LEARNING ROBUST TASK CONTEXT WITH HYPOTHETICAL ANALOGY-MAKING

Shinyoung Joo[1], **Sang Wan Lee**[1,2]
[1] Department of Bio and Brain Engineering
[2] Center for Neuroscience-inspired AI
Korea Advanced Institute of Science and Technology (KAIST)
{sineong.joo,sangwan}@kaist.ac.kr

ABSTRACT

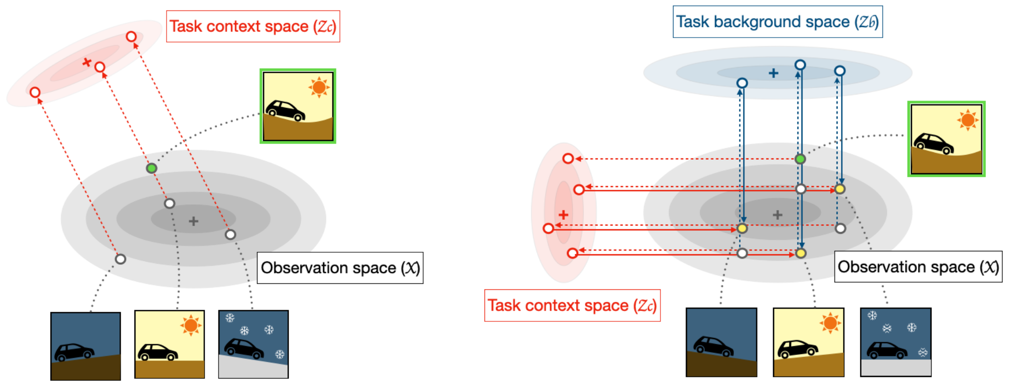
Learning compact state representations from high dimensional and noisy observations is the cornerstone of reinforcement learning (RL). However, these representations are often biased toward the current task context and overfitted to context-irrelevant features, making it hard to generalize to other tasks. Inspired by the human analogy-making process, we propose a novel representation learning framework called Hypothetical Analogy-Making (HAM) for learning robust task contexts and generalizable policy for RL. It consists of task context and background encoding, hypothetical observation generation, and analogy-making between the original and hypothetical observations. Our model introduces an auxiliary objective that maximizes the mutual information between the generated observation and existing labels of codes used to generate the observation. Experiments on various challenging RL environments showed that our model helps the RL agent's learned policy generalize by revealing a robust task context space.

1 INTRODUCTION

Learning policies directly from observations is a gateway to successful real-life applications such as auto-driving and robotics. However, current deep reinforcement learning (RL) algorithms trained from raw pixels have several issues undermining their applicability to real-world problems. First, they are vulnerable to common noises such as background changes (Gamrian & Goldberg, 2019; Zhang et al., 2021). They also often fail to adapt to small semantic changes, e.g., the height of the platform or the position of stars in the Climber environment in the OpenAI ProcGen benchmark suite (Cobbe et al., 2020). Both are because their representations are strongly biased toward the current task context and fail to learn the robust task context space (see Figure 1(a)). In fact, there exists a substantial discrepancy between the train and test performance of the vanilla PPO agent (Cobbe et al., 2020). Although a large amount of training data ($\geq$ 10k samples) could reduce the generalization gap (Cobbe et al., 2019), many practical applications cannot meet this condition for various reasons, such as cost and safety issues for data acquisition (Levine et al., 2020).

Unlike machine learning algorithms, humans have an outstanding ability to learn with a limited amount of experience (Lake et al., 2016; Kaiser et al., 2020). This ability is often ascribed to abstraction and analogy-making. Analogy-making is a central mechanism for revealing meaning from perception (Anderson, 1980; Bartha, 2019). We learn compact abstractions from noisy perception through numerous comparisons with other perceptions or imaginations by making analogies (Lakoff, 2009; Mitchell, 2021). In this paper, we hypothesize that applying the analogy-making process to learning task context space can help RL models learn generalizable policy. Figure 1(b) shows the example case when analogy-making can find proper task context space to improve generalization.

Inspired by human analogy-making, we propose a novel representation learning framework for learning robust task context in RL. To establish broadly generalizable policies in RL, we aim to separate the **task context** from **task background** by performing analogy-making between experienced and hypothetical observations. Our framework called Hypothetical Analogy-Making (HAM) consists of three phases: task context and task background encoding, hybrid observation generation, and analogy-making with discriminators. (i) It decomposes an observation into two independent


Task context space ($\mathcal{Z}_c$)
Observation space ($\mathcal{X}$)
Task background space ($\mathcal{Z}_b$)
Task context space ($\mathcal{Z}_c$)
Observation space ($\mathcal{X}$)

(a) Task background correlated task context space (b) Robust task context space with analogy-making

Figure 1: The analogy-making process can help learn the proper task context space: when the training dataset is small, as in (a), the learned task context space is easily biased toward task background space. In (a), there are three training data points (white-colored), and we try to encode task context that determines the control of the vehicle. We can find an undesirable correlation between learned task context space and the task background feature, weather, resulting in the wrong context encoding of the unseen observation (green-colored). By adopting the analogy-making process, we can relieve this problem. By generating hypothetical observations (yellow-colored) using learned task context and task background space and making analogies with the original observations (white-colored) such as ”Do A and B share the same control of the vehicle?” or ”Are A and B in the same weather condition?”, we can find proper task context space and task background space and we can map the unseen observation correctly as in (b).

components: a task context code and a task background code. (ii) By combining the codes from different tasks, it generates a hybrid observation, a hypothetical but realistic observation. In doing so, (iii) it maximizes the mutual information between the hypothetical observation and the original labels, enforcing the context-related and background-related information being embedded in each code. Our model learns robust task context across varying task background features by making analogies between the original and hypothetical observations generated with different task background codes.

The key contributions of our work are as follows:

- Motivated by human analogy-making, we propose a novel representation learning framework for learning robust task context in RL. For this, we combine context and background relevant encoding, hypothetical observation generation, and analogy-making between the original and hypothetical observations.
- Our model introduces an auxiliary objective that maximizes the mutual information between the generated image and the existing labels of codes used to generate the image. Using an information-theoretic approach, we show that human analogy-making process can be implemented with a simple encoder-decoder-discriminator architecture.
- We conducted several experiments in Jumping Task (des Combes et al., 2018) as a proof of concept, where the observation space has relatively simple generative factors. Furthermore, we empirically show that our model outperforms prior state-of-the-art baselines on several environments in the OpenAI ProcGen benchmark suite (Cobbe et al., 2020), which was intended to evaluate the generalization performance of RL algorithms.
- We show that our model successfully reveals a task context space which is invariant to task background changes using both quantitative metrics and qualitative visual results. The learned RL policy built on this robust task context space can generalize to the unseen observations.

2 Problem formulation

2.1 Preliminaries

We use a Markov Decision Process (MDP) setting, in which the MDP denoted as $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$ with a state space $\mathcal{X}$, an action space $\mathcal{A}$, a reward function R, transition dynamics P, and a discount factor $\gamma \in [0, 1)$. A policy $\pi(\cdot|x)$ represents a probability distribution over actions given state x. In RL, the basic goal is learning an optimal policy that maximizes the expected cumulative discounted rewards $\mathbb{E}_{a_t \sim \pi(\cdot|x_t)}[\sum_t \gamma^t R(x_t, a_t)]$ starting from an initial state x_0. The ultimate goal of RL is to learn a policy that generalizes across environments that share similar structure.

To formalize this, we consider a distribution of tasks, each defined as an MDP, $\mathcal{M}^i \in \mathcal{M}$ where $i \in I$ and $|I|$ defines the size of the task distribution. Those MDPs share an action space $\mathcal{A}$, a reward function R, and transition dynamics P but are with disjoint state spaces, $\mathcal{X}^i \cap \mathcal{X}^j = \emptyset$. For instance, different MDPs correspond to different task levels in the same environment in the OpenAI ProcGen benchmark suite (Cobbe et al., 2020) (see Appendix C.2.1). We define an union state space $\mathcal{S}$ of all possible state spaces, where $\mathcal{S} = \bigcup_{i \in I} \mathcal{X}^i$. We assume the RL agent has access to a collection of training MDPs $\{\mathcal{M}^i\}_{i=1}^N$ and the index i of each. After training, the RL agent applies its policy π over the entire state space $\mathcal{S}$ including unseen MDPs. Note that we evaluate the learned policy's zero-shot performance without any meta-learning phase.

2.2 Broadly generalizable policy

Intuitively, the agent should have as compact state representations as possible to make optimal choices. From this intuition, we define *key generative factors of state space* and formalize the goal of *broadly generalizable policy*. The formulation uses the notion of entropy H and mutual information $\mathcal{I}$.

Definition 2.1 (Key generative factors of state-space). *For any $x \in \mathcal{X}^i$, the latent feature group z_c^* and z_b^* are said to be key generative factors of a state space $\mathcal{X}^i$ when*

$$H(x) = \mathcal{I}(c, x) + H(x|c) = \mathcal{I}(c, x) + \mathcal{I}(b, x) = H(z_c^*) + H(z_b^*) \tag{1}$$

, where c and b indicate labels for task context (e.g. optimal action) and background (e.g. task id) respectively. Then we can define an ideal encoder E^*, which transforms x onto information relevant to task context and background, z_c^* and z_b^*, respectively. We can also define an ideal generator G^* that generates observation $G^*([z_c^*, z_b^*]) \in \mathcal{X}^i$.

Definition 2.2 (Broadly generalizable policy). *The goal of learning representations for broadly generalizable policy is to glean task context information from background. This is formulated as maximizing $\mathcal{I}(c, z)$ while minimizing $\mathcal{I}(b, z)$, so that a policy built upon the representation z becomes robust against task-irrelevant changes (i.e., changes in the task background space). The optimal solution of z is z_c^* where $[z_c^*, z_b^*]$ is the key generative factors of the state space $\mathcal{X}^i$.*

Here we claim that an RL agent can achieve the *broadly generalizable policy* by using our **hypothetical analogy-making module**, which (i) makes imaginary hypothetical observations with different combinations of the generative factors of inputs and (ii) performs analogy-making to achieve an optimal behavioral policy. Details are provided in the next section.

3 Hypothetical Analogy-Making

We implemented the Hypothetical Analogy-Making (HAM) process with mutual information(MI) regularized GAN structure. In addition to the original GAN objective aimed at making images as realistic as possible, we train our encoder E, generator G, and discriminator D to maintain a large amount of MI between the hypothetical images $G([z_c^i, z_b^j])$ and the original labels: task context label c and background label b.

The MI regularized GAN objective is as follows. For any $x^i \sim \mathcal{X}^i$ and $x^j \sim \mathcal{X}^j$,

$$\min_{E,G} \max_{D} V_{\mathcal{I}}(D, G, E) = V(D, G, E) - \lambda_1 \mathcal{I}(c^i, G([z_c^i, z_b^j])) - \lambda_2 \mathcal{I}(b^j, G([z_c^i, z_b^j])) \tag{2}$$

,where $[z_c^i, z_b^i] = E(x^i)$, $[z_c^j, z_b^j] = E(x^j)$, and $V(D, G, E)$ is an encoder-added version of the original GAN loss (Chen et al., 2016), which is $\mathbb{E}_{x \sim P_{data}}[logD(x)] + \mathbb{E}_{x \sim P_{data}}[log(1 - D(G(E(x))))]$.

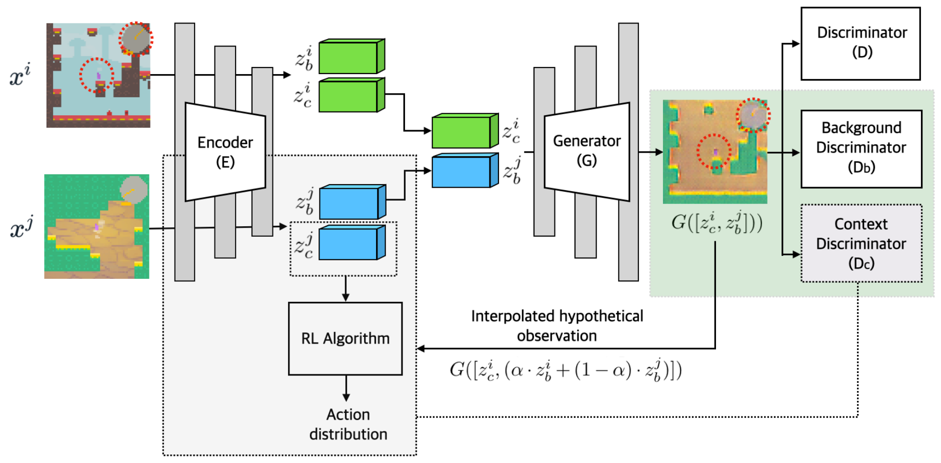

x^i
x^j
Encoder (E)
z_b^i
z_c^i
z_b^j
z_c^j
z_c^i
z_b^j
Generator (G)
$G([z_c^i, z_b^j])$
Discriminator (D)
Background Discriminator (Db)
Context Discriminator (Dc)
RL Algorithm
Action distribution
Interpolated hypothetical observation
$G([z_c^i, (\alpha \cdot z_b^i + (1-\alpha) \cdot z_b^j)])$

Figure 2: Architecture of Hypothetical Analogy Making model

The encoder part of the above objective corresponds to the first step of HAM: decomposing the original observation into the task context-relevant part and the others. It is followed by generating the hypothetical observation $G([z_c^i, z_b^j])$ as a second step: imagining a hypothetical situation by replacing the background part with what we have experienced before while maintaining the original context. The MI terms, $\mathcal{I}(c^i, G([z_c^i, z_b^j]))$ and $\mathcal{I}(b^j, G([z_c^i, z_b^j]))$, correspond to the last step: keeping the context and background information the same with the original observations where each code came from. We hypothesize that the RL agent mimicking the human analogy-making process with hypothetical observations can achieve a *broadly generalizable policy*.

Suppose we generated a new image with task context code z_c^i from an image x^i and the task background code z_b^j from another image x^j. We claim that by maximizing the mutual information between the generated image and the task context label from which its task context code came, $\mathcal{I}(c^i, G([z_c^i, z_b^j]))$, and the task background label of which its task background code came from, $\mathcal{I}(b^j, G([z_c^i, z_b^j]))$ with regard to the encoder E and generator G, we can find the optimal solution z_c^* by maximizing the lower bound of $\mathcal{I}(c, z_c)$ while minimizing $\mathcal{I}(b, z_c)$ (see Theorem 3.1).

Theorem 3.1. *The problem of maximizing the mutual information between the generated image and existing labels of codes used to generate the image, $\mathcal{I}(c^i, G([z_c^i, z_b^j])) + \mathcal{I}(b^j, G([z_c^i, z_b^j]))$, is equivalent to the problem of maximizing the lower bound of $\mathcal{I}(c, z_c)$ while minimizing $\mathcal{I}(b, z_c)$.*

Corollary 3.1.1. *In Theorem 3.1, the lower bound is tight when z_c and z_b are mutually independent.*

With Theorem 3.1 and Corollary 3.1.1, by maximizing the MI term $\mathcal{I}(c^i, G([z_c^i, z_b^j])) + \mathcal{I}(b^j, G([z_c^i, z_b^j]))$ and making z_c and z_b be mutually independent, we can have *broadly generalizable policy*. However, it is impossible to calculate the MI term $\mathcal{I}(a|x)$ directly because it requires the posterior $P(a|x)$. To circumvent this issue, we use a variational lower bound of the MI term by replacing $P(a|x)$ with an accessible distribution $Q(a|x)$. Finally, our original objective can be recast as a problem of minimizing a simple classification loss (see Corollary 3.1.2). Proof for Theorem 3.1, Corollary 3.1.1 and Corollary 3.1.2 is in Appendix B.

Corollary 3.1.2. *Let a be any label for the generated image x. The mutual information between the label a and the generated image x, $\mathcal{I}(a, x)$, can be maximized by minimizing the classification loss of the auxiliary classifier $Q(a|x; \theta)$.*

3.1 Model Architecture

The proposed HAM architecture is illustrated in Figure 2. All the components are trained jointly during the RL policy learning process. The following sections describe each component.

Image Generation with Disentangled Features. Given an input pair $(x^i, x^j) \sim (\mathcal{X}^i, \mathcal{X}^j)$, we apply our encoder E to each input. Then the encoder outputs task context code and background code for each input, $E(x^i) = [z_c^i, z_b^i]$ and $E(x^j) = [z_c^j, z_b^j]$, respectively. Then we apply the generator G to the hybrid pair $[z_c^i, z_b^j]$. The generated image $G([z_c^i, z_b^j])$ is fed into the basic discriminator D, which evaluates how realistic the generated image is. Furthermore, we train our encoder, generator, and discriminator to generate a realistic image while learning independent features that can be combined to create a new hybrid image. The loss $J_{GAN,hybrid}(E, G, D)$ is calculated as follows using the non-saturating adversarial loss (Mirza & Osindero, 2014):

$$\mathbb{E}_{x^i \sim \mathcal{X}^i, x^j \sim \mathcal{X}^j, x^i \neq x^j}[-log(D(G([z_c^i, z_b^j])))]. \tag{3}$$

The detailed structures of our encoder, generator and discriminator are provided in Appendix C.1.2 and Appendix C.2.2.

Analogy Making. The green shaded area of Figure 2 makes predictions about the hypothetical observations and uses the labels of the original images for the prediction loss. Note that this is the core component of our Hypothetical analogy-making process. The generated images $G([z_c^i, z_b^j]))$ are fed into the three modules: basic discriminator D, task background discriminator D_b, and task context discriminator D_c. A task background discriminator D_b outputs how realistic the given generated image is compared to the reference image x^j (the generated image got its task background code from x^j). The loss is given by:

$$J_{background}(E, G, D_b) = \mathbb{E}_{x^i \sim \mathcal{X}^i, x^j \sim \mathcal{X}^j}[-log(D_b(\texttt{crop}(G([z_c^i, z_b^j]))), \texttt{crops}(x^j))], \tag{4}$$

where $[z_b^j, z_c^j] = E(x^j)$ and `crop` randomly selects a fixed-sized patch of the full image and `crops` is a collection of multiple patches. Note that in experiments with Jumping task (des Combes et al., 2018), we use only the full image as an input of D_b without making patches because the task background features of Jumping task (e.g., the height of the floor and position of the obstacle) are lost in the cropping process.

The task context discriminator D_c outputs whether the hybrid image shares the same action context with the original image x^i (the generated image got its task context code from x^i) or not. In experiments with ProcGen benchmark (Cobbe et al., 2020), we utilize the action output of the RL part's policy network Q_π without using a separate action discriminator (gray shaded area in Figure 2). We took the action output of the policy network and conducted the learning in the direction of minimizing Jensen-Shannon (JS) divergence between the action probabilities of the original and hypothetical observations. The loss is given by:

$$J_{context}(E, G, Q_\pi) = \mathbb{E}_{x^i \sim \mathcal{X}^i, x^j \sim \mathcal{X}^j}[D_{JS}(Q_\pi(G([z_c^i, z_b^j])) \| Q_\pi(x^i)))]. \tag{5}$$

By using the above losses, our encoder E and generator G learn to generate a hybrid image, which contains the original task context of x^i and the task background of x^j. The task context of x^i is highlighted with red dotted circles in Figure 2.

Policy Learning. The RL agent learns its policy with the task context code z_c. In experiments with the ProcGen benchmark, the RL agent also utilizes hybrid images to facilitate learning task background-invariant context features. In ProcGen, we use a standard Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm for training the RL agent; note that our framework can exploit any model-free RL algorithm. PPO algorithm utilizes action-advantages $A_t = A^\pi(a_t, s_t) = Q^\pi(a_t, s_t) - V^\pi(s_t)$, and minimizes a clipped probability loss as follows:

$$\mathcal{J}_\pi(\theta) = -\mathbb{E}_{\tau \sim \pi}[\min(\rho_t(\theta)A_t, clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]. \tag{6}$$

We use hypothetical observations when calculating the clipped-ratio $\rho_t(\theta)$ over the recent experience collected with $\pi_{\theta_{old}}$ and updating a state-value estimator $V_\phi(s)$. We replace a certain ratio $\lambda_{mix} \in (0, 1]$ of the original observation s with hypothetical observation generated by performing background interpolation $G([z_c^i, (\alpha \cdot z_b^i + (1-\alpha) \cdot z_b^j)])$ using a random rate $\alpha \in [0, 1)$. See Appendix E.2.2 to check the quality of interpolated hypothetical observations.

4 Experimental Results

We ran simulations to show that our model improves generalization performance in challenging scenarios by separating task context-relevant visual features from task background-relevant ones. As a proof of concept, we conducted several experiments, including an ablation study in the Jumping task (des Combes et al., 2018), where the observation space has relatively simple generative factors. Then we conducted evaluations in more challenging tasks in ProcGen benchmark suite (Cobbe et al., 2020), where context-related visual features are intertwined with background-relevant ones. We carried out evaluations in three different ways: (i) generalization performance with a small size of the training dataset, (ii) qualitative results of generated hypothetical observations showing the separation of GAN features into the task context and background-related group, and (iii) further evaluation to show the robustness of learned task context space (in Appendix D).

4.1 Jumping task

In the Jumping task, a white agent has to jump over a gray obstacle without touching it. The evaluation measures test performance for 286 tasks of 26 different obstacle locations and 11 floor heights while the training proceeds with only 18 of them. Depending on the distribution of training samples on the evaluation grid, the training type is classified into Wide, Narrow, and Random. Each grid measures different types of generalization and see Appendix C.1.1 for the details. Experiments in the Jumping task are intended to confirm whether our model can separate relatively simple task context-relevant visual features from task background-relevant ones. In the Jumping task, the **distance between the agent and the obstacle** is the task context-relevant visual feature. The height of the floor and the location of the obstacle are the task background-relevant visual features.

4.1.1 Generalization test

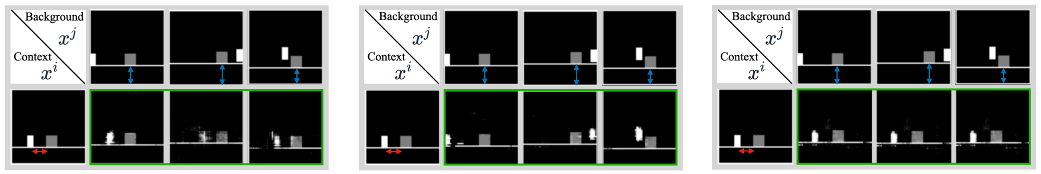
The following are the generalization results compared with other baseline methods and ablated models. In Table 1, we present the ratio of how many of the total 286 tasks the learned policy succeeded. We found that our model shows the highest performance in Wide and Narrow grid configurations, notably with highly significant performance improvement in the Narrow training setting. Results in Random show relatively low performance, presumably the case where our model has difficulty finding the visual feature that determines whether the two images share the same action context or not in input data with irregular intervals. The lower performance of ablated models implies that the two losses; context discriminator loss and background discriminator loss have synergistic effects. Generated hypothetical images of each ablated model are in Section 4.1.1.

Method	Success ratio (%)		
	Wide	Narrow	Random
L_2 reg.	20.0 (1.6)	15.7 (2.8)	8.7 (2.0)
PSEs	32.4 (8.2)	9.7 (5.2)	**34.1** (9.4)
GAN	22.9 (2.2)	16.5 (3.0)	9.9 (2.1)
HAM (ours)	**34.9** (3.5)	**37.5** (7.1)	19.3 (2.8)
HAM w/o CD	25.6 (4.9)	32.9 (6.0)	13.3 (2.9)
HAM w/o BD	26.7 (4.2)	31.1 (6.4)	15.2 (4.7)

Table 1: Generalization results on Jumping task: we present the generalization results of baseline methods, our model, and ablated model; without context discriminator (HAM w/o CD) and without background discriminator (HAM w/o BD) with three types of evaluation grid configuration. We measure the test performances after 20k timesteps training with an imitation agent as in PSEs (Agarwal et al., 2021). The results show the mean success ratio (%) among 286 tasks over 20 runs with different seeds and the values in parentheses represent standard deviations.

4.1.2 Disentanglement of visual features

The following results show hypothetical observations generated at the end of the training. Green-bordered images are generated by using task context code z_c^i and task background code z_b^j, respectively, from the ith observation x^i and jth observation x^j. We can find that task background-relevant visual features; the floor height and the location of the obstacle of generated hypothetical observation are similar to its first row x^j in our model. We can also see that task context-relevant feature; the distance between the agent and the obstacle that determines whether to jump is identical to its first column x^i. In other words, our model can effectively separate task context-relevant visual features from others, which is in line with the significant performance improvement in the Narrow configuration



(a) HAM (b) HAM w/o CD (c) HAM w/o BD

Figure 3: We show the role of each discriminator in our model by visualizing the generated hypothetical observations $G([z_c^i, z_b^j])$ (green-bordered images) in ablated models. We can find that HAM without context discriminator cannot separate context information (highlighted with red arrows) from x^i , and HAM without background discriminator cannot separate background information (highlighted with blue arrows) from x^j while our model separates both information successfully.

(see Table 1) quantifying the extrapolation ability. We also evaluated the robustness of our learned task context space with both quantitative metrics and qualitative visual results in Appendix D.1.

4.2 ProcGen benchmark

ProcGen (Cobbe et al., 2020) is a collection of unique environments designed to measure both sample efficiency and generalization in RL. In ProcGen, the train and test environments differ broadly in visual appearance and structure. We followed the environment setup in ProcGen, and we use Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm for training our policy network.

4.2.1 Generalization test

We conduct the generalization tests with 9 different models, including PPO, PPO with L_2 regularization, HAM (ours), HAM w/o CD (context discriminator ablated version), HAM w/o BD (background discriminator ablated version), and RAD with different augmentations. We chose the four representative and best-performing data augmentation techniques; gray, random crop, cutout, and color-jitter in (Laskin et al., 2020). We excluded models based on other representation learning techniques (e.g., contrastive learning (Srinivas et al., 2020), and deepMDP (Gelada et al., 2019)) for the fair comparison because both models are based on different RL algorithms; SAC (Haarnoja et al., 2018), and Distributional Q-learning (Bellemare et al., 2017). We also found the degraded generalization performance for both models in the ProcGen benchmark challenge (Mohanty et al., 2020).

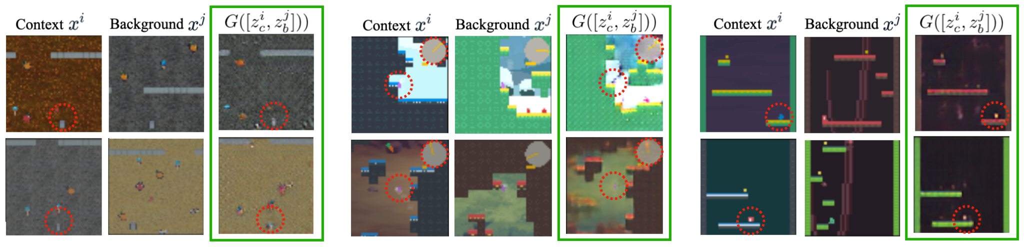
We use three OpenAI ProcGen environments: Fruitbot, Jumper, and Climber. For more information about each environment, refer to Appendix C.2.1. We found that our model significantly outperforms all the baselines when the training dataset is small (see Table 2). Notably, our model beats all the baselines trained on two times the number of training levels in Fruitbot and Jumper environments when the training dataset is the smallest; this is the most challenging scenario. We also plot the generalization gap over training level (see Appendix E.2.1). Vanilla PPO shows a significant gap with a small amount of training data, and the gap gradually reduces as the number of training data increases. On the other hand, our model maintains a small gap from the most challenging training condition.

4.2.2 Disentanglement of visual features

We present the generated observations from HAM around the end of the training process (see Figure 4). The results imply that our model successfully separate task context-relevant latent features from the task background-relevant ones. The task background code reflects to which task level the generated observation belongs. The task context code reflects to visual features that influence action selection, such as the structure of walls and the placement of fruits in the Fruitbot environment, the direction of the compass needle that points to the location of the carrot in the Jumper environment, and the arrangement of stairs and stars in the Climber environment. One intriguing aspect of our findings is that our model can disentangle not only style-relevant or structural generative factors like StyleGAN (Karras et al., 2020) but also important action-relevant visual features such as the direction of the

Table 2: Generalization results on ProcGen environments: we present the generalization results of our model and baseline methods on the three OpenAI ProcGen environments: Fruitbot, Jumper, and Climber. We measure the test performances after 20M timesteps. The results show the mean and standard deviation averaged over three runs with different seeds.

	# of levels	PPO	PPO $+L_2$ reg.	HAM	HAM -CD	HAM -BD	RAD (gray)	RAD (crop)	RAD (cutout)	RAD (color-jitter)
Fruitbot	50	6.6 ± 1.1	10.4 ±2.5	**17.3** ± 0.4	16.4 ± 1.6	9.7 ± 4.9	4.6 ±2.7	4.4 ±2.5	8.1 ±0.3	-1.4 ±0.9
	100	14.5 ±2.9	16.6 ±2.3	**19.6** ±2.1	18.4 ±1.5	18.0 ±0.9	7.6 ±2.0	11.2 ±3.8	14.7 ±0.4	5.3 ±4.5
	200	19.4 ±1.9	20.3 ±0.6	**21.3** ±0.6	20.5 ±1.4	19.9 ±0.6	14.1 ±0.6	16.1 ±4.7	18.5 ±2.6	19.4 ±2.9
Jumper	50	4.8 ±0.2	5.3 ±0.3	**6.0** ±0.1	5.6 ±0.3	5.4 ±0.3	5.0 ±0.2	4.0 ±0.2	5.2 ±0.2	5.6 ±0.2
	100	5.2 ±0.5	5.8 ±0.2	**6.2** ±0.3	6.0 ±0.1	6.1 ±0.2	5.2 ±0.1	5.1 ±0.2	5.6 ±0.1	6.1 ±0.2
	200	6.0 ±0.2	6.3 ±0.1	**6.4** ±0.1	6.4 ±0.1	6.3 ±0.2	5.6 ±0.1	5.2 ±0.7	5.4 ±0.1	5.9 ±0.1
Climber	50	3.4 ±0.2	3.6 ±0.2	**3.7** ±0.1	2.8 ±0.2	2.7 ±0.3	3.3 ±0.1	2.7 ±0.6	3.3 ±0.2	3.4 ±0.1
	100	4.2 ±0.3	**4.4** ±0.2	4.1 ±0.2	3.3 ±0.2	3.2 ±0.2	3.6 ±0.1	2.8 ±0.1	4.1 ±0.3	4.0 ±0.4
	200	4.5 ±0.1	**4.9** ±0.3	4.5 ±0.7	3.8 ±0.2	3.7 ±0.2	4.4 ±0.3	3.2 ±0.2	4.6 ±0.4	4.2 ±0.5



(a) Fruitbot (b) Jumper (c) Climber

Figure 4: Generated hypothetical observations $G([z_c^i, z_b^j])$ (green-bordered) with the task context code of x^i and background code of x^j.

compass needle in the Jumper environment. We also evaluated the robustness of our learned task context space using t-SNE in Appendix D.2.

5 Conclusion

Motivated by the human analogy-making process, this paper presents a novel auxiliary process called Hypothetical Analogy Making (HAM), which enables RL agents to learn compact and explainable context-relevant features that can generalize to unseen tasks. The HAM consists of three parts: task context and background relevant encoding, hypothetical observation generation, and analogy-making. By using an information-theoretic approach, we show that this process can be translated onto a simple encoder-decoder-discriminator architecture while maximizing the lower bound of the objective function of learning broadly generalizable policy. In simulations with the Jumping task and the OpenAI ProcGen benchmark, we show that our model can learn a generalizable behavioral policy by revealing the robust task context space. Our model outperforms other state-of-the-arts in challenging settings with less training data. In subsequent analyses, we show that HAM learns the proper task context-relevant visual features by visualizing generated hypothetical observations. We also analyze how robust our learned task context space is by comparing training and test samples' task context code distribution. Our approach opens up a new possibility of learning generalizable inductive bias by mimicking human cognition. Furthermore, our work is free from the limitations of existing studies that utilize invariance through hand-crafted transformations by learning task background invariance itself. Applying the attention mechanism when dividing GAN features into task context and background relevant groups will be interesting directions for the future work.

Acknowledgments

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01371, Development of brain-inspired AI with human-like intelligence), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2019M3E5D2A01066267), IITP grant funded by the Korean government (No.2017-0-00451), and Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-TC1603-52.

References

Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G. Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

John R. Anderson. Cognitive psychology and its implications. 1980.

Paul Bartha. Analogy and analogical reasoning. *The Stanford Encyclopedia of Philosophy (Spring 2019 Edition)*, 2019. URL `https://plato.stanford.edu/archives/spr2019/entries/reasoning-analogy`.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*, 2017.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016.

Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*, 2020.

Remi Tachet des Combes, Philip Bachman, and Harm van Seijen. Learning invariances for policy generalization. In *ICLR Workshop Track*, 2018.

Debidatta Dwibedi, Jonathan Tompson, Corey Lynch, and Pierre Sermanet. Learning actionable representations from visual observations. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1577–1584, 2018.

T. Fetaya, Elias Wang, K.-C. Welling, Michelle Zemel, Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. Neural relational inference for interacting systems. *arXiv: Machine Learning*, 2018.

Chelsea Finn, Xin Yu Tan, Yan Duan, Trevor Darrell, Sergey Levine, and P. Abbeel. Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. In *Conference on Robot Learning*, 2015.

James M. Foster and Matt Jones. Analogical reinforcement learning. *Cognitive Science*, 35, 2013.

Xiang Fu, Ge Yang, Pulkit Agrawal, and T. Jaakkola. Learning task informed abstractions. *ArXiv*, abs/2106.15612, 2021.

Shani Gamrian and Yoav Goldberg. Transfer learning for related reinforcement learning tasks via image-to-image translation. *ArXiv*, abs/1806.07377, 2019.

Yaroslav Ganin and Victor S. Lempitsky. Unsupervised domain adaptation by backpropagation. *ArXiv*, abs/1409.7495, 2015.

Marta Garnelo, Kai Arulkumaran, and Murray Shanahan. Towards deep symbolic reinforcement learning. *ArXiv*, abs/1609.05518, 2016.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *ArXiv*, abs/1906.02736, 2019.

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

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cynthia H Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, 2019.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, K. Czechowski, D. Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, G. Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *ArXiv*, abs/1903.00374, 2020.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8107–8116, 2020.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.

Brenden M. Lake, Tomer David Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2016.

George Lakoff. The neural theory of metaphor. *Law & Rhetoric eJournal*, 2009.

Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. In *Advances in Neural Information Processing Systems*, 2020.

Alex X. Lee, Anusha Nagabandi, P. Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. In *34th Conference on Neural Information Processing Systems*, 2020.

Sergey Levine, Aviral Kumar, G. Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *ArXiv*, abs/2005.01643, 2020.

Bogdan Mazoure, Rémi Tachet des Combes, Thang Van Doan, Philip Bachman, and R. Devon Hjelm. Deep reinforcement and infomax learning. *ArXiv*, abs/2006.07217, 2020.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *ArXiv*, abs/1411.1784, 2014.

Melanie Mitchell. Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 2021.

Sharada Prasanna Mohanty, Jyotish Poonganam, Adrien Gaidon, Andrey Kolobov, Blake Wulfe, Dipam Chakraborty, Gravzvydas vSemetulskis, João Schapke, Jonas Kubilius, Jurgis Pavsukonis, Linas Klimas, Matthew J. Hausknecht, Patrick MacAlpine, Quang Nhat Tran, Thomas Tumiel, Xiaocheng Tang, Xinwei Chen, Christopher Hesse, Jacob Hilton, William H. Guss, Sahika Genc, John Schulman, and Karl Cobbe. Measuring sample efficiency and generalization in reinforcement learning benchmarks: Neurips 2020 procgen benchmark. *ArXiv*, abs/2103.15332, 2020.

Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation. In *34th Conference on Neural Information Processing Systems*, 2020.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

A. Srinivas, Michael Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15: 1929–1958, 2014.

Chris Tensmeyer and Tony R. Martinez. Improving invariance and equivariance properties of convolutional neural networks. 2017.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. *ArXiv*, abs/1910.12827, 2019.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations*, 2021.

A Related work

A.1 Analogical Reasoning and RL

Analogical reasoning is a high-level cognitive process that finds a common relational system between two exemplars, domains, or situations (Bartha, 2019). Many attempts have been made to incorporate human analogical reasoning to encode relational similarity for RL (Foster & Jones, 2013; Fetaya et al., 2018). They often make object-level abstraction of observations to find symbolic rules between them (Garnelo et al., 2016; Veerapaneni et al., 2019). On the contrary, we focus on the role of analogy-making in high-level abstraction and concept learning (Mitchell, 2021). We intend to make observation-level analogies between the original and hypothetical observations, thereby gleaning from observations the **task context** information that corresponds to the core *concept* of an RL task.

A.2 Generalization in RL

Representation learning. Many RL algorithms incorporated a representation learning process to improve generalization performance. This approach obtains robust policies by learning compact vector representations from images (Finn et al., 2015; Dwibedi et al., 2018; Lee et al., 2020; Mazoure et al., 2020). DBC (Zhang et al., 2021) measures behavioral similarity between observations using reward signal and state transition probability, and PSE (Agarwal et al., 2021) compares optimal behaviors of an agent for learning robust representations. CURL (Srinivas et al., 2020) adopts a contrastive auxiliary task that leverages different views of an augmented image. In this paper, we obtain robust representations for generalizable policy by decomposing the latent space of observation space into task context and task background space. Our model presents a new perspective on representation learning in RL. Moreover, the learned hypothetical observation generative model has further possibilities for various applications, such as virtual observation generation in the metaverse.

Regularization and data augmentation. Regularization techniques, known to be effective in supervised learning contexts such as L_2 regularization and dropout (Srivastava et al., 2014), also help RL agents generalize to unseen contexts (Cobbe et al., 2019; Igl et al., 2019). As another attempt, data augmentation techniques in RL such as RAD (Laskin et al., 2020) improve generalization to unseen tasks or levels by simply training on more diversely augmented samples. RAD utilizes different views of the same input and maximizes the MI between features taken from them. While these methods are simple and effective, the background invariance of the learned policy is due to several predetermined factors such as translation, rotation, and color. This limits the applicability of these methods to situations that deal with a specific type of invariance; for example, the color of an object affects the probability of reward acquisition. On the other hand, our model learns task background code itself and makes analogies with hypothetical observations generated with different task background codes.

Task background modeling. A similar attempt for task background modeling was made in model-based learning. TIA (Fu et al., 2021) learns a distractor model using reward disassociation with negative gradient flow (Ganin & Lempitsky, 2015). TIA acquires reward-related code using the model and utilizes it for planning. On the other hand, our model obtains task context representation by dividing the latent space learned through mutual-information regularized GAN into task context-related and background-related ones using an additional analogy-making objective.

B Proof

We provide proof of each Theorem 3.1, Corollary 3.1.1, and Corollary 3.1.2 as follows.

B.1 Proof of Theorem 3.1 and Corollary 3.1.1

Theorem 3.1. *The problem of maximizing the mutual information between the generated image and existing labels of codes used to generate the image, $\mathcal{I}(c^i, G([z_c^i, z_b^j])) + \mathcal{I}(b^j, G([z_c^i, z_b^j]))$, is equivalent to the problem of maximizing the lower bound of $\mathcal{I}(c, z_c)$ while minimizing $\mathcal{I}(b, z_c)$.*

Corollary 3.1.1. *In Theorem 3.1, the lower bound is tight when z_c and z_b are mutually independent.*

Proof.

$$\begin{aligned}\mathcal{I}(c^i, G([z_c^i, z_b^j])) + \mathcal{I}(b^j, G([z_c^i, z_b^j])) &= \mathcal{I}(c^i, [z_c^i, z_b^j]) + \mathcal{I}(b^j, [z_c^i, z_b^j]) \\ &\leq \mathcal{I}(c^i, z_c^i) + \mathcal{I}(c^i, z_b^j) + \mathcal{I}(b^j, z_c^i) + \mathcal{I}(b^j, z_b^j) \\ &= \mathcal{I}(c^i, z_c^i) + \mathcal{I}(b^j, z_b^j) \\ &= \mathcal{I}(c, z_c) + \mathcal{I}(b, z_b) \\ &= H(z_b^*) + \mathcal{I}(c, z_c) \\ &= H(z_b^*) + H(z_c) - \mathcal{I}(b, z_c)\end{aligned} \tag{7}$$

The first equality holds due to $H(X) = H(f(X))$ for any bijective function f (Invariance under relabeling). The second inequality uses $H(X_1, ..., X_n) = \sum_{i=1}^{n} H(X_i|X^{i-1}) \leq \sum_{i=1}^{n} H(X_i)$ (Full chain rule) with equality iff $X_1, ..., X_n$ mutually independent (Corollary 3.1.1). The third equality holds because $\mathcal{I}(c^i, z_b^j) = 0$ and $\mathcal{I}(b^j, z_c^i) = 0$ when $i \neq j$. The last equality uses the definition of b, $H(x|c) = \mathcal{I}(b, x)$.

□

B.2 Proof of Corollary 3.1.2

Corollary 3.1.2. *Let a be any label for the generated image x. The mutual information between the label a and the generated image x, $\mathcal{I}(a, x)$, can be maximized by minimizing the classification loss of auxiliary classifier $Q(a|x;\theta)$.*

Proof.

$$\begin{aligned}\mathcal{I}(a, G([z_c^i, z_b^j])) &= H(a) - H(a|G([z_c^i, z_b^j])) \\ &= \mathbb{E}_{x\sim G([z_c^i, z_b^j])}[\mathbb{E}_{a'\sim P(a|x)}[log(P(a'|x))]] + H(a) \\ &= \mathbb{E}_{x\sim G([z_c^i, z_b^j])}[D_{KL}(P(\cdot|x)\|Q(\cdot|x)) + \mathbb{E}_{a'\sim P(a|x)}[log(Q(a'|x))]] + H(a) \\ &\geq \mathbb{E}_{x\sim G([z_c^i, z_b^j])}[\mathbb{E}_{a'\sim P(a|x)}[log(Q(a'|x'))]] + H(a) \\ &= \mathbb{E}_{a\sim P(a), x\sim G([z_c^i, z_b^j])}[log(Q(a|x))] + H(a) \\ &= -\mathbb{E}_{a\sim P(a), x\sim G([z_c^i, z_b^j])}[-log(Q(a|x))] + H(a)\end{aligned} \tag{8}$$

The fourth inequality uses variational lower bound (Kingma & Welling, 2014). The fifth equality uses generalized law of total expectation, $\mathbb{E}(X) = \mathbb{E}(\mathbb{E}(X|Y))$ derived in InfoGAN (Chen et al., 2016) lemma 5.1.

□

C Experiment Detail

C.1 Jumping task

We provide information about Jumping task (des Combes et al., 2018) and show our architecture designs and additional training details.

C.1.1 Environment description

In the Jumping task, a white agent has to jump over a gray obstacle without touching it. The agent can choose between two actions: right and jump. The environment is deterministic, with the agent observing a reward of $+1$ at each time step. If the agent successfully reaches the rightmost side of the screen, it receives a reward of $+100$; if the agent touches the obstacle, the episode terminates with a negative reward of -1. The observation space is the $60x60$ pixel representation of the environment, as depicted in Figure 5.

The evaluation measures test performance for 286 tasks of 26 different obstacle locations and 11 floor heights. The training proceeds with only 18 of them. Depending on the distribution of training

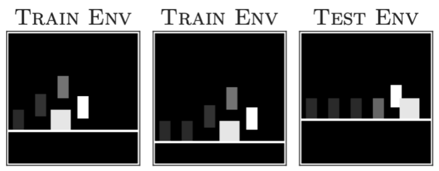
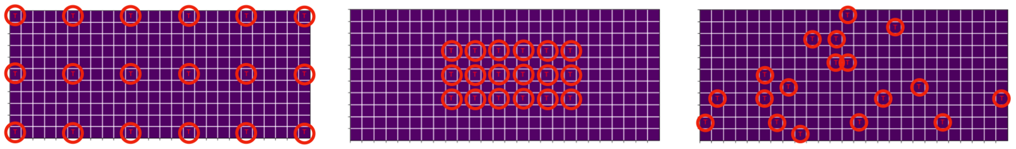


Figure 5: Example MDPs from Jumping task: In Jumping task, the agent (white block) should jump over the obstacle (gray block) without touching it. The shaded trajectory shows the optimal trajectory in training MDPs. The train and test MDPs differ in their observation space which is generated with different floor heights and obstacle positions of various ranges.



(a) Wide (b) Narrow (c) Random

Figure 6: Evaluation grids of Wide, Narrow, and Random grid configurations.

samples on the evaluation grid, the training type is classified into Wide, Narrow, and Random. The Wide type measures generalization through ”interpolation.” The Narrow type measures generalization outside the distribution through ”extrapolation.” Random type assumes a situation in which training and test data are sampled independently in the same distribution. Evaluation grid for each grid configuration is in Figure 6.

C.1.2 Architecture of HAM module

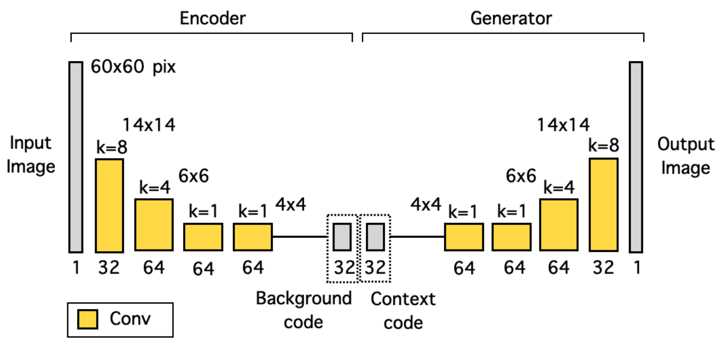


Figure 7: Architecture of encoder and generator

The **encoder** separates the input observation into task context and background codes, as shown in Figure 7 (left). For the context and background code, the network consists of 3 downsampling convolution layers and 1 convolution layer with kernel sizes 8x8, 4x4, 1x1, and 1x1 and strides 4, 2, 1, and 1. The code with 64 channels is divided into the background and context codes with 32 channels each. The divided context code is fed into a linear layer which computes the policy that outputs the probability of the jump and right actions. For the design of convolution layers, we referred to (Agarwal et al., 2021).

The **generator** maps the codes to an hypothetical observation, as shown in Figure 7 (right). The network consists of 1 convolution layer and 3 upsampling convolution layers. The kernel sizes and strides are symmetric to the encoder’s downsampling convolution layers.

The **discriminator** is designed to determine if the observation is realistic or not. Each observation is first encoded with3 downsampling convolution layers and 1 convolution layer with kernel sizes 8x8, 4x4, 1x1, and 1x1 and strides 4, 2, 1, and 1. the final prediction applies 3 dense layers to the flattened representation (see Figure 8). The **context discriminator** and **background discriminator** follow the identical architecture with the basic discriminator. Except that the **context discriminator** and **background discriminator** take an observation subtracted by the reference observation as an input to determine if the observation has the same background or context as the reference observation. They output logits with length 2 at the last dense layer.

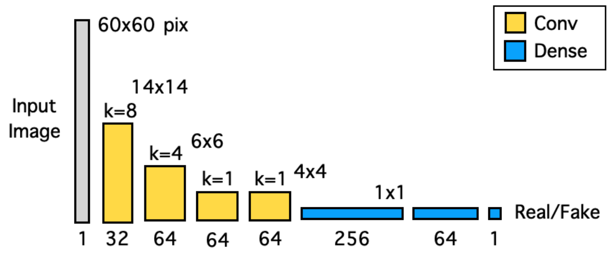


Figure 8: Architecture of discriminator: basic discriminator architecture consists of the feature extractor, which applies 3 downsampling convolution layers and 1 convolution layer with kernel sizes 8x8, 4x4, 1x1, and 1x1 and strides 4, 2, 1, and 1. And then we apply 3 dense layers to the flattened features and outputs the final prediction. The **context discriminator** and **background discriminator** shares the identical architecture with the basic discriminator except their final output is logits with length 2.

C.1.3 Training details

At each iteration, we conduct learning on the entire limitation data and train the limitation agent and the HAM module alternately in units of batches of size 256. Most hyperparameters for imitation agents with baseline methods are from (Agarwal et al., 2021), such as 256 batch size, 0.003 learning rate, 0.999 decay rate, and 256 hidden dimensions. We adjusted a few hyperparameters for imitation agent with HAM module as 0.001 learning rate (imitation agent), 0.0003 learning rate (HAM module), and 0.9999 decay rate.

C.2 ProcGen benchmark

We provide information about environments in ProcGen benchmark (Cobbe et al., 2020) and show our architecture designs and additional training details.

C.2.1 Environment descriptions

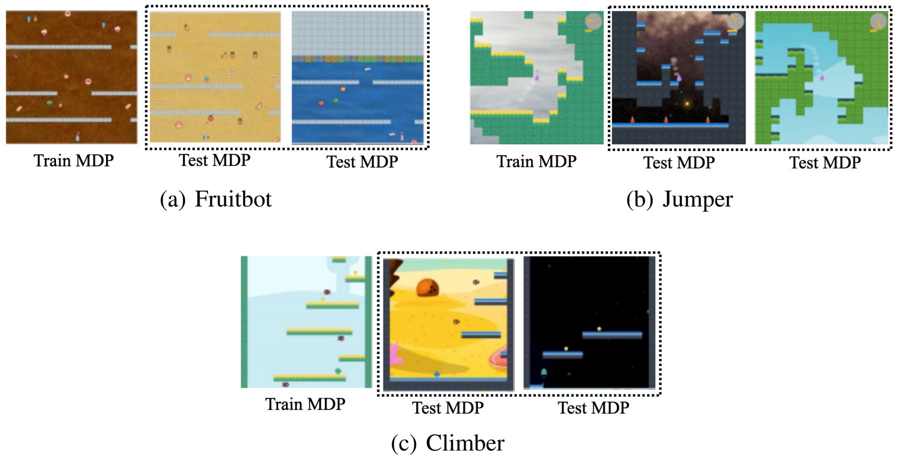


Figure 9: Example MDPs from Fruitbot, Jumper, and Climber environments:

A detailed description of Fruitbot, Jumper, Climber environments that we used in ProcGen benchmark (Cobbe et al., 2020) is as follows. Example MDPs of each environment are in Figure 9.

Fruitbot. A simple scrolling game where the player must navigate between gaps in walls and collect fruit along the way. The player receives a positive reward for collecting a piece of fruit and a more significant negative reward for mistakenly collecting a non-fruit object.

Jumper. An open world environment where the player, a bunny, should find the carrot which is randomly located in the map. The screen includes a compass with a needle that displays direction and distance to the carrot. Style of background, map structure, and location of enemy depending on the level.

Climber. A simple platformer where the player must climb a sequence of platforms, collecting stars along the way. Collecting a star gives a small reward, and a larger reward is given for collecting all the stars in a level. If all stars are collected, the episode ends.

C.2.2 Architecture of HAM module

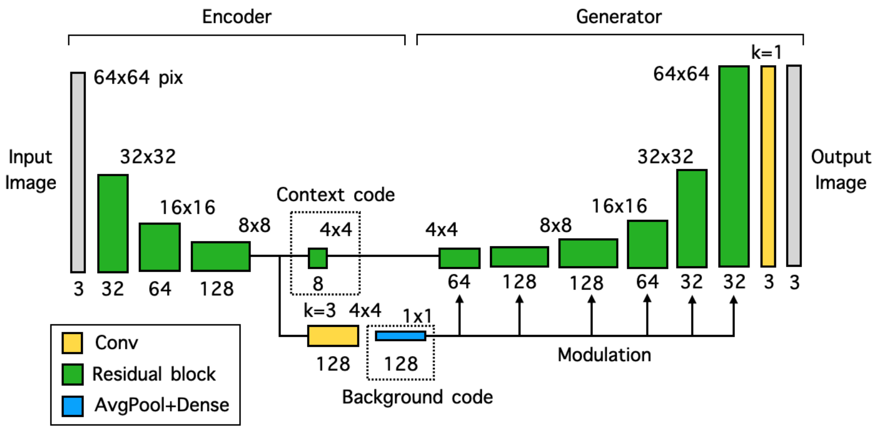


Figure 10: Architecture of encoder and generator

The **encoder** separates the input observation into task context and background codes, as shown in Figure 10 (left). For the context code, the network consists of 4 downsampling residual blocks (He et al., 2016). For the background code, the network branches off after 3 downsampling residual blocks followed by a convolution layer and an average pooling, and a dense layer. The design of the code shape is from Swap Autoencoder (Park et al., 2020) to impose an inductive bias that background information in ProcGen benchmark is agnostic to positional information like style features. The spatial dimension in the background code is removed using average pooling and no padding in the convolution layer.

The **generator** maps the codes to an hypothetical observation, as shown in Figure 10 (right). The network uses the context code in the main branch, followed by 2 residual blocks and 4 upsampling residual blocks. The background code is injected using the weight modulation layer from StyleGAN2 (Karras et al., 2020). We generate the hypothetical observation by applying a convolutional layer at the end of the residual blocks as in Swap Autoencoder.

The **discriminator** architecture is identical to Swap Autoencoder, except we omitted a few residual blocks as the observation size in ProcGen is 64x64, relatively smaller than the image dataset (e.g., 512x512) in Swap Autoencoder. The architecture of **background discriminator** is in Figure 11. The design is to determine whether an input patch has realistic background features compared to the reference patch.

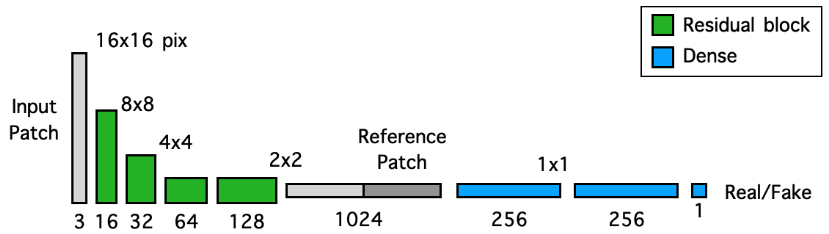


Figure 11: The architecture of background discriminator: the architecture consists of the feature extractor, which applies 3 downsampling residual blocks, 1 residual block, and the classifier with 3 dense layers. When fed into the classifier, we concatenate the flattened feature of the input patch with that of the reference patch in the channel dimension. The classifier outputs logit representing the background similarity between the input and reference images. For the reference patches, we average the extracted features of 4 patches over the batch dimension as in (Park et al., 2020).

C.2.3 Training details

For each ProcGen environment hyperparameters, we follow the environment setup in ProcGen (Cobbe et al., 2020), and hyperparameters in PPO (Schulman et al., 2017). We show a full list of hyperparameters for ProcGen experiments in Table 4 and Table 3. The hyperparameters values are unchanged across environments.

Table 3: Hyperparameters used for PPO algorithm and ProcGen environments

Hyperparameter	Value
PPO learning rate (α_{ppo})	0.0003
RMSprop optimizer epsilon	0.00001
RMSprop optimizer alpha	0.99
Discount factor for rewards	0.999
GAE coefficient	0.95
Entropy coefficient	0.01
Value loss coefficient	0.5
Max norm of gradients	0.5
Use unnormalized returns	FALSE
# of training CPU processes	32
# of forward steps in A2C	256
# of PPO epochs	3
# of batches for PPO	8
PPO clip parameter	0.2
# of environment steps to train	20M
distribution of environments	easy
Paint velocity vector	FALSE
Start level id	0

Table 4: Hyperparameters used for HAM module

Hyperparameter	Value
Encoder learning rate (α_{enc})	0.0003
Discriminator learning rate (α_{disc})	0.0001
# of batches for SRAR	8
Action code dimension	256
GAN loss coefficient (λ_{gan})	0.5
Style loss coefficient (λ_{style})	1.0
Action loss coefficient (λ_{action})	1.0
# of crop	8
reference # of crop	4
R1 coefficient	10
Style R1 coefficient	1.0

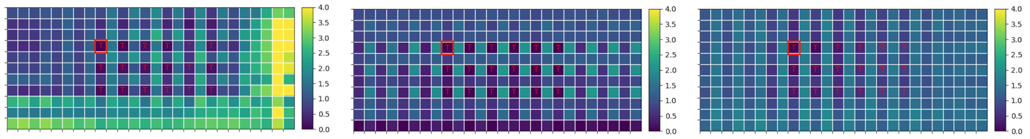
D Robustness of learned task context space

D.1 Jumping task

To evaluate how robust our learned task context code is against task background changes, we measure the normalized Euclidean distance (Tensmeyer & Martinez, 2017) in the representation space.

$$dist(z_c, z_c^*) = \frac{\|z_c - z_c^*\|_2}{\|z_c^*\|_2} \tag{9}$$

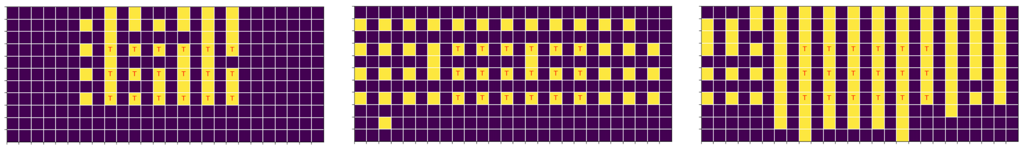
, where z^* is the left top sample in the training grid. Figure 12 shows the qualitative result that how similar our learned context space with training tasks (cells marked with $\top$) and the one with test tasks (cells in the entire grid) are so that learned policy can be deployed to unseen test data without errors. We can find that the color distributions of training and test task cells are much similar in our model compared to the baseline.



(a) GAN (b) L_2 reg. (c) HAM (Ours)

Figure 12: Invariance grid of learned task context on background changes: Grid cells marked with $\top$ indicate training tasks, and a cell with a red rectangle is a standard data point for normalization. The color of each cell shows the normalized Euclidean distance between the task context code of the current cell and the standard cell when the optimal action is *to jump*

We show the visualization of learned context space in Figure 12 and how it connects to the generalization performance in Figure 13. We use normalized euclidean distance as we mentioned above.



(a) GAN (b) L_2 reg. (c) HAM

Figure 13: Visualization of generalization performance of HAM and baseline methods in Narrow training grid configuration. Red letter $\top$ indicates the training tasks. Yellow tiles are solved tasks and violet tiles are tasks each method couldn't solve.

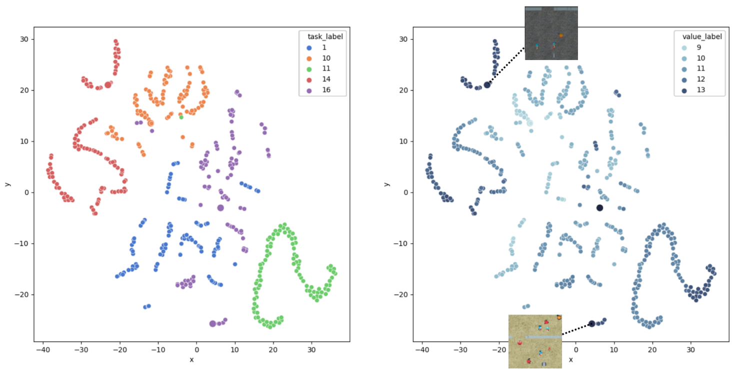
To quantitatively measure the distribution shift of the task context code between training and test tasks, we first assume that $dist(z_c, z_c^*)$ measured in each task set follows normal distribution $N(\mu, \sigma^2)$. We then measure the area under the curve (AUC) of probability density function of test context distribution overlapped with train context distribution with range of $[\mu_{train} - 3 * \sigma_{train}, \mu_{train} + 3 * \sigma_{train}]$. The higher value implies the test context distribution is mostly covered by the policy learned with the train context distribution. Appendix D.1 shows that our model has the highest AUC in all three grid configurations and learns the most robust context space against varying background features.

Method	Area under the curve		
	Wide	Narrow	Random
L_2 reg.	0.02 (0.06)	0.01 (0.02)	0.03 (0.14)
GAN	0.15 (0.09)	0.33 (0.26)	0.11 (0.07)
HAM	**0.72** (0.15)	**0.60** (0.19)	**0.33** (0.29)

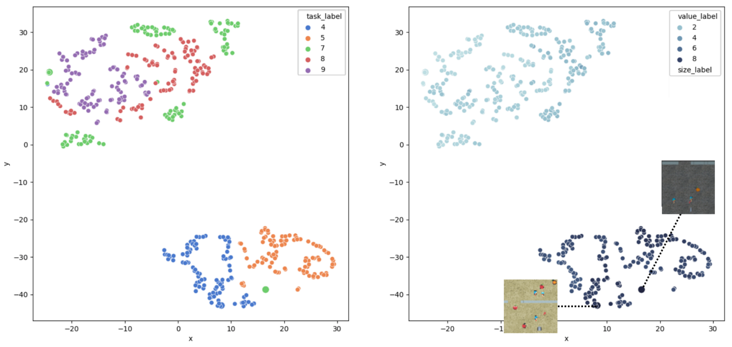
Table 5: Quantitative measure for the robustness of learned context space: we measure the area under the curve of the probability distribution of test context space covered by train context space. The measured area can range between $[0, 1]$. The results show the mean and standard deviation averaged over 10 runs.

D.2 ProcGen benchmark

To further evaluate the robustness of the learned task context space, we visualized the task context code z_c of observations in $2d$ space using t-SNE (see Figure 14). Because there was no optimal training data in the ProcGen environment, we could not sample the training and test observations with the same task context, as in the Jumping task. A sufficiently trained models can serve as an agent with the optimal policy, however, observations can be asynchronously sampled because the training performance differs by each baseline method. Instead, we plot the task context code z_c of observations in various tasks with color codes representing its task and value label. We show that our learned task context space is more task background-invariant. Our task context codes are clustered based on value similarity, while the vanilla PPO's codes are based on background similarity.



(a) PPO



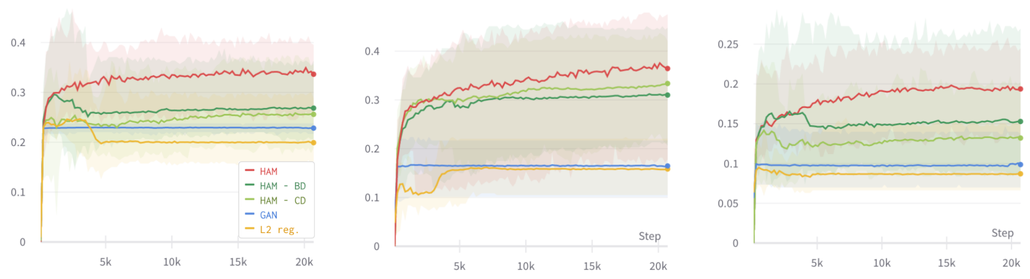
(b) HAM

Figure 14: T-SNE visualization of task context code: visualization of z_c trained with vanilla PPO and PPO + HAM module using 50 training levels in the Fruitbot environment. 100 observations from randomly sampled 5 different tasks. The figure on the left is color-coded with task label and the right figure is color-coded with value label. We can find that the learned context space with our HAM module is more task-invariant compared to the vanilla PPO model.

E Additional results

E.1 Jumping task

Generalization curves



(a) Wide (b) Narrow (c) Random

Figure 15: Generalization performance over training timestep on the entire grid including unseen floor heights and obstacle positions. HAM outperforms other baseline methods and ablated versions of HAM. We plot the average success ratio (# of success # of entire tasks in the grid), and the shaded region shows the standard deviation.

E.2 ProcGen benchmark

E.2.1 Generalization gap over training steps

We found HAM shows the smallest gap during the whole training process when the training dataset is small by plotting the generalization gap over timestep (see Figure 16).

We also plot the generalization gap over training level in the Fruitbot environment (see Figure 17). Vanilla PPO shows a significant gap with a small amount of training data, and the gap gradually reduces as the number of training data increases. On the other hand, our model maintains a small gap from the most challenging training condition. Furthermore, we found that the generalization gap of HAM on 50 training levels is compatible with that of PPO using four times more training data. PPO with L_2 regularization also shows a relatively small gap but not as small as HAM. In summary, the results show that the RL agent with HAM learns task background-invariant context features by virtue of the hypothetical analogy-making process.

E.2.2 Generated hypothetical observations

We show additional results of hypothetical observation generation on Fruitbot, Jumper and Climber in Figure 18. We also show the hypothetical observation generated by performing background interpolation $G([z_c^i, (\alpha \cdot z_b^i + (1-\alpha) \cdot z_b^j)])$ using a random rate $\alpha \in [0, 1)$ in Figure 19. We can find that the context-relevant features (highlighted with red-dotted circles) are preserved as its background-relevant features vary during the interpolation process.

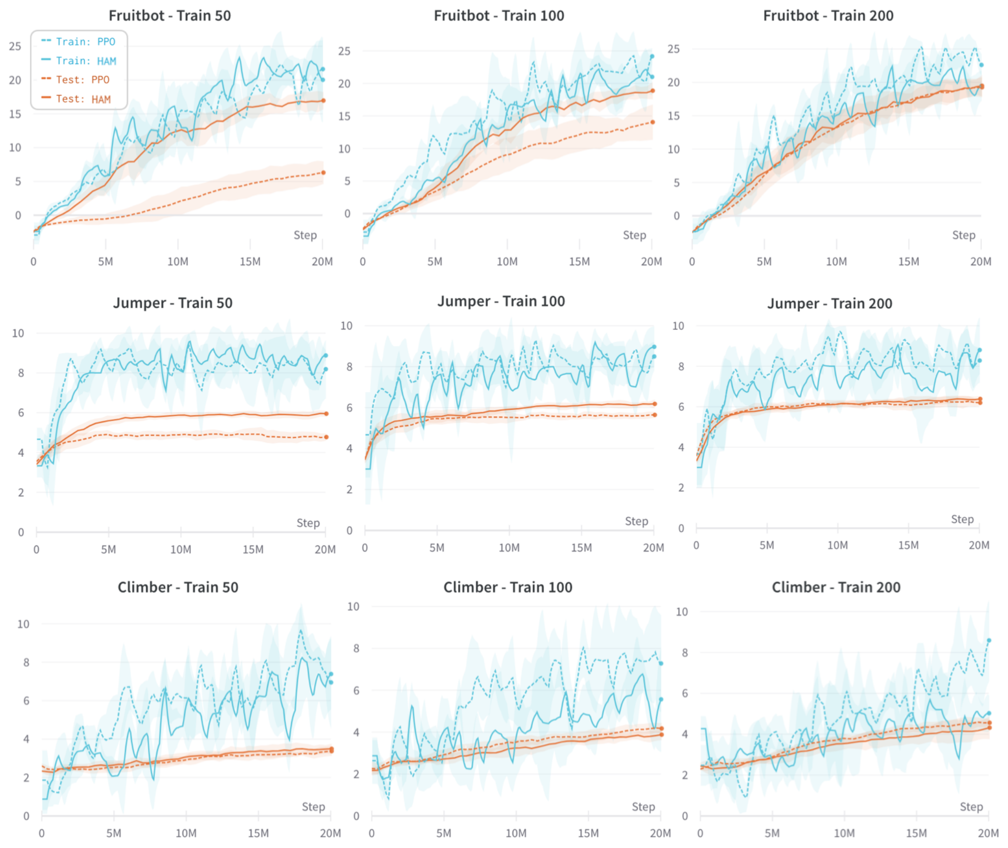
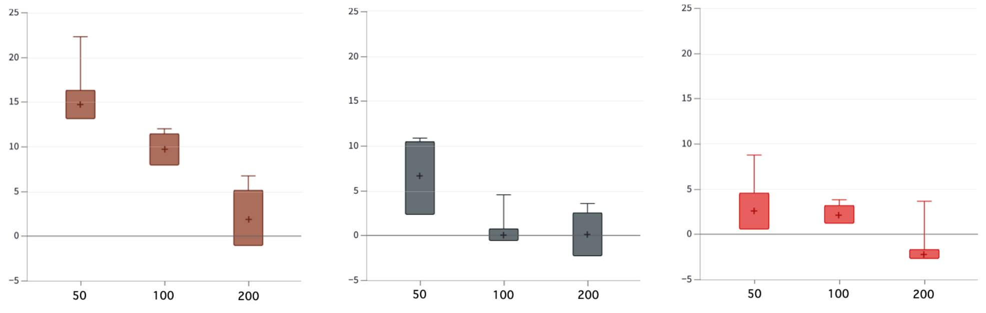


Figure 16: Generalization gap over timestep on Fruitbot, Jumper and Climber.



(a) PPO (b) PPO + L_2 reg. (c) HAM

Figure 17: Generalization gap over training level in the Fruitbot environment.

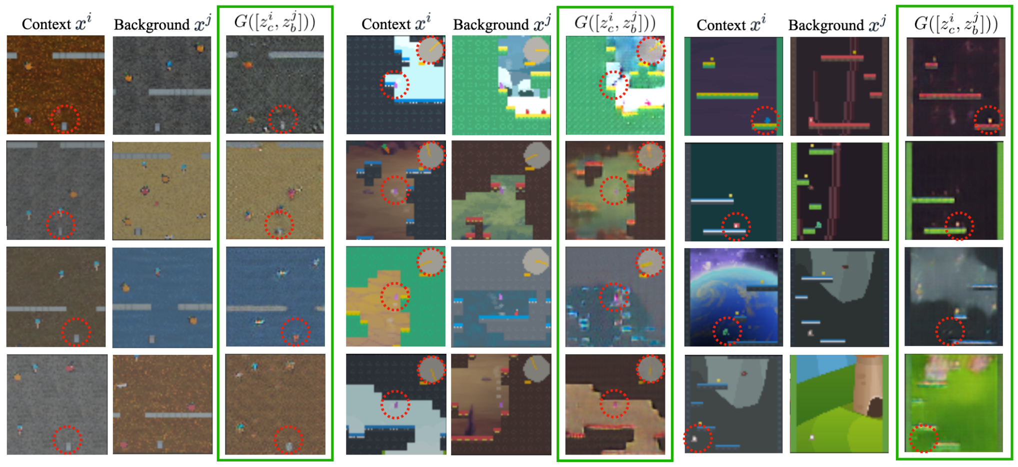


Figure 18: Generated hypothetical observations on Fruitbot, Jumper and Climber.

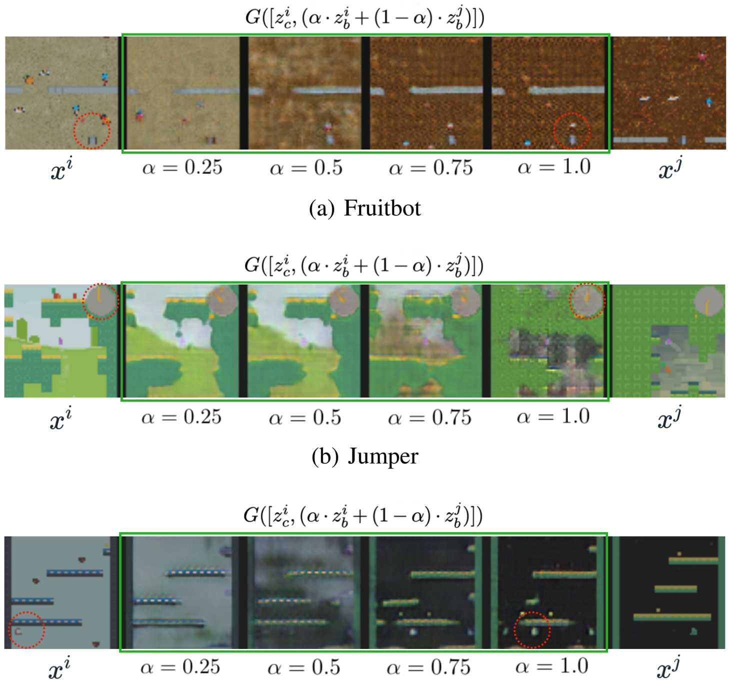


Figure 19: Generated hypothetical observations with background interpolation $G([z_c^i, (\alpha \cdot z_b^i + (1-\alpha) \cdot z_b^j)])$ on Fruitbot, Jumper and Climber.