Discovering Skills with Language

Arvind Rajaraman[1] **Vivek Myers**[1] **Anca Dragan**[1,2]
[1]UC Berkeley [2]Google DeepMind
`{arvind.rajaraman,vmyers,anca}@berkeley.edu`

Abstract

Unsupervised reinforcement learning (RL) algorithms promise to learn meaningful behaviors in the absence of a known reward function. In practice, these approaches fail to scale effectively to high-dimensional settings. The fundamental issue is that in the absence of additional structure, the unsupervised RL problem is underspecified: the best we can do is try and discover skills that cover the full space of behaviors, most of which will be meaningless. In many environments where we might wish to do skill discovery, language can provide precisely this additional structure, telling us which behaviors are *meaningful* to discover. We argue that applying the structure of pre-trained language embeddings as a prior for skill discovery enables us to learn more meaningful behaviors, demonstrating our approach (**DLSD**: **D**iverse **L**anguage-based **S**kill **D**iscovery) in the open-world Crafter environment. We make our code available at `https://github.com/arvindrajaraman/language-skills-diayn`.

1 Introduction

Deep reinforcement learning (RL) permits agents to learn behaviors in the presence of a well-shaped reward function. Unfortunately, in most real-world settings we do not closed-form rewards that are aligned with human preferences [1, 2]. For robots to be deployed in the real world and around humans, they must quickly generalize to unseen downstream tasks (with few-shot examples). Supervised RL is known to be brittle against task shift, and hand-designing reward functions for many downstream tasks is hard to scale [3].

Unsupervised skill discovery algorithms [4–8], aim to learn a set of diverse behaviors (known as *skills*), with the hope that these behaviors can be composed together or guide exploration for some unseen task at test-time. Skill discovery can also be used to shape exploration in sparse-reward environments. There is evidence that human children need language for early cognitive development and skill acquisition in childhood [9, 10]. Unfortunately, skill discovery algorithms fail to learn useful skills in high-dimensional environments, since agents cannot efficiently figure out what behaviors are semantically different. High-dimensional environments are known to contain sources of near-infinite novelty [11].

In this paper, we propose **DLSD** (**D**iverse **L**anguage-based **S**kill **D**iscovery), a skill discovery algorithm that maximizes the mutual information between language descriptions of observations and skills (see Figure 1). Language is inherently abstractable, recursive, context-sensitive, and is built to serve humans, making it an effective representation of observations for RL agents. DLSD produces behaviors that are *semantically* diverse by discerning between behaviors in the language embedding space.

Our core contributions are as follows.

1. We show our approach (DLSD) exploits language to improve unsupervised skill discovery in the absence of a ground-truth reward.

Preprint. Under review.

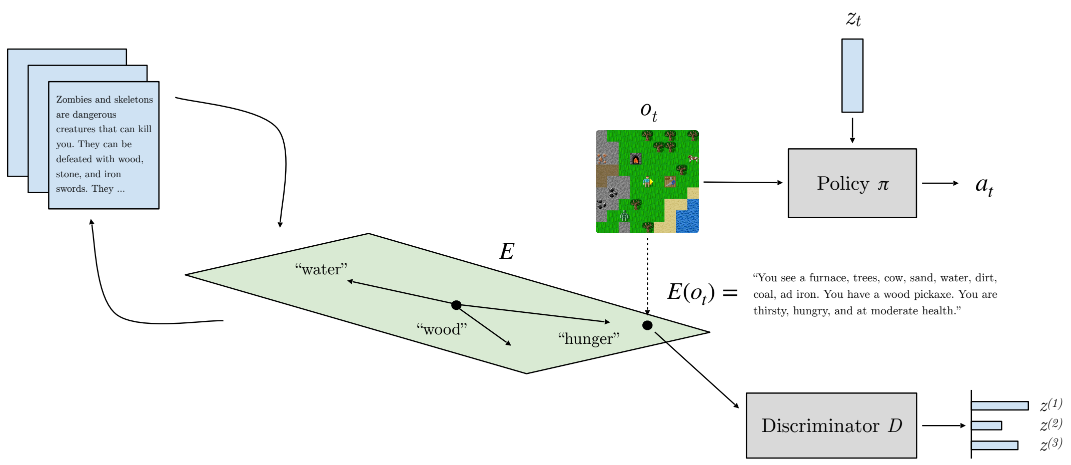

Zombies and skeletons are dangerous creatures that can kill you. They can be defeated with wood, stone, and iron swords. They ...
z_t
o_t
Policy π
a_t
E
"water"
"wood"
"hunger"
$E(o_t) =$
"You see a furnace, trees, cow, sand, water, dirt, coal, ad iron. You have a wood pickaxe. You are thirsty, hungry, and at moderate health."
Discriminator D
$z^{(1)}$
$z^{(2)}$
$z^{(3)}$

Figure 1: DLSD uses the structure of language to learn behaviors that are more useful to the task at hand. The language encoder is pre-trained on internet-scale data, including information about survival games like Crafter. DLSD's skill-conditioned policy uses raw observations for decision-making, while the discriminator learns from the language embedding space. By maximizing discernability in this space, the DLSD policy produces *semantically* different skills in Crafter.

2. DLSD serves as a more effective mechanism for pre-training a policy than traditional skill discovery, aligning the learned skills more closely to the distribution of downstream tasks that humans may request.
3. In the presence of a ground-truth reward function, our approach of maximizing mutual information between language embeddings and skills can be used as reward shaping bonus, making it easier to learn complex behaviors than with the ground-truth reward alone.

We demonstrate these claims in the Crafter environment [12], an open-ended 2-D survival game where an agent most unlock achievements organized in a tree, with harder achievements (e.g. making a wood sword) depending on the exploration of easier achievements (e.g. collecting wood) . Such a game requires agents to compose primitive behaviors to achieve high rewards, making it a useful environment to test the ability of unsupervised RL to shape exploration.

2 Related Work

Our approach lies at the intersection of unsupervised skill discovery and natural language in RL.

Unsupervised exploration. In the absence of a reward function, unsupervised exploration constructs a pseudo-reward that measures how novel or out-of-distribution a sample is [13, 14]. Recent methods include increasing state-space coverage [15] measuring the prediction error of a neural network trained on in-distribution samples [11], self-supervision over observed samples [16], maximizing mutual information between tasks and policy-induced states [17], and maximizing entropy in an embedding space [18]. Exploration can also be done by measuring the inability of the agent to predict the environment [19, 20], such as the forward/backward dynamics [21].

Unsupervised skill discovery. Skill discovery aims to learn a diverse set of behaviors. The most popular is DIAYN [5], which maximizes the mutual information between states and skills by learning a neural discriminator that associates states to skills. DIAYN's discriminator is prone to overfitting in complex environments though. Other methods include learning the dynamics of behaviors [22], seeking behaviors that are hard to achieve with the current set of skills [23], and learning skills that can can follow goals zero-shot [8]. Work has also been done in the discovery of skills within embedding spaces, such as those that encode behaviors [7] and increasing coverage in embedding space [6],

Using language in RL. Prior work has seen success in using language to associate high-level behaviors to primitive language instructions [24–27], using language to better shape rewards [28–

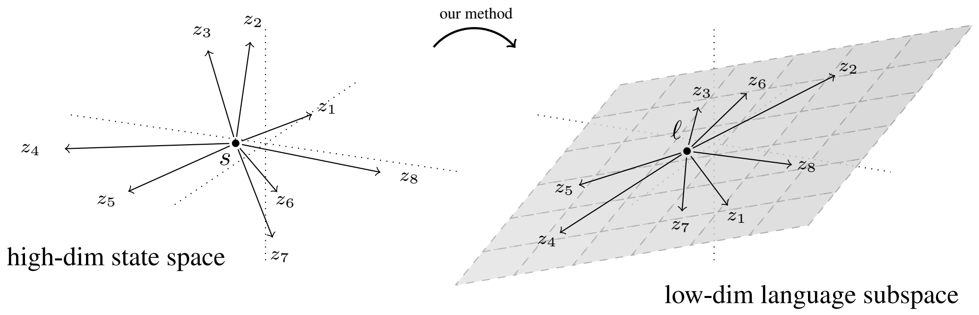


Figure 2: Traditional skill discovery methods that try and cover the full state space do not scale well to high-dimensional observations (left, skills z in state space). We propose to use language embeddings as a semantic prior to guide skill discovery (right, skills z in language space). By learning skills that project down to cover the lower-dimensional language (sub)space, we focus on discovering *meaningful* skills in the environment.

31], and using language models to suggest goals that may align to the unknown downstream task distribution [32, 33].

3 Using Language to Discover Skills

Traditional skill discovery algorithms fail to produce diverse behaviors in high-dimensional environments, because the agent is unable to associate complex behaviors with each other. Thus, conducting discriminator learning in a rich embedding space is necessary to learn a set of useful skills [7, 6].

We propose that language can precisely serve as an informative embedding space. The use of language to guide skill discovery in simulated and real-world settings is made easy due to the wide availability of LLMs [34]. Unsupervised RL algorithms try to manually impose structure to learn more useful behaviors, such as learning representation functions [6, 18] or imposing some novelty bias [11]. On the other hand, DLSD uses language as this structure, making our algorithm easier to scale and implement.

Language embeddings also exhibit the following qualities:

- *Abstraction*: Language embeddings provide an inherent weighting of the importance of certain features. For example, the embeddings for "iron sword" and "stone sword" are likely to be similar, while. the embeddings of "iron sword" and "iron pickaxe" will be more different, since the identity of the object is more semantically different than its material (see Figure 3). Using one-hot embeddings on the above three terms may yield equidistant embeddings, which is not ideal. Thus, a language-based skill discovery agent can focus on important features.
- *Human Usefulness*: Since language embeddings are pre-trained on internet-scale, human-produced data, these embeddings contain information about what humans do and do not find useful. This has implications for skill discovery, since language can guide the agent to behaviors that humans find useful. Also language will contain information on the downstream distribution of tasks that an agent may be asked to perform by a human.
- *Context Sensitivity*: The meaning of language changes depending on the information around it. When conducting skill discovery, language embeddings consisting of lists of entities/items can quickly learn whether combinations of entities/items are useful. Thus, language gleans insight into what primitive behaviors are relevant for complex behaviors, a quality important for semantic skill discovery.

Additionally, the data that LLMs are pre-trained on likely includes information about survival, exploration, common-sense, and even dynamics of games similar to (or exactly) Crafter. This information is also useful for real-world exploration and navigation, making language embeddings easy to scale in real-world settings.

4 Approach

Supervised RL problem. Consider a partially observable Markov decision process (POMDP) defined by the state space $\mathcal{S}$, action space $\mathcal{A}$, observation space Ω, reward function $\mathcal{R}$, and discount factor γ. Observations $o \in \mathcal{O}$ are derived from states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ via the observation dynamics $\Omega(o \mid s)$. States s are evolved to s' via the transition dynamics $\mathcal{T}(s' \mid s, a)$. The ground-truth reward function $\mathcal{R}_{gt}(o, a, o')$ returns a scalar given the current observation, action taken, and next observation. A reinforcement learning algorithm then trains a policy π to maximize expected rewards.

Unsupervised skill discovery. Unsupervised skill discovery modifies this problem by learning a set of diverse *skills* z sampled from some distribution of skills $p_{\mathcal{Z}}(z)$. The goal is to learn a skill-conditioned policy $\pi(a \mid o, z)$ that produces distinct behaviors depending on z. The ground-truth reward function $\mathcal{R}_{gt}$ is replaced with a pseudo-reward function $\mathcal{R}_{pr}(o, a, o', z)$ that also intakes the current skill.

We adopt the DIAYN framework from Eysenbach et al. [5], which tries to maximize the mutual information between observations and skills:

$$I(\mathcal{O}; Z) = D_{KL}(p_{\mathcal{O},\mathcal{Z}}(o, z) \parallel p_{\mathcal{O}}(o)p_{\mathcal{Z}}(z))$$

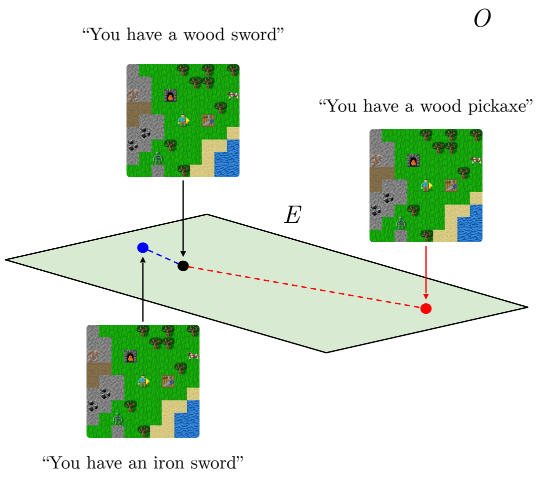


Figure 3: Language provides the necessary structure for discriminator learning to produce semantically different behaviors. While the distances between the 3 observations might be equidistant in one-hot observation space $\mathcal{O}$, "wood sword" and iron sword" will be much closer in language space E than the former and "wood pickaxe". Language weights the importance of the item's identity over its material. This may facilitate discovering distinct skills for attacking enemies and mining ore.

DIAYN introduce a discriminator $D_\phi(z \mid o)$ that classifies an observation to a specific skill, parametrized by ϕ. The pseudo-reward function is:

$$\mathcal{R}_{pr}(o', z) = \log D(z \mid o') - \log p(z)$$

Thus, the agent is rewarded for visiting states that the discriminator sees as highly associated with the current skill z. The discriminator is trained via regression between predicted and ground-truth skills. The distribution over skills $p_{\mathcal{Z}}(z)$ in DIAYN is uniform.

Our Method: DLSD. We introduce an observation captioner E that maps observations to *language* embeddings. The captioner E is pre-trained on some large, internet-scale corpora of text (details in appendix A). It is kept frozen during skill discovery.

In our method, the discriminator $D_\phi(z \mid E(o))$ learns from language embeddings while the policy $\pi(a \mid o, z)$ learns from the raw observations. Language associates states that are semantically similar, despite being different under primitive distance metrics (like L_2 norm). Large distances in the language embedding space are likely to consistently correspond to observations reached from semantically different behaviors. While discriminators trained on raw observations achieve high accuracy early-on in training, they are prone to overfitting to artifacts in the dataset. These artifacts are unlikely to lead the policy to a diverse set of behaviors.

We do not use language embeddings as input to our policy though, because language is too reductive to be used for decision-making (e.g. the agent might need to know specific locations and counts of entities in its environment). We found that passing in both the observation and language embedding $(o, E(o))$ degraded performance of the discriminator (see 5.2)

Figure 4: The Crafter environment. The agent must unlock achievements organized in a tree, with harder achievements depending on the exploration of easier achievements.

In environments where a ground-truth reward function is easily accessible, skill discovery can be used as an intrinsic motivator to learn a set of primitive yet diverse behaviors. In such a setting, we trade-off the pseudo- and ground-truth reward to produce the following hybrid reward function:

$$\begin{aligned}\mathcal{R}_h(o, a, o', z) &= \mathcal{R}_{gt}(o, a, o') + \lambda \mathcal{R}_{pr}(o, a, o', z) \\ &= \mathcal{R}_{gt}(o, a, o') + \lambda(\log D(z \mid o') - \log p(z))\end{aligned}$$

We also augment the DIAYN training dynamics with ϵ-greedy exploration, warmup steps, a replay buffer, and periodic model updates. We found that all of these were necessary for more stable discovery of skills. Pseudocode can be seen in Algorithm 1. If one does not have access to a ground-truth reward function, purely the psuedo-reward may be used. We run experiments in both settings.

Algorithm 1: DLSD: **D**iverse **L**anguage-based **S**kill **D**iscovery

```
Initialize policy $\pi$, discriminator $D$, replay buffer $\mathcal{B}$, and captioner $E$.
Initialize skill $z \sim p_{\mathcal{Z}}(z)$, state $s$, and observation $o \sim \Omega(\cdot \mid s)$.
for $t \leftarrow 0$ to num_steps do
    Sample action: $a \sim \pi(\cdot \mid s, z)$.
    Step environment: $s' \sim \mathcal{T}(\cdot \mid s, a)$.
    Get next observation: $o' \sim \Omega(\cdot \mid s')$.
    Add transition to replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(o, a, o', z)\}$.
    if terminated then
        Reinitialize skill $z$, state $s$, and observation $o$.
    end
    if $t \mod \text{update\_freq} = 0$ then
        Sample a batch of transitions $\{(o_i, a_i, o'_i, z_i)\}_i \sim \mathcal{B}$.
        Update $\pi$ using DQN [35] with reward $\mathcal{R}_{pr}(o_i, a_i, o'_i, z_i) + \lambda \mathcal{R}_{gt}(o_i, a_i, o'_i)$.
        Update $D$ using Adam [36] by minimizing $\|D(\hat{z}_i \mid E(o)) - z_i\|_2$.
    end
end
```

5 Experiments

We run experiments in the Crafter environment [12], a 2-D Minecraft-like survival game where an agent must complete achievements like crafting tools, defeating enemies, and collecting materials (Fig. 4). Crafter achievements are organized in a tree (see Figure 4 of [12]), where advanced achievements like making a stone pickaxe depend on primitive achievements like collecting wood. Crafter worlds are partially observable and procedurally generated, so Crafter agents must learn general survival tactics and complete long sequences of actions. Such an environment is a perfect place to explore the performance of unsupervised skill discovery algorithms, where primitive behaviors like collecting items and navigating need to be composed together to complete the advanced achievements.

Table 1: Comparison with baselines and ablations

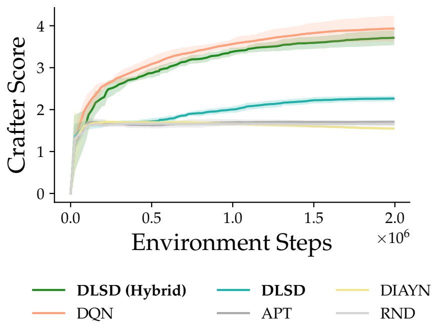
	No Reward				Ground Truth Reward	
Achievement	**DLSD**	DIAYN	APT	RND	**DLSD Hybrid**	DQN
Wake up	72.4	85.1	**85.8**	65.1	83.7	**87.0**
Eat cow	**1.14**	0.15	0.383	0.409	2.44	**5.01**
Collect wood	**59.1**	38.5	44.3	42.6	83.4	**88.4**
Collect drink	9.33	11.1	19.9	**20.1**	16.5	**29.4**
Collect sapling	**51.2**	32.6	32.1	28.3	64.4	**73.3**
Defeat zombie	**0.353**	0.11	0.147	0.12	1.23	**8.85**
Defeat skeleton	**0.036**	0.002	0.013	0.014	**0.0690**	0.052
Place table	**22.5**	7.23	9.37	9.37	52.5	**61.4**
Place plant	**36.8**	26.2	22.3	20.3	54.2	**67.8**
Make wood pickaxe	**2.46**	0.413	0.631	0.693	**11.0**	8.65
Make wood sword	**2.55**	0.4	0.591	0.68	**9.89**	7.91
Eat plant	0.0	0.0	0.0	0.0	0.0	0.0
Collect coal	**0.058**	0.003	0.008	0.002	**0.23**	0.097
Collect stone	**0.6**	0.055	0.114	0.093	**3.0**	1.3
Place stone	**0.225**	0.018	0.047	0.038	**1.21**	0.5
Place furnace	**0.259**	0.023	0.045	0.046	**1.42**	0.477
Make stone pickaxe	**0.005**	0.0	0.0	0.002	**0.022**	0.01
Make stone sword	0.0	0.0	0.002	0.0	**0.019**	0.006
Collect iron	**0.002**	0.0	0.0	0.0	**0.002**	0.0
Make iron pickaxe	0.0	0.0	0.0	0.0	0.0	0.0
Make iron sword	0.0	0.0	0.0	0.0	0.0	0.0
Collect diamond	0.0	0.0	0.0	0.0	0.0	0.0
Crafter Score	**2.27**	1.55	1.71	1.65	3.72	**3.93**
NLPSV	**23.4**	24.5	363	364	-	362

Our experiments include DLSD on both the hybrid reward and solely the pseudo-reward. These are the baselines we benchmark DLSD against:

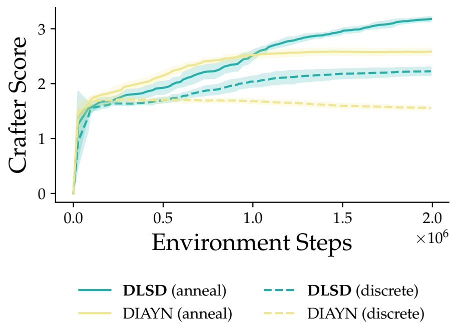
- **DIAYN** [5] - This method performs mutual information (MI) maximization over the *raw observations* and skills. We compare this against DLSD, which maximizes (MI) over the language embeddings and skills, in order to measure the ability of language to help the agent learn "meaningful" behaviors (specifically, the 22 achievements outlined in Crafter) without directly training on them.
- **RND** [11] - This unsupervised exploration method computes novelty by passing observations into a randomly initialized network and then computing the prediction error of a fine-tuned network on this random embedding. The latter network provides a measure of how in-distribution an observation is.
- **APT** [18] with ICM [21] - APT is an unsupervised exploration method that trains an observation encoder and produces a pseudo-reward that is precisely the distance to an observation's k-nearest neighbors in the encoder's output space. We train this encoder by measuring how well the agent can predict the forward/backward environment dynamics given samples so far. Further implementation details are in Appendix C. RND and APT serve as novelty-based exploration baselines to demonstrate whether DLSD's skills seek *useful* novelty.
- **DQN** [35] - We train DQN on the ground-truth reward to benchmark how effectively DLSD with hybrid reward can shape exploration.

We report the following metrics for each of our experiments:

- **Achievement rates** A_i: For a specific achievement, this represents the percentage of training episodes for which this achievement was unlocked at least once.



(a) DLSD outperforms DIAYN in terms of overall Crafter score by using language embeddings as a proxy for "human usefulness."



(b) Comparison of DLSD and DIAYN as pre-training. DLSD adapts more readily to the ground-truth reward both directly (discrete) and when annealed.

Figure 5: Comparison of approaches by overall score.

- **Crafter score**: This is the original environment reward from Hafner [12]:

$$\text{Crafter score} \triangleq \exp\left(\frac{1}{22}\sum_{i=1}^{22}\log\left(100A_i + 1\right)\right) - 1 \tag{1}$$

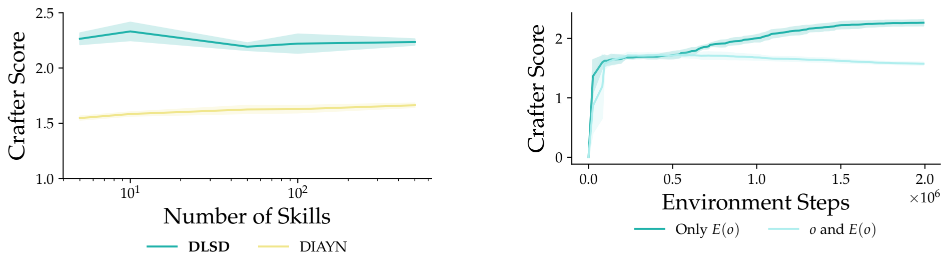
  In other words, the Crafter score is a geometric mean over the 22 achievement rates in Crafter. This is chosen over the arithmetic mean to reward agents for achieving new achievements instead of exploiting already-solved achievements. The $+1$ and -1 terms are added for numeric stability and 100 is added to provide a more human-readable score.

- **NLPSV** (**N**egative **L**og **P**roduct of **S**ingular **V**alues) **metric**: A diverse set of skills ideally produce behaviors that complete different sets of achievements. This metric aims to measure exactly that. Our motivation is that the determinant of a square matrix (which is the product of its singular values) measures the orthogonality of its rows. Extending this idea to our setting, consider the matrix of achievement rates per skill. The product of singular values of this matrix represents the ability of an algorithm to produce more diverse beahviors. We take the negative log of this value for readability purposes, making a lower quantity more desirable.

Our results are summarized in Fig. 5, and Table 1 shows the achievement rates, Crafter score, and NLPSV for each algorithm, grouped into those with exposure to the ground-truth reward and those without. The highest achievement rate, highest Crafter score, and lowest NLPSV are bolded among each group. The achievement rates are topologically sorted based on how early they are in the Crafter achievement tree. All rates are averaged over 5 random seeds. The best value is bolded amongst the algorithms without exposure to the ground-truth reward (DLSD, DIAYN, APT, and RND) and amongst those with exposure (DLSD Hybrid and DQN). DLSD considerably outperforms DIAYN on the Crafter score, even when not exposed to the ground-truth reward. Though the novelty-seeking baselines (APT and RND) outperform DIAYN in terms of Crafter score, language permits DLSD to considerably beat those baselines. For individual achievements as well, DLSD consistently outperforms the other baselines.

In addition, the NLPSV of DLSD is the lowest of all methods, providing evidence that our method is more effectively learning diverse behaviors under a set of behaviors that humans find useful. This furthers our claim (1) that language is a more effective modality than raw observations for skill discovery.

The achievement rates of DLSD hybrid also provides evidence of claim (3). While the DQN agent is able to exploit more primitive achievements effectively, DLSD trained on hybrid reward is able to unlock more difficult achievements at higher rates, showing that mutual information between language embeddings and skills serves as a scalable intrinsic reward in open-ended environments.



(a) We check the robustness of our DLSD approach across different numbers of learned skills. By using language, DLSD consistently outperforms the non-language DIAYN.

(b) We test whether the skill discovery agent can learn more effectively when learning from both the language embedding and raw observation.

Figure 6: Ablations of our DLSD approach.

5.1 Pre-Training Experiments

We additionally test the ability of DLSD to pre-train a policy and fine-tune on the ground-truth reward once it is known. We test two pre-training regimes:

- **Discrete**: For the first half of training, the agent discovers skills using pseudo-reward. For the second half of training, the agent then solely optimizes on the ground-truth reward.
- **Anneal**: The reward is linearly annealed between fully using the pseudo-reward in the beginning of training and fully using the ground-truth reward by the end of training. We use $\lambda = 5$.

As seen in Figure 5, DLSD outperforms DIAYN by a marginal amount in both the discrete and annealed settings. The annealed settings do better overall likely because the Q-network can slowly adjust to the change in magnitudes of rewards.

This furthers claim (2) that DLSD learns a more easily adaptable and generalizable policy initialization when the downstream task distribution is human-aligned but unknown.

5.2 Ablations

We performed the following ablations:

- Quantifying how the number of skills DLSD uses impacts the ability to learn human-useful diverse behaviors.
- Whether the discriminator can learn more effectively when learning from both the language embedding and raw observation.

The results of these ablations are shown in Figure 6. Increasing the number of skills provides a slight improvement to DIAYN but not for DLSD, showing that the language embedding space allow even a small skill size to permit the discovery of diverse behaviors.

Looking at the next ablation, surprisingly providing both the raw observation and language embedding degrades skill discovery. The raw observation in a complex, open-ended environment likely contains much more noise than signal, so much so that it is unable to learn with an informative language embedding. Thus, we omit the raw observation to the discriminator in our experiments.

6 Conclusion

DLSD provides evidence that language-based skill discovery can help shape exploration in open-ended environments both in the presence of and without ground-truth reward. This is because the language embedding space guides the discovery towards skills that humans likely deem useful.

Some future work of interest includes testing the use of language with other value learning or policy optimization algorithms, increasing the diversity of captions (by prompting an LLM) to adapt to a wider variety downstream tasks, utilizing multi-modals to effectively learn from both visual and language inputs, fine-tuning the language model on information from survival games like Minecraft (where there is a plethora of online content), and testing skill discovery on robots traversing real-world terrain. Other ways to bring language in the loop with some inductive bias on the task (like language-based goals and instructions) are also promising.

Limitations and Societal Impact. Some limitations of DLSD include the length of training time (as compared to imitation learning approaches) and reliance on a large language model, making discriminator learning slower. Our paper shows that DLSD exhibits more human-aligned skill discovery than DIAYN, which may help in assistive settings. It should be noted though that deploying skill discovery systems in the real world will require guardrails and safety, since unsupervised RL more weakly learns from human intent than explicit language instructions in supervised settings.

References

[1] Dylan Hadfield-Menell, Stuart J. Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

[2] Cassidy Laidlaw, Shivam Singhal, and Anca Dragan. Preventing Reward Hacking with Occupancy Measure Regularization, March 2024.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135. PMLR, July 2017.

[4] Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning, November 2019.

[5] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function, 2018.

[6] Seohong Park, Oleh Rybkin, and Sergey Levine. Metra: Scalable unsupervised rl with metric-aware abstraction, 2024.

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

[8] Seohong Park, Jongwook Choi, Jaekyeom Kim, Honglak Lee, and Gunhee Kim. Lipschitz-constrained unsupervised skill discovery, 2022.

[9] Virginia A Marchman, Elizabeth Bates, Amy Burkardt, and Alice B Good. Children's acquisition of the passive: Revealed preference for the semantic mapping of patient. *Language Acquisition*, 1(1):67–104, 1991.

[10] Kathleen McClure, Julian M Pine, and Elena VM Lieven. Putting object-knowledge to work: Tracking the roles of object representations for language learning. *Language Learning and Development*, 2(3):201–218, 2006.

[11] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation, 2018.

[12] Danijar Hafner. Benchmarking the spectrum of agent capabilities, 2022.

[13] Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation, 2016.

[14] Manuel Lopes, Tobias Lang, Marc Toussaint, and Pierre-yves Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper_files/paper/2012/file/a0a080f42e6f13b3a2df133f073095dd-Paper.pdf`.

[15] Vitchyr H. Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning, 2020.

[16] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning, 2017.

[17] Hao Liu and Pieter Abbeel. Aps: Active pretraining with successor features, 2021.

[18] Hao Liu and Pieter Abbeel. Behavior from the void: Unsupervised active pre-training, 2021.

[19] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration, 2017.

[20] Bradly C. Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models, 2015.

[21] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017.

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

[23] Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery, 2023.

[24] Suvir Mirchandani, Siddharth Karamcheti, and Dorsa Sadigh. Ella: Exploration through learned language abstraction, 2021.

[25] Kevin Black, Mitsuhiko Nakamoto, Pranav Atreya, Homer Walke, Chelsea Finn, Aviral Kumar, and Sergey Levine. Zero-Shot Robotic Manipulation with Pretrained Image-Editing Diffusion Models, October 2023.

[26] Vivek Myers, Andre Wang He, Kuan Fang, Homer Rich Walke, Philippe Hansen-Estruch, Ching-An Cheng, Mihai Jalobeanu, Andrey Kolobov, Anca Dragan, and Sergey Levine. Goal Representations for Instruction Following: A Semi-Supervised Language Interface to Control. In *Proceedings of The 7th Conference on Robot Learning*, pages 3894–3908. PMLR, December 2023.

[27] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding Pretraining in Reinforcement Learning with Large Language Models, February 2023.

[28] Prasoon Goyal, Scott Niekum, and Raymond J. Mooney. Using natural language for reward shaping in reinforcement learning, 2019.

[29] Dipendra Misra, John Langford, and Yoav Artzi. Mapping instructions and visual observations to actions with reinforcement learning, 2017.

[30] Nicholas Waytowich, Sean L. Barton, Vernon Lawhern, and Garrett Warnell. A narration-based reward shaping approach using grounded natural language commands, 2019.

[31] Ademi Adeniji, Amber Xie, Carmelo Sferrazza, Younggyo Seo, Stephen James, and Pieter Abbeel. Language Reward Modulation for Pretraining Reinforcement Learning, August 2023.

[32] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models, 2023.

[33] Jesse Zhang, Jiahui Zhang, Karl Pertsch, Ziyi Liu, Xiang Ren, Minsuk Chang, Shao-Hua Sun, and Joseph J. Lim. Bootstrap Your Own Skills: Learning to Solve New Tasks with Large Language Model Guidance, October 2023.

[34] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[37] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[38] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. Flax: A neural network library and ecosystem for JAX, 2023. URL `http://github.com/google/flax`.

[39] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. The DeepMind JAX Ecosystem, 2020. URL `http://github.com/google-deepmind`.

[40] Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Jackson, Samuel Coward, and Jakob Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning, 2024.

A Captioner Design

We created a hard-coded observation captioner for Crafter with the following format:

```
"""You see {blocks}.

You see {mobs}.

You have in your inventory {items}.

You feel {vitals}.

You are at {health level} health."""
```

The terms "blocks," "mobs," and "items" are comma-separated lists of what blocks, mobs, and items are visible to the player. Note that because Crafter is partially observable, the blocks/mobs listed are those near the agent.

The term "vitals" is populated with "hungry", "thirsty", and/or "tired", if any of these vitals are below 10.

The term "health level" is populated with "low" if the agent's health is less than 5, "moderate" if less than 10 but greater than or equal to 5, and "full" if equal to 10.

If any of the lists above were empty, then we omitted that sentence. For example, if there were no items in the player's inventory, then the term "You have in your inventory {items}." would be omitted from the caption.

After constructing the caption for a specific observation, we pass in the caption to HuggingFace's `sentence-transformers/all-mpnet-base-v2` LLM [34] as our embedding model. This model outputs language embeddings of size 768.

B Experimental Details

We use the JAX framework [37–39], a JAX-based implementation of Crafter [40], and the `sentence-transformers/all-mpnet-base-v2` model from HuggingFace [34] as the observation captioner.

We trained our models on NVIDIA RTX A6000 GPUs within an internal cluster. Each seed took approximately 5-10 hours to train and evaluate.

C Algorithmic Details

DIAYN. We use DQN [35] as our policy learning method, with a Polyak-averaged target network ($\tau = 0.001$) and discount factor $\gamma = 0.99$. All experiments were run for 2M steps (which includes 10K warmup steps where the networks are not trained). Each episode was allowed to run for a maximum of 1K steps. We used ϵ-greedy exploration for all experiments, where ϵ was initialized to 1 and exponentially decayed to 0.1 by 75% of training. The last 25% of training continued with $\epsilon = 0.1$.

We use a symbolic representation of Crafter (provided by the Craftax implementation [40]) in our experiments, which returned observations of size 1345. The observations include 21 one-hot grids of whether a specific mob/item exists at a certain position relative to the user, the item counts in the agent's inventory, player vitals/health levels, direction, light level, and whether the agent is sleeping. We opted for this symbolic representation to focus our experiments on the ability of DIAYN to discern between language embeddings and not worry about learning a good image representation.

The policy (and discriminator in the DIAYN experiments) first separates the observation into the 21 one-hot grids and other metadata. The grids are then passed through the following layers:

- Convolutional layer: 32 output channels, (3, 3) kernel size
- ReLU activation
- Convolutional layer: 64 output channels, (3, 3) kernel size
- ReLU activation

This grid representation is then flattened and concatenated with the other metadata, after which it is fed to downstream layers.

The Q-networks take as input the processed grids/metadata, followed by a dense layer of output size 768, ReLU, dense layer of output size 768, ReLU, and a dense layer of output size 17 (corresponding to the number of actions). The discriminator either takes the language embedding or processed grids/metadata, followed by dense layer of output size 768, ReLU, dense layer of output size 768, ReLU, and a dense layer of output size 5 (corresponding to the number of skills). The discriminator's and policy's learning rate are both $1e-3$, with the models being updated every 20 environment steps. Both are trained on samples from a replay buffer with size 50K and batch size of 512. In our hybrid reward experiments, we use $\lambda = 0.2$.

In order to enjoy the benefits of JAX parallelization, we vectorized our rollouts and model updates to occur in batches of 100. This meant that we took 100 environment steps in parallel and performed several model updates per iteration. Our experiments were trained on a cluster of 8 NVIDIA RTX A6000's.

RND. We set RND's [11] embedding size to 768. The RND network follows the same settings as the discriminator above.

APT. We set APT's [18] embedding size to 768 and $k = 5$ (where k is used in the algorithm's k-nearest neighbors subroutine). The neural encoder is trained by predicting the forward and backward dynamics of Crafter. The forward and backward dynamics model follow the same architecture as the discriminator. Pseudocode is provided in Algorithm 2.

ELLM. We use the default settings used in the ELLM repository [32].

Hyperparameter sweeps. We swept over the following values for DIAYN:

Algorithm 2: APT by Learning Environment Dynamics

```
Initialize policy $\pi$, encoder $E$, forward dynamics model $F$, backward dynamics model $B$, and
  replay buffer $\mathcal{B}$.
Initialize state $s$ and observation $o \sim \Omega(\cdot \mid s)$.
for $t \leftarrow 0$ to num_steps do
    Sample action: $a \sim \pi(\cdot \mid s, z)$.
    Step environment: $s' \sim \mathcal{T}(\cdot \mid s, a)$.
    Get next observation: $o' \sim \Omega(\cdot \mid s')$.
    Add transition to replay buffer: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(o, a, o')\}$.
    if current episode is terminated then
        Reinitialize state $s$ and observation $o$.
    end
    if $t \mod 0 =$ model_update_frequency then
        Sample a batch of transitions $\{(o_i, a_i, o_i')\}_i \sim \mathcal{B}$.
        Compute embeddings of the observations: $E(o_i)$ and $E(o_i')$.
        Forward error := $\|F(o_i, a_i) - E(o_i')\|_2$
        Backward error := $\|B(o_i, o_i') - a_i\|_2$
        Update $\pi$ using DQN [35] with the $k$-nearest neighbors reward detailed in APT [18].
        Update $E, F, B$ using Adam [36] by minimizing forward error + backward error.
    end
end
```

Hyperparameter	**Values**
Policy learning rate	$[0.0001, 0.0005, 0.001, 0.005, 0.01]$
Discriminator learning rate	$[0.0001, 0.0005, 0.001, 0.005, 0.01]$
Policy hidden size	$[768, 1024, 2048]$
Discriminator hidden size	$[768, 1024, 2048]$
Percentage of training to end ϵ-decay	$[25, 50, 75, 90]$
Model update frequency	[1, 5, 10, 20]
Pseudo-reward coefficient λ	[0.01, 0.02, 0.1, 0.2, 1, 2, 10, 20, 100]

D Achievement Success Rates

Figure 7 shows the achievement success rates for DLSD (Hybrid), DLSD, DIAYN, RND, APT, and the DQN. The achievements are topologically sorted based on Crafter's achievement tree, so harder achievements are shown later. DLSD (Hybrid) outperforms even direct training on the ground-truth reward (DQN) on several late-stage achievements, and DLSD consistently outperforms DIAYN and the unsupervised exploration baselines.

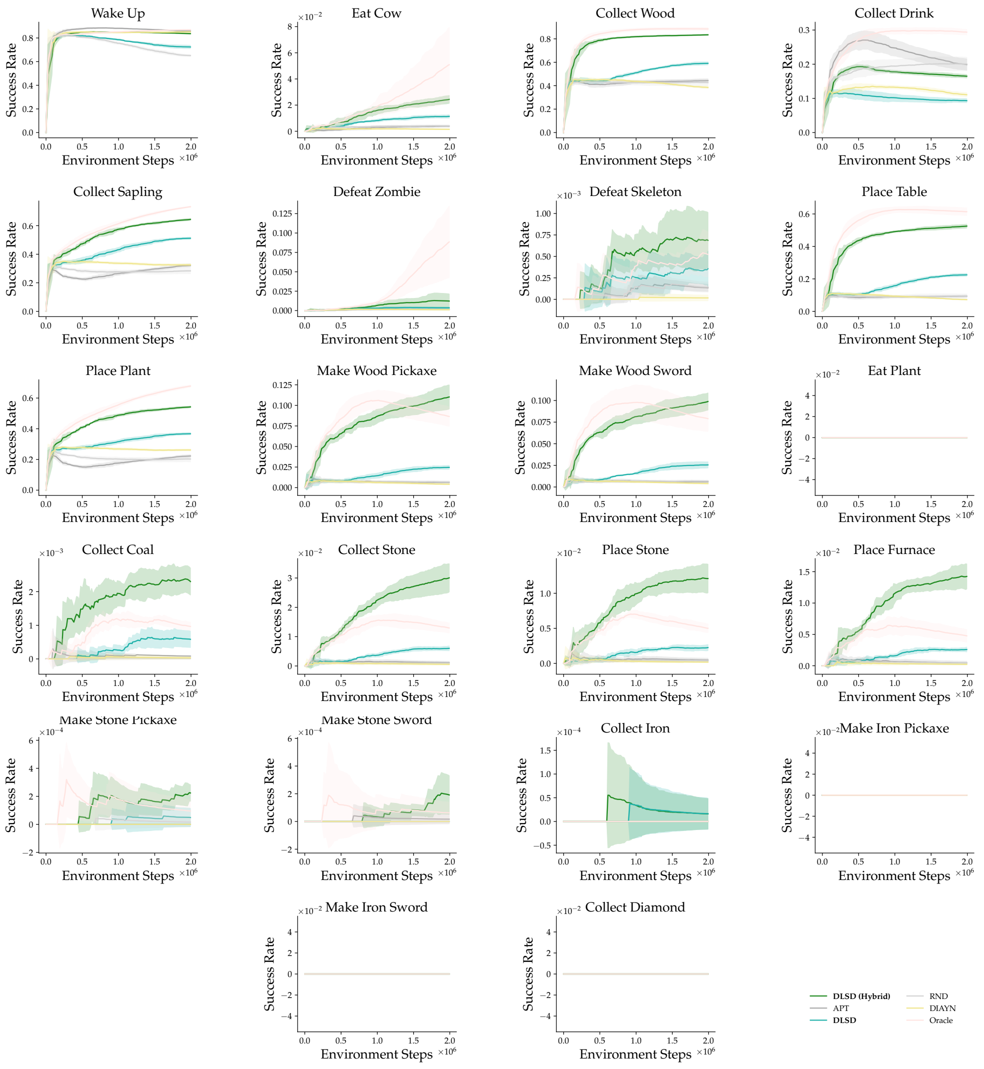


Figure 7: Achievement success rates per method across training.