Deep Surrogate Assisted Generation of Environments

Varun Bhatt[*]
University of Southern California
Los Angeles, CA
vsbhatt@usc.edu

Bryon Tjanaka[*]
University of Southern California
Los Angeles, CA
tjanaka@usc.edu

Matthew C. Fontaine[*]
University of Southern California
Los Angeles, CA
mfontain@usc.edu

Stefanos Nikolaidis
University of Southern California
Los Angeles, CA
nikolaid@usc.edu

Abstract

Recent progress in reinforcement learning (RL) has started producing generally capable agents that can solve a distribution of complex environments. These agents are typically tested on fixed, human-authored environments. On the other hand, quality diversity (QD) optimization has been proven to be an effective component of environment generation algorithms, which can generate collections of high-quality environments that are diverse in the resulting agent behaviors. However, these algorithms require potentially expensive simulations of agents on newly generated environments. We propose Deep Surrogate Assisted Generation of Environments (DSAGE), a sample-efficient QD environment generation algorithm that maintains a deep surrogate model for predicting agent behaviors in new environments. Results in two benchmark domains show that DSAGE significantly outperforms existing QD environment generation algorithms in discovering collections of environments that elicit diverse behaviors of a state-of-the-art RL agent and a planning agent. Our source code and videos are available at `https://dsagepaper.github.io/`

1 Introduction

We present an efficient method of automatically generating a collection of environments that elicit diverse agent behaviors. As a motivating example, consider deploying a robot agent at scale in a variety of home environments. The robot should generalize by performing robustly not only in test homes, but in any end user's home. To validate agent generalization, the test environments should have good coverage for the robot agent. However, obtaining such coverage may be difficult, as the generated environments would depend on the application domain, e.g. kitchen or living room, and on the specific agent we want to test, since different agents exhibit different behaviors.

To enable generalization of autonomous agents to new environments with differing levels of complexity, previous work on open-ended learning [1, 2] has integrated the environment generation and the agent training processes. The interplay between the two processes acts as a natural curriculum for the agents to learn robust skills that generalize to new, unseen environments [3–5]. The performance of these agents has been evaluated either in environments from the training distribution [1, 2, 5] or in suites of manually authored environments [3, 6, 4].

As a step towards testing generalizable agents, there has been increasing interest in competitions [7, 8] that require agents to generalize to new game layouts. Despite the recent progress of deep learning

[*]Equal contribution

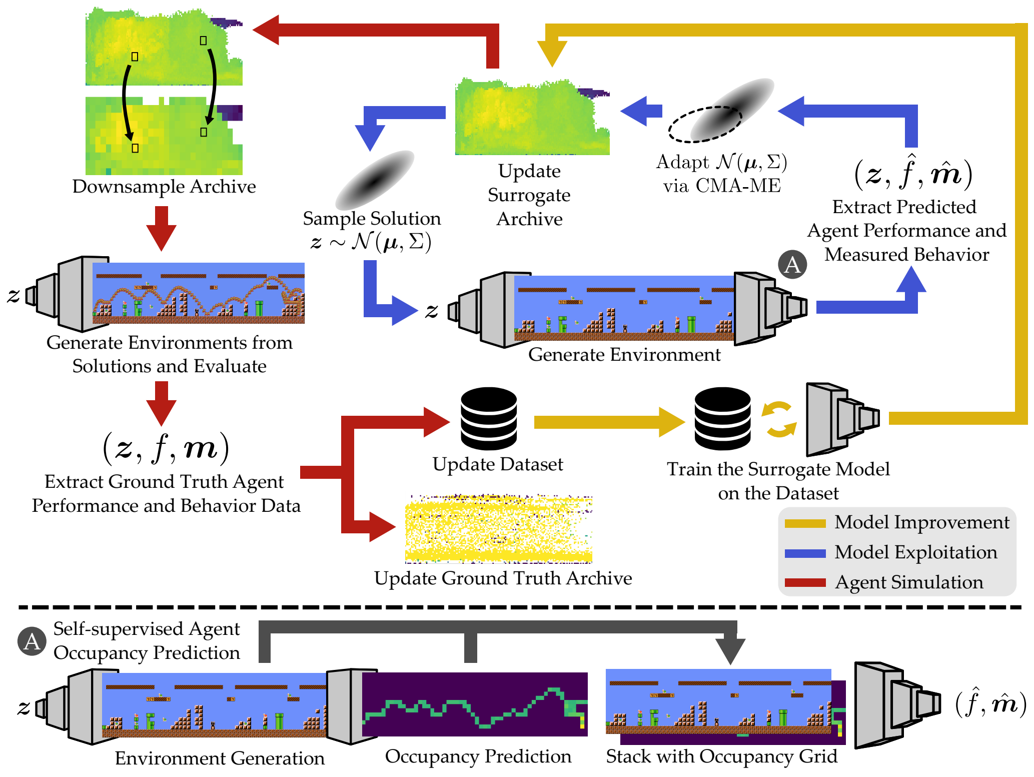

Downsample Archive
Update Surrogate Archive
Adapt $\mathcal{N}(\mu, \Sigma)$ via CMA-ME
$(z, \hat{f}, \hat{m})$
Extract Predicted Agent Performance and Measured Behavior
Sample Solution $z \sim \mathcal{N}(\mu, \Sigma)$
z
Generate Environments from Solutions and Evaluate
Generate Environment
A
(z, f, m)
Extract Ground Truth Agent Performance and Behavior Data
Update Dataset
Train the Surrogate Model on the Dataset
Update Ground Truth Archive
Model Improvement
Model Exploitation
Agent Simulation
Self-supervised Agent Occupancy Prediction
Environment Generation
Occupancy Prediction
Stack with Occupancy Grid
$(\hat{f}, \hat{m})$

Figure 1: An overview of the Deep Surrogate Assisted Generation of Environments (DSAGE) algorithm. The algorithm begins by generating and evaluating random environments to initialize the dataset and the surrogate model (not shown in the figure). An archive of solutions is generated by exploiting a deep surrogate model (**blue arrows**) with a QD optimizer, e.g., CMA-ME [14]. A subset of solutions from this archive are chosen by downsampling and evaluated by generating the corresponding environment and simulating an agent (**red arrows**). The surrogate model is then trained on the data from the simulations (**yellow arrows**). While the images show Mario levels, the algorithm structure is similar for mazes.

agents in fixed game domains, e.g. in Chess [9], Go [10], Starcraft [11], and Poker [12, 13], it has been rule-based agents that have succeeded in these competitions [8]. Such competitions also rely on manually authored game levels as a test set, handcrafted by a human designer.

While manually authored environments are important for standardized testing, creating these environments can be tedious and time-consuming. Additionally, manually authored test suites are often insufficient for eliciting the diverse range of possible agent behaviors. Instead, we would like an interactive test set that proposes an environment, observes the agent's performance and behavior, and then proposes new environments that diversify the agent behaviors, based on what the system has learned from previous execution traces of the agent.

To address collecting environments with diverse agent behaviors, prior work frames the problem as a quality diversity (QD) problem [15–17]. A QD problem consists of an objective function, e.g. whether the agent can solve the environment, and measure functions, e.g. how long the agent takes to complete their task. The measure functions quantify the behavior we would like to vary in the agent, allowing practitioners to specify the case coverage they would like to see in the domain they are testing. While QD algorithms can generate diverse collections of environments, they require a large number of environment evaluations to produce the collection, and each of these evaluations requires multiple time-consuming simulated executions of potentially stochastic agent policies.

We study how *deep surrogate models that predict agent performance can accelerate the generation of environments that are diverse in agent behaviors*. We draw upon insights from model-based quality diversity algorithms that have been previously shown to improve sample efficiency in design optimization [18] and Hearthstone deckbuilding [19]. Environments present a much more complex prediction task because the evaluation of environments involves simulating stochastic agent policies, and small changes in the environment may result in large changes in the emergent agent behaviors [20].

We make the following contributions: (1) We propose the use of deep surrogate models to predict agent performance in new environments. Our algorithm, Deep Surrogate Assisted Generation of Environments (DSAGE) (Fig. 1), integrates deep surrogate models into quality diversity optimization to efficiently generate diverse environments. (2) We show in two benchmark domains from previous work, a Maze domain [3, 4] with a trained ACCEL agent [4] and a Mario domain [21, 16] with an A* agent [22], that DSAGE outperforms state-of-the-art QD algorithms in discovering diverse agent behaviors. (3) We show with ablation studies that training the surrogate model with ancillary agent behavior data and downsampling a subset of solutions from the surrogate archive results in substantial improvements in performance, compared to the surrogate models of previous work [19].

2 Problem Definition

Quality diversity (QD) optimization. We adopt the QD problem definition from previous work [23]. A QD optimization problem specifies an objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ and a joint measure function $\boldsymbol{m} : \mathbb{R}^n \rightarrow \mathbb{R}^m$. For each element $s \in S$, where $S \subseteq \mathbb{R}^m$ is the range of the measure function, the QD goal is to find a solution $\boldsymbol{\theta} \in \mathbb{R}^n$ such that $\boldsymbol{m}(\boldsymbol{\theta}) = s$ and $f(\boldsymbol{\theta})$ is maximized.

Since the range of the measure function can be continuous, we restrict ourselves to algorithms from the MAP-Elites family [24, 25] that discretize this space into a finite number of M cells. A solution $\boldsymbol{\theta}$ is mapped to a cell based on its measure $\boldsymbol{m}(\boldsymbol{\theta})$. The solutions that occupy cells form an *archive* of solutions. Our goal is to find solutions $\boldsymbol{\theta}_i, i \in \{1, ..., M\}$ that maximize the objective f for all cells in the measure space.

$$\max_{\boldsymbol{\theta}_i} \sum_{i=1}^{M} f(\boldsymbol{\theta}_i) \tag{1}$$

The computed sum in Eq. 1 is defined as the QD-score [26], where empty cells have an objective value of 0. A second metric of the performance of a QD algorithm is coverage of the measure space, defined as the proportion of cells that are filled in by solutions: $\frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{\boldsymbol{\theta}_i}$.

QD for environment generation. We assume a single agent acting in an environment parameterized by $\boldsymbol{\theta} \in \mathbb{R}^n$. The environment parameters can be locations of different objects or latent variables that are passed as inputs to a generative model [27].[2] A QD algorithm generates new solutions $\boldsymbol{\theta}$ and evaluates them by simulating the agent on the environment parameterized by $\boldsymbol{\theta}$. The evaluation returns an objective value f and measure values $\boldsymbol{m}$. The QD algorithm attempts to generate environments that maximize f but are diverse with respect to the measures $\boldsymbol{m}$.

3 Background and Related Work

Quality diversity (QD) optimization. QD optimization originated in the genetic algorithm community with diversity optimization [28], the predecessor to QD. Later work introduced objectives to diversity optimization and resulted in the first QD algorithms: Novelty Search with Local Competition [29] and MAP-Elites [25, 24]. The QD community has grown beyond its genetic algorithm roots, with algorithms being proposed based on gradient ascent [23], Bayesian optimization [30], differential evolution [31], and evolution strategies [14, 32, 33]. QD algorithms have applications in damage recovery in robotics [24], reinforcement learning [34–36], and generative design [18, 37].

Among the QD algorithms, those of particular interest to us are the model-based ones. Current model-based [38, 39] QD algorithms either (1) learn a surrogate model of the objective and measure functions [18, 40, 41], e.g. a Gaussian process or neural network, (2) learn a generative model of the representation parameters [42, 43], or (3) draw inspiration from model-based RL [44, 45]. In particular, Deep Surrogate Assisted MAP-Elites (DSA-ME) [19] trains a deep surrogate model on a diverse dataset of solutions generated by MAP-Elites and then leverages the model to guide MAP-Elites. However, DSA-ME has only been applied to Hearthstone deck building, a simpler prediction problem than predicting agent behavior in generated environments. Additionally, DSA-ME is specific to MAP-Elites only and cannot run other QD algorithms to exploit the surrogate model.

[2]For consistency with the generative model literature, we use $\mathbf{z}$ instead of $\boldsymbol{\theta}$ when denoting latent vectors

Furthermore, DSA-ME is restricted to direct search and cannot integrate generative models to generate environments that match a provided dataset.

Automatic environment generation. Automatic environment generation algorithms have been proposed in a variety of fields. Methods between multiple communities often share generation techniques, but differ in how each community applies the generation algorithms.

For example, in the procedural content generation (PCG) field [46], an environment generator produces video game levels that result in player enjoyment. Since diversity of player experience and game mechanics is valued in games, many level generation systems incorporate QD optimization [47, 16, 48–52]. The procedural content generation via machine learning (PCGML) [53, 54] subfield studies environment generators that incorporate machine learning techniques such as Markov Chains [55], probabilistic graphical models [56], LSTMs [57], generative models [58–61], and reinforcement learning [62, 63]. Prior work [64] has leveraged surrogate models trained on offline data to accelerate search-based PCG [65].

Environment generation methods have also been proposed by the scenario generation community in robotics. Early work explored automated methods for generating road layouts, vehicle arrangements, and vehicle behaviors for testing autonomous vehicles [66–72]. Outside of autonomous vehicles, prior work [73] evaluates robot motion planning algorithms by generating environments that target specific motion planning behaviors. In human-robot interaction, QD algorithms have been applied as environment generators to find failures in shared autonomy systems [17] and human-aware planners tested in the collaborative Overcooked domain [15]

Environment generation can also help improve the generality of RL agents. Prior work proposes directly applying PCG level generation algorithms to improve the robustness of RL [74, 75] or to benchmark RL agents [76]. Paired Open-ended Trailblazer (POET) [1, 2] coevolves a population of both agents and environments to discover specialized agents that solve complex tasks. POET inspired a variety of open-ended coevolution algorithms [77–79, 5]. Later work proposes the PAIRED [3], PLR [80, 6], and ACCEL [4] algorithms that train a single generally capable agent by maximizing the regret between a pair of agents. These methods generate environments in parallel with an agent to create an automatic training curriculum. However, the authors validate these methods on human-designed environments [81]. Our work proposes a method that automatically generates valid environments that reveal diverse behaviors of these more general RL agents.

4 Deep Surrogate Assisted Generation of Environments (DSAGE)

Algorithm. We propose the Deep Surrogate Assisted Generation of Environments (DSAGE) algorithm for discovering environments that elicit diverse agent behaviors. Akin to the MAP-Elites family of QD algorithms, DSAGE maintains a *ground-truth archive* where solutions are stored based on their ground-truth evaluations. Simultaneously, DSAGE also trains and exploits a deep surrogate model for predicting the behavior of a fixed agent in new environments. The QD optimization occurs in three phases that take place in an outer loop: model exploitation, agent simulation, and model improvement (Fig. 1). Algorithm 1 provides the pseudocode for the DSAGE algorithm.

The model exploitation phase (lines 11–20) is an inner loop that leverages existing QD optimization algorithms and the predictions of the deep surrogate model to build an archive – referred to as the *surrogate* archive – of solutions. The first step of this phase is to query a list of B candidate solutions through the QD algorithm's *ask* method. These solutions are environment parameters, e.g., latent vectors of a GAN, which are passed through the environment generator, e.g., a GAN, to create an environment (line 15). Next, we make predictions with the surrogate model. The surrogate model first predicts data representing the agent's behavior, e.g., the probability of occupying each discretized tile in the environment (line 16), referred to as "ancillary agent behavior data" (y). The predicted ancillary agent behavior data ($\hat{y}$) then guides the surrogate model's downstream prediction of the objective ($\hat{f}$) and the measure values ($\hat{\boldsymbol{m}}$) (line 17). Finally, the QD algorithm's *tell* method adds the solution to the surrogate archive based on the predicted objective and measure values.

Note that since DSAGE is independent of the QD algorithm, the *ask* and *tell* methods abstract out the QD algorithm's details. For example, when the QD algorithm is MAP-Elites or CMA-ME, *tell* adds solutions if the cell in the measure space that they belong to is empty or if the existing solution in that cell has a lower objective. For CMA-ME, *tell* also includes updating internal CMA-ES parameters.

The agent simulation phase (lines 21–28) inserts a subset of solutions from the surrogate archive into the ground-truth archive. This phase begins by selecting the subset of solutions from the surrogate archive (line 21). The selected solutions are evaluated by generating the corresponding environment (line 23) and simulating a fixed agent to obtain the true objective and measure values, as well as ancillary agent behavior data (line 24). Evaluation data is appended to the dataset, and solutions that improve their corresponding cell in the ground-truth archive are added to that archive (lines 25, 26).

In the model improvement phase (line 29), the surrogate model is trained in a self-supervised manner through the supervision provided by the agent simulations and the ancillary agent behavior data.

The algorithm is initialized by generating random solutions and simulating the agent in the corresponding environments (lines 2-8). Subsequently, every outer iteration (lines 10-30) consists of model exploitation followed by agent simulation and ending with model improvement.

Algorithm 1: Deep Surrogate Assisted Generation of Environments (DSAGE)

```
Input: $N$: Maximum number of evaluations, $n_{rand}$: Number of initial random solutions,
       $N_{exploit}$: Number of iterations in the model exploitation phase, $B$: Batch size for the
       model exploitation QD optimizer
Output: Final version of the ground-truth archive $\mathcal{A}_{gt}$
1  Initialize the ground-truth archive $\mathcal{A}_{gt}$, the dataset $\mathcal{D}$, and the deep surrogate model $sm$
2  $\boldsymbol{\Theta} \leftarrow generate\_random\_solutions(n_{rand})$
3  for $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ do
4      $env \leftarrow g(\boldsymbol{\theta})$
5      $f, \boldsymbol{m}, \boldsymbol{y} \leftarrow evaluate(env)$
6      $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{\theta}, f, \boldsymbol{m}, \boldsymbol{y})$
7      $\mathcal{A}_{gt} \leftarrow add\_solution(\mathcal{A}_{gt}, (\boldsymbol{\theta}, f, \boldsymbol{m}))$
8  end
9  $evals \leftarrow n_{rand}$
10 while $evals < N$ do
11     Initialize a QD optimizer $qd$ with the surrogate archive $\mathcal{A}_{surrogate}$
12     for $itr \in \{1, 2, \ldots, N_{exploit}\}$ do
13         $\boldsymbol{\Theta} \leftarrow qd.ask(B)$
14         for $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ do
15             $env \leftarrow g(\boldsymbol{\theta})$
16             $\hat{y} \leftarrow sm.predict\_ancillary(env)$                  } Model Exploitation
17             $\hat{f}, \hat{\boldsymbol{m}} \leftarrow sm.predict(env, \hat{y})$
18             $qd.tell(\boldsymbol{\theta}, \hat{f}, \hat{\boldsymbol{m}})$
19         end
20     end
21     $\boldsymbol{\Theta} \leftarrow select\_solutions(\mathcal{A}_{surrogate})$
22     for $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ do
23         $env \leftarrow g(\boldsymbol{\theta})$
24         $f, \boldsymbol{m}, \boldsymbol{y} \leftarrow evaluate(env)$
25         $\mathcal{D} \leftarrow \mathcal{D} \cup (\boldsymbol{\theta}, f, \boldsymbol{m}, \boldsymbol{y})$      } Agent Simulation
26         $\mathcal{A}_{gt} \leftarrow add\_solution(\mathcal{A}_{gt}, (\boldsymbol{\theta}, f, \boldsymbol{m}))$
27         $evals \leftarrow evals + 1$
28     end
29     $sm.train(\mathcal{D})$                                               } Model Improvement
30 end
```

Self-supervised prediction of ancillary agent behavior data. By default, a surrogate model directly predicts the objective and measure values based on the initial state of the environment and the agent (provided in the form of a one-hot encoded image). However, we anticipate that direct prediction will be challenging in some domains, as it requires understanding the agent's trajectory in the environment. Thus, we provide additional supervision to the surrogate model in DSAGE via a two-stage self-supervised process.

First, a deep neural network predicts ancillary agent behavior data. In our work, we obtain this data by recording the expected number of times the agent visits each discretized tile in the environment, resulting in an “occupancy grid.” We then concatenate the predicted ancillary information, i.e., the predicted occupancy grid, with the one-hot encoded image of the environment and pass them through another deep neural network to obtain the predicted objective and measure values. We use CNNs for both predictors and include architecture details in Appendix B. As a baseline, we compare our model with a CNN that directly predicts the objective and measure values without the help of ancillary data.

Downsampling to select solutions from the surrogate archive. After the model exploitation phase, the surrogate archive is populated with solutions that were predicted to be high-performing and diverse. Hence, a basic selection mechanism (line 21) would select all solutions from the surrogate archive, identical to DSA-ME [19]. However, if the surrogate archive is overly populated, full selection may result in a large number of ground-truth evaluations per outer-loop iteration, leading to fewer outer loops and less surrogate model training. To balance the trade-off between evaluating solutions from the surrogate archive and training the surrogate model, we only select a subset of solutions for evaluation by downsampling the surrogate archive. Downsampling uniformly divides the surrogate archive into sub-regions of cells and selects a random solution from each area.

5 Domains

We test our algorithms in two benchmark domains from prior work: a Maze domain [82, 3, 4] with a trained ACCEL agent [4] and a Mario domain [83, 16] with an A* agent [22]. We select these domains because, despite their relative simplicity (each environment is represented as a 2D grid of tiles), agents in these environments exhibit complex and diverse behaviors.

In the Maze domain, we directly search for different mazes, with the QD algorithm returning the layout of the maze. In the Mario domain, we search for latent codes that are passed through a pre-trained GAN, similar to the corresponding previous work.

We select the objective and measure functions as described below. Since the agent or the environment dynamics are stochastic in each domain, we average the objective and measure values over 50 episodes in the Maze domain and 5 episodes in the Mario domain.

Maze. We set a binary objective function f that is 1 if the generated environment is solvable and 0 otherwise, indicating the validity of the environment. Since we wish to generate visually diverse levels that offer a range of difficulty level for the agent, we select as measures (1) *number of wall cells* (range: $[0, 256]$), and (2) *mean agent path length* (range: $[0, 648]$, where 648 indicates a failure to reach the goal).

Mario. Since we wish to generate playable levels, we set the objective as the *completion rate*, i.e., the proportion of the level that the agent completes before dying. We additionally want to generate environments that result in qualitatively different agent behaviors, thus we selected as measures: (1) *sky tiles*, the number of tiles of a certain type that are in the top half of the 2D grid (range: $[0, 150]$), (2) *number of jumps*, the number of times that the A* agent jumps during its execution (range: $[0, 100]$).

See Appendix A for further environment details.

6 Experiments

6.1 Experiment Design

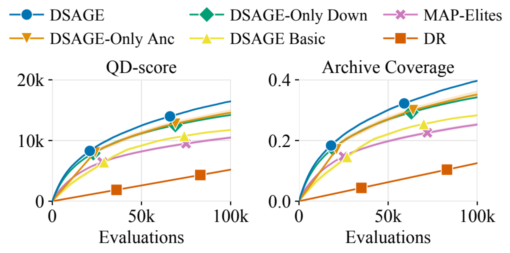
Independent variables. In each domain (Maze and Mario), we follow a between-groups design, where the independent variable is the algorithm. We test the following algorithms:

DSAGE: The proposed algorithm that includes predicting ancillary agent behavior data and downsampling the surrogate archive (Sec. 4).

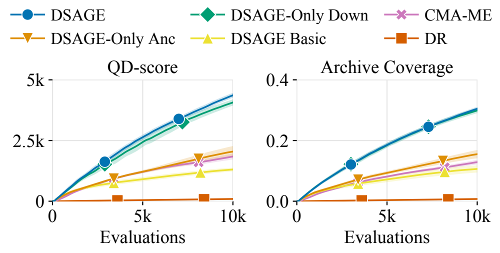
DSAGE-Only Anc: The proposed algorithm with ancillary data prediction and no downsampling, i.e., selecting all solutions from the surrogate archive.

DSAGE-Only Down: The proposed algorithm with downsampling and no ancillary data prediction.

(a) Maze



(b) Mario



Algorithm	QD-score	Archive Coverage
DSAGE	**16,446.60 ± 42.27**	**0.40 ± 0.00**
DSAGE-Only Anc	14,568.00 ± 434.56	0.35 ± 0.01
DSAGE-Only Down	14,205.20 ± 40.86	0.34 ± 0.00
DSAGE Basic	11,740.00 ± 84.13	0.28 ± 0.00
MAP-Elites	10,480.80 ± 150.13	0.25 ± 0.00
DR	5,199.60 ± 30.32	0.13 ± 0.00

Algorithm	QD-score	Archive Coverage
DSAGE	**4,362.29 ± 72.54**	**0.30 ± 0.00**
DSAGE-Only Anc	2,045.28 ± 201.64	0.16 ± 0.01
DSAGE-Only Down	4,067.42 ± 102.06	**0.30 ± 0.01**
DSAGE Basic	1,306.11 ± 50.90	0.11 ± 0.01
CMA-ME	1,840.17 ± 95.76	0.13 ± 0.01
DR	92.75 ± 3.01	0.01 ± 0.00

Figure 2: QD-score and archive coverage attained by baseline QD algorithms and DSAGE in the Maze and Mario domains over 5 trials. Tables and plots show mean and standard error of the mean.

DSAGE Basic: The basic version of the proposed algorithm that selects all solutions from the surrogate archive and does not predict ancillary data.

Baseline QD: The QD algorithm without surrogate assistance. We follow previous work [16] and use CMA-ME for the Mario domain. Since CMA-ME operates only in continuous spaces, we use MAP-Elites in the discrete Maze domain.

Domain Randomization (DR) [84–86]: Algorithm that generates and evaluates random solutions, i.e., wall locations in the maze domain and the latent code to pass through the GAN in the Mario domain.

Dependent variables. We measure the quality and diversity of the solutions with the QD-score metric [26](Eq. 1). As an additional metric of diversity, we also report the archive coverage. We run each algorithm for 5 trials in each domain.

Hypothesis. *We hypothesize that DSAGE will result in a better QD-score than DSAGE Basic in all domains, which in turn will result in better performance than the baseline QD algorithm. DSAGE, DSAGE Basic, and the baseline QD algorithm will all exceed DR.* We base this hypothesis on previous work [25, 17] which shows that QD algorithms outperform random sampling in a variety of domains, as well as previous work [18, 19] which shows that surrogate-assisted MAP-Elites outperforms standard MAP-Elites in design optimization and Hearthstone domains. Furthermore, we expect that the additional supervision through ancillary agent behavior data and downsampling will result in DSAGE performing significantly better than DSAGE Basic.

6.2 Analysis

Fig. 2 summarizes the results obtained by the six algorithms on the Maze and the Mario domains.

One-way ANOVA tests showed a significant effect of the algorithm on the QD-score for the Maze ($F(5, 24) = 430.98, p < 0.001$) and Mario ($F(5, 24) = 238.09, p < 0.001$) domains.

Post-hoc pairwise comparisons with Bonferroni corrections showed that DSAGE outperformed DSAGE Basic, Baseline QD, and DR in both the Maze and the Mario domains ($p < 0.001$). Additionally, DSAGE Basic outperformed MAP-Elites and DR in the Maze domain ($p < 0.001$), while it performed significantly worse than the QD counterpart, CMA-ME, in the Mario domain ($p = 0.003$). Finally, Baseline QD outperformed DR in both the Maze and Mario domains ($p < 0.001$).

These results show that deep surrogate assisted generation of environments results in significant improvements compared to quality diversity algorithms without surrogate assistance. They also show that adding ancillary agent behavior data and downsampling is important in both domains. Without these components, DSAGE Basic has limited or no improvement compared to the QD algorithm

Table 1: Number of evaluations required to reach a QD-score of 10480.8 in the Maze domain and 1306.11 in the Mario domain.

(a) Maze

Algorithm	Evaluations
DSAGE	**33,930.40 ± 1,411.04**
DSAGE-Only Anc	51,919.60 ± 8,254.24
DSAGE-Only Down	42,816.60 ± 691.38
DSAGE Basic	85,328.60 ± 2,947.24
MAP-Elites	100,000

(b) Mario

Algorithm	Evaluations
DSAGE	**2,464.40 ± 356.36**
DSAGE-Only Anc	7,727.40 ± 1,433.33
DSAGE-Only Down	**2,768.60 ± 586.34**
DSAGE Basic	10,000
CMA-ME	5,760.00 ± 516.14

Table 2: Mean absolute error of the objective and measure predictions by the surrogate models.

	Maze			Mario		
Algorithm	Objective MAE	Number of Wall Cells MAE	Mean Agent Path Length MAE	Objective MAE	Number of Sky Tiles MAE	Number of Jumps MAE
DSAGE	0.03	0.37	96.58	0.10	1.10	7.16
DSAGE-Only Anc	0.04	0.96	95.14	0.20	1.11	9.97
DSAGE-Only Down	0.10	0.95	151.50	0.11	0.87	6.52
DSAGE Basic	0.18	5.48	157.69	0.20	2.16	10.71

without surrogate assistance. Additionally, domain randomization is significantly worse than DSAGE as well as the baselines. The archive coverage and consequently the QD-score is negligible in the Mario domain since randomly sampled latent codes led to little diversity in the levels.

Table 1 shows another metric of the speed-up provided by DSAGE: the number of evaluations (agent simulations) required to reach a fixed QD-score. We set this fixed QD-score to be 10480.8 in the Maze domain and 1306.11 in the Mario domain, which are the mean QD-scores of MAP-Elites and DSAGE Basic, respectively, in those domains. DSAGE reaches these QD-scores faster than the baselines do.

To assess the quality of the trained surrogate model, we create a combined dataset consisting of data from one run of each surrogate assisted algorithm. We use this dataset to evaluate the surrogate models trained from separate runs of DSAGE and its variants. Table 2 shows the mean absolute error (MAE) of the predictions by the surrogate models. The model learned by DSAGE Basic fails to predict the agent-based measures well. It has an MAE of 157.69 for the mean agent path length in Maze and MAE = 10.71 for the number of jumps in Mario. In contrast, the model learned by DSAGE makes more accurate predictions, with MAE = 96.58 for mean agent path length and MAE = 7.16 for number of jumps. We provide detailed results of the surrogate model predictions in Appendix B.1.

6.3 Ablation Study

Sec. 4 describes two key components of DSAGE: (1) self-supervised prediction of ancillary agent behavior data, and (2) downsampling to select solutions from the surrogate archive. We perform an ablation study by treating the inclusion of ancillary data prediction (ancillary data / no ancillary data) and the method of selecting solutions from the surrogate archive (downsampling / full selection) as independent variables. A two-way ANOVA for each domain showed no significant interaction effects. We perform a main effects analysis for each independent variable.

Inclusion of ancillary data prediction. A main effects analysis for the inclusion of ancillary data prediction showed that algorithms that predict ancillary agent behavior data (DSAGE, DSAGE-Only Anc) performed significantly better than their counterparts with no ancillary data prediction (DSAGE-Only Down, DSAGE Basic) in both domains ($p < 0.001$).

Fig. 2 shows that predicting ancillary agent behavior data also resulted in a larger mean coverage for Maze, while it has little or no improvement for Mario. Additionally, as shown in Table 2, predicting ancillary agent behavior data helped improve the prediction of the mean agent path length in the Maze domain but provided little improvement to the prediction of the number of jumps in the Mario domain. The reason is that in the Maze domain, the mean agent path length is a scaled version of the sum of the agent's tile occupancy frequency, hence the two-stage process which predicts the occupancy grid first is essential for improving the accuracy of the model. On the other hand, the presence of a jump in Mario depends not only on cell occupancy, but also on the structure of the level and the sequence of the occupied cells.

Method of selecting solutions from the surrogate archive. A main effects analysis for the method of selecting solutions from the surrogate archive showed that the algorithms with downsampling (DSAGE, DSAGE-Only Down) performed significantly better than their counterparts with no downsampling (DSAGE-Only Anc, DSAGE Basic) in both domains ($p < 0.001$).

A major advantage of downsampling is that it decreases the number of ground-truth evaluations in each outer iteration. Thus, for a fixed evaluation budget, downsampling results in a greater number of outer iterations. For instance, in the Maze domain, runs without downsampling had only 6-7 outer iterations, while runs with downsampling had approximately 220 outer iterations. More outer iterations leads to more training and thus higher accuracy of the surrogate model. In turn, a more accurate surrogate model will generate a better surrogate archive in the inner loop.

We include an ablation in Appendix D to test between two possible explanations for why having more outer iterations helps with performance: (1) larger number of training epochs, (2) more updates to the dataset allowing the surrogate model to iteratively correct its own errors. We observed that iterative correction accounted for most of the performance increase with downsampling.

The second advantage of downsampling is that it selects solutions evenly from all regions of the measure space, thus creating a more balanced dataset. This helps train the surrogate model in parts of the measure space that are not frequently visited. We include an additional baseline in Appendix E in which we select a subset of solutions uniformly at random from the surrogate archive instead of downsampling. We observe that downsampling has a slight advantage over uniform random sampling in the Maze domain.

Furthermore, if instead of downsampling we sampled multiple solutions from nearby regions of the surrogate archive, the prediction errors could cause the solutions to collapse to a single cell in the ground-truth archive, resulting in many solutions being discarded.

Overall, our ablation study shows that both predicting the occupancy grid as ancillary data and downsampling the surrogate archive independently help improve the performance of DSAGE.

6.4 Qualitative Results

Fig. 3 and Fig. 4 show example environments generated by DSAGE in the Maze and Mario domains.

Having the mean agent path length as a measure in the Maze domain results in environments of varying difficulty for the ACCEL agent. For instance, we observe that the environment in Fig. 3(a) has very few walls, yet the ACCEL agent gets stuck in the top half of the maze and is unable to find the goal within the allotted time. On the other hand, the environment in Fig. 3(d) is cluttered and there are multiple dead-ends, yet the ACCEL agent is able to reach the goal.

Fig. 4 shows that the generated environments result in qualitatively diverse behaviors for the Mario agent too. Level (b) only has a few sky tiles and is mostly flat, resulting in a small number of jumps. Level (c) has a “staircase trap” on the right side, forcing the agent to perform continuous jumps to escape and complete the level. We include videos of the playthroughs in the supplemental material.

7 Societal Impacts

By introducing surrogate models into quality diversity algorithms, we can efficiently generate environments that result in diverse agent behaviors. While we focused on an RL agent in a Maze domain and a symbolic agent in a Mario game domain, our method can be applied to a variety of agents and domains. This can help with testing the robustness of agents, attaining insights about their behavior, and discovering edge cases before real-world deployment. Furthermore, we anticipate that in the future, closing the loop between environment generation and agent training can improve the ability of agents to generalize to new settings and thus increase their widespread use.

Our work may also have negative impacts. Training agents in diverse environments can be considered as a step towards open-ended evolution [87], which raises concerns about the predictability and safety of the emergent agent behaviors [88, 89]. Discovering corner cases that result in unwanted behaviors or catastrophic failures may also be used maliciously to reveal vulnerabilities in deployed agents [90].

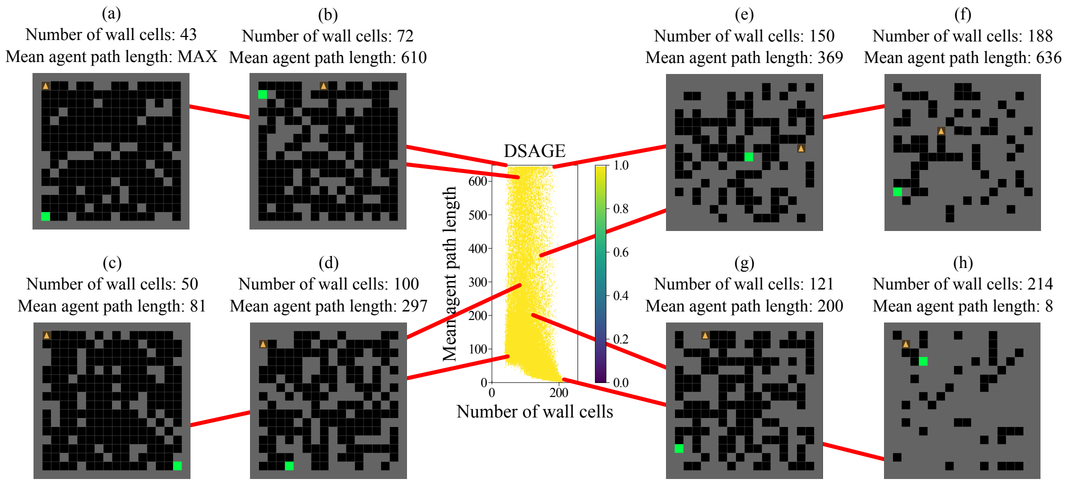


Figure 3: Archive and levels generated by DSAGE in the Maze domain. The agent's initial position is shown as an orange triangle, while the goal is a green square.

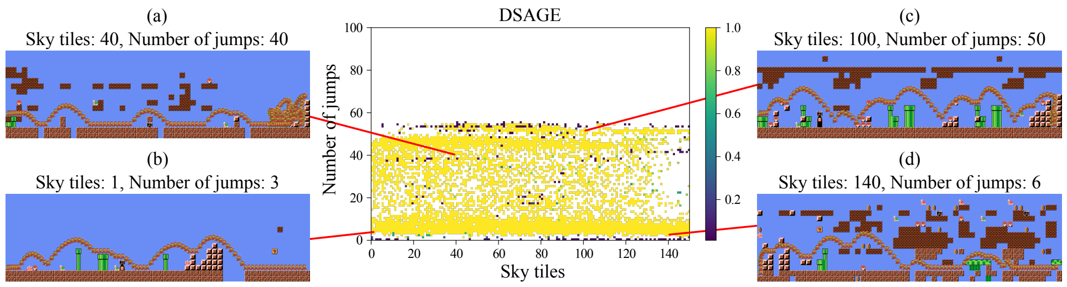


Figure 4: Archive and levels generated by DSAGE in the Mario domain. Each level shows the path Mario takes, starting on the left of the level and finishing on the right.

8 Limitations and Future Work

Automatic environment generation is a rapidly growing research area with a wide range of applications, including designing video game levels [46, 53, 54], training and testing autonomous agents [74, 76, 1, 3, 4], and discovering failures in human-robot interaction [17, 16]. We introduce the DSAGE algorithm, which efficiently generates a diverse collection of environments via deep surrogate models of agent behavior.

Our paper has several limitations. First, occupancy grid prediction does not encode temporal information about the agent. While this prediction allows us to avoid the compounding error problem of model-based RL [91], forgoing temporal information makes it harder to predict some behaviors, such as the number of jumps in Mario. We will explore this trade-off in future work.

Furthermore, we have studied 2D domains where a single ground-truth evaluation lasts between a few seconds and a few minutes. We are excited about the use of surrogate models to predict the performance of agents in more complex domains with expensive, high-fidelity simulations [92].

Acknowledgments and Disclosure of Funding

This work was partially funded by NSF CAREER (#2145077) and NSF GRFP (#DGE-1842487). One of the GPUs used in the experiments was awarded by the NVIDIA Academic Hardware Grant. We thank J. Parker-Holder et al., the authors of the ACCEL agent, for providing a pre-trained model of the agent for our experiments. We also thank Ya-Chuan Hsu for providing invaluable feedback.

References

[1] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions," *CoRR*, vol. abs/1901.01753, 2019.

[2] R. Wang, J. Lehman, A. Rawal, J. Zhi, Y. Li, J. Clune, and K. O. Stanley, "Enhanced POET: open-ended reinforcement learning through unbounded invention of learning challenges and their solutions," in *Proceedings of the 37th International Conference on Machine Learning, ICML*, 2020.

[3] M. Dennis, N. Jaques, E. Vinitsky, A. M. Bayen, S. Russell, A. Critch, and S. Levine, "Emergent complexity and zero-shot transfer via unsupervised environment design," in *Advances in Neural Information Processing Systems 33*, 2020.

[4] J. Parker-Holder, M. Jiang, M. Dennis, M. Samvelyan, J. N. Foerster, E. Grefenstette, and T. Rocktäschel, "Evolving curricula with regret-based environment design," *CoRR*, vol. abs/2203.01302, 2022.

[5] A. Dharna, A. K. Hoover, J. Togelius, and L. Soros, "Transfer dynamics in emergent evolutionary curricula," *IEEE Transactions on Games*, 2022.

[6] M. Jiang, M. Dennis, J. Parker-Holder, J. N. Foerster, E. Grefenstette, and T. Rocktäschel, "Replay-guided adversarial environment design," in *Advances in Neural Information Processing Systems 34*, 2021.

[7] D. P. Liebana, S. Samothrakis, J. Togelius, T. Schaul, and S. M. Lucas, "General video game AI: competition, challenges and opportunities," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.

[8] E. Hambro, S. P. Mohanty, D. Babaev, M. Byeon, D. Chakraborty, E. Grefenstette, M. Jiang, *et al.*, "Insights from the NeurIPS 2021 NetHack challenge," *CoRR*, vol. abs/2203.11889, 2022.

[9] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, 2018.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, 2016.

[11] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, 2019.

[12] M. Moravcík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. H. Bowling, "Deepstack: Expert-level artificial intelligence in no-limit poker," *Science*, 2017.

[13] N. Brown and T. Sandholm, "Superhuman AI for multiplayer poker," *Science*, 2019.

[14] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance matrix adaptation for the rapid illumination of behavior space," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[15] M. C. Fontaine, Y. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the importance of environments in human-robot coordination," in *Robotics: Science and Systems*, 2021.

[16] M. C. Fontaine, R. Liu, A. Khalifa, J. Modi, J. Togelius, A. K. Hoover, and S. Nikolaidis, "Illuminating mario scenes in the latent space of a generative adversarial network," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

[17] M. C. Fontaine and S. Nikolaidis, "A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy," in *Robotics: Science and Systems*, 2021.

[18] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-efficient design exploration through surrogate-assisted illumination," *Evolutionary Computation*, 2018.

[19] Y. Zhang, M. C. Fontaine, A. K. Hoover, and S. Nikolaidis, "Deep surrogate assisted MAP-Elites for automated hearthstone deckbuilding," *CoRR*, vol. abs/2112.03534, 2021.

[20] N. Sturtevant, N. Decroocq, A. Tripodi, and M. Guzdial, "The unexpected consequence of incremental design changes," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

[21] S. Karakovskiy and J. Togelius, “The mario AI benchmark and competitions,” *IEEE Transactions on Computational Intelligence and AI in Games*, 2012.

[22] R. Baumgarten, “Infinite super mario AI,” 2009.

[23] M. C. Fontaine and S. Nikolaidis, “Differentiable quality diversity,” *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[24] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, “Robots that can adapt like animals,” *Nature*, 2015.

[25] J. Mouret and J. Clune, “Illuminating search spaces by mapping elites,” *CoRR*, vol. abs/1504.04909, 2015.

[26] J. K. Pugh, L. B. Soros, and K. O. Stanley, “Quality diversity: A new frontier for evolutionary computation,” *Frontiers in Robotics and AI*, 2016.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in Neural Information Processing Systems*, 2014.

[28] J. Lehman and K. O. Stanley, “Abandoning objectives: Evolution through the search for novelty alone,” *Evolutionary Computation*, 2011.

[29] J. Lehman and K. O. Stanley, “Evolving a diversity of virtual creatures through novelty search and local competition,” in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, 2011.

[30] P. Kent and J. Branke, “Bop-elites, a bayesian optimisation algorithm for quality-diversity search,” *CoRR*, vol. abs/2005.04320, 2020.

[31] T. J. Choi and J. Togelius, “Self-referential quality diversity through differential map-elites,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[32] E. Conti, V. Madhavan, F. P. Such, J. Lehman, K. O. Stanley, and J. Clune, “Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents,” in *Advances in Neural Information Processing Systems 31*, 2018.

[33] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, “Scaling map-elites to deep neuroevolution,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[34] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis, “Approximating gradients for differentiable quality diversity in reinforcement learning,” *CoRR*, vol. abs/2202.03666, 2022.

[35] O. Nilsson and A. Cully, “Policy gradient assisted map-elites,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[36] G. Cideron, T. Pierrot, N. Perrin, K. Beguir, and O. Sigaud, “QD-RL: efficient mixing of quality and diversity in reinforcement learning,” *CoRR*, vol. abs/2006.08505, 2020.

[37] A. Hagg, S. Berns, A. Asteroth, S. Colton, and T. Bäck, “Expressivity of parameterized and data-driven representations in quality diversity search,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[38] T. Bartz-Beielstein, “A survey of model-based methods for global optimization,” *Bioinspired Optimization Methods and Their Applications*, 2016.

[39] T. M. Moerland, J. Broekens, and C. M. Jonker, “Model-based reinforcement learning: A survey,” *CoRR*, vol. abs/2006.16712, 2020.

[40] A. Hagg, D. Wilde, A. Asteroth, and T. Bäck, “Designing air flow with surrogate-assisted phenotypic niching,” in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, 2020.

[41] L. Cazenille, N. Bredeche, and N. Aubert-Kato, “Exploring self-assembling behaviors in a swarm of bio-micro-robots using surrogate-assisted map-elites,” in *IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.

[42] A. Gaier, A. Asteroth, and J.-B. Mouret, “Discovering representations for black-box optimization,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[43] N. Rakicevic, A. Cully, and P. Kormushev, “Policy manifold search: Exploring the manifold hypothesis for diversity-based neuroevolution,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[44] L. Keller, D. Tanneberg, S. Stark, and J. Peters, “Model-based quality-diversity search for efficient robot learning,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2020.

[45] B. Lim, L. Grillotti, L. Bernasconi, and A. Cully, “Dynamics-aware quality-diversity for efficient learning of skill repertoires,” *CoRR*, vol. abs/2109.08522, 2021.

[46] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*. Springer, 2016.

[47] D. Gravina, A. Khalifa, A. Liapis, J. Togelius, and G. N. Yannakakis, “Procedural content generation through quality diversity,” in *Proceedings of the IEEE Conference on Games (CoG)*, 2019.

[48] S. Earle, J. Snider, M. C. Fontaine, S. Nikolaidis, and J. Togelius, “Illuminating diverse neural cellular automata for level generation,” *CoRR*, vol. abs/2109.05489, 2021.

[49] A. Khalifa, S. Lee, A. Nealen, and J. Togelius, “Talakat: Bullet hell generation through constrained map-elites,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.

[50] K. Steckel and J. Schrum, “Illuminating the space of beatable lode runner levels produced by various generative adversarial networks,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021.

[51] J. Schrum, V. Volz, and S. Risi, “CPPN2GAN: combining compositional pattern producing networks and gans for large-scale pattern generation,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[52] A. Sarkar and S. Cooper, “Generating and blending game levels via quality-diversity in the latent space of a variational autoencoder,” in *Proceedings of the 16th International Conference on the Foundations of Digital Games*, 2021.

[53] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, “Procedural content generation via machine learning (PCGML),” *IEEE Transactions on Games*, 2018.

[54] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, “Deep learning for procedural content generation,” *Neural Computing and Applications*, 2021.

[55] S. Snodgrass and S. Ontañón, “Experiments in map generation using markov chains,” in *Proceedings of the 9th International Conference on the Foundations of Digital Games, FDG*, 2014.

[56] M. Guzdial and M. Riedl, “Game level generation from gameplay videos,” in *Proceedings of the 12th Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016.

[57] A. Summerville and M. Mateas, “Super mario as a string: Platformer level generation via LSTMs,” in *Proceedings of the First Joint International Conference of Digital Games Research Association and Foundation of Digital Games, DiGRA/FDG*, 2016.

[58] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi, “Evolving mario levels in the latent space of a deep convolutional generative adversarial network,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018.

[59] E. Giacomello, P. L. Lanzi, and D. Loiacono, “DOOM level generation using generative adversarial networks,” in *Proceedings of the IEEE Games, Entertainment, Media Conference (GEM)*, 2018.

[60] R. R. Torrado, A. Khalifa, M. C. Green, N. Justesen, S. Risi, and J. Togelius, “Bootstrapping conditional gans for video game level generation,” in *Proceedings of the IEEE Conference on Games*, 2020.

[61] A. Sarkar, Z. Yang, and S. Cooper, “Conditional level generation and game blending,” *CoRR*, vol. abs/2010.07735, 2020.

[62] A. Khalifa, P. Bontrager, S. Earle, and J. Togelius, “PCGRL: procedural content generation via reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

[63] S. Earle, M. Edwards, A. Khalifa, P. Bontrager, and J. Togelius, “Learning controllable content generators,” in *Proceedings of the IEEE Conference on Games (CoG)*, 2021.

[64] D. Karavolos, A. Liapis, and G. N. Yannakakis, “A multifaceted surrogate model for search-based procedural content generation,” *IEEE Transactions on Games*, 2021.

[65] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, “Search-based procedural content generation: A taxonomy and survey,” *IEEE Transactions on Computational Intelligence and AI in Games*, 2011.

[66] J. Arnold and R. Alexander, “Testing autonomous robot control software using procedural content generation,” in *Proceedings of the 32nd International Conference on Computer Safety, Reliability, and Security*, 2013.

[67] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, “Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles,” *Journal of Systems and Software*, 2018.

[68] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, “Generating adversarial driving scenarios in high-fidelity simulators,” in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2019.

[69] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, “Automated scenario generation for regression testing of autonomous vehicles,” in *Proceedings of the 20th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017.

[70] A. Gambi, M. Mueller, and G. Fraser, “Automatically testing self-driving cars with search-based procedural content generation,” in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2019.

[71] D. Sadigh, S. S. Sastry, and S. A. Seshia, “Verifying robustness of human-aware autonomous cars,” *IFAC-PapersOnLine*, 2019.

[72] D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, and S. A. Seshia, “Scenic: a language for scenario specification and scene generation,” in *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.

[73] Y. Zhou, S. Booth, N. Figueroa, and J. Shah, “RoCUS: robot controller understanding via sampling,” in *Proceedings of the Conference on Robot Learning*, 2021.

[74] S. Risi and J. Togelius, “Increasing generality in machine learning through procedural content generation,” *Nature Machine Intelligence*, 2020.

[75] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, “Illuminating generalization in deep reinforcement learning through procedural level generation,” *arXiv preprint arXiv:1806.10729*, 2018.

[76] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, “Leveraging procedural generation to benchmark reinforcement learning,” in *Proceedings of the International Conference on Machine Learning*, 2020.

[77] T. Gabor, A. Sedlmeier, M. Kiermeier, T. Phan, M. Henrich, M. Pichlmair, B. Kempter, C. Klein, H. Sauer, R. S. AG, *et al.*, “Scenario co-evolution for reinforcement learning on a grid world smart factory domain,” in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[78] D. M. Bossens and D. Tarapore, “QED: using quality-environment-diversity to evolve resilient robot swarms,” *IEEE Transactions on Evolutionary Computation*, 2020.

[79] A. Dharna, J. Togelius, and L. B. Soros, “Co-generation of game levels and game-playing agents,” in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2020.

[80] M. Jiang, E. Grefenstette, and T. Rocktäschel, “Prioritized level replay,” in *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.

[81] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, “A survey of generalisation in deep reinforcement learning,” *CoRR*, vol. abs/2111.09794, 2021.

[82] M. Chevalier-Boisvert, L. Willems, and S. Pal, “Minimalistic gridworld environment for OpenAI gym.” `https://github.com/maximecb/gym-minigrid`, 2018.

[83] J. Togelius, S. Karakovskiy, and R. Baumgarten, “The 2009 mario AI competition,” in *Proceedings of the IEEE Congress on Evolutionary Computation, CEC*, 2010.

[84] N. Jakobi, “Evolutionary robotics and the radical envelope-of-noise hypothesis,” *Adaptive Behavior*, 1998.

[85] F. Sadeghi and S. Levine, "CAD2RL: real single-image flight without a single real image," in *Proceedings of Robotics: Science and Systems XIII*, 2017.

[86] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2017.

[87] K. O. Stanley, J. Lehman, and L. Soros, "Open-endedness: The last grand challenge you've never heard of," 2017.

[88] A. Ecoffet, J. Clune, and J. Lehman, "Open questions in creating safe open-ended AI: tensions between control and creativity," *CoRR*, vol. abs/2006.07495, 2020.

[89] D. Hendrycks, N. Carlini, J. Schulman, and J. Steinhardt, "Unsolved problems in ML safety," *CoRR*, vol. abs/2109.13916, 2021.

[90] A. Roy, N. Memon, J. Togelius, and A. Ross, "Evolutionary methods for generating synthetic masterprint templates: Dictionary attack in fingerprint recognition," in *2018 International Conference on Biometrics (ICB)*, pp. 39–46, IEEE, 2018.

[91] C. Xiao, Y. Wu, C. Ma, D. Schuurmans, and M. Müller, "Learning to combat compounding-error in model-based reinforcement learning," *CoRR*, vol. abs/1912.11206, 2019.

[92] P. R. Wurman, S. Barrett, K. Kawamoto, J. MacGlashan, K. Subramanian, T. J. Walsh, R. Capobianco, A. Devlic, F. Eckert, F. Fuchs, *et al.*, "Outracing champion gran turismo drivers with deep reinforcement learning," *Nature*, 2022.

[93] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2022.

[94] A. Khalifa, "Mario AI framework." `https://github.com/amidos2006/Mario-AI-Framework`, 2019.

[95] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[96] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein GANs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[97] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML*, 2015.

[98] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[99] B. Tjanaka, M. C. Fontaine, D. H. Lee, T. T. M. Vu, Y. Zhang, S. Sommerer, N. Dennler, and S. Nikolaidis, "pyribs: A bare-bones python library for quality diversity optimization." `https://github.com/icaros-usc/pyribs`, 2021.

[100] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015.

[101] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, *et al.*, "PyTorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019.

[102] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra, "Grokking: Generalization beyond overfitting on small algorithmic datasets," *CoRR*, vol. abs/2201.02177, 2022.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] We claim that our algorithm significantly outperforms QD environment generation algorithms. The claim is supported by results and the statistical tests shown in Sec. 6.2.

   (b) Did you describe the limitations of your work? [Yes] We list the limitations of our work in Sec. 8.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We discuss the potential positive and negative impacts of this work in Sec. 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide the code in the supplemental material for running all the experiments specified in Sec. 6. The code contains a README file with the instructions to reproduce the experiments.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] The main details are presented in Sec. 5 and Sec. 6. Other details are provided in Appendix C and also available in the attached code.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] All results shown in Sec. 6.2 and Sec. 6.3 have the corresponding standard error and the inferences are supported by relevant statistical tests.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We include the description of the computational resources used for the runs in Appendix C.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We have used existing open source code for the Mario simulator and the A* agent, the Maze environment, and QD algorithm implementation. We have cited the creators either in Sec. 5 or in Appendix A. We obtained ACCEL agents directly from the original authors with their consent. We have cited the paper which introduced the algorithm and have acknowledged the authors for providing the pre-trained agents.
   (b) Did you mention the license of the assets? [Yes] The licenses to the external code and assets are added to the README file in the code that we provide with the supplemental material.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We include videos of the agent's playthrough in the supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]