HOW THE LEVEL SAMPLING PROCESS IMPACTS ZERO-SHOT GENERALISATION IN DEEP REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

A key limitation preventing the wider adoption of autonomous agents trained via deep reinforcement learning (RL) is their limited ability to generalise to new environments, even when these share similar characteristics with environments encountered during training. In this work, we investigate how a non-uniform sampling strategy of individual environment instances, or levels, affects the zero-shot generalisation (ZSG) ability of RL agents, considering two failure modes: overfitting and over-generalisation. As a first step, we measure the mutual information (MI) between the agent's internal representation and the set of training levels, which we find to be well-correlated to instance overfitting. In contrast to uniform sampling, adaptive sampling strategies prioritising levels based on their value loss are more effective at maintaining lower MI, which provides a novel theoretical justification for this class of techniques. We then turn our attention to unsupervised environment design (UED) methods, which adaptively *generate* new training levels and minimise MI more effectively than methods sampling from a fixed set. However, we find UED methods significantly *shift* the training distribution, resulting in over-generalisation and worse ZSG performance over the distribution of interest. To prevent both instance overfitting and over-generalisation, we introduce *self-supervised environment design* (SSED). SSED generates levels using a variational autoencoder, effectively reducing MI while minimising the shift with the distribution of interest, and leads to statistically significant improvements in ZSG over fixed-set level sampling strategies and UED methods.

1 INTRODUCTION

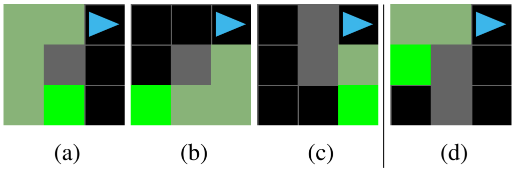


Figure 1: The agent (blue) must navigate to the goal (lime green) but cannot pass through walls (grey) and only observes tiles directly adjacent to itself. An agent trained over levels (a)-(c) will transfer zero-shot to level (d) if it has learnt a behavior adapted to the task semantics of following pale green tiles to reach the goal location.

A central challenge facing modern reinforcement learning (RL) is learning policies capable of zero-shot transfer of learned behaviours to a wide range of environment settings. Prior applications of RL algorithms (Agostinelli et al., 2019; Lee et al., 2020; Rudin et al., 2021) indicate that strong zero-shot generalisation (ZSG) can be achieved through an adaptive sampling strategy over the set of environment instances available during training, which we refer to as the set of training *levels*. However the relationship between ZSG and the level sampling process remains poorly understood. In this work, we draw novel connections between this process and the minimisation of an upper bound on the generalisation error derived by Bertran et al. (2020), which depends on the *mutual information* (MI) between the agent's internal representation and the identity of individual training levels.

An agent learning level-specific policies implies high MI between its internal representation and the level identities, and, in general, will not transfer zero-shot to new levels. To build an understanding of the relationship between MI and ZSG, consider the minimal gridworld navigation example in Figure 1. A "shortcut" exists in level (a), and a model with high MI is able to first predict the level identity from its initial observation to learn an ensemble of level-specific policy optimal over the training set. When deployed on (d) the model will predict it is in (a) since under the agent's restricted field of view (a) and (d) share the same initial observation. As a result the agent will attempt to follow the (a)-specific policy, which will not transfer. We discover that the reduced generalisation error achieved by adaptive level sampling strategies over uniform sampling can be attributed to their effectiveness in reducing the MI between the agent's internal representation and the level identity. In particular, we find that strategies de-prioritising levels with low value loss, as proposed in prioritised level replay (PLR, Jiang et al., 2021b), implicitly minimise mutual information as they avoid training on levels in which the value function is accurately estimated through level identification.

While some adaptive sampling strategies reduce the generalisation gap, their effectiveness is ultimately limited by the number of training levels. We propose *Self-Supervised Environment Design* (SSED) which augments the set of training levels to further reduce generalisation error. We find training on an augmented set can negatively impact performance when the augmented set is not drawn from the same distribution as the training set. We show it induces a form of *over-generalisation*, in which the agent learns to solve levels incompatible with the targeted task, and performs poorly at test time. There is therefore a trade-off between augmenting the training set to prevent instance-overfitting, i.e. to avoid learning level-specific policies, and ensuring that this augmented set comes from the same distribution to avoid distributional shift and over-generalisation. In our experiments, we show that SSED strikes this trade-off more effectively than other adaptive sampling and environment design methods. SSED achieves significant improvements in the agent's ZSG capabilities, reaching 1.25 times the returns of the next best baseline on held-out levels, and improving performance by two to three times on more difficult instantiations of the target task.

2 RELATED WORK

Buffer-free sampling strategies. Domain randomisation (DR, Tobin et al., 2017; Jakobi, 1997), is one of the earliest proposed methods for improving the generalisation ability of RL agents by augmenting the set of available training levels, and does so by sampling uniformly between manually specified ranges of environment parameters. Subsequent contributions introduce an implicit prioritisation over the generated set by inducing a minimax return (robust adversarial RL Pinto et al., 2017) or a minimax regret (unsupervised environment design (UED), Dennis et al., 2020) game between the agent and a *level generator*, which are trained concurrently. These adversarial formulations prioritise levels in which the agent is currently performing poorly to encourage robust generalisation over the sampled set, with UED achieving better Nash equilibrium theoretical guarantees. CLUTR (Azad et al., 2023) removes the need for domain-specific RL environments and improves sample efficiency by having the level generator operate within a low dimensional latent space of a generative model pre-trained on randomly sampled level parameters. However the performance and sample efficiency of these methods is poor when compared to a well calibrated DR implementation or to the buffer-based sampling strategies discussed next.

Buffer-based sampling strategies. Prioritised sampling is often applied to off-policy algorithms, where individual transitions in the replay buffer are prioritised (Schaul et al., 2015) or resampled with different goals in multi-goal RL (Andrychowicz et al., 2017; Zhang et al., 2020). Prioritised Level Replay (PLR, Jiang et al., 2021b) instead affects the sampling process of *future* experiences, and is thus applicable to both on- and off-policy algorithms. PLR maintains a buffer of training levels and empirically demonstrates that prioritising levels using a scoring function proportional to high value prediction loss results in better sample efficiency and improved ZSG performance. Robust PLR (RPLR, Jiang et al., 2021a) extends PLR to the UED setting, using DR as its level generation mechanism, whereas ACCEL (Parker-Holder et al., 2022) gradually evolves new levels by performing random edits on high scoring levels in the buffer. SAMPLR (Jiang et al., 2022) proposes to eliminate the covariate shift induced by the prioritisation strategy by modifying *individual transitions* using a second simulator that runs in parallel. However SAMPLR is only applicable to settings in which the level parameter distribution is provided, whereas SSED can approximate this distribution from a dataset of examples.

Mutual-information minimisation in RL. In prior work, mutual information has been minimised in order to mitigate instance-overfitting, either by learning an ensemble of policies (Bertran et al., 2020; Ghosh et al., 2021), performing data augmentation on observations (Raileanu et al., 2021; Kostrikov et al., 2021), an auxiliary objective (Dunion et al., 2023) or introducing information bottlenecks through selective noise injection on the agent model (Igl et al., 2019; Cobbe et al., 2019).

In contrast, our work is the first to draw connections between mutual-information minimisation and adaptive level sampling and generation strategies.

3 PRELIMINARIES

Reinforcement learning. We model an individual level as a Partially Observable Markov Decision Process (POMDP) $\langle A, O, S, \mathcal{T}, \Omega, R, p_0, \gamma\rangle$ where A is the action space, O is the observation space, S is the set of states, $\mathcal{T} : S \times A \rightarrow \Delta(S)$ and $\Omega : S \rightarrow \Delta(O)$ are the transition and observation functions (we use $\Delta(\cdot)$ to indicate these functions map to distributions), $R : S \rightarrow \mathbb{R}$ is the reward function, $p_0(\mathrm{s})$ is the initial state distribution and γ is the discount factor. We consider the episodic RL setting, in which the agent attempts to learn a policy π maximising the expected discounted return $V^\pi(s_t) = \mathbb{E}_\pi[\sum_{\bar{t}=t}^{T} \gamma^{t-\bar{t}} r_t]$ over an episode terminating at timestep T, where s_t and r_t are the state and reward at step t. We use V^π to refer to $V^\pi(s_0)$, the expected episodic returns taken from the first timestep of the episode. In this work, we focus on on-policy actor-critic algorithms (Mnih et al., 2016; Lillicrap et al., 2016; Schulman et al., 2017) representing the agent policy $\pi_{\boldsymbol{\theta}}$ and value estimate $\hat{V}_{\boldsymbol{\theta}}$ with neural networks (in this paper we use $\boldsymbol{\theta}$ to refer to model weights). The policy and value networks usually share an intermediate state representation $b_{\boldsymbol{\theta}}(o_t)$ (or for recurrent architectures $b_{\boldsymbol{\theta}}(H_t^o)$, $H_t^o = \{o_0, \cdots, o_t\}$ being the history of observations o_i).

Contextual MDPs. Following Kirk et al. (2023), we model the set of environment instances we aim to generalise over as a Contextual-MDP (CMDP) $\mathcal{M} = \langle A, O, S, \mathcal{T}, \Omega, R, p_0(\mathrm{s}|\mathbf{x}), \gamma, X_C, p(\mathbf{x})\rangle$. The CMDP may be viewed as a POMDP, except that the reward, transition and observation functions now also depend on the *context set* X_C with associated distribution $p(\mathbf{x})$, that is $\mathcal{T} : S \times X_C \times A \rightarrow \Delta(S)$, $\Omega : S \times X_C \rightarrow \Delta(O)$, $R : S \times X_C \rightarrow \mathbb{R}$. Each element $\boldsymbol{x} \in X_C$ is not observable by the agent and instantiates a *level* $i_{\boldsymbol{x}}$ of the CMDP with initial state distribution $p_0(\mathrm{s}|\mathbf{x})$. The optimal policy of the CMDP maximises $V_C^\pi = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[V_{i_{\boldsymbol{x}}}^\pi]$, with $V_{i_{\boldsymbol{x}}}^\pi$ referring to the expected return in level $i_{\boldsymbol{x}}$ instantiated by $\boldsymbol{x}$ (we use L to refer to a set of levels i and X to refer to a set of level parameters $\boldsymbol{x}$). We assume access to a parametrisable simulator with parameter space $\mathbb{X}$, with $X_C \subset \mathbb{X}$. While prior work expects X_C to correspond to all solvable levels in $\mathbb{X}$, we consider the more general setting in which there may be more than one CMDP within $\mathbb{X}$, whereas we aim to solve a specific target CMDP. We refer to levels with parameters $\boldsymbol{x} \in X_C$ as *in-context* and to levels with parameters $\boldsymbol{x} \in \mathbb{X} \setminus X_C$ as *out-of-context*. As we show in our experiments, training on out-of-context levels can induce distributional shift and cause the agent to learn a different policy than the optimal CMDP policy.

Generalisation bounds. We start training with access to a limited set of level parameters $X_{\text{train}} \subset X_C$ sampled from $p(\mathbf{x})$, and evaluate generalisation using a set of held-out level parameters X_{test}, also sampled from $p(\mathbf{x})$. Using the unbiased value estimator lemma from Bertran et al. (2020),

Lemma 3.1 *Given a policy π and a set $L = \{i_{\boldsymbol{x}}|\boldsymbol{x} \sim p(\mathbf{x})\}_n$ of n levels from a CMDP with context distribution $p(\mathbf{x})$, we have $\forall H_t^o$ $(t < \infty)$ compatible with L (that is the observation sequence H_t^o occurs in L), $\mathbb{E}_{L|H_t^o}[V_{i_{\boldsymbol{x}}}^\pi(H_t^o)] = V_C^\pi(H_t^o)$, with $V_{i_{\boldsymbol{x}}}^\pi(H_t^o)$ being the expected returns under π given observation history H_t^o in a given level $i_{\boldsymbol{x}}$, and $V_C^\pi(H_t^o)$ being the expected returns across all possible occurrences of H_t^o in the CMDP.*

we can estimate the generalisation gap using a formulation reminiscent of supervised learning,

$$\text{GenGap}(\pi) := \frac{1}{|X_{\text{train}}|}\sum_{\boldsymbol{x}\in X_{\text{train}}} V_{i_{\boldsymbol{x}}}^\pi - \frac{1}{|X_{\text{test}}|}\sum_{\boldsymbol{x}\in X_{\text{test}}} V_{i_{\boldsymbol{x}}}^\pi. \tag{1}$$

Using this formulation, Bertran et al. (2020) extend generalisation results in the supervised setting (Xu & Raginsky, 2017) to derive an upper bound for the GenGap.

Theorem 3.2 *For any CMDP such that $|V_C^\pi(H_t^o)| \leq D/2, \forall H_t^o, \pi$, then for any set of training levels L, and policy π*

$$\textit{GenGap}(\pi) \leq \sqrt{\frac{2D^2}{|L|} \times \textit{MI}(L, \pi)}. \tag{2}$$

With $\text{MI}(L, \pi) = \sum_{i\in L} \text{MI}(i, \pi)$ being the mutual information between π and the identity of each level $i \in L$. In this work, we show that minimising the bound in Theorem 3.2 is an effective surrogate objective for reducing the GenGap.

Adaptive level sampling. We study the connection between $\mathrm{MI}(L, \pi)$ and adaptive sampling strategies over L. PLR introduce a scoring function $\textbf{score}(\tau_i, \pi)$ compute level scores from a rollout trajectory τ_i. Scores are used to define an adaptive sampling distribution over a level buffer Λ, with

$$P_\Lambda = (1 - \rho) \cdot P_S + \rho \cdot P_R, \tag{3}$$

where P_S is a distribution parametrised by the level scores and ρ is a coefficient mixing P_S with a staleness distribution P_R that promotes levels replayed less recently. Jiang et al. (2021b) experiment with different scoring functions, and empirically find that the scoring function based on the ℓ_1-value loss $S_i^V = \textbf{score}(\tau_i, \pi) = (1/|\tau_i|) \sum_{H_t^o \in \tau_i} |\hat{V}(H_t^o) - V_i^\pi(H_t^o)|$ incurs a significant reduction in the GenGap at test time.

In the remaining sections, we draw novel connections between the ℓ_1-value loss prioritisation strategy and the minimisation of $\mathrm{MI}(L, \pi)$. We then introduce SSED, a level generation and sampling framework training the agent over an augmented set of levels. SSED jointly minimises $\mathrm{MI}(L, \pi)$ while increasing $|L|$ and as such is more effective at minimising the bound from Theorem 3.2.

4 MUTUAL-INFORMATION MINIMISATION UNDER A FIXED SET OF LEVELS

We begin by considering the setting in which L remains fixed. We make the following arguments: 1) as the contribution of each level to $\mathrm{MI}(L, \pi)$ is generally *not uniform* across L *nor constant* over the course of training, an adaptive level sampling strategy yielding training data with low $\mathrm{MI}(L, \pi)$ can reduce the GenGap over uniform sampling; 2) the value prediction objective promotes learning internal representations informative of the current level identity and causes overfitting; 3) deprioritising levels with small value loss implicitly reshapes the training data distribution to yield smaller $\mathrm{MI}(L, \pi)$, reducing GenGap. We substantiate our arguments with a comparison of different sampling strategies in the Procgen benchmark (Cobbe et al., 2020).

4.1 MAINTAINING LOW MUTUAL INFORMATION CONTENT VIA ADAPTIVE SAMPLING

The following lemma enables us to derive an upper bound for $\mathrm{MI}(L, \pi)$ that can be approximated using the activations of the state representation shared between the actor and critic networks.

Lemma 4.1 *(proof in appendix) Given a set of training levels L and an agent model $\pi = f \circ b$, where $b(H_t^o) = h_t$ is an intermediate state representation and f is the policy head, we can bound $MI(L, \pi \circ b)$ by $MI(L, b)$, which in turn satisfies*

$$\begin{aligned} MI(L, \pi) \leq MI(L, b) &= \mathcal{H}(p(\mathrm{i})) + \sum_{i \in L} \int dh p(\mathrm{h}, \mathrm{i}) \log p(\mathrm{i}|\mathrm{h}) && (4) \\ &\approx \mathcal{H}(p(\mathrm{i})) + \frac{1}{|B|} \sum_{(i, H_t^o) \in B} \log p(\mathrm{i}|b(H_t^o)) && (5) \end{aligned}$$

where $\mathcal{H}(p)$ is the entropy of p and B is a batch of trajectories collected from levels $i \in L$.

This result applies to any state representation function b, including the non-recurrent case where $b(H_t^o) = b(o_t), \forall (o, H_t^o) \in (O, O^{\otimes t})$. To remain consistent with the CMDP we must set $p(\mathrm{i})$ to $p(\mathbf{x})$, making the entropy $\mathcal{H}(p(\mathrm{i}))$ a constant. However the second term in Equation (5) depends on the representation b learned under the training data. We hypothesise that minimising $\mathrm{MI}(L, b)$ in the training data is an effective data regularisation technique against instance-overfitting. We can isolate level-specific contributions to $\mathrm{MI}(L, b)$ as

$$\sum_{(i, H_t^o) \in B} \log p(\mathrm{i}|b(H_t^o)) = \sum_{i \in L} \sum_{H_t^o \in B_i} \log p(\mathrm{i}|b(H_t^o)), \tag{6}$$

where B_i indicates the batch trajectories collected from level i. As sampled trajectories depend on the behavioral policy, and as the information being retained depends on b, each level's contribution to $\mathrm{MI}(L, b)$ is in general not constant over the course of training nor uniform across L. There should therefore exist adaptive distributions minimising $\mathrm{MI}(L, b)$ more effectively than uniform sampling.

4.2 ON THE EFFECTIVENESS OF VALUE LOSS PRIORITISATION STRATEGIES

From a representation learning perspective, the value prediction objective may be viewed as a self-supervised auxiliary objective shaping the intermediate state representation b. This additional signal

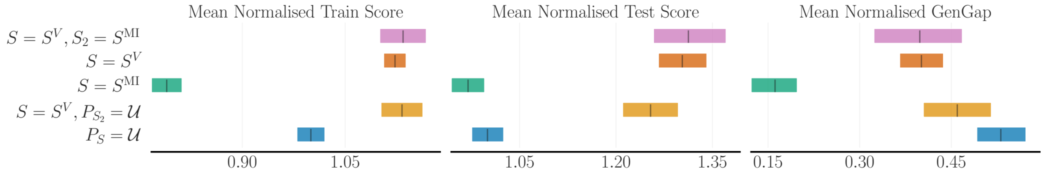


Figure 2: Mean aggregated train and test scores and GenGap of different sampling strategies. Scores are normalised for each game using the mean score achieved by the uniform sampling strategy (we use the test set scores to normalise GenGap).

is often necessary for learning, and motivates sharing b across the policy and value networks. However, in a CMDP the value prediction loss

$$L_V(\boldsymbol{\theta}) = \frac{1}{|B|} \sum_{(i,H_t^o)\in B} (\hat{V}_{\boldsymbol{\theta}}(H_t^o) - V_i^\pi(H_t^o))^2 \tag{7}$$

uses level-specific functions V_i^π as targets, which may be expressed as $V_i^\pi = V_C^\pi + v_i^\pi$, where v_i^π is a component specific to i. While Lemma 3.1 guarantees convergence to an unbiased estimator for V_C^π when minimising Equation (7), reaching zero training loss is only achievable by learning the level specific components v_i^π. Perfect value prediction requires learning an intermediate representation from which the current level i is identifiable, which implies high MI(i, b). Conversely, we can characterise PLR's ℓ_1-value loss sampling as a data regularisation technique minimising MI(L, b) when generating the training data. By de-prioritising levels with low L_{V_i}, PLR prevents the agent from generating training data for which its internal representation has started overfitting to.

4.3 COMPARING MUTUAL INFORMATION MINIMISATION SAMPLING STRATEGIES

We aim to establish how PLR under value loss scoring S^V compares to a scoring function based on Equation (6). We define this function as $S_i^{\text{MI}} = \sum_{t=0}^{T} \log p_{\boldsymbol{\theta}}(\text{i}|b(H_t^o))$, where $p_{\boldsymbol{\theta}}$ is a linear classifier. We also introduce a secondary scoring strategy S_2 with associated distribution P_{S_2}, letting us *mix* different sampling strategies and study their interaction. P_Λ then becomes

$$P_\Lambda = (1-\rho)\cdot((1-\eta)\cdot P_S + \eta \cdot P_{S_2}) + \rho \cdot P_R, \tag{8}$$

with η being a mixing parameter. We compare different sampling strategies in Procgen, a benchmark of 16 games designed to measure generalisation in RL. We train the PPO (Schulman et al., 2017) baseline employed in (Cobbe et al., 2020), which uses a non-recurrent intermediate representation $b_{\boldsymbol{\theta}}(o_t)$ in the "easy" setting ($|L| = 200$, $25M$ timesteps). We report a complete description of the experimental setup in Appendix D.1.

Figure 2, compares value loss scoring ($S = S^V$), uniform sampling ($P_S = \mathcal{U}$), $\mathcal{U}(\cdot)$ being the uniform distribution, direct MI minimisation ($S = S^{\text{MI}}$) as well as mixed strategies ($S = S^V, S_2 = S^{\text{MI}}$) and ($S = S^V, P_{S_2} = \mathcal{U}$). While ($S = S^{\text{MI}}$) reduces GenGap the most, the degradation it induces in the training performance outweigh its regularisation benefits. This result is consistent with Theorem 3.2 and Lemma 4.1, as MI(L, b) bounds the GenGap and not the test returns.[1] On the other-hand, $(S = S^V)$ slightly improves training efficiency while reducing the GenGap. As denoted by its smaller GenGap when compared to ($P_S = \mathcal{U}$), the improvements achieved by $(S = S^V)$ are markedly stronger over the test set than for the train set, and indicate that the main driver behind the stronger generalisation performance is not a higher sample efficiency but a stronger regularisation. We tested different mixed strategies ($S = S^V, S_2 = S^{\text{MI}}$) using different η, and the best performing configuration (reported in Figure 2) only achieves a marginal improvement over $(S = S^V)$ (on the other hand, mixing S^V and uniform sampling ($S = S^V, P_{S_2} = \mathcal{U}$) noticeably reduces the test set performance). This implies that $(S = S^V)$ strikes a good balance between training efficiency and regularisation within the space of mutual information minimisation adaptive sampling strategies. In Appendix B.1 we analyse the correlation between MI(L, b), the ℓ_1-value loss and the GenGap across all procgen games and methods tested. We find MI(L, b) to be positively correlated to the GenGap $(p < 1\text{e}{-34})$ and inversely correlated with the ℓ_1-value loss $(p < 1\text{e}{-16})$.

[1] Exclusively focusing on data regularisation can be problematic: in the most extreme case, destroying all information contained within the training data would guarantee MI(L, π) = GenGap = 0 but it would also make the performance on the train and test sets equally bad.

5 SELF-SUPERVISED ENVIRONMENT DESIGN

We have established that certain adaptive sampling strategies effectively minimise $\mathrm{MI}(L, b)$, which in turn reduces GenGap. However our experiments in Section 4 and appendices B.1 and B.2 indicate GenGap may still be significant when training the agent over a fixed level set, even with an adaptive sampling strategy. We now introduce SSED, a framework designed to more aggressively minimise the generalisation bound in Theorem 3.2 by jointly minimising $\mathrm{MI}(L, b)$ and increasing $|L|$. SSED does so by generating an augmented set of training levels $\tilde{L} \supset L$, while still employing an adaptive sampling strategy over the augmented set.

SSED shares UED's requirement of having access to a parametrisable simulator allowing the specification of levels through environment parameters $\boldsymbol{x}$. In addition, we assume that we start with a limited set of level parameters X_{train} sampled from $p(\mathbf{x})$. SSED consists of two components: a *generative phase*, in an augmented set $\tilde{X}$ is generated using a batch $X \sim \mathcal{U}(X_{\text{train}})$ and added to the buffer Λ, and a *replay phase*, in which we use the adaptive distribution P_Λ to sample levels from Λ. We alternate between the generative and replay phases, and only perform gradient updates on the agent during the replay phase. Algorithm 1 describes the full SSED pipeline, and we provide further details on each phase below.

Algorithm 1 Self-Supervised Environment Design

```
Input: Pre-trained VAE encoder and decoder networks $\psi_{\boldsymbol{\theta}_E}$, $\phi_{\boldsymbol{\theta}_D}$, level parameters $X_{\text{train}}$, number of pairs $M$, number of interpolations per pair $K$
Initialise agent policy $\pi$ and level buffer $\Lambda$, adding level parameters in $X_{\text{train}}$ to $\Lambda$
Update $X_{\text{train}}$ with variational parameters $(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_n \leftarrow \psi_{\boldsymbol{\theta}_E}(\boldsymbol{x}_n)$ for $\boldsymbol{x}_n$ in $X_{\text{train}}$
while not converged do
    Sample batch $X$ using $P_\Lambda$                                          ▷ Replay phase
    for $\boldsymbol{x}$ in $X$ do
        Collect rollouts $\tau$ from $i_{\boldsymbol{x}}$ and compute scores $S, S_2$
        Update $\pi$ according to $\tau$
        Update scores $S, S_2$ of $\boldsymbol{x}$ in $\Lambda$
    Randomly sample $2M$ $(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ from $X_{\text{train}}$ and arrange them into $M$ pairs.
    for $((\boldsymbol{x}, \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_i, (\boldsymbol{x}, \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_j)$ in pairs do        ▷ Generative phase
        Compute $K$ interpolations $\{(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})\}_K$ between $((\boldsymbol{x}, \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_i, (\boldsymbol{x}, \boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_j)$
        for $(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})_k$ in $\{(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})\}_K$ do
            Sample embedding $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}})$
            $\tilde{\boldsymbol{x}} \leftarrow \phi_{\boldsymbol{\theta}_D}(\boldsymbol{z})$
            Collect $\pi$'s trajectory $\tau$ from $\boldsymbol{x}$ and compute scores $S, S_2$
            Add $\langle \boldsymbol{x}, S, S_2 \rangle$ to $\Lambda$ at $\arg\min_{\{\Lambda \setminus X_{\text{train}}\}} S_2$ if $S_2 > \min_{\{\Lambda \setminus X_{\text{train}}\}} S_2$
```

5.1 THE GENERATIVE PHASE

While SSED is not restricted to a particular approach to obtain $\tilde{X}$, we chose the VAE (Kingma & Welling, 2014; Rezende et al., 2014) due its ability to model the underlying training data distribution $p(\mathbf{x})$ as stochastic realisations of a latent distribution $p(\mathbf{z})$ via a generative model $p(\mathbf{x} \mid \mathbf{z})$. The model is pre-trained on X_{train} by maximising the variational ELBO

$$\mathcal{L}_{\text{ELBO}} = \mathop{\mathbb{E}}_{\boldsymbol{x} \sim p(\mathbf{x})} \left\{ \mathop{\mathbb{E}}_{\boldsymbol{z} \sim q(\mathbf{z} \mid \mathbf{x}; \psi_{\boldsymbol{\theta}_E})} [\log p(\mathbf{x} \mid \mathbf{z}; \phi_{\boldsymbol{\theta}_D})] - \beta D_{\text{KL}}(q(\mathbf{z} \mid \mathbf{x}; \psi_{\boldsymbol{\theta}_E}) \mid\mid p(\mathbf{z})) \right\}, \tag{9}$$

where $q(\mathbf{z} \mid \mathbf{x}; \psi_{\boldsymbol{\theta}_E})$ is a variational approximation of an intractable model posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ and $D_{\text{KL}}(\cdot \mid\mid \cdot)$ denotes the Kullback–Leibler divergence, which is balanced using the coefficient β, as proposed by Higgins et al. (2017). The generative $p(\mathbf{x} \mid \mathbf{z}; \phi_{\boldsymbol{\theta}_D})$ and variational $q(\mathbf{z} \mid \mathbf{x}; \psi_{\boldsymbol{\theta}_E})$ models are parametrised via encoder and decoder networks $\psi_{\boldsymbol{\theta}_E}$ and $\phi_{\boldsymbol{\theta}_D}$.

We use $p(\mathbf{x}; \phi_{\boldsymbol{\theta}_D})$ to generate augmented level parameters $\tilde{\boldsymbol{x}}$. As maximising Equation (9) fits the VAE such that the marginal $p(\mathbf{x}; \phi_{\boldsymbol{\theta}_D}) = \int p(\mathbf{x} \mid \mathbf{z}; \phi_{\boldsymbol{\theta}_D}) p(\mathbf{z}) \, \mathrm{d}\mathbf{z}$ approximates the data distribution $p(\mathbf{x})$, and sampling from it limits distributional shift. This makes out-of-context levels less frequent, and we show in Section 6 that this aspect is key in enabling SSED-trained agents to outperform UED-agents. To improve the quality of the generated $\tilde{\boldsymbol{x}}$, we interpolate in the latent space between the latent representations of pair of samples $(\boldsymbol{x}_i, \boldsymbol{x}_j) \sim X_{\text{train}}$ to obtain $\mathbf{z}$, instead of sampling from $p(\mathbf{z})$, as proposed by White (2016). We evaluate the agent (without updating its weights) on the levels obtained from a batch of levels parameters $\tilde{X}$, adding to the buffer Λ any level scoring higher than the lowest scoring generated level in Λ. We provide additional details on the architecture, hyperparameters and pre-training process in Appendix D.3.

5.2 THE REPLAY PHASE

All levels in X_{train} originate from $p(\mathbf{x})$ and are in-context, whereas generated levels, being obtained from an approximation of $p(\mathbf{x})$, do not benefit from as strong of a guarantee. As training on out-of-context levels can significantly harm the agents' performance on the CMDP, we control the ratio between X_{train} and augmented levels using Equation (8) to define P_Λ. P_S and P_R only sample from X_{train} levels, whereas P_{S_2} supports the entire buffer. We set both S_1 and S_2 to score levels according to the ℓ_1-value loss. We initialise the buffer Λ to contain X_{train} levels and gradually add generative phase levels over the course of training. A level gets added if Λ is not full or if it scores higher than the lowest scoring level in the buffer. We only consider levels solved at least once during generative phase rollouts to ensure unsolvable levels do not get added in. We find out-of-context levels to be particularly harmful in the early stages of training, and reduce their frequency early on by linearly increasing the mixing parameter η from 0 to 1 over the course of training.

6 EXPERIMENTS

As it only permits level instantiation via providing a random seed, Procgen's level generation process is both uncontrollable and unobservable. Considering the seed space to be the level parameter space $\mathbb{X}$ makes the level generation problem trivial as it is only possible to define a single CMDP with $X_C = \mathbb{X}$ and $p(\mathbf{x}) = \mathcal{U}(\mathbb{X})$. Instead, we wish to demonstrate SSED's capability in settings where X_C spans a (non-trivial) manifold in $\mathbb{X}$, i.e. only specific parameter semantics will yield levels of the CMDP of interest. As such we pick Minigrid, a partially observable gridworld navigation domain (Chevalier-Boisvert et al., 2018). Minigrid levels can be instantiated via a parameter vector describing the locations, starting states and appearance of the objects in the grid. Despite its simplicity, Minigrid qualifies as a parametrisable simulator capable of instantiating multiple CMDPs. We define the context space of our target CMDP as spanning the layouts where the location of green "moss" tiles and orange "lava" tiles are respectively positively and negatively correlated to their distance to the goal location. We employ procedural generation to obtain a set X_{train} of 512 level parameters, referring the reader to Figure 10 for a visualisation of levels from X_{train}, and to Appendix C for extended details on the CMDP specification and level set generation process.

As the agent only observes its immediate surroundings and does not know the goal location a priori, the optimal CMDP policy is one that exploits the semantics shared by all levels in the CMDP, exploring first areas with high perceived moss density and avoiding areas with high lava density. Our CMDP coexists alongside a multitude of other potential CMDPs in the level space and some correspond to incompatible optimal policies (for example levels in which the correlation of moss and lava tiles with the goal is reversed). As such, it is important to maintain consistency with the CMDP semantics when generating new levels.

We compare SSED to multiple baselines, sorted in two sets. The first set of baselines is restricted to sample from X_{train}, and consists of uniform sampling ($\mathcal{U}$) and PLR with the ℓ_1-value loss strategy. The second set incorporates a generative mechanism and as such are more similar to SSED. We consider domain randomisation (DR) (Tobin et al., 2017) which generates levels by sampling uniformly between pre-determined ranges of parameters, RPLR (Jiang et al., 2021a), which combines PLR with DR used as its generator, and the current UED state-of-the-art, ACCEL Parker-Holder et al. (2022), an extension of RPLR replacing DR by a generator making local edits to currently high scoring levels in the buffer. All experiments share the same PPO (Schulman et al., 2017) agent, which uses the LSTM-based architecture and hyperparameters reported in Parker-Holder et al. (2022), training over 27k updates.

6.1 GENERALISATION PERFORMANCE

As shown in Figure 3, SSED achieves statistically significant improvements in its IQM (inter-quantile mean), mean score, optimality gap and mean solved rate over other methods on held-out levels from the CMDP. SSED's ZSG performance on held-out levels from X_{train} demonstrates it al-

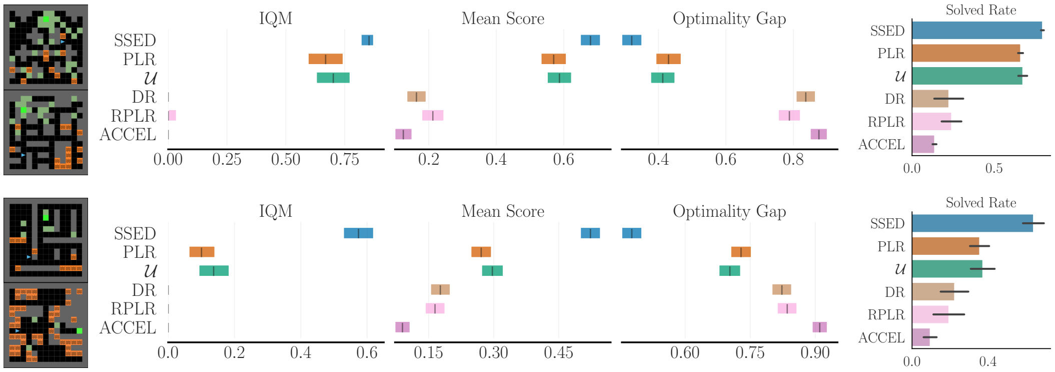


Figure 3: Center: aggregate test performance on 200 held-out levels from X_{train} (top) and in-context edge cases (bottom). Right: zero-shot solved rate on the same levels, the bars indicate standard error for 3 training seeds (some example levels are provided for reference, refer to Appendix C for additional details on our evaluation sets).

leviates instance-overfitting while remaining consistent with the target CMDP. This is thanks to its generative model effectively approximating $p(\mathbf{x})$, and to its mixed sampling strategy ensuring many training levels originate from X_{train}, which are guaranteed to be in-context. We next investigate whether SSED's level generation improves robustness to *edge cases* which are in-context but would have a near zero likelihood of being in X_{train} in a realistic setting. We model edge cases as levels matching the CMDP semantics but generated via different procedural generation parameters. We find SSED to be particularly dominant is this setting, achieving a solved rate and IQM respectively two- and four-times $\mathcal{U}$, the next best method, and a mean score 1.6 times PLR, the next best method for that metric. SSED is therefore capable of introducing additional diversity in the level set in a manner that remains semantically consistent with the CMDP. In Figure 4, we measure transfer to levels of increased complexity using a set of layouts 9 times larger in area than X_{train} levels and which would be impossible to instantiate during training. We find that SSED performs over twice as well as the next best method in this setting.

To better understand the importance of using a VAE as a generative model we introduce SSED-EL, a version of SSED replacing the VAE with ACCEL's level editing strategy. SSED-EL may be viewed as an SSED variant of ACCEL augmenting X_{train} using a non-parametric generative method, or equivalently as an ablation of SSED that does not approximate $p(\mathbf{x})$ and is therefore less grounded to the target CMDP. In Figure 4, we compare the two methods across level sets, and find that SSED improves more significantly over its ablation for level sets that are most similar to the original training set. This highlights the significance of being able to approximate $p(\mathbf{x})$ through the VAE to avoid distributional shift, which we discuss next.

6.2 DISTRIBUTIONAL SHIFT AND THE OVER-GENERALISATION GAP

Despite poor test scores, the UED baselines achieve small GenGap (as shown in Figure 8), as they perform poorly on both the test set and on X_{train}. Yet the fact they tend to perform well on the subset of $\mathbb{X}$ spanning their own training distribution means that they have over-generalised to an out-of-context set. As such, we cannot qualify their poor performance on X_C as a lack of capability but instead as a form of misgeneralisation not quantifiable by the GenGap, and which are reminiscent of goal misgeneralisation failure modes reported in Di Langosco et al. (2022); Shah et al. (2022). Instead, we propose the *over-generalisation gap* as a complementary metric, which we define as

$$\text{OverGap}(\pi) := \sum_{\tilde{\boldsymbol{x}} \in \Lambda} P_\Lambda(i_{\tilde{\boldsymbol{x}}}) \cdot V^{\pi}_{i_{\tilde{\boldsymbol{x}}}} - \frac{1}{|X_{\text{train}}|} \sum_{\boldsymbol{x} \in X_{\text{train}}} V^{\pi}_{i_{\boldsymbol{x}}}. \tag{10}$$

Note that OverGap compares the agent's performance with X_{train} and as such is designed to measure over-generalisation induced by distributional shift.[2] Based on further analysis conducted in Appendix B.3, high OverGap coincides with the inclusion of out-of-context levels coupled with a significant shift in the level parameter distribution with respect to $p(\mathbf{x})$, and we find that SSED is the only level generation method tested able to maintain both low distributional shift and OverGap.

[2] using X_{test} would make OverGap $\equiv$ GenGap if $P_\Lambda = \mathcal{U}(X_{\text{train}})$, whereas it should be 0.

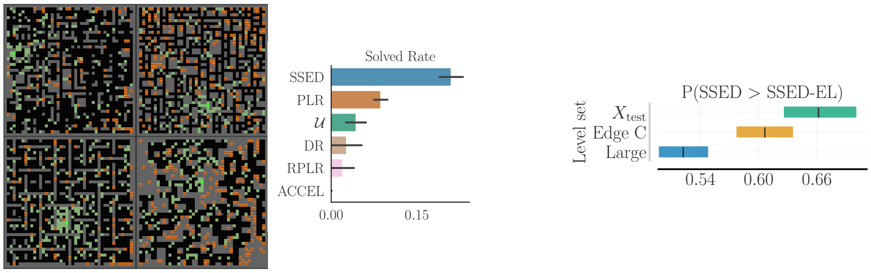


Figure 4: Left: zero-shot solved rate on a set of 100 levels with larger layouts. SSED's success rate is over twice PLR's, the next best performing method. Right: probability ($p < 0.05$) of SSED achieving higher zero-shot returns than its ablation SSED-EL, evaluated in each level set.

7 CONCLUSION

In this work, we investigated the impact of the level sampling process on the ZSG capabilities of RL agents. We found adaptive sampling strategies are best understood as data regularisation techniques minimising the mutual information between the agent's internal representation and the identity of training levels. In doing so, these methods minimise an upper bound on the generalisation gap, and our experiments showed it to act as an effective proxy for reducing this gap in practice. This theoretical framing allowed us to understand the mechanisms behind the improved generalisation achieved by value loss prioritised level sampling, which had only been justified empirically in prior work. We then investigated the setting in which the set of training levels is not fixed and where the generalisation bound can be minimised by training over an augmented set. We proposed SSED, a level generation and sampling framework restricting level generation to an approximation of the underlying distribution of a starting set of level parameters. We showed that this restriction lets SSED mitigates the distributional shift induced by UED methods. By jointly minimising the generalisation and over-generalisation gaps, we demonstrated that SSED achieves strong generalisation performance on in-distribution test levels, while also being robust to in-context edge-cases.

In future work, we plan to investigate how SSED scales to more complex environments. In a practical setting, the level parameter space is often high dimensional, and levels are described by highly structured data corresponding to specific regions of the parameter space. Depending on the simulator used, level parameters may consist of sets of values, configuration files or any other modality specific to the simulator. For example, they could be 3D scans of indoor environments (Li et al., 2021) or a vector map describing a city's road infrastructure (Wilson et al., 2021), which are often costly to collect or prescribe manually, and thus are limited in supply. Augmenting the number of training environments is therefore likely to play a role in scaling up RL in a cost effective manner. Our experiments show that unsupervised environment generation is problematic even in gridworlds, whereas the SSED framework is designed to scale with the amount of data being provided.

Lastly, we are interested in further exploring the synergies between SSED and mutual information minimisation frameworks. SSED performs data augmentation uptream of level sampling, whereas Jiang et al. (2021b) report significant improvements in combining PLR with data augmentation on the agent's observations (Raileanu et al., 2021) and thus acting downstream of level sampling. There may be further synergies in combining mutual information minimisation techniques at different points of the level-to-agent information chain, which is an investigation we leave for future work.

8 REPRODUCIBILITY STATEMENT

We believe parametrisable simulators are better suited to benchmark ZSG than procedural environments, as they provide a fine degree of control over the environment and are more consistent with a realistic application setting, as argued by Kirk et al. (2023). For example, our study of over-generalisation would not have been possible in Procgen, due to each game supporting a singular and non-parametrisable level distribution. Having access to the level parameters used in experiments also facilitates the reproducibility of ZSG research, and we make the train and evaluation sets of level parameters used in this work, as well as the code for running experiments, publicly available.[3] To encourage this practice in future work, we open-source our code[3] for specifying arbitrary CMDPs

[3] Available upon de-anonymisation of this publication

in Minigrid and generate their associated level sets, describing the generation process in detail in Appendix C. We also provide a dataset of 1.5M procedurally generated minigrid base layouts to facilitate level set generation.[3]

Proofs and derivations are included in Appendix A and additional implementation details, including the hyperparameters used and searched over are included in Appendix D.

REFERENCES

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in Neural Information Processing Systems*, 2021.

Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1(8):356–363, 8 2019. ISSN 2522-5839. doi: 10.1038/s42256-019-0070-z. URL `https://doi.org/10.1038/s42256-019-0070-z`.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Abdus Salam Azad, Izzeddin Gur, Jasper Emhoff, Nathaniel Alexis, Aleksandra Faust, Pieter Abbeel, and Ion Stoica. Clutr: Curriculum learning via unsupervised task representation learning. In *International Conference on Machine Learning*, pp. 1361–1395. PMLR, 2023.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents (extended abstract). In *IJCAI*, 2015.

Martin Bertran, Natalia Martinez, Mariano Phielipp, and Guillermo Sapiro. Instance based generalization in reinforcement learning. *CoRR*, abs/2011.01089, 2020. URL `https://arxiv.org/abs/2011.01089`.

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

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1282–1289. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/cobbe19a.html`.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. Emergent complexity and zero-shot transfer via unsupervised environment design. *NIPS*, 2020.

Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal misgeneralization in deep reinforcement learning. In *International Conference on Machine Learning*, pp. 12004–12019. PMLR, 2022.

Mhairi Dunion, Trevor McInroe, Kevin Sebastian Luck, Josiah P. Hanna, and Stefano V. Albrecht. Conditional mutual information for disentangled representations in reinforcement learning, 2023.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why generalization in RL is difficult: Epistemic pomdps and implicit partial observability. *CoRR*, abs/2107.06277, 2021. URL `https://arxiv.org/abs/2107.06277`.

M. Gumin. Wave function collapse algorithm. `https://github.com/mxgmn/`, 2016.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Neural Information Processing Systems*, 2019. URL `https://api.semanticscholar.org/CorpusID:202778414`.

Nick Jakobi. Evolutionary robotics and the radical envelope-of-noise hypothesis. *Adaptive Behavior*, 6:325 – 368, 1997.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Replay-guided adversarial environment design. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 1884–1897. Curran Associates, Inc., 2021a. URL `https://proceedings.neurips.cc/paper/2021/file/0e915db6326b6fb6a3c56546980a8c93-Paper.pdf`.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. *ArXiv*, abs/2010.03934, 2021b.

Minqi Jiang, Michael Dennis, Jack Parker-Holder, Andrei Lupu, Heinrich Küttler, Edward Grefenstette, Tim Rocktäschel, and Jakob Foerster. Grounding aleatoric uncertainty in unsupervised environment design. *arXiv preprint arXiv:2207.05219*, 2022.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *ICLR*, 2014.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of zero-shot generalisation in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 76:201–264, 2023.

Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *ArXiv*, abs/2004.13649, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

J. Lee, J. Hwango, et al. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 2020.

Z. Li, T. Yu, S. Sang, S. Wang, M. Song, Y. Liu, Y. Yeh, R. Zhu, N. Gundavarapu, J. Shi, S. Bi, H. Yu, Z. Xu, K. Sunkavalli, M. Hasan, R. Ramamoorthi, and M. Chandraker. Openrooms: An open framework for photorealistic indoor scene datasets. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7186–7195, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society. doi: 10.1109/CVPR46437.2021.00711. URL `https://doi.ieeecomputersociety.org/10.1109/CVPR46437.2021.00711`.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2016.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktaschel. Evolving curricula with regret-based environment design. *ArXiv*, abs/2203.01302, 2022.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Kumar Gupta. Robust adversarial reinforcement learning. In *ICML*, 2017.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. *Advances in Neural Information Processing Systems*, 34:5402–5415, 2021.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In Eric P. Xing and Tony Jebara (eds.), *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 1278–1286, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/rezende14.html.

Nikita Rudin, David Hoeller, Philipp Reist, and Marco Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning. *arXiv preprint arXiv:2109.11978*, 2021.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2015. URL https://api.semanticscholar.org/CorpusID:13022595.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 2017.

Rohin Shah, Vikrant Varma, Ramana Kumar, Mary Phuong, Victoria Krakovna, Jonathan Uesato, and Zac Kenton. Goal misgeneralization: Why correct specifications aren't enough for correct goals. *arXiv preprint arXiv:2210.01790*, 2022.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 23–30. IEEE, 2017.

Tom White. Sampling generative networks. *arXiv preprint arXiv:1609.04468*, 2016.

Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, Deva Ramanan, Peter Carr, and James Hays. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf.

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. *Advances in Neural Information Processing Systems*, 30, 2017.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. Automatic curriculum learning through value disagreement. *Advances in Neural Information Processing Systems*, 33:7648–7659, 2020.

A THEORETICAL RESULTS

Lemma A.1 *(proof in appendix) Given a set of training levels L and an agent model $\pi = f \circ b$, where $b(H_t^o) = h_t$ is an intermediate state representation and f is the policy head, we can bound MI$(L, \pi \circ b)$ by MI(L, b), which in turn satisfies*

$$MI(L, \pi) \leq MI(L, b) = \mathcal{H}(p(\mathrm{i})) + \sum_{i \in L} \int dh p(\mathrm{h}, \mathrm{i}) \log p(\mathrm{i}|\mathrm{h}) \tag{11}$$

$$\approx \mathcal{H}(p(\mathrm{i})) + \frac{1}{|B|} \sum_{(i, H_t^o) \in B} \log p(\mathrm{i}|b(H_t^o)) \tag{12}$$

where $\mathcal{H}(p)$ is the entropy of distribution p and B is a batch of trajectories collected from individual levels $i \in L$.

proof:

Given that the information chain of our model follows $H_t^o \rightarrow b \rightarrow f$, we have MI$(L, f \circ b) \leq$ MI(L, b) following the data processing inequality. MI(L, b) can then be manipulated as follows

$$MI(L, b) = \sum_{i \in L} \int d\boldsymbol{h} p(\mathbf{h}, i) \log \frac{p(\mathbf{h}, i)}{p(\mathbf{h}) p(\mathrm{i})} \tag{13}$$

$$= - \sum_{i \in L} \int d\mathbf{h} p(\mathbf{h}, \mathrm{i}) \log p(i) + \sum_{i \in L} \int d\boldsymbol{h} p(\mathbf{h}, i) \log p(\mathrm{i}|\mathbf{h}) \tag{14}$$

$$\approx \mathcal{H}(p(\mathrm{i})) + \frac{1}{|B|} \sum_{n}^{|B|} \log p(i^{(n)}|\boldsymbol{h}^{(n)}) \tag{15}$$

where in eq. (15) *we approximate $p(\boldsymbol{h}, i)$ as the empirical distribution*

$$\tilde{p}(\boldsymbol{h}, i) = \begin{cases} \frac{1}{|B|} & \textit{if } (H_t^o, i) \in B, \textit{ with } \boldsymbol{h} = b(H_t^o) \\ 0 & \textit{otherwise.} \end{cases} \tag{16}$$

Note that we can treat non-recurrent architectures as a particular case, setting $H_t^o = o_t$ without loss of generality.

B ADDITIONAL EXPERIMENTAL RESULTS

B.1 PROCGEN ADDITIONAL EXPERIMENTAL RESULTS

From Equation (5), we estimate MI(L, b) modelling $p_{\boldsymbol{\theta}}$ as a linear classifier. We plot this estimate against the GenGap and the ℓ_1 value loss for all methods tested and across Procgen games in Figure 5. As expected under our theoretical framework, we measure a positive correlation between MI(L, b) and GenGap with Kendall rank correlation coefficient $\tau = 0.53$ ($p <$ 1e-34), and a negative correlation with the ℓ_1 value loss with Kendall rank correlation coefficient $\tau = -0.28$ ($p <$ 1e-16).

In order to provide a more intuitive quantification of our mutual information estimates, we consider the classification accuracy of the linear classifier used to compute our estimate for MI(L, b), as these two quantities are proportional with each other. Out of 200 training levels, the classifier correctly predicts the current level 49% of the times under uniform sampling, 34% under ($S = S^V$) and 23% under S^{MI}. Adaptive sampling strategies are therefore able to reduce ($S = S^{\text{MI}}$) across ProcGen games, and ranking different methods according to their level classification accuracy will also sort them according to their respective GenGap. To understand how likely it is for a given sampling strategy to improve over another, we report the probability of improvement in test scores and GenGap for different pairs of strategies in Figure 6.

Nevertheless, the mean classifier accuracy remains 68 times random guessing for ($S = S^V$) and 46 times random guessing for ($S = S^{\text{MI}}$). As the classifier makes a prediction using an internal

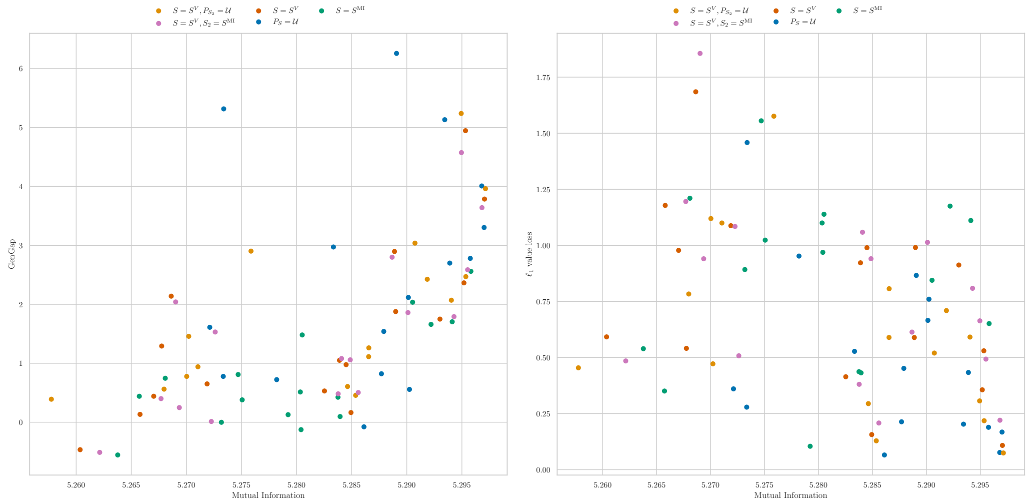


Figure 5: Scatter plot displaying the relationship between $\mathrm{MI}(L, b)$ and the (unormalised) GenGap (left) and with the ℓ_1 average value loss (right), measured across all methods and Procgen games at the end of training. Each point represents 5 seeds of a level sampling method in a particular game.

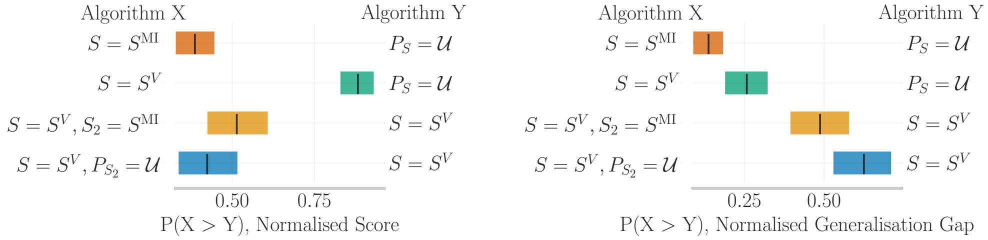


Figure 6: Probability of algorithm X incurring a higher normalised test score (left) and GenGap (right) than algorithm Y. Evaluation performed over 5 seeds across all Procgen games, using the rliable library(Agarwal et al., 2021). Colored bars indicate the 95% confidence interval.

representation obtained from a single observation we find these results surprising, and demonstrate adaptive sampling strategies can only reduce $\mathrm{MI}(L, b)$ up to a point. To further reduce $\mathrm{MI}(L, b)$, adaptive sampling should be combined with other data regularisation techniques, such as the level augmentation technique proposed by SSED and/or additional data augmentation techniques. Indeed Jiang et al. (2021b) report a significant improvement in test scores when combining PLR with UCB-DrAC Raileanu et al. (2021), an observation augmentation method.

B.2 COMPARING THE EFFECTIVENESS OF ADAPTIVE SAMPLING STRATEGIES ACROSS PROCGEN GAMES

We observe that both the classifier accuracy under uniform level sampling and the potential improvement induced by adaptive sampling is highly dependent on the procgen game tested. To better understand why, we compare the measured accuracy with a qualitative analysis of the observations and levels encountered in the Maze and Bigfish games, which we provide a sample of in Figure 7.

In Maze, the accuracy remains over 80% (160× random) for all methods tested and the reduction is GenGap insignificant. On the other hand, in Bigfish all adaptive sampling strategies tested lead to a significant reduction in classifier accuracy, dropping from 80% to under 20%, and they are associated with a significant drop in GenGap and improvement in test scores.

In Maze, the observation space is set up such that the agent observes the full layout at each timestep. The layout is unique to each level and provides many features for identification that are straightfor-

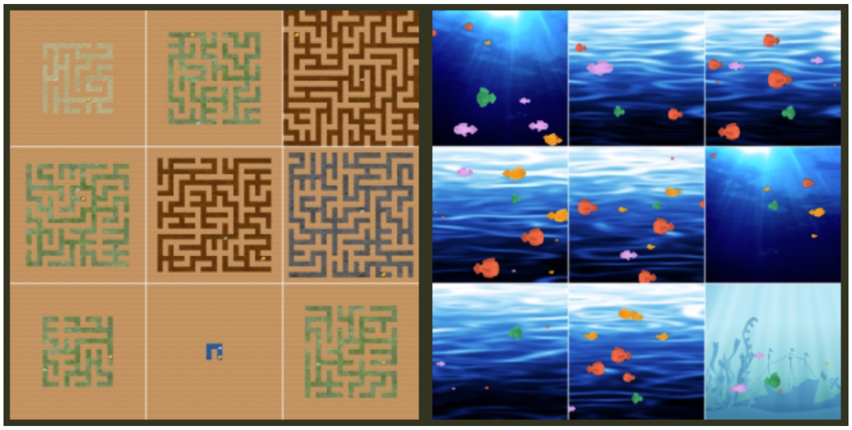


Figure 7: Agent observations sampled from 9 levels from the Maze (left) and Bigfish (right) games of the Procgen benchmark.

ward to learn by the agent's ResNet architecture. In addition, these features cannot be ignored by the agent model in order to solve the task. Intuitively, we can hypothesise that adaptive sampling strategies will not be effective if all the levels are easily identifiable by the agent, which appears to be the case in Maze. In these cases, other data regularisation techniques, such as augmenting the observations, can be more effective, and in fact Jiang et al. (2021b) report that Maze is one of the games where combining PLR with UCB-DrAC leads to a significant improvement in test scores.

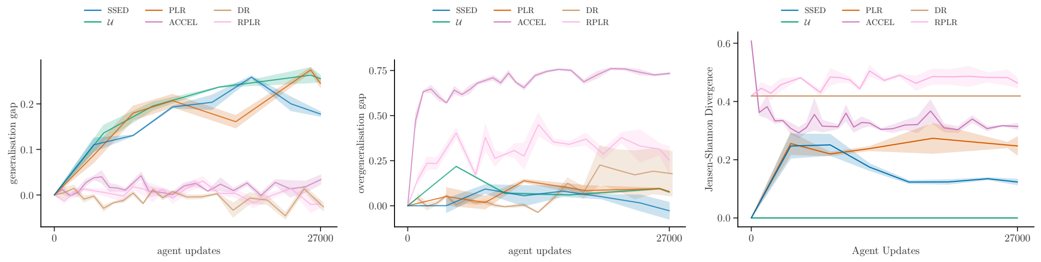
On the other hand, we observe that many of the Bigfish levels yield similar observations. Indeed, both the features relevant to the task (the fish) and irrelevant (the background) are similar in many of the training levels. Furthermore, there's significant variation in the observations encountered during an episode, as fish constantly appear and leave the screen. Yet, some levels (top left, middle and bottom right) are easily identifiable thanks to their background, and we can hypothesise that adaptive sampling strategies will tend to de-prioritise them more often, essentially performing data regularisation via a form of rejection sampling.

B.3 QUANTIFYING THE DISTRIBUTIONAL SHIFT IN MINIGRID

In Figure 8, we report the generalisation and over-generalisation gaps in the Minigrid experiments. We observe that UED methods tend to exhibit lower generalisation gaps than SSED, PLR or uniform sampling. We find that introducing an additional metric in the form of the OverGap Equation (10) necessary to quantify this form of misgeneralisation. We next study the correlation between the OverGap and distributional shifts between the underlying CMDP level parameter distribution $p(\mathbf{x})$ and $p_\Lambda(\mathbf{x})$, the distribution of level parameters existing in the level replay buffer Λ.

We approximate $p(\mathbf{x})$ as $\tilde{p}(\mathbf{x}) \approx \mathcal{U}(X_\text{train})$ and we use the Jensen-Shannon Divergence (JSD) between $\tilde{p}(\mathbf{x})$ and $p_\Lambda(\mathbf{x})$, defining the JSD in a space consistent with the CMDP semantics. To do so, we measure the distribution of distances between the goal location and each other tile type. The JSD is therefore expressed as $\text{JSD}(c_p || c_q)$, where $c(t, d|\mathbf{x})$ is the categorical distribution measuring the probability of tile type t occurring at distance d from the goal location in a given level $\boldsymbol{x}$ and c_p is the marginal $c_p = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[c(t, d|\mathbf{x})]$, setting $p_\Lambda(\mathbf{x})$ and $q = \mathcal{U}(X_\text{train})$.

We report how the JSD evolves over the course of training for different methods in Figure 8c. We observe that distributional shift occurs early on during training and remains relatively stable afterwards in all methods. JSD and OverGap tend to be positively correlated for most methods, except for DR and PLR, which both present high JSD but low OverGap. SSED is the only generative method to maintain a low JSD throughout trainingIn fig. 9, we report additional metrics on the levels sampled by each method. We find that SSED tends to be as proficient as PLR in maintaining consistency with X_train with the occurence of different tile types, as well as higher order task-relevant properties such as the shortest path length between the start and goal locations.



(a) (b) (c)

Figure 8: Generalisation gap ((a), Equation (1)) and over-generalisation gap ((b), Equation (10)) of different methods during training. Fixed set sampling strategies experience higher generalisation gap, while UED methods are dominated by the over-generalisation gap. SSED tends to follow a similar profile as fixed set sampling methods, with a moderate generalisation gap and a low (and even at times negative) over-generalisation gap. SSED also exhibit small distributional shift in its level parameters, as demonstrated by the evolution of the JSD over the course of training (c). Surprisingly, SSED demonstrates a smaller divergence than PLR, even when PLR only has access to X_{train} and as such can only affect the JSD by changing the prioritisation of individual levels $i_{\boldsymbol{x}}$, $\boldsymbol{x} \in X_{\text{train}}$.

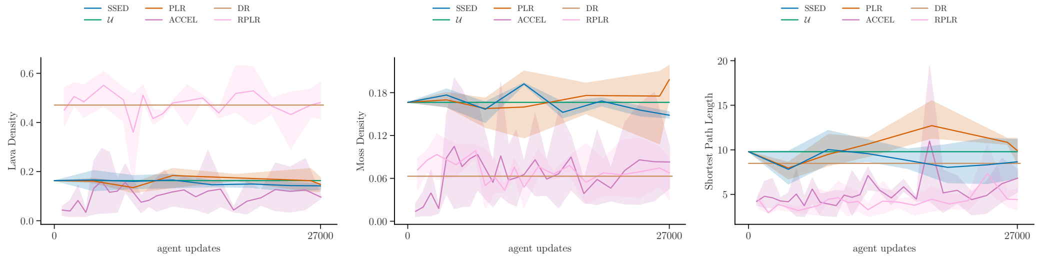


Figure 9: Left and middle: evolution of lava and moss tile densities encountered in sampled levels over the course of training. Right: evolution of the shortest path length between the start and goal location in sampled levels over the course of training.

C CMDP SPECIFICATION AND THE LEVEL GENERATION PROCESS

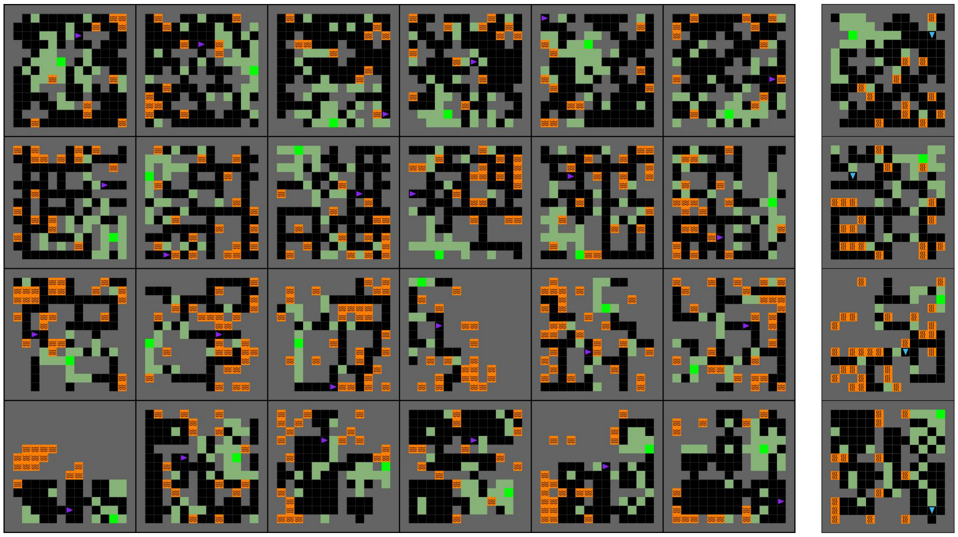
In Minigrid Chevalier-Boisvert et al. (2018), the agent receives as an observation a partial view of its surroundings (in our experiments it is set to two tiles to each side of the agent and four tiles in front) and a one-hot vector representing the agent's heading. The action space consists of 7 discrete actions, however in our setting only the actions moving the agent forward and rotating it to the left or right have an effect on the environment. The episode starts with the agent at the start tile and facing a random direction. The episode terminates successfully when the agent reaches the goal tile and receives a reward between 0 and 1 based on the number of timesteps it took to get there. The episode will terminate without a reward if the agent steps on a lava tile, or when the maximum number of timesteps is reached.

Levels are parameterised as 2D grids representing the overall layout, with each tile type represented by an unique ID. Tiles can be classified as navigable (for example, moss or empty tiles) or non-navigable (for example, walls and lava, as stepping into lava terminates the episode). To be valid, a level must possess exactly one goal and start tile, and to be solvable there must exist a navigable path between the start and the goal location. We provide the color palette of tiles used in Figure 12.

In this work, we define and train within the "Cave Escape" CMDP, which corresponds to a subset of the solvable levels in which moss and lava node placement is respectively positively and negatively correlated with the geodesic distance to the goal.[4] Under partial observability, the minimax regret

[4] To measure the distance-to-goal of a non-navigable node, we first find the navigable node that is closest from it and measure its geodesic distance to the goal. We then add to it the distance between this navigable

policy for this CMDP would leverage moss and lava locations as context cues, seeking regions with perceived higher moss density avoiding regions with perceived high lava density, and thus the CMDP possesses an attributable goal and optimal behavior. We provide example levels of the CMDP in Figure 10.



(a) Example training levels from X_{train}

(b) Held-out levels

Figure 10: Sample levels from X_{train} and from held-out test levels. Wall tiles are rendered in gray, empty tiles in black, moss tiles in green and the goal tile in lime green. The agent is rendered as a blue or purple triangle, and is depicted at its start location. Each row corresponds to levels generated with a specific wave function collapse base pattern. Four different base patterns are used in X_{train}.

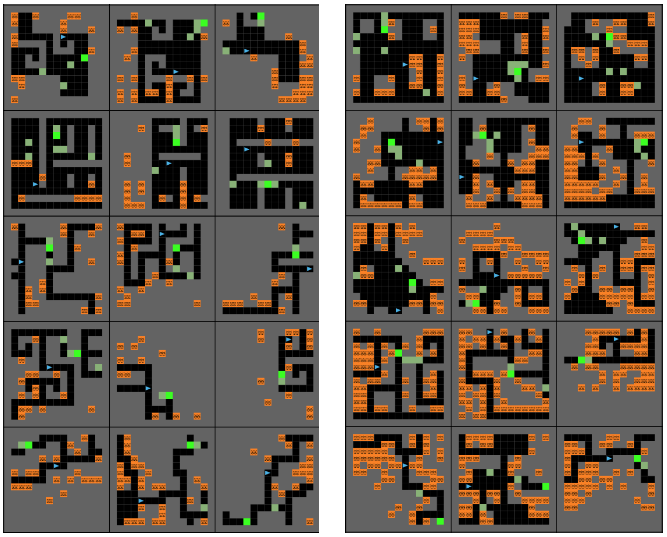
C.1 Generating Highly Structured Levels

We employ the wave function collapse (WFC) algorithm (Gumin, 2016) as our procedural generation algorithm to obtain highly structured but still diverse gridworld layouts. WFC takes as an input a basic pattern and gradually collapses a superposition of all possible level parameters into a layout respecting the constraints defined by the input pattern. By doing so, it is possible to generate a vast number of tasks from a small number of starting patterns. Given suitable base patterns the obtained layouts provide a high degree of structure, and guarantee that both task structure and diversity scale with the gridworld dimensions. We provide 22 different base patterns and allow for custom ones to be defined. After generating a layout using WFC, we convert the navigable nodes of a layout into a graph, choose its largest connected component as the layout and convert any unreachable nodes to non-navigable nodes. We place the goal location at random and place the start at a node located at the median geodesic distance from the goal in the navigation graph. By doing so we ensure that the complexity of generated layouts is relatively consistent given a specific grid size and base pattern. Finally we sample tiles according to parameterisable distributions defined over the navigable or non-navigable node sets. In this work the tile set consists of the $\{$ moss, empty, start, goal $\}$ tiles as the navigable set and the $\{$wall, lava$\}$ tiles as the non-navigable set, and we define distributions for the moss and lava node types over the navigable and non-navigable node sets, respectively.

C.2 Controlling Level Complexity

We provide two options to vary the complexity of the level distribution. The first is to change the gridworld size, which directly results in an increase in complexity. The second, which is specific to the Cave Escape CMDP, is to change the sampling probability of moss and lava nodes. Since the

node and the non-navigable node of interest. If there are multiple equally close navigable nodes, we select the navigable node with the smallest geodesic distance to the goal.



(a) Example levels from the first set of edge cases

(b) Example levels from the second set of edge cases

Figure 11: We generate 2 separate sets of edge cases, using 14 different base patterns not used to generate X_{train}. The moss density of the first set (a) is 3 times as small as in X_{train}, making finding the goal using CMDP contextual cues more challenging. (b) is the same as (a) but also has the lava density multiplied by a factor of 3, making it more difficult to avoid walking into lava and failing the episode. Both sets are combined when evaluating the agent on edge cases.

environment is partially observable, reducing the fraction of moss to navigable nodes, or increasing their entropy diminishes the usefulness of moss tiles as context cues. On the other hand, increasing the density of lava tiles increases the risk associated with selecting the wrong action during play. Thus it is straightforward assess the agent's performance on edge-cases by defining level sets with a larger layout size, or with shifted moss and lava tile distributions.

D IMPLEMENTATION DETAILS

D.1 PROCGEN

The Procgen Benchmark is a set of 16 diverse PCG environments that echo the gameplay variety seen in the ALE benchmark Bellemare et al. (2015). The game levels, determined by a random seed, can differ in visual design, navigational structure, and the starting locations of entities. All Procgen environments use a common discrete 15-dimensional action space and generate $64 \times 64 \times 3$ RGB observations. A detailed explanation of each of the 16 environments is given by Cobbe et al. (2020). Leading RL algorithms such as PPO reveal significant differences between test and training performance in all games, making Procgen a valuable tool for evaluating generalisation performance.

We conduct our experiment on the easy setting of Procgen, which employs 200 training levels and a budget of 25M training steps, and evaluate the agent's ZSG performance on the full range of levels, excluding the training levels. We calculate normalised test returns using the formula $\frac{(R - R_{\text{min}})}{(R_{\text{max}} - R_{\text{min}})}$, where R is the non normalised return and R_{min} and R_{max} are the minimum and maximum returns for each game as provided in (Cobbe et al., 2020).

We employ the same ResNet policy architecture and PPO hyperparameters used across all games as Cobbe et al. (2020), which we reference in Table 1. To compute the MI based scoring strategy S^{MI} used in our experiments, we parametrise $p_{\boldsymbol{\theta}}(i|b(o_t)$ as a linear classifier and we ensure the training processes of the agent and the classifier remain independent from one-another by employing a separate optimiser and stopping the gradients from propagating through the agent's network.

D.2 MINIGRID RL AGENT

We use the same PPO-based agent as reported in Parker-Holder et al. (2022). The actor and critic share the same initial layers. The first initial layer consists of a convolutional layer with 16 output channels and kernel size 3 processes the agent's view and a fully connected layer that processes its directional information. Their output is concatenated and fed to an LSTM layer with hidden size 256. The actor and critic heads each consist of two fully connected layers of size 32, the actor outputs a categorical distribution over action probabilities while the critic outputs a scalar. Weights are optimized using Adam and we employ the same hyperparameters in all experiments, reported in Table 1. Trajectories are collected via 36 worker threads, with each experiment conducted using a single GPU and 10 CPUs.

Following Parker-Holder et al. (2022), the non dataset based methods employ domain randomisation as their standard level generation process, in which the start and goal locations, alongside a random number between 0 and 60 moss, wall or lava tiles are randomly placed. The level editing process of ACCEL and SSED-EL remains unchanged from Parker-Holder et al. (2022), consisting of five steps. The first three steps may change a randomly selected tile to any of its counterparts, whereas the last two are reserved to replacing the start and goal locations if they had been removed in prior steps.

We train three different seeds for each baseline. We use the same hyperparameters as reported in Parker-Holder et al. (2022) for the DR, RPLR and ACCEL methods and as Jiang et al. (2021b) for PLR, as an extensive hyperparameter search was conducted in a similar-sized Minigrid environment in each case. SSED employs the same hyperparameters as PLR for its level buffer, with the additional secondary sampling strategy hyperparameters introduced by SSED. We did not perform an hyperparameter search for these additional hyperparameters as we found that the initial values worked adequately. We report all hyperparameters in Table 1.

D.3 VAE ARCHITECTURE AND PRE-TRAINING PROCESS

We employ the $\beta-$VAE formulation proposed in Higgins et al. (2017), and we parametrise the encoder as a Graph Convolutional Network (GCN), a generalisation of the Convolutional Neural Network (CNN) (Krizhevsky et al., 2012) to non Euclidian spaces. Our choice of a GCN architecture is motivated by the fact that the level parameter space $\mathbb{X}$ is simulator-specific. Employing a graph as an input modality for our encoder gives our model additional flexibility to by applicable to different simulators. Using a GCN, some of the inductive biases that would be internal in a traditional architecture can be defined through the wrapper encoding the environment parameter $\boldsymbol{x}$ into the graph $\mathcal{G}_{\boldsymbol{x}}$. For example, in minigrid, we represent each layout as a grid graph, each gridworld cell being an individual node. Doing so makes our GCN equivalent to a traditional CNN in the Minigrid domain. We select the GIN architecture (Xu et al., 2019) for the GCN, which we connect to an MLP network that outputs latent distribution parameters $\boldsymbol{\mu_z}, \boldsymbol{\sigma}_v z$. The decoder is a fully connected network with three heads. The *layout* head outputs the parameters of Categorical distributions for each grid cell, predicting the tile identity between [Empty, Moss, Lava, Wall]. The *start* and *goal* heads output the parameters of Categorical distributions predicting the identity of the start and goal locations across grid cells, which matches the inductive bias of a single goal or start node being present in any given level.

We pre-train the VAE for 200 epochs on X_{train}, using cross-validation for hyperparameter tuning. During training, we formulate the reconstruction loss as a weighted sum of the cross-entropy loss for each head.[5] At deployment, we guarantee *valid* layouts (layouts containing a unique start and goal location, but not necessarily solvable) by masking the non-passable nodes sampled by the layout head when sampling the start location, and masking the generated start and non-passable nodes when sampling the goal location. In this way, we guarantee unique start and goal locations that will not override one-another. Note that our generative model may still generate *unsolvable* layouts, which do not have a passable path between start and goal locations, and therefore it must learn to generate solvable layouts from X_{train} in order to be useful. We do not explicitly encourage the VAE to generate solvable layouts, but we find that optimising for the ELBO in Equation (9) is an effective proxy. Layouts reconstructed from X_{train} have over 80% solvability rate, while layouts generated

[5] To compute the cross-entropy loss of the layout head, we replace the start and goal nodes in the reconstruction targets by a uniform distribution across {moss, empty}.

via latent space interpolations have over 70% solvability rate. This indicate that maximising the ELBO results in the generative model learning to reproduce the high-level abstract properties that are shared across the level parameters in the training set, which we also observe in Figure 9.

We conduct a random sweep over a budget of 100 runs, jointly sweeping architectural parameters (number of layers, layer sizes) and the β coefficient, individual decoder head reconstruction coefficients and the learning rate. Each run takes 20 minutes, which means that our sweeping procedure takes less time to complete than a single seed of our Minigrid experiment. We report the chosen hyperparameters in Table 2.

Table 1: Hyperparameters used for Minigrid experiments. Hyperparameters shared between methods are only reported if they change from the method above.

Parameter	Procgen	MiniGrid
PPO		
γ	0.999	0.995
λ_{GAE}	0.95	0.95
PPO rollout length	256	256
PPO epochs	3	5
PPO minibatches per epoch	8	1
PPO clip range	0.2	0.2
PPO number of workers	64	32
Adam learning rate	5e-6	1e-4
Adam ϵ	1e-5	1e-5
PPO max gradient norm	0.5	0.5
PPO value clipping	yes	yes
return normalisation	yes	no
value loss coefficient	0.5	0.5
student entropy coefficient		0.0
generator entropy coefficient		0.0
PLR		
Scoring function		ℓ_1 value loss
Replay rate, p		1.0
Buffer size, K		512
Prioritisation,		rank
Temperature,		0.1
Staleness coefficient, ρ		0.3
RPLR		
Scoring function,		positive value loss
Replay rate, p		0.5
Buffer size, K		4000
ACCEL		
Edit rate, q		1.0
Replay rate, p		0.8
Buffer size, K		4000
Edit method,		random
Levels edited,		easy
SSED		
Replay rate, p		1.0
Scoring function support,		dataset
Staleness support,		dataset
Secondary Scoring function,		ℓ_1 value loss
Secondary Scoring function support,		buffer
Secondary Temperature,		1.0
Mixing coefficient, η		linearly increased from 0 to 1

Table 2: Hyperparameters used for pre-training the VAE.

Parameter	
VAE	
β	0.0448
layout head reconstruction coefficient	0.04
start and goal heads reconstruction coefficients	0.013
number of variational samples	1
Adam learning rate	4e-4
Latent space dimension	1024
number of encoder GCN layers	4
encoder GCN layer dimension	12
number of encoder MLP layers (including bottleneck layer)	2
encoder MLP layers dimension	2048
encoder bottleneck layer dimension	256
number of decoder layers	3
decoder layers dimension	256

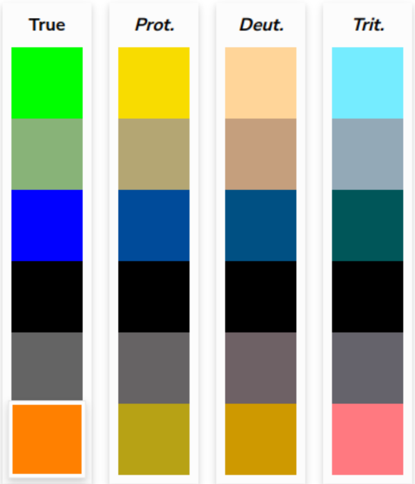


Figure 12: Color palette used for rendering minigrid layouts in this paper and their equivalent for Protanopia (Prot.), Deuteranopia (Deut.) and Tritanopia (Trit.) color blindness. We refer to each row in the main text as, in order: green (goal tiles), pale green (moss tiles), blue (agent), black (empty/floor tiles), grey (wall tiles) and orange (lava tiles).