Novelty Search for Deep Reinforcement Learning Policy Network Weights by Action Sequence Edit Metric Distance

Ethan C. Jackson
The University of Western Ontario
Vector Institute
ejacks42@uwo.ca

Mark Daley
The University of Western Ontario
Vector Institute
mdaley2@uwo.ca

ABSTRACT

Reinforcement learning (RL) problems often feature deceptive local optima, and methods that optimize purely for reward often fail to learn strategies for overcoming them [2]. Deep neuroevolution and novelty search have been proposed as effective alternatives to gradient-based methods for learning RL policies directly from pixels. We introduce and evaluate the use of novelty search over agent action sequences by Levenshtein distance as a means for promoting innovation. We also introduce a method for stagnation detection and population regeneration inspired by recent developments in the RL community [5], [1] that is derived from novelty search. Our methods extend a state-of-the-art method for deep neuroevolution using a simple genetic algorithm (GA) designed to efficiently learn deep RL policy network weights [6]. Results provide further evidence that GAs are competitive with gradient-based algorithms for deep RL in the Atari 2600 benchmark. Results also demonstrate that novelty search over agent action sequences can be effectively used as a secondary source of evolutionary selection pressure.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; **Genetic algorithms**;

KEYWORDS

deep reinforcement learning, genetic algorithms, novelty search

ACM Reference Format:
Ethan C. Jackson and Mark Daley. 2019. Novelty Search for Deep Reinforcement Learning Policy Network Weights by Action Sequence Edit Metric Distance. In *Genetic and Evolutionary Computation Conference Companion (GECCO '19 Companion), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3321956

1 INTRODUCTION

"*Can the history of actions performed by agents be used to promote innovative behaviour in benchmark RL problems?*" Towards answering this, we introduce two new methods for using Lehman and Stanley's *novelty search* [2] — an evolutionary framework for RL in which the reward signal is completely replaced by a behavioural metric — to train deep RL networks. The base algorithm is an approximate replication of Such et al.'s genetic algorithm (GA) for learning DQN network [4] weights. This is a very simple yet effective gradient-free approach for learning DQN policies that are competitive with those produced by deep Q-learning [6].

Novelty search has been shown to promote innovation in RL [6]. Here we introduce the use of approximate Levenshtein distance [3] — a form of *string edit metric distance* — as the behavioural distance function in two methods that directly apply or derive from novelty search for deep RL.

In contrast to previous uses of novelty search for deep RL, the characterization used here is highly general and *does not require environment-specific knowledge*: it can be used in any problem for which agent actions can be encoded as a sequence of discrete values.

2 SEED-BASED GENETIC ALGORITHM

The Base GA is an approximate re-implementation of Such et al.'s GA introduced in [6]. A network instance is defined by:

$$\Theta^n = \Theta^{n-1} + \sigma\epsilon(\tau_n) \tag{1}$$

$$\Theta^0 = \phi(\tau_0) \tag{2}$$

where Θ^n denotes network weights at generation n, τ denotes the encoding of Θ^n as a list of seeds, ϕ denotes a seeded, deterministic initialization function, $\epsilon(\tau_n) \sim \mathcal{N}(0,1)$ denotes a seeded, deterministic, normally-distributed pseudo-random number generator seeded with τ_n and σ denotes a constant scaling factor (mutation power). The GA does not implement crossover, and mutation appends a randomly-generated seed to τ. The GA performs *truncated selection* — a process whereby the top T individuals are selected as reproduction candidates (parents) for the next generation. From these T parents, the next generation's population is uniformly, randomly sampled with replacement, and mutated. Elitism is also used in conjunction with *validation episodes* to improve generalizability.

3 DQN ARCHITECTURE AND PREPROCESSING

We used the DQN neural network architecture [4] in all experiments: three convolutional layers with 32, 64, and 64 filters, respectively, followed by one dense layer with 512 units. The convolutional filter sizes are 8×8, 4×4, and 3×3, respectively. The strides are 4, 2, and 1, respectively. All weights are initialized using Glorot normal initialization. All network layer outputs use rectified linear unit (ReLU) activation. All game observations (frames) are downsampled to $84\times84\times4$ arrays. The third dimension reflects separate intensity channels for red, green, blue, and luminosity. Consecutive game observations are summed to rectify sprite flickering.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
GECCO '19 Companion, July 13–17, 2019, Prague, Czech Republic
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3321956

Hyperparameter	Method I	Method II
Population Size (N)	100 + 1	1,000 + 1
Generations	500	1000
Truncation Size (T)	20	20
Mutation Power (σ)	0.002	0.002
Archive Probability	0.1	0.01
Max Frames Per Episode (F)	20,000	20,000
Training Episodes	1	1
Validation Episodes	5	30
Improvement Generations (*IG*)		10

Table 1: Hyperparameters for all experiments. Population sizes are incremented to account for elites.

4 NOVELTY SEARCH

Novelty search requires two main components defined as follows. 1) We define the *behaviour characteristic* to be the sequence of discrete actions performed by an agent in response to consecutive environment observations. These action sequences are encoded as strings of length F, where F is the maximum number of frames available during training. Characters are either elements of a game's *action space* or the character x, which is reserved to encode a non-consumed frame. 2) We define the *behavioural distance function* as an approximation of the Levenshtein distance [3] between action sequences encoded by strings.

$$d(A, B) = \sum_{s=0}^{S-1} L(A_{sn \cdots sn+n-1}, B_{sn \cdots sn+n-1}) \tag{3}$$

where A and B are two action sequences encoded by strings, S is the number of segments, n is the length of each segment, and L computes the Levenshtein distance between two strings. The number of segments n is determined by computing $\lceil F/s \rceil$, where F is the number of characters in A and B, equal to the maximum number of frames available during training.

5 RESULTS AND CONCLUSIONS

All experiments use Assault, Asteroids, MsPacman, and Space Invaders. Source code and supplementary material are available at https://github.com/ethancjackson/NoveltySearchLevenshtein. Hyperparameters are shown in Table 1. For testing, 30 episodes not used in training or validation are used to evaluate the Base GA and either Method I or II. In tables, means (± standard deviations) are reported in game score units.

5.1 Method I

Method I implements a pure novelty search using the behaviour characteristic and behavioural distance function described in Section 4 and the GA as described in [6]. Learning results are summarized by Table 2.

Game	Base GA	Method I
Assault	**812 (± 228)**	488 (± 158)
Asteroids	**1321 (± 503)**	736 (± 426)
MsPacman	**2325 (± 351)**	1437 (± 527)
Space Invaders	500 (± 303)	474 (± 195)

Table 2: Method I experiment results. Bolded results denote significantly better testing scores ($p < 0.05$ in a two-tailed t-test). In terms of game score, the Base GA outperforms Method I in all but one game.

5.2 Method II

Method II applies novelty search to provide secondary selection pressure when premature convergence or *stagnation* is detected. Like Method I, this method also uses an archive of agent action sequences and the same behaviour characteristic and behavioural distance function as described in Section 4. In contrast, novelty scores are only computed when stagnation has occurred: defined as periods of *IG* generations with no net gains in validation performance. These scores are used to regenerate the population with individuals whose behaviours are maximally different from those in the stagnant population. Results using Method II are summarized by Table 3.

Game	Base GA	DQN	Method II
Assault	1219 (± 676)	**3359 (± 775)**	1007 (± 413)
Asteroids	1263 (± 590)	1629 (± 542)	1476 (± 640)
MsPacman	3385 (± 633)	2311 (± 525)	**3700 (± 209)**
Space Invaders	615 (± 323)	**1976 (± 893)**	1211 (± 244)

Table 3: Method II experiment results. DQN scores are taken from [4]. Bolded results denote significantly better testing scores ($p < 0.05$ in a two-tailed t-test) than other methods reported. Method II significantly improves learning in Space Invaders over the Base GA (also $p < 0.05$), and in MsPacman over both DQN and the Base GA.

5.3 Conclusions

Results suggest that novelty search by string edit metric distance on agent action sequences is likely not a suitable form of primary selection pressure for learning in Atari games. Conversely, results suggest that it provides an effective source of secondary selection pressure, making it a viable GA extension for deep RL.

REFERENCES

[1] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2019. Go-Explore: a New Approach for Hard-Exploration Problems. *arXiv:1901.10995* (2019).

[2] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[3] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10-8. 707–710.

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and Others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[5] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. 2018. Episodic curiosity through reachability. *arXiv preprint arXiv:1810.02274* (2018).

[6] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2017. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning. *arXiv preprint arXiv:1712.06567* (2017).