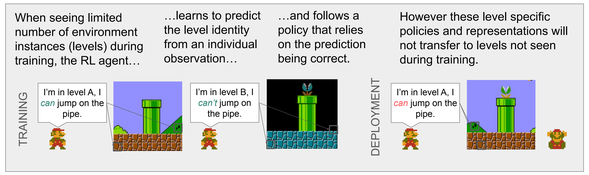
DRED: Zero-Shot Transfer in Reinforcement Learning via Data-Regularised Environment Design

Samuel Garcin s.garcin@ed.ac.uk · James Doran · Shangmin Guo · Christopher G. Lucas · Stefano Albrecht

ICML International Conference On Machine Learning · Autonomous Agents Research Group · THE UNIVERSITY of EDINBURGH

What is "instance overfitting" in RL?

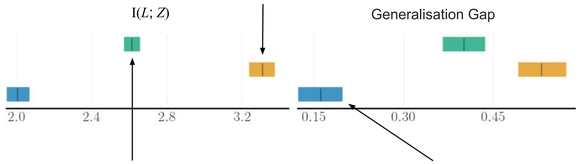


The **generalisation gap** quantifies the drop in performance between training and unseen levels. It can be reduced by minimising its upper bound,

The number of **training levels** $\sqrt{\frac{\text{const}}{|L|} \cdot \mathrm{I}(L;Z)}$ The **mutual information** between the set of training levels and the agent's learned representation.

Why do adaptive level sampling strategies help?

In the Procgen generalisation benchmark, we discover that representations learned under **uniform level sampling** tend to be highly informative of the level identity. On average, a linear classifier conditioning on the agent's representation is 50% accurate at predicting the current level (out of 200).



Prioritising levels with **high value loss** during training reduces mutual information, and results in a smaller generalisation gap.

Explicitly prioritising the **least informative levels** reduces the generalisation gap most effectively, but also impacts training efficiency.

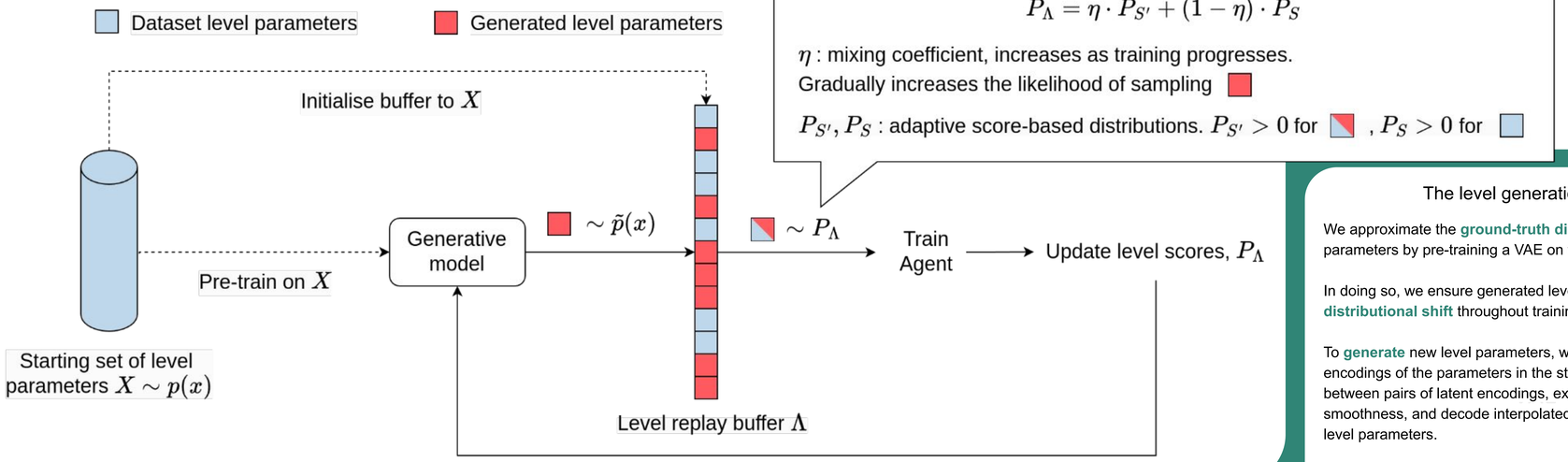
Why does value loss level prioritisation reduce $\mathrm{I}(L;Z)$?

The critic is trained to predict **level specific** value targets. When the value function has a component specific to individual levels, performing **level identification** becomes necessary to achieve zero value prediction loss. Learning level specific representations (i.e. with high mutual information) is therefore an effective strategy for value prediction.

By prioritising high value loss levels, we also de-prioritise levels with low value loss, i.e. levels for which the agent's representation has started to overfit.

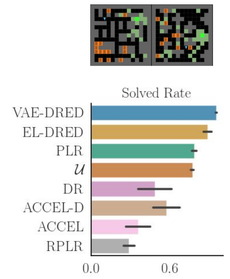
Value loss prioritisation therefore prevents instance overfitting by regularising mutual information in the **future training data**.

RL agents often require access to upwards of 10,000 training levels to not significantly overfit. Even with an access to a parametrisable simulator, obtaining the parameters necessary for level instantiation is rarely practical. With Data Regularised Environment Design (DRED), we maximise the generalisation potential of a limited starting set of level parameters by combining **adaptive sampling** with the generation of **synthetic level parameters**.

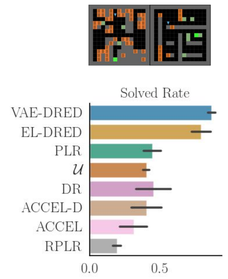


How does DRED compare to other environment design techniques?

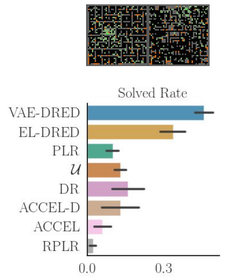
Transfer to held-out levels



Transfer to edge cases



Transfer to hard levels



In this partially observable navigation task, the agent has to make use of contextual cues shared across all levels to transfer zero-shot to unseen levels.

The proposed DRED approaches prevent overfitting and remain grounded to the starting level set distribution. In doing so, they successfully generalise to held-out levels, and perform up to 3 times better than the next best baseline on more difficult edge cases.

Methods restricted to the starting set (512 levels) tend to overfit, and are not robust to edge cases.

Due to not being grounded to any particular distribution, UED methods (DR, ACCEL and RPLR) will shift the learning problem away from desirable policies when levels inconsistent with the starting set can be generated.

The level generation process

We approximate the **ground-truth distribution** of the level parameters by pre-training a VAE on the starting set.

In doing so, we ensure generated levels induce **minimal distributional shift** throughout training.

To **generate** new level parameters, we first compute the latent encodings of the parameters in the starting set. We **interpolate** between pairs of latent encodings, exploiting the latent space's smoothness, and decode interpolated points to obtain synthetic level parameters.

Generated levels via latent space interpolation

Additional resources

Code, models, datasets and experimental data

Tutorial on Unsupervised Environment Design (blog)