e^{Cosine(E_s, E_{text1})} + e^{Cosine(E_s, E_{text2})}}, \tag{13}$$

where E_s is the embedding of the current pixel-based state, E_{text1} is the embedding of textual descriptions of the desirable intention and E_{text2} is the embedding of textual descriptions of the undesirable intention.

[2] `https://github.com/seohongpark/METRA`

F.4 Computation Usage

We run our experiments on an internal cluster consisting of A100 and H100 GPUs. Each run takes no more than 24 hours.

F.5 Experimental Setup for Human Judge

In Section 4.2, we train FoG to be twisted in DMC Humanoid task. However, it is difficult for human to design a reward function to measure if learned skills contains more twisted postures or not. Thus, we ask human to be the judge to tell if FoG learns more twisted skills than the ones learned by the baselines.

We pick ten skills of each method randomly, in this case, FoG and METRA, and then pair them randomly. Participates are asked to select the video that shows the most 'twisted' behaviors without given any other information. Please see the full questionnaire we used in `https://sites.google.com/view/iclr-fog/questionnaire-of-humanoid`.

F.6 Foundation Models

For state-based tasks, we query ChatGPT[3] or Cluade[4] to generate score functions that meet our requirements. For pixel-based tasks, we use pre-trained CLIP (clip-vit-large-patch14) from huggingface[5].

[3] `https://chatgpt.com`
[4] `https://claude.ai/new`
[5] `https://huggingface.co/openai/clip-vit-large-patch14`

F.7 Prompts Used

F.7.1 State-based Tasks

Input to foundation models for HalfCheetah: `[Descriptions of the task, state space and action space] According to the given info, could you please write a python function to check if the cheetah is flipped over or not. If yes, output 1 otherwise 0.`

Output:

python Copy code

```python
import numpy as np

def is_cheetah_flipped(observation):
    """
    Check if the HalfCheetah is flipped over.

    Args:
    observation (np.array): Observation array from the environment.

    Returns:
    int: 1 if the cheetah is flipped over, 0 otherwise.
    """
    # The second element in the observation array is the angle of the front tip
    angle = observation[1]

    # Define the threshold for flipping (e.g., |angle| > π/2)
    threshold = np.pi / 2

    if abs(angle) > threshold:
        return 1
    else:
        return 0

# Example usage:
# Assume `obs` is the current observation from the environment
# obs = env.reset()
# print(is_cheetah_flipped(obs))
```

Input to foundation models for Ant: `[Descriptions of the task, state space and action space] The agent always starts at [0, 0]. According to the given info, could you please write a python function to check if the agent is in the north part or not. If yes, output 1 otherwise 0.`

Output:

```python
def is_in_north(state):
    """
    Check if the agent is in the northern part (y > 0).

    Parameters:
    state (list or numpy array): The state space observation vector. The y-posit

    Returns:
    int: 1 if the agent is in the northern part, 0 otherwise.
    """
    # Assuming the second element in the state space corresponds to the y-coordi
    y_position = state[1]

    # Check if the y-coordinate is greater than 0
    if y_position > 0:
        return 1
    else:
        return 0
```

F.7.2 Pixel-based Tasks

Textual descriptions of intentions we used for Cheetah:

- Section 4.2.1: `'The simulated two-leg robot flips over'`, `'The simulated two-leg robot stands normally'`
- FR-SAC agent in Section 4.2.1: `'The simulated two-leg robot flips over'`, `'The simulated two-leg robot is running normally'`
- Section 4.2.2: `'The underneath plane is Yellow-Orange'`, `'The underneath plane is Green-Blue'`

Textual descriptions of intentions we used for Quadruped in Section 4.2.2: `'The underneath plane is Pink-Purple'`, `'The underneath plane is Green-Blue'`.

Textual descriptions of intentions we used for Humanoid in Section 4.2.3: `'The simulated humanoid robot is stretched'`, `'The simulated humanoid robot is twisted'`.

G Impact Statement

As we integrate foundation models into RL agents, the possibility of them acting in unexpected ways to maximize scores outputted by foundation models increases. As such, we expect research into safety to be paramount.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The statements made in the abstract and introduction are mainly supported by experiments in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The theoretical results can be in the main paper and details are in Appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The code is provided in the supplementary materials.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code is provided and will be open-sourced after the submission.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All hyper-parameters used are reported and the code is provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The results are averaged over multiple independent runs and standard errors are reported as well.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We reported compute usage in Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The codebase and website in the paper are both anonymous.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discussed the broader impacts in Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We properly cited all codebases we used in this work and with url provided in Appendix.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We provided a README to explain how to run our codebase in the supplementary.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: [NA]

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.