Do's and Don'ts: Learning Desirable Skills with Instruction Videos

Hyunseung Kim[1,2] **Byungkun Lee**[1] **Hojoon Lee**[1]
Dongyoon Hwang[1] **Donghu Kim**[1] **Jaegul Choo**[1]
[1]KAIST, [2]KRAFTON
{mynsng,byungkun.lee,joonleesky,godnpeter,quagmire,jchoo}@kaist.ac.kr

Abstract

Unsupervised skill discovery is a learning paradigm that aims to acquire diverse behaviors without explicit rewards. However, it faces challenges in learning complex behaviors and often leads to learning unsafe or undesirable behaviors. For instance, in various continuous control tasks, current unsupervised skill discovery methods succeed in learning basic locomotions like standing but struggle with learning more complex movements such as walking and running. Moreover, they may acquire unsafe behaviors like tripping and rolling or navigate to undesirable locations such as pitfalls or hazardous areas. In response, we present **DoDont** (Do's and Dont's), an instruction-based skill discovery algorithm composed of two stages. First, in instruction learning stage, DoDont leverages action-free instruction videos to train an instruction network to distinguish desirable transitions from undesirable ones. Then, in the skill learning stage, the instruction network adjusts the reward function of the skill discovery algorithm to weight the desired behaviors. Specifically, we integrate the instruction network into a distance-maximizing skill discovery algorithm, where the instruction network serves as the distance function. Empirically, with less than 8 instruction videos, DoDont effectively learns desirable behaviors and avoids undesirable ones across complex continuous control tasks. Code and videos are available at `https://mynsng.github.io/dodont/`

1 Introduction

Recent advancements in unsupervised pre-training methodologies have led to the creation of large-scale foundational models across diverse domains, including computer vision [3, 41] and natural language processing [7, 11, 22]. These methodologies exploit self-supervised learning objectives to extract meaningful representations without using explicit, supervised learning signals. In attempts to expand this paradigm to reinforcement learning, researchers have explored crafting self-supervised objectives to develop foundational policies capable of learning diverse behaviors without predefined reward signals, termed unsupervised skill discovery (USD) [1, 14, 9, 21, 43, 31, 24, 23, 27, 35, 38, 39].

Despite notable advancements in USD methodologies, acquiring foundational policies in environments with large state and action spaces (e.g., multi-jointed quadrupeds) remains a significant challenge. Two major issues arise when training agents with USD in these complex environments. First, the vast state and action spaces allow the agent to acquire a wide variety of behaviors. While learning simple behaviors like standing may be feasible, acquiring complex behaviors that require intricate joint movements, such as walking or running, can take an exceedingly long time. Second, agents can develop undesirable and risky behaviors during training, such as tripping, rolling, or navigating hazardous areas like pitfalls. These challenges raise a key question: *Is the purely unsupervised assumption of USD ideal for learning foundational policies in the real-world?*

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-1512

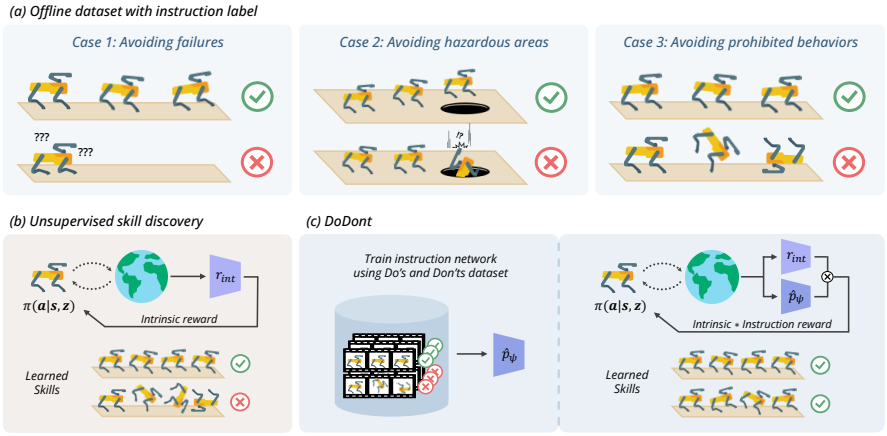

Figure 1: **(a)** The offline instruction video dataset includes videos of desirable behaviors (Do's) and undesirable behaviors (Don'ts). **(b)** Unsupervised skill discovery algorithms tend to learn undesirable behaviors. **(c)** In DoDont, an instruction network is first trained with the Do's and Don'ts videos to distinguish desirable and undesirable behaviors. Then, this instruction network adjusts the intrinsic reward of the skill discovery algorithm, promoting desirable skills while avoiding undesirable ones.

According to social learning theory, humans learn behaviors through both internal and external motivations [4, 5, 16]. Internally, they seek to perform diverse behaviors they haven't tried before. Externally, they adopt socially desirable behaviors and avoid those that are not. In the skill discovery algorithm, this can be mimicked by combining internal motivation (USD objective) with external motivation based on a human's intention. A straightforward way to incorporate external motivation in USD is to use a hand-designed reward function [26, 10]. However, hand-designing reward functions are not scalable for learning diverse and desirable behaviors for several reasons. One of the reasons is the complexity of designing an intricate reward signal for guiding a desired behavior, and another is the difficulty of balancing multiple behavioral intentions into a single reward function.

To address these challenges, we propose **DoDont**, a skill discovery algorithm that integrates USD objectives with intended behavioral goals. Instead of relying on a hand-designed reward, DoDont *learns a reward function from a small set of instruction videos* that demonstrate desirable and undesirable behaviors. Videos are chosen because they are inexpensive to collect and do not require action or reward labels [13, 46].

As illustrated in Figure 1, DoDont starts by collecting instruction videos of desirable (Do's) and undesirable behaviors (Don'ts). We then train an instruction network which assigns higher values to desirable behaviors and lower values to undesirable ones. This network re-weights the internal USD objective for the skill discovery phase. We utilize a distance-maximizing skill discovery algorithm as our main USD objective [35, 38, 39] where the instruction network serves as the distance metric.

To validate DoDont, we conduct experiments on various continuous control tasks that require complex locomotion (e.g., Cheetah and Quadruped [45]) or precise manipulation (e.g., Kitchen [17]). Our results show that with fewer than eight instruction videos, DoDont effectively learns complex locomotion skills (e.g., running quadruped), which are challenging to acquire with standard USD algorithms [39]. Moreover, our instruction network effectively captures human intentions better than hand-designed reward functions in balancing multiple behavioral objectives. Additionally, DoDont learns diverse skills while avoiding undesirable and unsafe behaviors (e.g., backflipping), which are difficult to prevent even with previous skill discovery algorithms which utilize prior knowledge [26].

2 Related work

2.1 Learning diverse behaviors without pre-defined task

Numerous unsupervised skill discovery (USD) methods have been developed to create foundational policies capable of learning a diverse behaviors without pre-defined task. The most com-

mon approach involves maximizing mutual information (MI) between states and skills ($I(S;Z) = D_{KL}(p(s,z)||p(s)p(z))$) [14, 43, 31, 9, 27]. This is typically done by using an auxiliary neural network $q_\theta(z|s)$ to estimate a lower bound of $I(S;Z)$ ($I(S;Z) \geq \mathbb{E}z, s[\log q_\theta(z|s)]$). This network acts as a skill discriminator, predicting the skill z from the state s, which encourages the policy to create distinct trajectories for different skills and promotes learning a wide range of skills. However, these methods often struggle to learn complex, dynamic skills, as the MI objective can be met with simple, static skills, leaving agents unmotivated to explore more intricate behaviors [44, 23, 30, 31, 24].

To overcome this problem, distance-maximizing skill discovery (DSD) methods have been introduced [35, 38, 39]. Instead of maximizing MI between states and skills, DSD explicitly maximizes a predefined distance function of states $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}_0^+$. For instance, LSD [35] uses Euclidean distance between states to encourage agents to move further distances ($d_{\text{LSD}} = ||s' - s||$). CSD [38] employs a density model based distance, promoting agents to visit less frequented states ($d_{\text{CSD}} = -\log q_\theta(s'|s)$, where q_θ is the density model). METRA [39] defines the distance temporally, pushing agents to learn skills that are temporally far apart (d_{METRA} = minimum number of environment steps to reach s' from s). DoDont is built upon the DSD framework, which will be elaborated in Sections 3 and 4.

2.2 Learning diverse behaviors with pre-defined task

In contrast to USD, there is a line of research focused on learning policies that exhibit diverse behaviors by integrating intrinsic motivation (e.g., MI-based rewards) with human intention, as the form of task-specific reward functions [10, 26, 55, 54] or using demonstration datasets [40]. While traditional RL methods typically aim to find a single optimal policy, these approaches aim to learn multiple policies $\pi_\theta(a|s, z_{i=1:N})$. This allows for multiple ways to solve given tasks, resulting in increased robustness to environmental changes and the development of policies with distinct characteristics.

SMERL [26] and DGPO [10] combine intrinsic and task rewards, maximizing their sum when task rewards exceed a given threshold. RSPO [55] iteratively finds novel policies by switching between task rewards and intrinsic diversity rewards, ensuring each new policy is distinct from previous ones. DOMiNO [54] addresses a constrained optimization problem by maximizing intrinsic diversity rewards while ensuring that all policies achieve sufficiently high performance (task rewards as constraints). ASE [40] utilizes a action-free demonstration dataset instead of a task reward function, minimizing a behavioral cloning loss while simultaneously maximizing a diversity intrinsic reward. DoDont also employs action-free demonstration datasets to extract human prior knowledge through an instruction network, pioneering the incorporation of both positive and negative behavior examples to guide learning towards desired behaviors and away from undesirable ones.

3 Preliminaries and problem setting

Markov decision process and skill-conditioned policy. Unsupervised skill discovery (USD) focuses on identifying a range of skills without pre-defined reward function. In this approach, we use a reward-free Markov Decision Process defined as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mu, p)$. Here, $\mathcal{S}$ represents the state space, $\mathcal{A}$ is the action space, μ denotes the initial state distribution, and p is transition dynamics function. We utilize a latent vector $z \in \mathcal{Z}$ and their associated policy $\pi(a|s, z)$, which we refer to as skills. To generate a skill trajectory (i.e., behavior), we first sample a skill z from the prior distribution, denoted as $p(z)$, and then execute a trajectory using the skill policy $\pi(a|s, z)$. For the skill prior, we utilize a standard normal distribution to represent continuous skills by following previous works [35, 38, 39].

Distance-maximizing skill discovery. As discussed in Section 2.1, numerous distance-maximizing skill discovery methods have been proposed [35, 38, 39]. Specifically, DSD introduces the Wasserstein dependency measure (WDM) as a learning objective for unsupervised skill discovery (USD) [34].

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

Here, $\mathcal{W}$ is the 1-Wasserstein distance on the metric space $(S \times Z, d)$, where d is a *distance metric*. Unlike MI-based skill discovery methods (maximizing $I(S;Z) = D_{KL}(p(s,z)||p(s)p(z))$) which does not explicitly motivate the agent to learn more complex behaviors (as discussed in Section 2.1), DSD maximizes the *distance-aware* $I_{\mathcal{W}}(S;Z)$. By doing so, DSD not only discovers a set of

https://doi.org/10.52202/079017-1512

distinguishable skills but also maximizes the distance d between different states. By defining the distance function to encourage desired behaviors, DSD effectively induces the agent to learn these behaviors.

In practice, by leveraging the Kantorovich-Rubenstein duality [48, 34] under some simplifying assumptions, the following concise objective can be derived:

$$I_{\mathcal{W}}(S_T; Z) \approx \sup_{\|\phi\|_L \leq 1} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right], \tag{2}$$

where $\phi : S \to \mathbb{R}^D$ is a function that maps states into D-dimensional space, $\|\phi\|_L$ denotes the Lipschitz constant for the function ϕ under the given distance metric d. Now, we can rewrite Equation 2 with an arbitrary distance function $d : S \times S \to \mathbb{R}_0^+$ as:

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right] \quad \text{s.t.} \quad \|\phi(x) - \phi(y)\|_2 \leq d(x, y), \quad \forall (x, y) \in S. \tag{3}$$

For a detailed derivation, we advise readers to refer to Section 4 of METRA [39]. It's important to note that a generic distance function, d, does not necessarily need to meet the criteria of a valid distance metric, such as symmetry or the triangle inequality. As shown in previous work [38], the original constraint in Equation 3 can be implicitly converted into one with a valid pseudometric, allowing us to use any non-negative function as a distance metric.

4 Do's and don'ts (DoDont)

Our goal is to integrate human intention into USD, where diverse policies are learned without predefined reward signals. The core concept of this work involves training an instruction network using video data to distinguish desired and undesired behvaiors (Section 4.1). Then, we set the trained instruction network as the distance metric in the distance-maximizing skill discovery framework (Section 4.2). Consequently, this will result in an agent that not only learns diverse behaviors but also incorporate human intention since the instruction network is set as the distance metric.

4.1 Instruction network

We aim to train an instruction network that assigns high values for desirable behaviors (Do's) and low values for undesirable behaviors (Don'ts). The distance metric in the WDM objective (in Equation 3) takes two states as input and should predict a non-negative value.

To meet this requirement, we first prepare videos depicting desirable behaviors (i.e., Do's) and videos illustrating behaviors to avoid (i.e., Don'ts) according to our intentions. We then label our video dataset by assigning $y = 1$ to adjacent state pairs (s_t, s_{t+1}) in Do's videos and $y = 0$ to those in Don'ts videos. This results in our Do's and Don't video dataset $\mathcal{D}_V$ which is comprised of triplets (s_t, s_{t+1}, y). After preparing our video dataset $\mathcal{D}_V$, we use it to train our instruction network $\hat{p}_\psi$, which is designed to classify whether a given pair of adjacent states is from a Do's video or a Don'ts video. The training objective for the instruction network is a simple binary classification loss:

$$\mathcal{L}^{\text{Instruction}} = -\mathbb{E}_{(s_t, s_{t+1}, y) \sim D_V} \left[y \log \hat{p}_\psi(s_t, s_{t+1}) + (1 - y) \log(1 - \hat{p}_\psi(s_t, s_{t+1})) \right]. \tag{4}$$

4.2 DoDont

Here, we integrate the learned instruction network into the online distance-maximizing skill discovery algorithm. As $\hat{p}_\psi$ is a non-negative function, we can directly use this instruction network as the distance metric of the WDM objective in the Equation 3 as:

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right] \quad \text{s.t.} \quad \|\phi(s) - \phi(s')\|_2 \leq \hat{p}_\psi(s, s'), \quad \forall (s, s') \in S_{adj}, \tag{5}$$

where S_{adj} represents the set of adjacent state pairs. By assigning higher values to behaviors deemed desirable by humans (i.e., large distances) and lower values to behaviors that should be avoided, it guides the agent to learn skills that correspond to human intentions.

Then, following previous work [38, 39, 49], we can optimize Equation 5 with dual gradient descent, incorporating a Lagrange multiplier λ and a small slack variable $\epsilon > 0$:

$$
\begin{aligned}
r^{\text{DoDont}} &:= (\phi(s') - \phi(s))^\top z, \\
\mathcal{J}^{\text{DoDont},\phi} &:= \mathbb{E}[(\phi(s') - \phi(s))^\top z] + \lambda \cdot \min(\epsilon, \hat{p}_\psi(s, s') - \|\phi(s) - \phi(s')\|), \\
\mathcal{J}^{\text{DoDont},\lambda} &:= -\lambda \cdot \mathbb{E}[\min(\epsilon, \hat{p}_\psi(s, s') - \|\phi(s) - \phi(s')\|)],
\end{aligned}
\tag{6}
$$

where r^{DoDont} is the intrinsic reward for the latent conditioned policy, and $\mathcal{J}^{\text{DoDont},\phi}$ and $\mathcal{J}^{\text{DoDont},\lambda}$ are the objectives for ϕ and λ. We note that since the instruction network $\hat{p}_\psi$ is already trained on D_V, we keep $\hat{p}_\psi$ frozen for training stability.

However, one drawback of Equation 5 is that r^{DoDont} can reflect instruction signals from $\hat{p}_\psi$ only after $\phi(s)$ has been sufficiently trained through $\mathcal{J}^{\text{DoDont},\phi}$. This indicates that in the early stages of training where $\phi(s)$ has not yet been sufficiently optimized, the agent may not properly receive instruction signals. Such delay in receiving instruction signals could potentially slow down the training process. Therefore, to ensure the agent receives instruction signals regardless of the training state of $\phi(s)$, we note that the following equation can be easily derived from Equation 5 :

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

For a detailed derivation, please refer to the Appendix B. Through this rephrasing, we now separate $\hat{p}_\psi$ from the training of $\phi(s)$. This allows the agent to receive a direct instruction signal, as $\hat{p}_\psi$ explicitly multiplies with the original intrinsic reward, independent of the training of $\phi(s)$. In our experiment, we found that this direct signal significantly enhances the agent's ability to learn diverse and desirable behaviors (Section 5.3). In addition, another advantage of Equation 7 is that it can be interpreted as simply multiplying instruction network $\hat{p}_\psi$ to the original learning objective function of METRA. This allows us to implement our algorithm by simply adding a single line of code on top of the METRA framework. We provide the pseudocode of DoDont Algorithm 1. Furthermore, our instruction network can be applied to zero-shot offline RL to learn diverse behaviors while prioritizing desirable behaviors within the offline unlabeled dataset. Detailed explanations and experiments are presented in Appendix A.

5 Experiments

5.1 Experimental setup

Our experiments aim to evaluate the effectiveness of the DoDont method in learning desirable behaviors while avoiding undesirable ones across various environments and instruction scenarios. We test DoDont in two contexts: complex locomotion tasks (Cheetah and Quadruped environments from the DeepMind Control (DMC) Suite [45]) and precise manipulation tasks (Kitchen environment [17, 33]) (Figure 2). For pixel-based inputs in the DMC environments, we colored the floors to help the agent be aware of its location from the pixel data, consistent with previous studies [36, 20, 39].

Figure 2: **Benchmark environments.**

For quantitative comparisons in locomotion tasks, we use two main metrics: (i) state coverage and (ii) zero-shot task reward. State coverage is measured by counting the x-axis coverage for Cheetah and x, y-axis coverage for Quadruped achieved by the learned skills at each evaluation epoch, following prior literature [39]. Zero-shot task reward evaluates the learned skills without task-specific training rewards to assess whether the agent efficiently learns or avoids specific behaviors. Our quantitative analysis uses four random seeds and provides 95% confidence intervals, represented by error bars or shaded areas. For each instruction scenario, we use eight videos (four "do" videos and four "don't" videos), unless otherwise specified. Additional details are elaborated in Appendix D.

https://doi.org/10.52202/079017-1512

To comprehensively evaluate DoDont's performance, we compare it against three types of baselines. First, to determine if an instruction network facilitates acquiring intended behavior, we compare DoDont with METRA [39], a state-of-the-art online unsupervised skill discovery algorithm. Second, to assess the effectiveness of our video-based intention network against a hand-designed reward function, we introduce METRA†, a modified version of METRA that uses hand-designed reward functions as the distance metric. Finally, to verify if using an instruction network as a distance function effectively integrates our intention with skill discovery algorithms, we compare it to SMERL [26] and DGPO [10], which integrate task rewards in specialized ways. In this comparison, we use our intention network as the task reward function for both SMERL and DGPO. Details are in Appendix D.

5.2 Main experiment

In this section, our experiments are designed to answer the following research questions:

- How efficiently does DoDont learn complex behaviors in locomotion tasks? (Section 5.2.1)
- Does DoDont learn diverse behaviors while avoiding hazardous areas? (Section 5.2.2)
- Can DoDont learn diverse behaviors without learning unsafe behaviors? (Section 5.2.3)
- Can DoDont be applied to a manipulation environment? (Section 5.2.4)

5.2.1 How efficiently does DoDont learn complex behaviors in locomotion tasks?

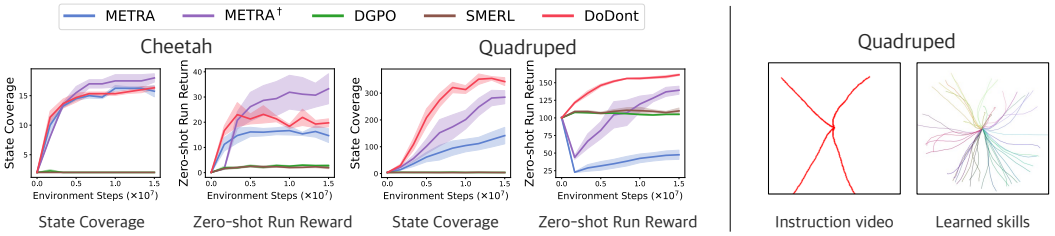

Figure 3: **Left**: State coverage and zero-shot task reward for Cheetah and Quadruped. **Right**: Visualization of Do videos in our instruction video dataset and learned skills by DoDont. We are able to observe that DoDont does not simply mimic instruction videos but extracts desirable behaviors (e.g., run) from the videos and learn diverse skills.

In this experiment, we aim to evaluate whether DoDont can effectively learn diverse complex behaviors by simply providing videos of desirable but complex behaviors. To accomplish this, we set videos of agents successfully running as Do's and random action videos as Don'ts for the Cheetah and Quadruped environment. As shown in Figure 3, DoDont achieves higher running rewards than METRA in both environments. Specifically, in the Quadruped environment, METRA primarily learns rolling movements for locomotion, while DoDont learns to run upright. Please refer to the project page videos for further details (link). Additionally, DoDont exhibits higher state coverage than METRA, since DoDont effectively learn running behaviors which allows the agent to cover longer distances whereas METRA only learn mediocre rolling behaviors.

We also observe that both SMERL and DGPO fail to learn diverse behaviors and only learns simple behaviors such as standing upright at the starting point with minimal movement. This limitation is likely due to their use of the MI objective, which appears insufficient for promoting diverse behavior learning in complex continuous control environments. METRA†, a variant of METRA where we set the run task reward as the distance metric, is effective for simple environments with low-dimensional action spaces (Cheetah), but struggle in learning effective behaviors for environments with high-dimensional action spaces (Quadruped). We speculate that as the environment complexity increases, designing a single reward function which captures a variety of desirable behaviors becomes increasingly challenging.

An important point which we would like to emphasize is that DoDont only requires a total of eight videos to learn running behaviors. Despite the direction of each running video is limited (instruction videos in Figure 3), DoDont learns skill which run in all directions. This indicates that DoDont is not

merely imitating the behaviors in the Do's videos but discover a variety of behaviors which resemble those seen in the Do's videos. In addition, to further evaluate the meaningfulness of the learned behaviors, we conduct downstream task evaluation for each method in the Appendix A.

5.2.2 Does DoDont learn diverse behaviors while avoiding hazardous areas?

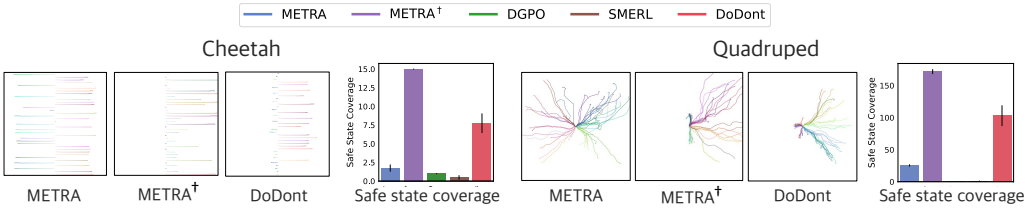

Figure 4: **Visualization and comparison of learned skills.** In both environments, the left side is hazardous and the right side is safe. Safe state coverage assesses the agent's ability to cover safe areas and avoid hazards.

In practical scenarios, it is crucial for agents to avoid hazardous areas. For instance, in navigation tasks, robots must avoid dangerous areas such as pitfalls or private property. However, traditional skill discovery methods, which aim to cover the entire state space, face difficulties in controlling the regions an agent explores. We now aim to evaluate whether the Don't videos in DoDont can reliably prevent agents from learning unwanted area navigation. To replicate real-world conditions, we perform experiments where we designate the left side of the environment as hazardous areas and the right side as safe areas. Thus, we use videos that move to the right as Do's and videos that move to the left as Don'ts, aiming to direct the agent away from hazardous zones and towards safer regions.

To evaluate the agent's ability to cover safe areas while avoiding hazardous ones, we introduced the concept of *safe state coverage*. This metric assigns a score of +1 for states in the safe area and -1 for states in the hazardous area. As illustrated in Figure 4, METRA without prior knowledge attempts to cover the entire state space, including hazardous areas, whereas DoDont effectively incorporates human intentions, avoiding dangerous areas while adequately covering the safe regions. We also observe that SMERL and DGPO fail to learn diverse skills, resulting in static behavior within the safe region, possibly due to the difficulty of learning diverse behaviors in pixel-based environments using the mutual information objective. Furthermore, METRA† demonstrates superior performance in this experiment, likely because designing a reward function that incorporates human intentions is straightforward (assigning +1 for the safe region and 0 for the hazardous region).

5.2.3 Can DoDont learn diverse behaviors without learning unsafe behaviors?

In real-world scenarios, agents should also avoid learning risky behaviors. For instance, actions such as rolling on the ground or flipping upside down can potentially damage the robot. Therefore, in this experiment, we aim to investigate whether DoDont can learn a variety of desirable behaviors while avoiding certain unwanted behaviors. We trained our agent using running videos as desirable behaviors (Do's) and rolling or flipping videos as undesirable behaviors (Don'ts) for both Quadruped and Cheetah environments.

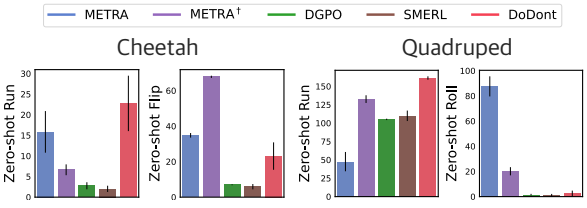

Figure 5: **Learning safe and diverse behaviors.** Zero-shot rewards assess how effectively each method learns desired behaviors while avoiding hazardous ones.

As shown in Figure 5, compared to METRA without any prior knowledge, DoDont effectively learns running behaviors while avoiding hazardous actions such as rolling or flipping using our instruction videos. Again, SMERL and DGPO fail to learn diverse behaviors, merely learning the simple behavior of standing upright at the starting point. For METRA†, we designed the reward function as $r_{\text{run}} - r_{\text{flip or roll}}$ to encourage running while avoiding flipping or rolling. We observe that METRA† cannot fully encapsulate human intention in Cheetah environment. We believe this issue

https://doi.org/10.52202/079017-1512

arises from the entanglement of the task signals in the reward function which makes the agent difficult from discerning which type of behaviors are beneficial and which are not. To receive the run reward, the agent must move forward, but as the body moves, it sometimes triggers the flip reward, which penalizes the agent, hindering the agent from properly learning the run behavior. This illustrates that balancing multiple behavioral intentions is challenging, making the creation of hand-crafted reward functions difficult and unscalable.

5.2.4 Manipulation task

In this experiment, we aim to demonstrate that our intention network can be applied to a manipulation environment to learn multiple tasks. We use six target tasks following previous works [39, 33]. To effectively learn these six tasks, we use the D4RL dataset as Do's and random action videos as Don'ts. As shown in Figure 6, through our instruction network, DoDont successfully performs more tasks compared to METRA. SMERL and DGPO only learn two to three tasks, which we believe is due to the failure of the MI-based objective to capture diverse behaviors in pixel-based environments. For METRA†, we use a hand-designed reward function by summing the rewards for all six tasks. Nevertheless, METRA† does not effectively learn the tasks, likely due to the entanglement of multiple behavioral signals within a single reward function.

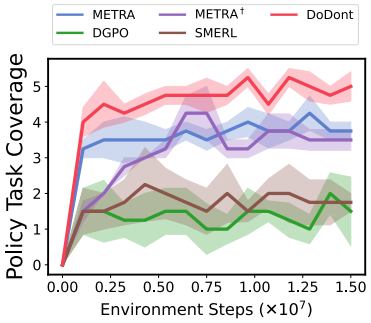

Figure 6: **Policy task coverage.** The number of tasks successfully completed by sampled skills.

5.3 Ablation Studies

5.3.1 Model components

We conduct an ablation analysis to demonstrate the importance of each component of our algorithm in achieving high performance. Specifically, we experimented with three variants: (i) using only one Do's video, (ii) optimizing Equation 5, referred to as delayed reward, and (iii) focusing solely on encouraging Do's behaviors. We tested these variants in a Quadruped environment, using run videos as Do's and random action videos as Don'ts.

Impact of data quantity. While we are already training with a relateively small amount of videos as our default setting (four Do's, four Don'ts), we also conduct an ablation study where we use only two videos (one Do, one Don't) to see how well our method performs under extremely limited data conditions. According to Figure 7, even with only one Do's video, the agent can mimic the running behavior, as evidenced by the zero-shot run reward. Remarkably, despite being exposed to just one video showing movement in a single direction, the agent is still able to learn movements in multiple directions; please refer to the project page videos (link).

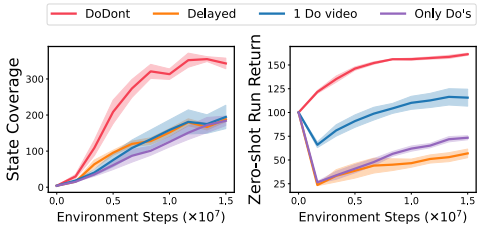

Figure 7: **Ablation study on model components.**

Direct reward signal. To evaluate the effectiveness of Equation 7, we trained an agent using Equation 5. As shown in Figure 7, the direct instruction signal effectively captures the desired behavior from the early stages of training. In contrast, the delayed instruction signal initially fails to capture the desired behavior, as evidenced by drop in the zero-shot reward at the beginning. Consequently, the agent takes a long time to overcome before it can stand up again.

Only encouraging Do's behaviors. To further analyze the importance of penalizing Don'ts behaviors, we trained an agent only to encourage Do's behaviors. As illustrated in Figure 7, solely encouraging Do's behaviors fails to capture desirable behaviors. We speculate that without penalizing unwanted behaviors, the agent cannot avoid these undesirable actions, leading to a drop in the zero-

https://doi.org/10.52202/079017-1512

shot Run return at the beginning. Consequently, it takes a significant amount of time for the agent to learn the desired behaviors. Further details of this experiment in Appendix C.

5.3.2 The importance of utilizing instruction network as the distance metric

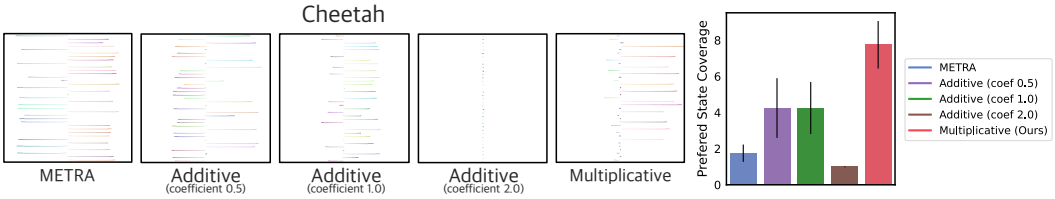

Figure 8: **Left:** Visualization of acquired skills, with the hazardous zone on the left and the safe zone on the right. **Right:** Quantitative comparison of each method.

In this study, we use the instruction network as the distance metric for the distance-maximizing skill discovery algorithm [39]. This setup is convenient because it only requires multiplying the instruction network with the intrinsic USD reward to train the policy. However, an alternative approach is to add the output of the instruction network with the intrinsic USD reward, similar to SMERL [26] and DGPO [10]. To compare these two approaches (multiplication and addition), we conducted experiments in the Cheetah environment following the setup from Section 5.2.2.

Our qualitative results, shown in Figure 8, indicate that none of the additive variants ($r_{\text{total}} = r_{\text{METRA}} + \alpha\hat{p}_\psi(s, s')$) match the performance of the multiplicative approach ($r_{\text{total}} = r_{\text{METRA}} \cdot \alpha\hat{p}_\psi(s, s')$). We attribute this to the fact that multiplying $\hat{p}_\psi(s, s')$ with the intrinsic reward function controls the scale of the total reward effectively since it directly impacts the influence of r_{METRA} while adding $\hat{p}_\psi(s, s')$ indirectly impacts the influence of r_{METRA}. Although additive variants might achieve similar performance, they would likely require careful tuning of coefficients to balance the multiple reward components.

6 Conclusions, limitations, and future work

In this paper, we introduce DoDont, an instruction-based skill discovery algorithm designed to combine human intention with unsupervised skill discovery. We empirically demonstrate that our instruction network enables the agent to effectively learn desirable behaviors and avoid undesirable ones across a range of realistic instruction scenarios.

However, our study faces certain limitations. Specifically, in this research, we trained the instruction network using in-domain video data, which is not a readily available resource in real-world applications. We propose that training the instruction model with general, in-the-wild video data represents a scalable and compelling direction for future investigation.

Despite these constraints, DoDont exhibits promising results even with a limited video dataset. Moreover, we believe that videos are a cost-effective form of data for representing behaviors as they do not require action and reward labels. Thus, preparing data is both practical and feasible. On the other hand, as video generation models advance, several recent works have utilized these models as simulators [50, 51, 12, 6]. There is potential to generate Do's and Don'ts videos with these models, which could allow for the use of generated data without the need for additional data collection.

As final remarks, in unsupervised skill discovery, there are many excellent prior works that have succeeded in learning diverse behaviors without any supervision, data, or prior knowledge. However, these methods often struggle to acquire complex behaviors and may learn hazardous behaviors, making them unsuitable for direct real-world application. Therefore, we believe that the next step in unsupervised reinforcement learning focuses on leveraging minimal supervision or data (i.e., resources that are easily obtainable in the real world) to become scalable and applicable in complex control tasks and practical for the real world.

https://doi.org/10.52202/079017-1512

Acknowledgments

We are grateful to Seohong Park for providing insightful discussions and valuable feedback. This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.RS-2019-II190075 Artificial Intelligence Graduate School Program(KAIST)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1A2B5B02001913).

References

[1] Joshua Achiam, Harrison Edwards, Dario Amodei, and Pieter Abbeel. Variational option discovery algorithms. *arXiv preprint arXiv:1807.10299*, 2018.

[2] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[3] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[4] Albert Bandura. Influence of models' reinforcement contingencies on the acquisition of imitative responses. *Journal of Personality and Social Psychology*, 1965.

[5] Albert Bandura, Ross Ross, Dorothea, and Sheila A. Transmission of aggression through imitation of aggressive models. *The Journal of Abnormal and Social Psychology*, 1961.

[6] Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. 2024.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

[9] Víctor Campos, Alexander Trott, Caiming Xiong, Richard Socher, Xavier Giró-i Nieto, and Jordi Torres. Explore, discover and learn: Unsupervised discovery of state-covering skills. In *International Conference on Machine Learning*, pages 1317–1327. PMLR, 2020.

[10] Wentse Chen, Shiyu Huang, Yuan Chiang, Tim Pearce, Wei-Wei Tu, Ting Chen, and Jun Zhu. Dgpo: discovering multiple strategies with diversity-guided policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11390–11398, 2024.

[11] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[12] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in Neural Information Processing Systems*, 36, 2024.

[13] Alejandro Escontrela, Ademi Adeniji, Wilson Yan, Ajay Jain, Xue Bin Peng, Ken Goldberg, Youngwoon Lee, Danijar Hafner, and Pieter Abbeel. Video prediction models as rewards for reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

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

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

[16] Anthony G Greenwald and Stuart M Albert. Observational learning: A technique for elucidating s-r mediation processes. *Journal of Experimental Psychology*, 1968.

[17] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey 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.

[19] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[20] Danijar Hafner, Kuang-Huei Lee, Ian Fischer, and Pieter Abbeel. Deep hierarchical planning from pixels. *Advances in Neural Information Processing Systems*, 35:26091–26104, 2022.

[21] Steven Hansen, Will Dabney, Andre Barreto, Tom Van de Wiele, David Warde-Farley, and Volodymyr Mnih. Fast task inference with variational intrinsic successor features. *Proc. the International Conference on Learning Representations (ICLR)*, 2019.

[22] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[23] Pierre-Alexandre Kamienny, Jean Tarbouriech, Alessandro Lazaric, and Ludovic Denoyer. Direct then diffuse: Incremental unsupervised skill discovery for state covering and goal reaching. *Proc. the International Conference on Learning Representations (ICLR)*, 2021.

[24] Hyunseung Kim, Byung Kun Lee, Hojoon Lee, Dongyoon Hwang, Sejik Park, Kyushik Min, and Jaegul Choo. Learning to discover skills through guidance. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Saurabh Kumar, Aviral Kumar, Sergey Levine, and Chelsea Finn. One solution is not all you need: Few-shot extrapolation via structured maxent rl. *Advances in Neural Information Processing Systems*, 33:8198–8210, 2020.

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

[28] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[29] Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

[30] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[31] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features. In *Proc. the International Conference on Machine Learning (ICML)*, 2021.

[32] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training. *Advances in Neural Information Processing Systems*, 34:18459–18473, 2021.

 https://doi.org/10.52202/079017-1512

[33] Russell Mendonca, Oleh Rybkin, Kostas Daniilidis, Danijar Hafner, and Deepak Pathak. Discovering and achieving goals via world models. *Advances in Neural Information Processing Systems*, 34:24379–24391, 2021.

[34] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron Van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[35] Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

[36] Seohong Park, Dibya Ghosh, Benjamin Eysenbach, and Sergey Levine. Hiql: Offline goal-conditioned rl with latent states as actions. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Seohong Park, Tobias Kreiman, and Sergey Levine. Foundation policies with hilbert representations. *arXiv preprint arXiv:2402.15567*, 2024.

[38] Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. *arXiv preprint arXiv:2302.05103*, 2023.

[39] Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction. *arXiv preprint arXiv:2310.08887*, 2023.

[40] Xue Bin Peng, Yunrong Guo, Lina Halper, Sergey Levine, and Sanja Fidler. Ase: large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions on Graphics*, 41(4):1–17, July 2022.

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

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

[43] Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. *Proc. the International Conference on Learning Representations (ICLR)*, 2019.

[44] DJ Strouse, Kate Baumli, David Warde-Farley, Vlad Mnih, and Steven Hansen. Learning more skills through optimistic exploration. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

[45] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[46] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[47] Ahmed Touati, Jérémy Rapin, and Yann Ollivier. Does zero-shot reinforcement learning exist? In *The Eleventh International Conference on Learning Representations*, 2022.

[48] Cédric Villani. Optimal transport: Old and new. 2008.

[49] Tongzhou Wang, Antonio Torralba, Phillip Isola, and Amy Zhang. Optimal goal-reaching reinforcement learning via quasimetric learning. In *International Conference on Machine Learning*, pages 36411–36430. PMLR, 2023.

[50] Mengjiao Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

[51] Sherry Yang, Jacob Walker, Jack Parker-Holder, Yilun Du, Jake Bruce, Andre Barreto, Pieter Abbeel, and Dale Schuurmans. Video as the new language for real-world decision making. *arXiv preprint arXiv:2402.17139*, 2024.

[52] Denis Yarats, David Brandfonbrener, Hao Liu, Michael Laskin, Pieter Abbeel, Alessandro Lazaric, and Lerrel Pinto. Don't change the algorithm, change the data: Exploratory data for offline reinforcement learning. *arXiv preprint arXiv:2201.13425*, 2022.

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

[54] Tom Zahavy, Yannick Schroecker, Feryal Behbahani, Kate Baumli, Sebastian Flennerhag, Shaobo Hou, and Satinder Singh. Discovering policies with domino: Diversity optimization maintaining near optimality. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

[55] Zihan Zhou, Wei Fu, Bingliang Zhang, and Yi Wu. Continuously discovering novel strategies via reward-switching policy optimization. *Proc. the International Conference on Learning Representations (ICLR)*, 2022.

 https://doi.org/10.52202/079017-1512

Appendix

A Additional Experiments

A.1 Improving zero-shot offline RL with instruction videos

Recently, HILP [37] has introduced a novel framework for pre-training generalist policies that learn diverse behaviors from task-agnostic unlabeled datasets [52]. HILP can be considered as an offline variant of METRA, as both methodologies focus on learning temporal distance-preserving representations to facilitate the acquisition of diverse behaviors. Similar to our previous application in METRA, we can easily integrate our instruction network into HILP's intrinsic reward function. The pseudocode for the offline version of DoDont is provided in Algorithm 2. Intuitively, by utilizing our instruction network, the policy is trained to learn diverse behaviors while prioritizing desirable behaviors and avoiding hazardous ones within the offline dataset.

In this experiment, we further investigate the effectiveness of our method when applied to an offline zero-shot reinforcement learning (RL) approach that learns diverse behaviors from task-agnostic, unlabeled datasets. Zero-shot RL is divided into two phases: reward-free pretraining with an offline dataset and maximizing an arbitrary reward function provided at test time without additional training. We primarily compared our results with HILP [37], FB [47], and FDM [37, 47], the leading methods in offline zero-shot RL. We conducted experiments in four environments (Walker, Cheetah, Quadruped, and Jaco) using two ExORL datasets [52] collected by APS [31] and RND [8]. We followed the experimental protocol outlined in the HILP paper [37], where detailed information is available in Appendix D.

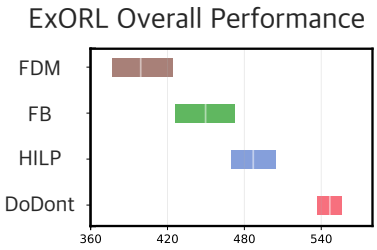

Figure 9: **Aggregated performance.** The overall results are aggregated over 4 environments, 4 tasks, 2 datasets, and 4 seeds (i.e., 128 values in total).

During the pre-training phase, we employed four downstream task videos as Do's and four random action videos as Don'ts for each environment. This strategy aims to prioritize the learning of task-relevant behaviors. We use the interquartile mean (IQM) [2] for overall aggregation to assess performance. As illustrated in Figure 9, DoDont is superior over other methods by a large margin in effectively capturing downstream task behaviors from a large, task-agnostic, unlabeled dataset. We would like to note that although DoDont benefits from downstream task relevant data, it significantly outperforms baselines while utilizing only a small set of instruction videos. Detailed results are in Appendix E.

A.2 Downstream task performance

In this section, we employed a hierarchical controller that selects (frozen) learned skills to maximize the downstream task reward to evaluate the meaningfulness of the learned behaviors. As depicted in Figure 10, behaviors learned through DoDont showed higher performance than METRA, indicating that DoDont effectively learned useful behaviors that can be applied to downstream tasks.

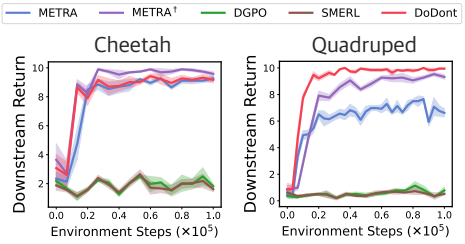

Figure 10: **Downstream task performance.**

https://doi.org/10.52202/079017-1512

B Derivation of equation 7

We first start with Eq. 5:

$$\sup_{\pi,\phi} \mathbb{E}_{p(\tau,z)} \left[\sum_{t=0}^{T-1} \left(\phi(s_{t+1}) - \phi(s_t)\right)^\top z \right] \quad \text{s.t. } \|\phi(s) - \phi(s')\|_2 \leq \hat{p}_\psi(s, s'), \ \ \forall (s, s') \in S_{adj}, \tag{8}$$

Let scaled state function $\tilde{\phi}(s) := \frac{\phi(s)}{\hat{p}_\psi(s,s')}$. Then, we can transform the constraint term in Eq. 8 as follows ($\hat{p}_\psi(s, s') \geq 0$ since it is the output value of binary classification network),

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

By replacing $\phi(s)$ with $\tilde{\phi}(s) \cdot \hat{p}_\psi(s, s')$ in Eq. 9, we derive

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

C Implementation Details of DoDont

Here, we provide further details regarding the implementation of DoDont. Our experiments are based on two different codebases: METRA[1] [39] for online learning and HILP[2] [37] for offline learning. Our experiments run on NVIDIA RTX 3090 GPUs, with each run taking no more than 28 hours.

C.1 Data curation

We now explain how the action-free Do's and Don't video dataset we used as our human intention dataset were collected. For DMC tasks, we first trained expert Soft Actor-Critic (SAC) [18] policies based on proprioceptive states with the ground truth task reward for over 2 million environment steps and generate Do videos by deploying the expert policy in the environment and saving the visual observations rendered by the simulator. In addition, we set the ExORL dataset [52] as our Don't videos for DMC tasks since the ExORL dataset is mainly composed of transitions generated via undesirable random behavior. On the other hand for the Kitchen environment, we set the the D4RL dataset [15] as our Do's video dataset and additionally gather videos of random action rollouts and set them as our Don't videos. The number of videos used for each environment can be found in Table 1.

Table 1: **The number of videos used for each environment.**

Environment	# of Do's videos	# of Don'ts videos
Online DMC	4	4
Kitchen	# of videos in D4RL dataset	10
Offline Zero-shot DMC	12 videos per task	# of videos in ExORL dataset

C.2 Instruction network

For the instruction network, we use four convolutional layers for the backbone and three fully connected layers for the prediction head. To improve stability, we bound the output using the Tanh function, following previous work [29]. Additionally, we employ simple random shift augmentation, a widely used data augmentation technique. The Adam optimizer [25] is employed with a learning rate of 1×10^{-4} and a batch size of 1024.

[1] https://github.com/seohongpark/METRA

[2] https://github.com/seohongpark/HILP

C.3 Skill learning

We implemented DoDont based on two algorithms: METRA [39] and HILP [37]. We used the hyperparameters from these algorithms without modification. Additionally, DoDont introduces only one extra hyperparameter, which is the coefficient for the instruction network. The complete list of hyperparameters can be found in Table 2, 3.

C.4 Pseudocode of DoDont

We present the pseudocode for the online version of DoDont in Algorithm 1 and the offline version of DoDont in Algorithm 2.

Algorithm 1: Do's and Don'ts (Online)

Initialize instruction network $\hat{p}_\psi(s, s')$, video dataset $\mathcal{D}_V$
for *number of epochs* **do**
  Sample batch (σ^0, σ^1, y) from video dataset $\mathcal{D}_V$
  Update $\hat{p}_\psi(s, s')$ to minimize $\mathcal{L}^{\text{Instruction}}$ in (4)
Initialize policy $\pi(a|s, z)$, representation function $\phi(s)$, Lagrange multiplier λ, replay buffer $\mathcal{D}$
for *number of policy epochs* **do**
  for *number of episodes per epochs* **do**
    Sample skill $z \sim p(z)$
    Sample trajectory τ with $\pi(a|s, z)$ and add to replay buffer $\mathcal{D}$
  Update $\phi(s)$ to maximize $\mathbb{E}_{(s,z,s')\sim\mathcal{D}}[(\phi(s') - \phi(s))^\top z + \lambda \cdot \min(\epsilon, 1 - \|\phi(s) - \phi(s')\|)]$
  Update λ to minimize $\mathbb{E}_{(s,z,s')\sim\mathcal{D}}[\lambda \cdot \min(\epsilon, 1 - \|\phi(s) - \phi(s')\|)]$
  Update $\pi(a|s, z)$ using SAC with reward $r(s, z, s') = \hat{p}_\psi(s, s')(\phi(s') - \phi(s))^\top z$

Algorithm 2: Do's and Don'ts (Offline)

Initialize representation $\phi(s)$
Initialize policy $\pi(a|s, z)$
while *not converged* **do**
  Sample $(s, s', g) \sim \mathcal{D}$
  Train $\phi(s)$
Train instruction network $\hat{p}_\psi(s, s')$
while *not converged* **do**
  Sample $(s, a, s') \sim \mathcal{D}$ and $z \sim \mathbb{S}^{D-1}$
  Compute intrinsic reward $r(s, z, s')$
  Train $\pi(a|s, z)$ with weighted reward $\hat{p}_\psi(s, s')r(s, z, s')$

D Experimental details

D.1 Environments

Benchmark environments. In this work, we perform experiments with the Cheetah and Quadruped environment from DMC [45] and a pixel-based version of the Kitchen environment [17, 33]. Following previous studies [39, 36, 20], for the pixel-based DMC environments we use gradient-colored floors which allows the agent to infer its location from pixels. For the Kitchen environment, we utilize the same camera settings as LEXA [33].

For Sections 5.2.1, 5.2.3, and 5.3, we use state information as input for both the policy and the critic. For Sections 5.2.2 and 5.2.4, we use pixel data as input for the policy and the critic. The reason for this experimental setup is that in pixel-based environments with gradient-colored floors, the discriminator might exploit the background color as a shortcut to distinguish between different states rather than observing the agent's embodiment. This makes it challenging to incorporate behavioral intentions. However, without colored floors, METRA cannot learn diverse behaviors. Therefore,

https://doi.org/10.52202/079017-1512

Table 2: **Hyperparameters used in Online DoDont.** We adopt default hyperparameters from METRA [39], introducing only one additional hyperparameter.

Hyperparameter	Value
Learning rate	0.0001
Optimizer	Adam [25]
# episodes per epoch	8
Encoder	CNN [28]
# hidden layers	2
# hidden units per layer	1024
Minibatch size	256
Target network smoothing coefficient	0.995
# gradient steps per epoch	50 (state-based DMC), 200 (pixel-based DMC), 100 (Kitchen)
Replay buffer size	10^6 (state-based DMC), 3×10^5 (pixel-based DMC), 10^5 (Kitchen)
Entropy coefficient	auto-adjust [19] (DMC), 0.01 (Kitchen)
ϵ	0.001
initial λ	30
Latent dimension	4 (continuous) (DMC), 24 (discrete) (Kitchen)
Instruction network coefficient	2

Table 3: **Hyperparameters used in Offline DoDont.** We adopt default hyperparameters from HILP [37], introducing only one additional hyperparameter.

Hyperparameter	Value
# gradient steps	10^6
Learning rate	0.0005 (ϕ), 0.0001(others)
Optimizer	Adam [25]
Minibatch size	1024
MLP dimensions	(512, 512) (ϕ), (1024, 1024, 1024) (others)
TD3 target smoothing coefficient	0.99
Latent dimension	50
# state samples for latent vector inference	10000
Successor feature loss	Q loss on {Quadruped, Jaco}, vector loss (others)
Hilbert discount factor	0.96 (Walker), 0.98 (others)
Hilbert expectile	0.5
Hilbert target smoothing coefficient	0.995
Instruction network coefficient	3 (Walker), 2 (others)

for Sections 5.2.1, 5.2.3, and 5.3, where behavioral intentions are included, we use state-based environments and use non-colored default pixel as a human intentions. Notably, all experiments use videos (i.e., pixel data) as a human intentions.

For offline zero-shot RL, we use four environments (Walker, Cheetah, Quadruped, and Jaco) and two different ExORL datasets [52] (APS [31], RND [8]). Although experiments in the HILP [37] paper is based on four ExORL datasets (APS [31], RND [8], Proto [53], APT [32]), due to limited computational resources, we performed experiments only with the APS and RND ExoRL dataset since these two dataset are where HILP achieved the best performance.

Metrics. For the state coverage in DMC, we count the number of 1×1 $x - y$ bins occupied by any of the target trajectories for Quadruped and 1-sized x bins for the Cheetah. In the Kitchen environment, we count the number of predefined tasks achieved by any of the target trajectories. We use the same six predefined tasks as LEXA [33] and METRA [39]: Kettle (K), Microwave (M),

Light Switch (LS), Hinge Cabinet (HC), Slide Cabinet (SC), and Bottom Burner (BB). Policy state coverage is computed using 48 deterministic trajectories with 48 randomly sampled skills at the current epoch.

For offline zero-shot RL performance, we follow the method outlined in HILP [37]. We sample a small number of (s, a, s') tuples from the dataset, compute the optimal latent z^* with respect to the test-time reward function, execute the corresponding policy $\pi(a|s, z^*)$, and compute the return value. We use all the hyperparameters from HILP without modification.

Downstream tasks. We utilize two downstream tasks: QuadrupedGoal and CheetahGoal, primarily following METRA [39]. For each task, the objective is to reach a target spot randomly sampled from $[-7.5, 7.5]^2$ for QuadrupedGoal and $[-10.0, 10.0]^2$ for CheetahGoal. The agent receives a reward of 10 upon reaching within a radius of 3 units from the target spot. We train a hierarchical high-level controller on top of the frozen skill policy. The high-level controller selects a skill z for every $K = 50$ environment steps, and the pre-trained skill policy $\pi(a|s, z)$ executes the same skill z for those K steps. We use PPO [42] for discrete skills and SAC [18] for continuous skills, maintaining all hyperparameters as in METRA [39].

D.2 Implementation details of baseline algorithms

In this work, we use four baselines: METRA [39], METRA†, DGPO [10], and SMERL [26]. For METRA, we used the open-source METRA codebase and retained the original hyperparameters. METRA† is a variant of METRA that incorporates hand-designed reward functions as a distance metric. As described in the main paper, we manually designed the reward function and rescaled it to be greater than zero since it is used as a distance metric. Like the DoDont method, we multiplied METRA's intrinsic reward by this hand-designed reward function.

DGPO introduces an intrinsic reward function $r^{\text{int}} = \min_{z' \neq z} \log \frac{q(z|s')}{q(z|s')+q(z'|s')}$. With this intrinsic reward, we use our instruction network as a human intention. Total reward function for DGPO is $r = \hat{p}_\psi(s, s') + \alpha \min_{z' \neq z} \log \frac{q(z|s')}{q(z|s')+q(z'|s')}$, where α is the balancing coefficient.

SMERL uses an intrinsic reward defined as $r^{\text{int}} = \mathbb{1}_{R_\mathcal{M}(\pi_\theta) \geq R_\mathcal{M}(\pi^*_\mathcal{M}) - \epsilon} \tilde{r}$. Like DGPO, we employed our instruction network as a human intention. To train SMERL, we first trained a baseline SAC agent with our instruction network and considered the maximum return achieved by the trained SAC agent as the optimal return $R_\mathcal{M}(\pi^*_\mathcal{M})$. We used the same $\tilde{r} = q_\phi(z|s)$ as described in the original paper. The total reward function for SMERL is $r = \hat{p}_\psi(s, s') + \alpha \mathbb{1}_{R_\mathcal{M}(\pi_\theta) \geq R_\mathcal{M}(\pi^*_\mathcal{M}) - \epsilon} q_\phi(z|s)$, where α is the balancing coefficient.

In Section 5.3, we introduce "Only Do's," a variant of our algorithm that exclusively promotes desired behaviors. We achieve this by eliminating the penalties for undesirable behaviors. Specifically, we define $p_{\text{dos}}(s, s')$ as follows:

$$p_{\text{dos}}(s, s') = \begin{cases} \hat{p}_\psi(s, s') & \text{if } \hat{p}_\psi(s, s') \geq 0.5 \\ 0.5 & \text{if } \hat{p}_\psi(s, s') < 0.5 \end{cases} \tag{11}$$

This approach ensures that the probability $p_{\text{dos}}(s, s')$ never falls below 0.5, thereby encouraging only the desired behaviors.

D.3 License

- Environment
  - Deepmind Control suite: We used the publicly available environment code in [39, 45][3].
  - Kitchen: We used the publicly available environment code in [39, 15][4].
- Dataset
  - ExORL: We used the publicly available dataset code in [52][5].

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

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

[5] `https://github.com/denisyarats/exorl`

https://doi.org/10.52202/079017-1512

E Full results

E.1 Zero-shot RL

Table 4: **Full results on the zero-shot RL performance.** The table shows the zero-shot RL performance averaged over four seeds in each setting. We adopted the results from HILP [37]

Dataset	Environment	Task	FDM	FB	HILP	DoDont
APS	Walker	Flip	426± 68	334 ± 178	573 ± 37	506 ± 70
		Run	248 ± 23	388 ± 27	348 ± 14	355 ± 46
		Stand	865 ± 77	824 ± 54	883 ± 42	953 ± 22
		Walk	634 ± 91	842 ± 105	862 ± 31	930 ± 34
	Cheetah	Run Backward	116 ± 116	250 ± 135	373 ± 72	383 ± 19
		Run	360 ± 17	251 ± 39	316 ± 21	290 ± 44
		Walk	396 ± 287	683 ± 267	939 ± 55	984 ± 2
		Walk Backward	982 ± 2	980 ± 3	985 ± 2	985 ± 2
	Quadruped	Jump	707 ± 30	757 ± 52	623 ± 149	721 ± 30
		Run	481 ± 4	474 ± 33	411 ± 62	489 ± 6
		Stand	961 ± 20	949 ± 30	797 ± 117	965 ± 4
		Walk	578 ± 145	583 ± 12	605 ± 75	568 ± 55
	Jaco	Reach Bottom Left	12 ± 18	14 ± 12	88 ± 41	61 ± 17
		Reach Bottom Right	28 ± 20	24 ± 7	48 ± 24	60 ± 24
		Reach Top Left	21 ± 16	23 ± 17	49 ± 18	88 ± 31
		Reach Top Right	34 ± 40	17 ± 15	51 ± 32	64 ± 22
RND	Walker	Flip	415± 20	548 ± 94	563± 136	644 ± 33
		Run	295 ± 70	409 ± 15	401 ± 30	436 ± 22
		Stand	821 ± 84	866 ± 120	800 ± 61	894 ± 31
		Walk	476 ± 259	811 ± 52	855 ± 34	872 ± 32
	Cheetah	Run Backward	107 ± 26	183 ± 83	262 ± 53	257 ± 6
		Run	177 ± 62	153 ± 41	187 ± 55	226 ± 10
		Walk	494 ± 122	636 ± 291	823 ± 141	944 ± 15
		Walk Backward	652 ± 274	677 ± 85	843 ± 184	955 ± 1
	Quadruped	Jump	758 ± 98	642 ± 36	556 ± 101	684 ± 16
		Run	491 ± 7	436 ± 26	393 ± 42	494 ± 35
		Stand	971 ± 11	797 ± 72	810 ± 97	859 ± 79
		Walk	601 ± 82	642 ± 202	542 ± 32	753 ± 58
	Jaco	Reach Bottom Left	53 ± 20	18 ± 11	19 ± 20	33 ± 14
		Reach Bottom Right	38 ± 14	29 ± 12	18 ± 17	29 ± 31
		Reach Top Left	36 ± 19	45 ± 16	8 ± 10	33 ± 15
		Reach Top Right	44 ± 24	22 ± 7	5 ± 4	9 ± 9

F Broader impact

Our research has a broader impact on the safety and reliability of unsupervised skill discovery (USD) algorithms. By incorporating instruction-based learning to differentiate between desirable and undesirable behaviors, our method significantly lowers the risk of agents developing unsafe behaviors, such as tripping or navigating hazardous areas. This improvement enhances the practical usability of USD in real-world scenarios where safety is critical, especially in applications like autonomous vehicles and robotic manipulation. However, a major issue with USD remains its sample inefficiency due to the need for extensive simulations.

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: We accurately reflect the paper's contribution and scope in the abstract and introduction.

   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: We discuss our limitation and future work 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]

https://doi.org/10.52202/079017-1512

Justification: Yes, we provide the full set of assumptions and a complete proof in Appendix B.

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: We provide pseudocode and hyperparameter list in Appendix C

   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: We provide our code link in Appendix C

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, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide the training and test details in Appendix C, D

   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: We report error bars in all experiments.

   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.

https://doi.org/10.52202/079017-1512

- 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 provide our GPU resources, and time of execution in Appendix C.

   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: We conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   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 discuss the broader impacts of our paper in Appendix F

    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: We do not use data or models that have a high risk for misuse.

    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: The creators or original owners of assets (e.g., code, data, models) used in the paper are properly credited, and the license and terms of use are explicitly mentioned and properly respected in Appendix D.3.

    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.

https://doi.org/10.52202/079017-1512

- 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: [NA]

    Justification: We do not release new assets.

    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: We do not involve crowdsourcing experiments.

    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: We do not involve crowdsourcing experiments.

    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.