

VINE: An Open Source Interactive Data Visualization Tool for Neuroevolution

Rui Wang, Jeff Clune, and Kenneth O. Stanley

Uber AI Labs

San Francisco, CA 94103

{ruiwang,jeffclune,kstanley}@uber.com

ABSTRACT

Recent advances in *deep neuroevolution* have demonstrated that evolutionary algorithms, such as evolution strategies (ES) and genetic algorithms (GA), can scale to train deep neural networks to solve difficult reinforcement learning (RL) problems. However, it remains a challenge to analyze and interpret the underlying process of neuroevolution in such high dimensions. To begin to address this challenge, this paper presents an interactive data visualization tool called VINE (Visual Inspector for NeuroEvolution) aimed at helping neuroevolution researchers and end-users better understand and explore this family of algorithms. VINE works seamlessly with a breadth of neuroevolution algorithms, including ES and GA, and addresses the difficulty of observing the underlying dynamics of the learning process through an interactive visualization of the evolving agent’s behavior characterizations over generations. As neuroevolution scales to neural networks with millions or more connections, visualization tools like VINE that offer fresh insight into the underlying dynamics of evolution become increasingly valuable and important for inspiring new innovations and applications.

CCS CONCEPTS

- Computing methodologies → Genetic algorithms; Neural networks; Artificial life; Evolutionary robotics;

KEYWORDS

Neuroevolution, visualization, deep learning

ACM Reference Format:

Rui Wang, Jeff Clune, and Kenneth O. Stanley. 2018. VINE: An Open Source Interactive Data Visualization Tool for Neuroevolution. In *GECCO ’18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan*. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3205651.3208236>

1 INTRODUCTION

Recent progress in *deep neuroevolution* [3, 7, 9–11] has shown that evolutionary algorithms, such as evolution strategies (ES) and genetic algorithms (GA), are capable of training deep neural networks [4] with millions or more parameters (weights) to solve difficult reinforcement learning (RL) problems. Figure 1 illustrates one such popular problem, Mujoco Humanoid Locomotion, which both ES and GA solve effectively [3, 9].

GECCO ’18 Companion, July 15–19, 2018, Kyoto, Japan

© 2018 Uber Technologies, Inc.

This is the author’s version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *GECCO ’18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan*, <https://doi.org/10.1145/3205651.3208236>.

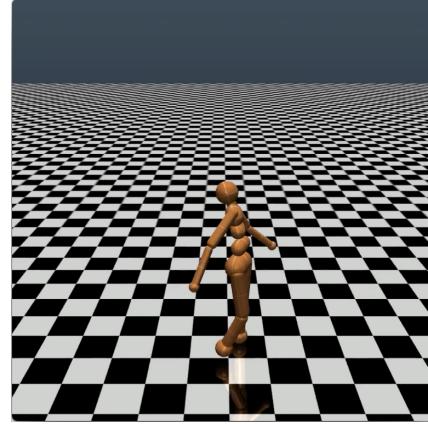


Figure 1: The Mujoco Humanoid Locomotion task. This benchmark is the basis of a number of examples in this paper and can be solved by both the ES and GA approaches to neuroevolution.

While it is possible to probe the properties of such algorithms, such as in recent investigations into the relationship of ES to finite-difference gradient approximation [6] and stochastic gradient descent [14], it is generally difficult to observe the underlying dynamics of the learning process in neuroevolution and neural network optimization. To address this gap and open up the process to observation, we introduce the *Visual Inspector for NeuroEvolution* (VINE), an interactive data visualization tool aimed at helping those who are interested in neuroevolution to better understand and explore its behavior. The source code for VINE is available at https://www.github.com/uber-common/deep-neuroevolution/tree/master/visual_inspector. We hope this technology will inspire new understanding, innovations, and applications of neuroevolution in the future.

VINE can illuminate both ES- and GA-style approaches. In this paper, we focus on visualizing the result of applying ES to the Mujoco Humanoid Locomotion [2, 12] task from Figure 1.

2 USING VINE

In the conventional application of the version of ES popularized by OpenAI [10], a group of neural networks called the pseudo-offspring cloud are optimized against an objective over generations. The parameters of each individual neural network in the cloud are generated by randomly perturbing the parameters of a single “parent” neural network. Each pseudo-offspring neural network is then evaluated against the objective: in the Humanoid Locomotion task,

each pseudo-offspring neural network controls the movement of a robot, and earns a score called its fitness based on how well it walks. The ES constructs the next parent by aggregating the parameters of pseudo-offspring based on these fitness scores (almost like a sophisticated form of multi-parent crossover, and also reminiscent of stochastic finite differences). The cycle then repeats. The full details of this technique are formalized in [10].

To take advantage of VINE, behavior characterizations (BCs) [8] for each parent and all pseudo-offspring are recorded during evaluation. Here, a BC can be any property of the agent's behavior when interacting with its environment. For example, in the Mujoco Humanoid Locomotion task we simply use the agent's final $\{x, y\}$ location as the BC, which indicates how far the agent has moved away from the origin and to what location.

The visualization tool then maps parents and pseudo-offspring onto 2D planes according to their BCs. For that purpose, it invokes a graphical user interface (GUI), whose major components consist of two types of interrelated plots: one or more pseudo-offspring cloud plots (on separate 2D planes), and one fitness plot. Illustrated in Figure 2, a pseudo-offspring cloud plot displays the BCs for the parent and pseudo-offspring in the cloud for every generation, while a fitness plot displays the parent's fitness score curve as a key indicator of progress over generations.

Users then interact with these plots to explore the overall trend of the pseudo-offspring cloud as well as the individual behaviors of any parent or pseudo-offspring over the evolutionary process: (1) users can visualize parents, top performers, and/or the entire pseudo-offspring cloud of any given generation, and explore the quantitative and spatial distribution on the 2D BC plane of pseudo-offspring with different fitness scores; (2) users can compare between generations, navigate through generations to visualize how the parent and/or the pseudo-offspring cloud is moving on the 2D BC plane, and how such moves relate to the fitness score curve (as illustrated in Figure 3, a full movie clip of the moving cloud can be generated automatically); (3) clicking on any point on the cloud plot reveals behavioral information and the fitness score of the corresponding pseudo-offspring.

3 ADDITIONAL USE CASES

The tool also supports advanced options and customized visualizations beyond the default features. For example, instead of just a single final $\{x, y\}$ point, the BC could instead be each agent's full trajectory (e.g., the concatenated $\{x, y\}$ for 1,000 time steps). In that case, where the dimensionality of the BC is above two, dimensionality reduction techniques (such as Principal Components Analysis (PCA) [5] or t-Distributed Stochastic Neighbor Embedding (t-SNE) [13]) are needed to reduce the dimensionality of BC data to 2D. Our tool automates these dimensionality-reduction procedures.

The GUI is capable of loading multiple sets of 2D BCs (perhaps generated through different reduction techniques) and displaying them in simultaneous and connected cloud plots, as demonstrated in Figure 4. This capability provides a convenient way for users to explore different BC choices and dimensionality reduction methods. Furthermore, users can also extend the basic visualization with customized functionality. Figure 4 exhibits one such customized

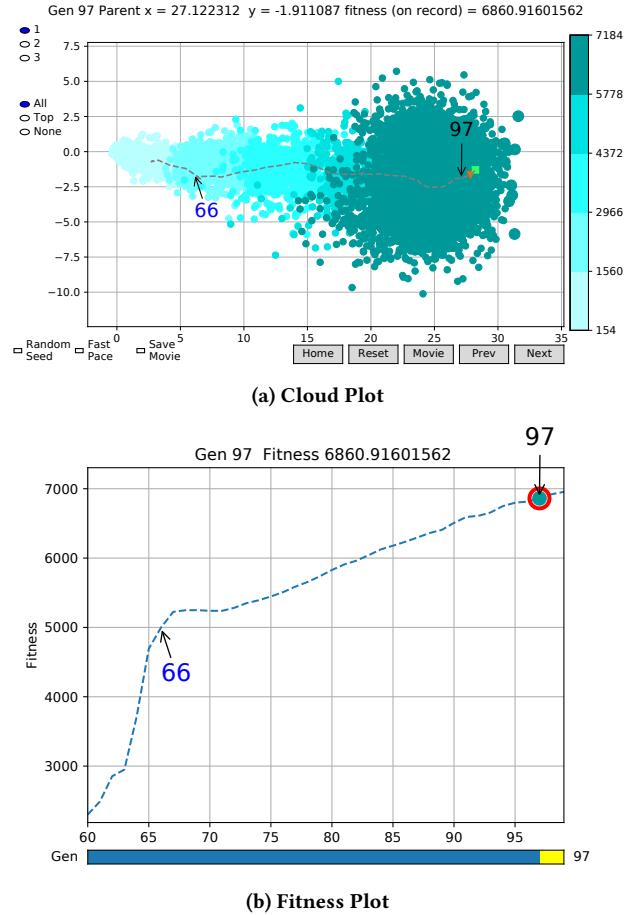


Figure 2: Examples of a pseudo-offspring cloud plot and a fitness plot.

cloud plot that can display certain types of domain-specific high-dimensional BCs (in this case, an agent's full trajectory) together with the corresponding reduced 2D BCs. Another example of a customized cloud plot, in Figure 5, allows the user to replay the agent's deterministic or stochastic behavior that results when it interacts with an environment.

The tool is also designed to work with domains other than locomotion tasks. Figure 6 demonstrates a cloud plot that visualizes ES agents trained to play Frostbite, one of the Atari 2600 games [1], where we use the final emulator RAM state (integer-valued vectors of length 128 that capture all the state variables in a game) as the BC and apply PCA to map the BC onto a 2D plane.

The plot shows that as evolution progresses, the pseudo-offspring cloud shifts towards the left and clusters there. The ability to see the corresponding video of each of these agents playing the game lets us infer that each cluster corresponds to semantically meaningful and distinct end states.

VINE also works seamlessly with other neuroevolution algorithms such as GAs, which maintain a population of offspring over generations. In fact, the tool works independently of any specific neuroevolution algorithm. Users only need to slightly modify their

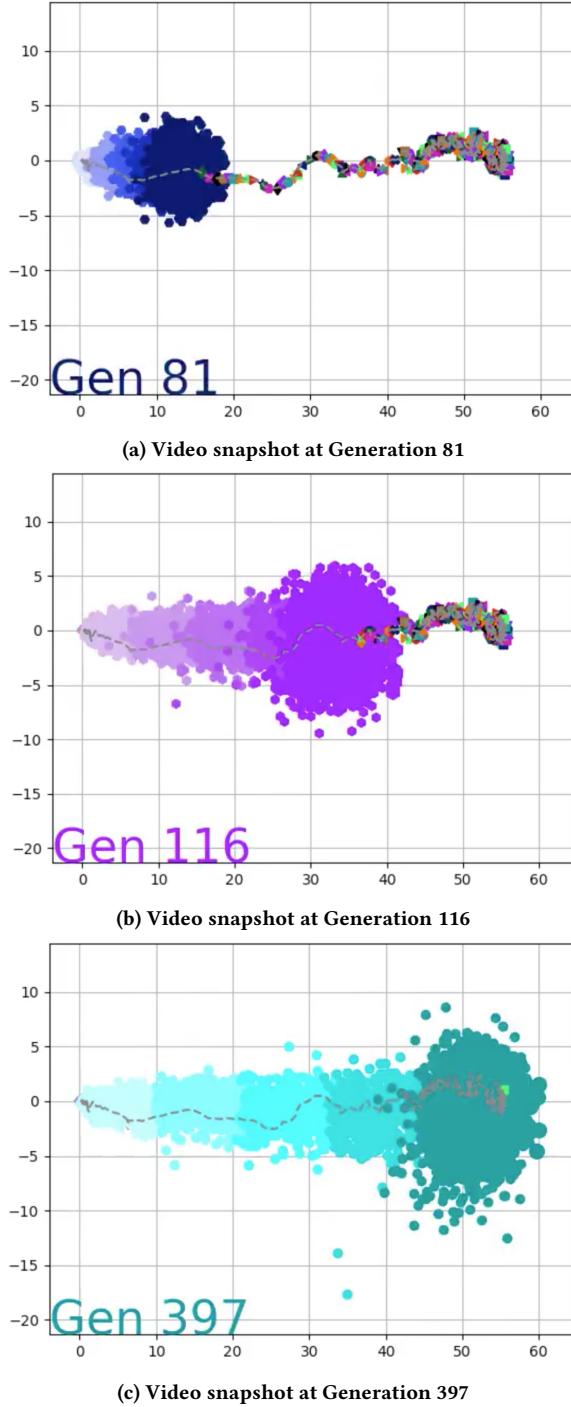


Figure 3: Frames taken from a VINE-generated video visualizing the evolution of behaviors over generations in Humanoid Walking. The color changes in each generation. Within a generation, the color intensity of each pseudo-offspring is based on the percentile of its fitness score in that generation (aggregated into five bins). The position of each point corresponds to the endpoint of an individual walker (which was the BC in this example).

neuroevolution code to save the BCs they pick for their specific problems. In the code release, we provide such modifications to our ES and GA implementations as examples.

4 CONCLUSION

Because evolutionary methods operate over a set of points, they present an opportunity for new types of visualization. Having implemented a tool that provides visualizations we found useful, we wanted to share it with the machine learning community so all can benefit. As neuroevolution scales to neural networks with millions or more connections, gaining additional insight through tools like VINE is increasingly valuable and important for further progress.

ACKNOWLEDGEMENTS

We thank Uber AI Labs, in particular Joel Lehman, Xingwen Zhang, Felipe Petroski Such, and Vashisht Madhavan for valuable suggestions and helpful discussions.

REFERENCES

- [1] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 47:253–279.
- [2] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.
- [3] Conti, E., Madhavan, V., Petroski Such, F., Lehman, J., Stanley, K. O., and Clune, J. (2017). Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. *arXiv preprint arXiv:1712.06560*.
- [4] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- [5] Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441.
- [6] Lehman, J., Chen, J., Clune, J., and Stanley, K. O. (2017a). ES is more than just a traditional finite-difference approximator. *arXiv preprint arXiv:1712.06568*.
- [7] Lehman, J., Chen, J., Clune, J., and Stanley, K. O. (2017b). Safe mutations for deep and recurrent neural networks through output gradients. *arXiv preprint arXiv:1712.06563*.
- [8] Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223.
- [9] Petroski Such, F., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. (2017). Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*.
- [10] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. (2017). Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints*, 1703.03864.
- [11] Stanley, K. O. (2017). Neuroevolution: A different kind of deep learning. *O'Reilly Online*, July 13.
- [12] Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *IROS*, pages 5026–5033. IEEE.
- [13] van der Maaten, L. and Hinton, G. (2008). Visualizing high-dimensional data using t-SNE. *The Journal of Machine Learning Research*, 9(Nov):2579–2605.
- [14] Zhang, X., Clune, J., and Stanley, K. O. (2017). On the relationship between the openai evolution strategy and stochastic gradient descent. *arXiv preprint arXiv:1712.06564*.

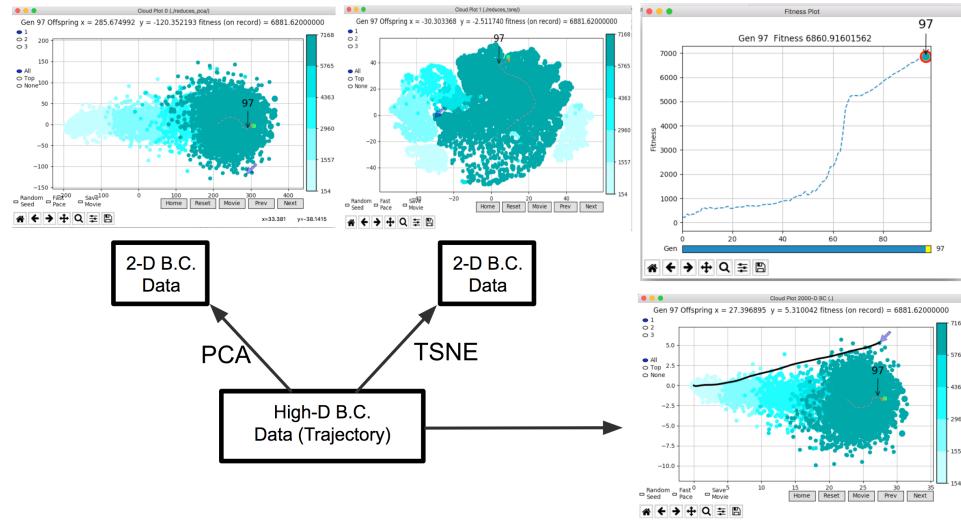
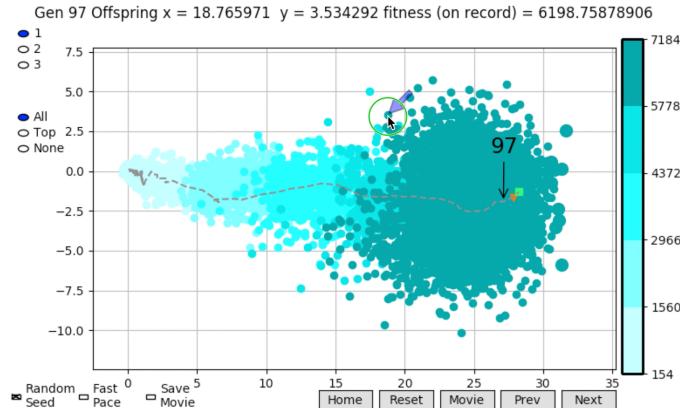
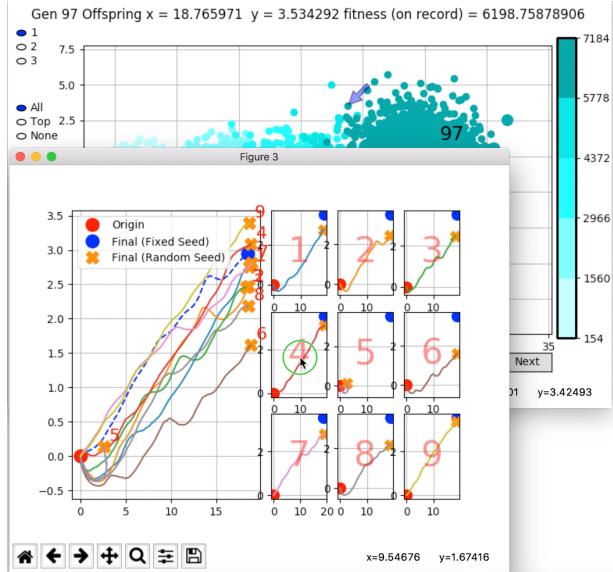


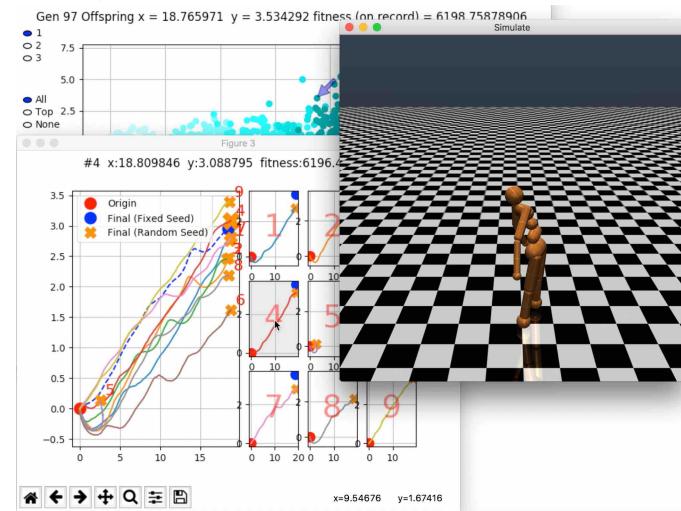
Figure 4: Visualizations of multiple 2D BCs and a high-dimensional BC along with a fitness plot. The three cloud plots show the same pseudo-offspring, but with their high-dimensional BCs reduced through different dimensionality reduction techniques, giving multiple perspectives on the space as it is searched.



(a) Right click a pseudo-offspring to invoke nine stochastic roll-outs.



(b) Right click one of the trajectories as a result of nine roll-outs.



(c) Visualize the agent's behavior that corresponds to the trajectory in (b).

Figure 5: Users can view videos of any agent's deterministic and stochastic behaviors through a video pop-up (at bottom).

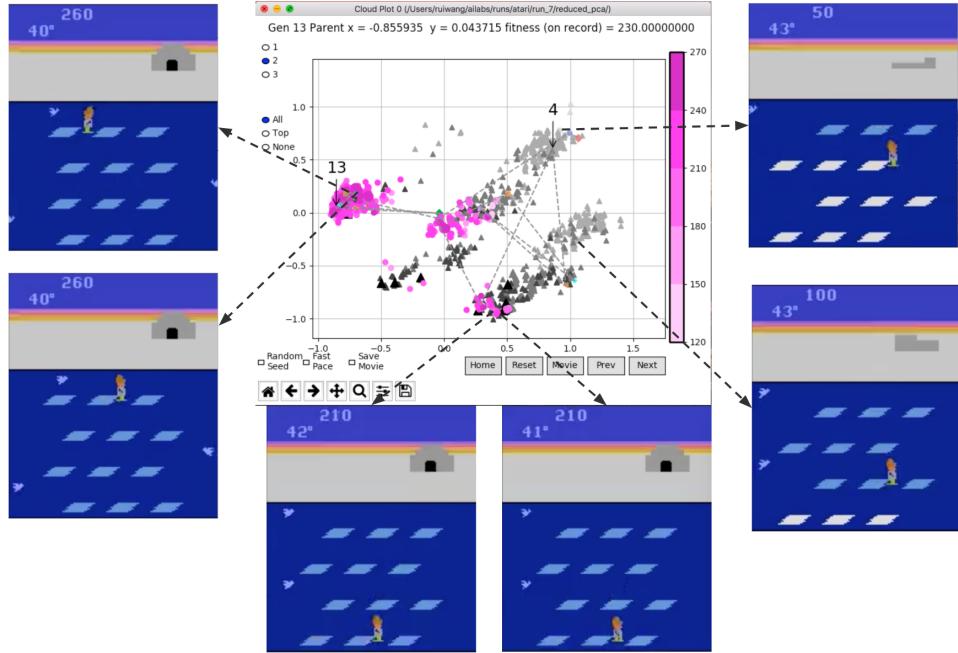


Figure 6: Visualizing agents learning to play Frostbite. Each point is a 2D reduction of a high-dimensional representation of the end-state of the game for a particular pseudo-offspring. Users can click on any point to see the rollout of the game that leads to this endpoint, revealing the underlying semantics of the space.