MINIMUM DESCRIPTION LENGTH SKILLS FOR ACCELERATED REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Humans can quickly learn new tasks by reusing a large number of previously acquired skills. How can we discover such reusable skills for artificial agents when given a large dataset of prior experience? Past works leverage extensive human supervision to define skills or use simple skill heuristics that limit their expressiveness. In contrast, we propose a principled, unsupervised objective for skill discovery from large, offline datasets based on the Minimum Description Length principle: we show that a "code book" of skills that can maximally compress the training data can be reused to efficiently learn new tasks. By minimizing description length we strike an optimal balance between the number of extracted skills and their complexity. We show that our approach outperforms alternative approaches that heuristically define skills on a complex, long-horizon maze navigation task.

1 INTRODUCTION

Humans are able to learn new tasks quickly by leveraging a large repertoire of previously learned skills. Recent work on skill-based reinforcement learning (RL) uses this intuition to accelerate RL by extracting skills from large offline datasets and leveraging them for learning new tasks (Pertsch et al., 2020a; Ajay et al., 2021). One key question is: how do we discover an appropriate set of skills from the data? Some prior work relies on costly human supervision that defines subskills (Schaal, 2006; Lee et al., 2018; Shiarlis et al., 2018), but this is not scalable. Others employ simple heuristics like defining skills as randomly sampled, fixed-length sub-trajectories from the training data (Gupta et al., 2019; Pertsch et al., 2020a; Ajay et al., 2021), thereby limiting the expressiveness of the learned skills.

In this work, we aim to formulate a more principled *unsupervised* objective for learning to discover a useful set of skills. We propose to learn skills such that they most compactly represent solutions to a wide range of tasks. Thus, a policy that leverages these skills to solve a new, related task can learn more efficiently, since its solution can be compactly represented with the learned set of skills.

To learn such skills, we leverage the Minimum Description Length principle (Rissanen, 1978): we propose to learn a "code book" of skills such that we can maximally compress the action trajectories from the large, offline training dataset. We then leverage the extracted skill library for efficient RL on unseen tasks. By minimizing description length we can strike an optimal balance between the number of extracted skills and their complexity.

We first validate that skills which minimize description length lead to more efficient learning of downstream tasks in a simple grid-world environment. We then propose a deep latent variable model for learning to extract minimum description length skills. In experiments on a challenging, long-horizon maze navigation task we show that our approach leads to more efficient learning than alternative approaches that uses heuristically determined skills.

2 RELATED WORK

A large body of prior work leverages pre-trained skills for accelerating the learning of new tasks. Often, approaches require manual definition of skills, e.g. as motion primitives (Schaal, 2006; Pastor

et al., 2009), subskill polies (Oh et al., 2017; Lee et al., 2018; Xu et al., 2018) or task sketches (Andreas et al., 2017; Shiarlis et al., 2018), making them challenging to scale to large skill sets. Others use simple heuristics to extract skills from unstructured data, like defining skills as fixed-length action trajectories (Hausman et al., 2018; Merel et al., 2020; Gupta et al., 2019; Mandlekar et al., 2020; Lynch et al., 2020; Pertsch et al., 2020a; Ajay et al., 2021), limiting their expressiveness. The options framework, in contrast, aims to learn useful skills without skill supervision while learning to solve a task (Sutton et al., 1999; Bacon et al., 2017), but jointly learning skills and policy purely from sparse reward supervision is challenging. Recently, multiple works have explored discovering variable-length skills from offline datasets, but they have focused on modeling high-dimensional observation sequences (Kipf et al., 2019; Kim et al., 2019; Pertsch et al., 2020c;b) or require multi-stage training procedures to stabilize training (Shankar et al., 2019; Shankar & Gupta, 2020), while we propose an end-to-end model for learning skills that can directly be used as behavior primitives by downstream RL agents.

3 Approach

Given a large dataset $\mathcal{D} = \{s_t, a_t, ...\}$ of meaningful agent behaviors, collected via teleoperation or from prior training tasks, we aim to extract a set of reusable *skills*, temporally extended action sequences, that can be leveraged to accelerate learning on new target tasks. We first briefly summarize the minimum description length (MDL) principle, and then describe our approach for learning minimum description length skills.

3.1 Preliminaries: the Minimum Description Length Principle

The minimum description length (MDL) principle (Rissanen, 1978) defines a model that best fits a given data sample x as the model that minimizes the data's description length $L(x)$. Using a two-part code, the "crude" MDL principle (Grunwald, 2004) defines $L(x) = L(\mathcal{M}) + L(x|\mathcal{M})$, i.e. the description length equals the summed description lengths of the model $L(\mathcal{M})$ and of the data using the model $L(x|\mathcal{M})$. Shannon's source coding theorem (Shannon, 1948) states that the MDL of a variable x that is distributed under $p(x)$ equals $\mathbb{E}[-\log p(x)]$. Therefore, when using an encoder $q(z|x)$ to translate the data into an encoded message z and a decoding model $p(x|z)$, the description length of the data is $L(x) = \mathbb{E}_{z\sim q(z|x)}\big[-\log p(x|z) - \log p(z)\big]$, where we treat the message z as part of the model and omit the one-time cost of transmitting the decoder model parameters (Hinton & Zemel, 1994). In their work on *bits-back coding*, Hinton & Zemel (1994) showed that we can use a more sophisticated coding scheme to gain back "free bits" and achieve an even lower description length:

$$L(x) = \mathbb{E}_{z\sim q(z|x)}\big[\underbrace{-\log p(x|z)}_{\text{data reconstruction}} \underbrace{-\log p(z)}_{\text{message length}} + \underbrace{\log q(z|x)}_{\text{"free" bits}}\big]. \tag{1}$$

This is equal to a negated evidence lower bound (ELBO) on x. We can leverage tools from the literature on amortized variational inference (Kingma & Welling, 2014; Rezende et al., 2014) to minimize this bound. Thus, we can *learn* a model $p(x|z)$ that minimizes the description length of the data. We will use this to learn a model of minimum description length skills.

3.2 Minimum Description Length Skills

Our goal is to extract transferrable skills from the large dataset $\mathcal{D}$, which accelerate the learning of new target tasks. Following the MDL principle from Section 3.1 we aim to find a set of skills such that the description length of the action trajectories in the dataset is minimized when encoded with this skill "code book". Put differently, we aim to find a set of skills such that they efficiently represent the solution trajectories to a wide variety of tasks present in the dataset. This will allow a policy which leverages these skills to efficiently find a solution to a new downstream task.[1] Intuitively, minimizing description length allows our approach to strike an optimal balance between the number of extracted skills and their complexity.

[1] We assume that the solution trajectory to the downstream task lies within the distribution of trajectories observed in the training dataset $\mathcal{D}$, i.e. that a code book optimized for $\mathcal{D}$ is also adequate for downstream task solutions. This can be assured by learning from a *rich* training dataset that covers a wide range of behaviors.

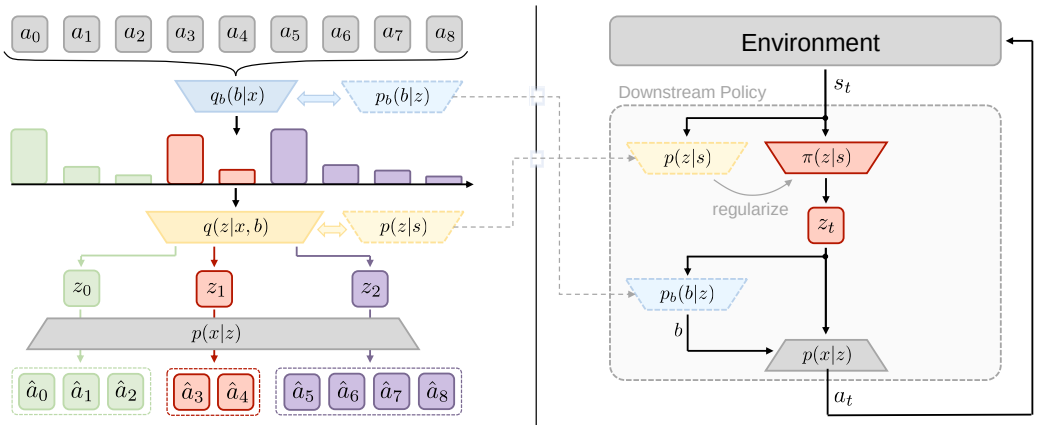

Figure 2: Our model for learning and using MDL skills. **Left**: During model training, the boundary encoder $q_b(b|x)$ predicts a step-wise boundary probability, from which we sample to split the input action sequence into skills. The skill encoder $q(z_n|x, b)$ predicts a latent skill representation z for each skill, which get separately decoded into reconstructed action sequences using the skill decoder $p(x|z_n)$. We further train a skill prior $p(z_n|s)$ and a skill boundary prior $p_b(b_n|z_n)$ for use during downstream learning. **Right**: We efficiently learn a skill policy $\pi(z|s)$ for solving a new task by decoding its outputs into executable actions using skill decoder and boundary prior. Following Pertsch et al. (2020a), we regularize the skill policy with the learned skill prior.

Crucial for the learning efficiency of a downstream task policy is the size of its search space: the smaller the search space, the more efficient the learning. For a policy that uses a vocabulary of learned skills instead of primitive actions, the size of this search space is proportional to $|V|^T$, where $|V|$ is the size of the skill vocabulary and T is the average number of skills per episode. Crucially, both $|V|$ and T are directly related to the *message length* in the two-part code formulation of MDL introduced in Section 3.1: the larger the number of skills $|V|$ the more bits are required to encode any single skill; the more skills per episode T the more symbols need to be encoded per message. Finding skills that minimize the description length of the data strikes a balance between the total number of skills and the average number of skills per message, i.e. the skill complexity, and thus reduces the search space for downstream task policies, making learning more efficient.

We experimentally validate this intuition in a simple grid-world environment (see Figure 1). We randomly sample sets of skills in the form of variable-length movement primitives, compute the description lengths of a large dataset of goal-reaching trajectories using these skill code books and then measure the efficiency of learning to solve an unseen goal-reaching task, using both a search-based optimizer (A*) and RL-based policy optimization. We find that skill sets which result in shorter data description lengths, lead to more efficient learning of unseen tasks across both optimizers, supporting our previous intuition. Next, we will describe how to *learn* MDL skill code books.

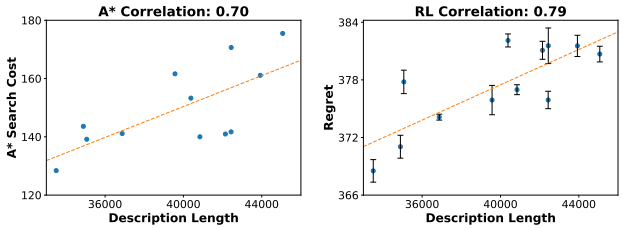

Figure 1: Description length vs. efficiency of downstream task learning using A* (**left**, measuring number of expanded nodes) and RL (**right**, measuring regret) on the grid world environment. The results show that skills which lead to smaller description length result in increased downstream learning efficiency. For qualitative results, see Figure 4.

3.3 Learning Minimum Description Length Skills

Our goal is to learn a set of skills such that we can represent the training action trajectories $x = a_1, \ldots, a_H$ for a large H efficiently. In Section 3.1 we summarized how the ELBO objective can be used to learn a MDL representation of the training data with an encoder $q(z|x)$ and a decoder $p(x|z)$. Yet, this formulation assumed that the encoder compresses the *whole* input data sequence x into *a single* message symbol z which gets transmitted and decoded. However, we aim to encode

trajectories from the training dataset into *sequences* of skills $z_{1:N}$. Thus, we need to introduce an additional component to our model: a boundary encoder $q_b(b|x)$ that determines how to split the input sequence into variable-length skills. An overview of our model is depicted in Figure 2 (left).

We can formulate the ELBO for our *factorized* skill learning model as:

$$\mathbb{E}_{z\sim q(z|x), b\sim q_b(b|x)} \sum_{n=1}^{N} \log p(x_{t_n:t_{n+1}}|z_n) - \beta D_{\mathrm{KL}}\big(q(z_n|x,b), p(z_n)\big) - \beta_b D_{\mathrm{KL}}\big(q_b(b_n|x), p_b(b_n)\big) \tag{2}$$

Here, $p(z_n)$ and $p_b(b_n)$ are unit Gaussian and uniform priors respectively. D_{KL} denotes the Kullback-Leibler divergence and the coefficients β, β_b are introduced to weight the KL regularization terms, as is common in the literature on amortized variational inference (Higgins et al., 2017).

In practice, the boundary encoder outputs a per-timestep discrete variable which indicates whether this timestep is the beginning of a new skill. We instantiate all components of our model with neural networks and optimize the objective in Equation 2 with gradient descent, using Gumbel softmax reparametrization (Jang et al., 2017; Maddison et al., 2017) and the straight-through gradient estimator (Bengio et al., 2013) for the discrete boundary variable b. For further implementation details, see Section A.2.

Leveraging MDL Skills for Downstream Learning. To leverage the learned skills during learning of a new task, we train a policy $\pi(z|s)$ which takes in the current state and outputs a skill z (see Figure 2, right). To decode the skill, we additionally learn a boundary decoder $p_b(b|z)$ which is trained to match the boundary encoder's output. Given its predicted skill length, we roll out the skill decoder $q(x|z)$ for the predicted number of steps and execute the decoded actions. Following SPiRL (Pertsch et al., 2020a) we train a skill prior $p(z|s)$ jointly with our model and regularize the policy towards it during downstream task learning to guide exploration. For policy optimization we use the SPiRL algorithm, which is a modified version of Soft Actor-Critic (Haarnoja et al., 2018a;b).

4 Experiments

We evaluate our approach in a 2D continuous control maze navigation environment based on Fu et al. (2020) (see Figure 3, left), in which the agent is controlled with planar velocity commands to reach a fixed goal location. Learning this task is challenging since the agent needs to execute hundreds of actions before reaching the goal, at which it only receives a sparse binary reward signal. We collect a dataset of $10,000$ trajectories between random start-goal pairs for training of our MDL skills and then train a policy that leverages the skills to navigate to an *unseen* goal. Analogous to Section 3.2, we compare our approach to an agent that uses a random sampling heuristic for discovering skill code books without optimizing the skill's description length, as well as a state-of-the-art model-free RL agent (SAC, (Haarnoja et al., 2018a)) that does not leverage pre-trained skills.

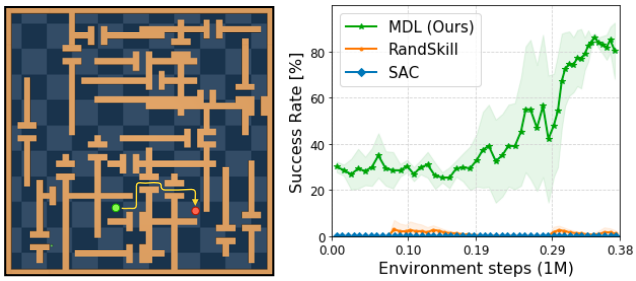

Figure 3: **Left**: Maze navigation task: the agent needs to navigate from the green start to the red goal position and only receives reward upon reaching the goal. **Right**: Performance on the maze navigation task: only the MDL skill model is able to learn the task, showing the benefits of MDL skills for effective transfer to new tasks.

Our results are summarized in Figure 3 (right): we find that the agent leveraging MDL skills can effectively learn the challenging, long-horizon task while the agent that does not optimize the description length of its skills fails to make learning progress. The visualization of the rollouts during training in Figure 5 shows, that only the MDL skills lead to effective exploration of the maze environment, supporting our intuition from Section 3.2. The SAC agent completely fails at the task since it is unable to leverage pre-trained skills to effectively explore the maze and receive the sparse reward signal, underlining the challenge the tested task presents for current RL algorithms.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. {OPAL}: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN. 1, 2

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175. PMLR, 2017. 2

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017. 2

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. 4

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018. 7

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. 4

Peter Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004. 2

Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *CoRL*, 2019. 1, 2

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *ICML*, 2018a. 4

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*, 2018b. 4

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *ICLR*, 2018. 2

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 4

Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pp. 3–10, 1994. 2

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 7

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 4

Taesup Kim, Sungjin Ahn, and Yoshua Bengio. Variational temporal abstraction. In *Advances in Neural Information Processing Systems 32*, pp. 11566–11575. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9332-variational-temporal-abstraction.pdf. 2

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014. 2

Thomas Kipf, Yujia Li, Hanjun Dai, Vinicius Zambaldi, Edward Grefenstette, Pushmeet Kohli, and Peter Battaglia. Compositional imitation learning: Explaining and executing one task at a time. *ICML*, 2019. 2

Youngwoon Lee, Shao-Hua Sun, Sriram Somasundaram, Edward S Hu, and Joseph J Lim. Composing complex skills by learning transition policies. In *International Conference on Learning Representations*, 2018. 1, 2

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *CoRL*, 2020. 2

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017. 4

Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *ICRA*, 2020. 2

Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: Reusable neural controllers for vision-guided whole-body tasks. *ACM. Trans. Graph.*, 2020. 2

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *International Conference on Machine Learning*, pp. 2661–2670. PMLR, 2017. 2

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768. IEEE, 2009. 1

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning (CoRL)*, 2020a. 1, 2, 3, 4

Karl Pertsch, Oleh Rybkin, Frederik Ebert, Chelsea Finn, Dinesh Jayaraman, and Sergey Levine. Long-horizon visual planning with goal-conditioned hierarchical predictors. *arXiv preprint arXiv:2006.13205*, 2020b. 2

Karl Pertsch, Oleh Rybkin, Jingyun Yang, Shenghao Zhou, Kosta Derpanis, Joseph Lim, Kostas Daniilidis, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. *L4DC*, 2020c. 2

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014. 2

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978. 1, 2

Stefan Schaal. *Dynamic Movement Primitives - A Framework for Motor Control in Humans and Humanoid Robotics*. Springer Tokyo, 2006. 1

Tanmay Shankar and Abhinav Gupta. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020. 2

Tanmay Shankar, Shubham Tulsiani, Lerrel Pinto, and Abhinav Gupta. Discovering motor programs by recomposing demonstrations. In *ICLR*, 2019. 2

Claude E Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 2

Kyriacos Shiarlis, Markus Wulfmeier, Sasha Salter, Shimon Whiteson, and Ingmar Posner. Taco: Learning task decomposition via temporal alignment for control. *ICML*, 2018. 1, 2

Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999. 2

Danfei Xu, Suraj Nair, Yuke Zhu, Julian Gao, Animesh Garg, Li Fei-Fei, and Silvio Savarese. Neural task programming: Learning to generalize across hierarchical tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3795–3802. IEEE, 2018. 2

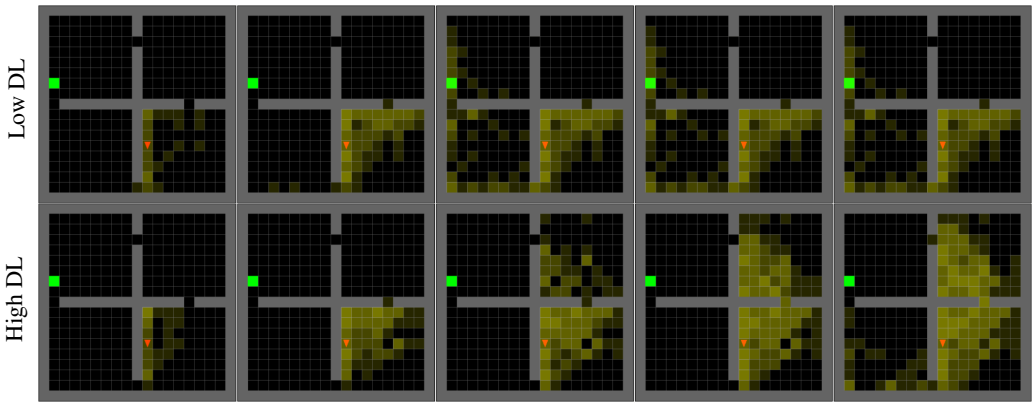

Figure 4: A* search state visitation heatmaps, taken at equal intervals during A* search, of two codebooks. The low description length (DL) codebook (top) allows A* to reach the goal much faster, finishing the search by the third picture. The high DL codebook (bottom) results in A* spending more time in each room.

A Appendix

A.1 Four Room Experiment Details

The 4-room environment (Chevalier-Boisvert et al., 2018) is a grid-world goal-reaching environment with four rooms connected by randomly placed doors. We sample initial agent and goal positions in the diagonal rooms (the most difficult configurations) for testing and all other position combinations for training. To build codebooks, we first sample a large number (2000) of training environment setups and collect trajectories by running A*. The heuristic we use for A* is the sum of absolute horizontal and vertical distances plus the number of turns required to reach the goal. We then randomly dissect each trajectory into skill segments. We assign a different skill length range for each codebook such that its DL is distinct enough from others. These length ranges have different variances, but always the same mean (5). For example, there is a codebook with 2, 6, 7-length skills and another one with 4, 5, 6-length skills. For evaluation, we only use the first 15 most frequent skills from each codebook.

The first downstream task is running the same A* solver but enables it with skills from the codebook. We define the search cost as the total number of visited nodes and compute the average search cost over a large number of runs (500). We are able to get a 0.70 correlation between DL and A* search cost, see Figure 1. This means codebooks with smaller DL have lower average cost on A*. The difference is more apparent when looking at visitation heatmaps over time, see Figure 4.

The second downstream task is training a standard DQN agent with the codebooks. The 4-room environment reward is defined as `1-0.9*(step count/max steps)`. Our network hidden size is 128. We use an Adam optimizer with a learning rate of 3e-5. We have a discount factor of 0.99. We also employ an epsilon greedy exploration with a 0.1 epsilon. Our replay buffer size is 1e6. For each codebook, we train DQN with 8 random seeds. To evaluate the performance of a codebook, we measure the average regret over all seeds and study its standard error. We find a 0.79 correlation between DL and DQN regret, which confirms our theory that smaller description length helps downstream learning efficiency.

A.2 Skill Model Implementation Details

Our MDL skill model consists of 5 major networks: a prior boundary detector $p_b(b|z)$, a posterior boundary detector $q_b(b|x)$, a skill prior predictor $p(z|s)$, a skill posterior encoder $q(z|x, b)$, and a decoder $p(x|z)$. The posterior boundary detector $q_b(b|x)$ is a bidirectional LSTM (Hochreiter & Schmidhuber, 1997) that predicts skill segmentations based on the input action trajectories x. It predicts logits for every time step, from which we then sample a single discrete posterior boundary length for the entire skill (casting the logits as those of a Categorical distribution). The posterior

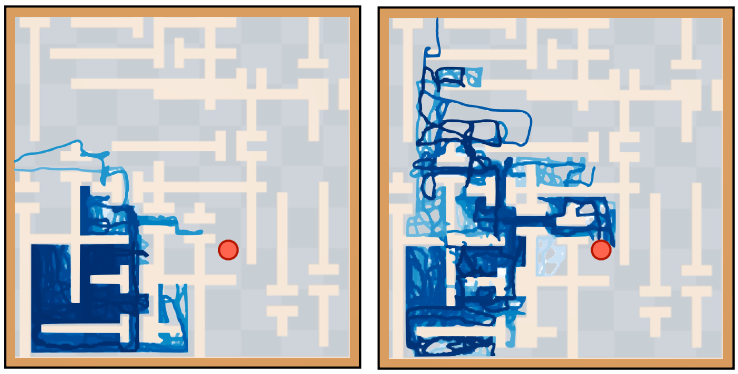

Figure 5: Rollouts using randomly sampled skill code book (**left**) and MDL skills (**right**) in the maze environment. The goal of the target task is indicated in red. Only the MDL skills allow for more effective exploration of the maze and thereby allow the agent to learn to solve new tasks.

boundary network marks timesteps in which new skills start with a 1, while all other steps are marked with a 0. The resulting output of the boundary detector is a (batched) array of 1's and 0's at each timestep, with the same length as the input sequence length. The skill posterior encoder $q(z|x, b)$ is an LSTM that scans the input action sequence and outputs the means and log-variances of a posterior skill (Gaussian) distribution for each skill segmentation as instructed by the posterior boundary indicator (the input is the concatenation of input action trajectories x and the boundary posterior's outputs). The skill prior predictor $p(z|s)$ is a multi-layer perceptron (MLP) that takes in the current state and predicts a prior skill distribution for every timestep. We then sample the latent skill z for each skill using either the posterior skill distribution. The decoder reconstructs actions from each encoded skill z, and its recurrent hidden state is masked out by the output of the posterior boundary network (so that whenever a new skill starts the decoder's hidden state is reset). The prior boundary detector $p_b(b|z)$ is an MLP takes the skill z as input and generates logits for each timestep which are formed into a categorical distribution.

During RL training, a higher level policy outputs latent z values which are decoded by the skill decoder for the amount of timesteps as indicated by sampling the prior boundary detector with z as its input.

For training loss, we compute the discrete KL divergence between the posterior boundary and a fixed prior boundary, which we select to be a Gamma distribution with rate 10 and concentration 0.5. We also compute the discrete KL divergence between the prior boundary and the detached posterior boundary samples, masked by the output of the posterior boundary detector as the prior boundary detector's predictions are only used at the start of each skill during RL training. Similarly, we compute the KL divergence between the skill posterior and a fixed skill prior (unit Gaussian), and the KL divergence between the detached skill posterior's outputs z, and the learned skill prior's predictions of z. Action reconstruction loss is computed as the mean squared error between the output action trajectories produced by the decoder and the ground truth input actions.