Evolving HyperNetworks for Game-Playing Agents

Christian Carvelli
modl.ai
chris@modl.ai

Djordje Grbic
IT University of Copenhagen
djgr@itu.dk

Sebastian Risi
IT University of Copenhagen
sebr@itu.dk

ABSTRACT

This work investigates the evolution of indirectly-encoded neural networks through a hypernetwork approach. We find that for some Atari games, a hypernetwork with over 14 times fewer parameters, can compete or even outperform directly-encoded policy networks. While hypernetworks perform worse than directly encoded networks in the game Frostbite, in the game Gravitar, the approach reaches a higher score than any other evolutionary method and outperforms complicated deep reinforcement learning setups such as Rainbow.

KEYWORDS

Neuroevolution, indirect encoding, hypernetworks

ACM Reference Format:
Christian Carvelli, Djordje Grbic, and Sebastian Risi. 2020. Evolving HyperNetworks for Game-Playing Agents. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389961

1 INTRODUCTION

Evolving deep neural networks (i.e. deep neuroevolution) has recently shown to be a competitive alternative to deep reinforcement learning in a variety of different domains [6, 8]. Interestingly, recent breakthroughs in this area came from pairing a very simple genetic algorithm (GA) with a deep neural network where the network's weights are all mutated at the same time through additive Gaussian noise [5, 8]. It is an open question how more advanced neuroevolution algorithms, such as indirect encodings, could further improve on the performance of these simple GAs. In an indirect encoding, not every parameter is encoded separately in the genotype but a genotypic bottleneck forces reuse of information in the decoding from genotype to phenotype.

A particularly promising indirect encoding is HyperNEAT [7], in which the connectivity pattern of a policy network is described by an evolved function that takes as input the geometry of a network and outputs the weights of a policy network. However, one potential drawback of HyperNEAT is that the location of nodes need to be decided by an experimenter or require extensions such as evolvable-substrate HyperNEAT [4]. While the placement of the network's inputs and outputs might be straightforward in a domain such as robot locomotion, in which it is easy to correlate sensors with their position in space, it is less obvious how the geometry of a network for games such as Atari should look like.

In this work we investigate a method without the aforementioned limitation. Hypernetworks [1], a recent HyperNEAT-inspired encoding that does not require nodes to have locations in space, has shown promise when being optimized end-to-end through backpropagation for supervised learning tasks. However, it is currently not clear how evolvable hypernetworks are, which is the focus of this paper.

2 EVOLVING HYPERNETWORKS

A hypernetwork [1] is an indirect encoding that generates the parameters for a policy network using a smaller network, the "hypernetwork", and a set of learned parameters called "embeddings" (Fig. 1). In a typical policy network, the parameters of each layer are represented by a tensor K^j of arbitrary size and shape, for each layer $j = 1, ..., D$. Following Ha et al. [1] for each layer j the hypernetwork receives an embedding $z^j \in \mathbb{R}^{N_z}$ as input and generates the weights for that particular layer K^j: $K^j = g(z^j), \forall j = 1, ..., D$.

To limit the number of hypernetwork parameters and its number of outputs, the hypernetwork generates a small set of weights at a time, called "tile", one for each z vector. The number of tiles needed for a layer j is computed as $T^j = \frac{K^j_{size}}{t}$, where K^j_{size} is the number of parameter of the layer j and t is the tile size (an hyperparameter).

Each embedding z^j_i, the i-th embedding of the j-th layer, is linearly projected in an intermediate vector a^j_i that will be projected into the final tile T^j_i. Finally, each group of T^j tiles can be concatenated, reshaped and trimmed to fit into K^j:

$$\begin{array}{ll} a^j_i = W_i z^j + B_i & \forall i = 1, ..., T^j, \forall j = 1, ..., D \\ K^j_i = \langle W_out, a^j_i \rangle + B_{out} & \forall i = 1, ..., T^j, \forall j = 1, ..., D \\ K^j = (K^j_1 K^j_2 ... K^j_i ... K^j_{N_i n}) & \forall j = 1, ..., D \end{array}$$

The policy network in this paper is a deep convolutional network with a dense head and ReLU non-linearity, based on the original DQN architecture [3]. The hypernetwork is a simple dense network with a single hidden layer with 8 neurons and ReLU non-linearity. The embedding size is 32, with a total number of 1199 embeddings.

3 EXPERIMENTS

We compare the hypernetwork setup (43240 trainable parameters) with a directly-encoded network with the traditional DQN architecture (612018 parameters) and against a version where all the inner layers have been scaled down by a constant factor of 0.25 (41658 parameters). For the hypernetwork approach, Gaussian noise is either applied to the hypernetwork weights or the embeddings, with 50/50 probability. In the case of embeddings, a layer in the policy network is randomly chosen and only those embeddings are mutated. For the direct encoding, Gaussian noise is added to all the parameters of the policy network [8].

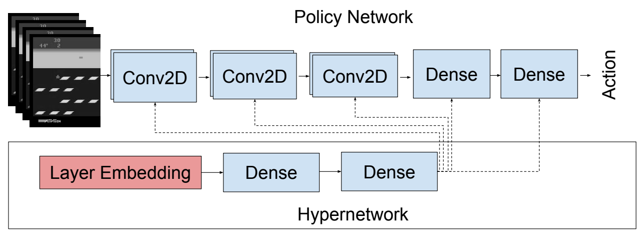


Figure 1: Hypernetwork Architecture. The hypernetwork produces the weights for each layer in the policy network, conditioned on different layer embeddings.

Experimental details. The GA evolves a population of 1000 individuals, evaluating them on one episode capped at 20K frames. The best 20 individuals are evaluated 30 more times to find the true elite. We experimented with two different parent selection strategies, where a parent is either: (1) uniform randomly selected from the best 20 individuals of the previous generation (**top**), or (2) the best of two randomly selected individuals determined via tournament selection (**2way**). The mutation operator applies additive Gaussian noise drawn from $\mathcal{N}(0, 0.002)$. The directly encoded individuals are initialized using the Xavier initialization for the weights and zero biases. Indirectly encoded networks are initialized with Kaiming normal initialization for the weights and zero for the biases, while the initial z vector is drawn from a normal distribution $\mathcal{N}(0, 0.1)$.

4 RESULTS & DISCUSSION

On the three games tested (Amidar, Frostbite, Gravitar), the hypernetwork approach performs equally well to a direct encoding in one game, reaches a lower performance in another, and outperforms evolutionary methods and state-of-the-art RL methods such as Rainbow in the game Gravitar (Table 1). Interestingly, 2-way tournament selection works better for indirectly encoded networks but penalizes the direct encodings in most cases.

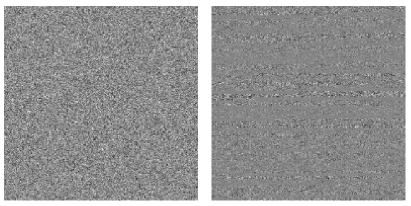
The small-sized direct encoding did perform slightly worse than the full-sized version on all three games. The indirect encoding performed worse than the normal and small direct encoding on Frostbite but managed to outperform the small directly encoded network (which has around the same number of parameters as the hypernetwork) in two out of three tests. Importantly, it also outperformed the full-sized network, which has over 14 times the number of parameters, in Gravitar and performed as well in Amidar.

Fig. 2 shows evolved and generated weights for the kernels of the last convolutional layer (conv3) in the champion agent for Gravitar. The kernels, concatenated in row-major order, can be recognized as groups of 3×3 pixels in the images. In the indirect encoding, the hypernetwork generates multiple "tiles" that are concatenated together, reshaped, and eventually cut to fit the appropriate layer. While it is difficult to distinguish particular features in these kernels, it is interesting how the hypernetwork-encoded kernels show a greater regularity and structure than the directly encoded ones.

This work opens interesting future research directions in indirect encodings, which include studying in more depth what types of tasks they excel in and how compression through a genotypic bottleneck can hurt or improve performance in different settings.

	Rainbow[2]	Direct		Direct small		Hypernet	
Frames	200M	500M		500M		500M	
Selection		top	2way	top	2way	top	2way
Amidar	**5,131.2**	372	263	270	244	270	364
Frostbite	**9,590.5**	5,926	3,456	5,880	1,178	2,520	3,719
Gravitar	1,419.3	732	476	590	875		**2,009**

Table 1: Testing Performance. We take the best agent found during three independent evolutionary runs and report its average performance across 200 episodes.



(a) Conv3 direct (b) Conv3 HN

Figure 2: Weight Patterns. Comparison of the kernels of the last convolutional layer of the direct encoding (a) and encoded by an evolved hypernetwork (b).

ACKNOWLEDGEMENTS

DG and SR were supported by the Lifelong Learning Machines program from DARPA/MTO under Contract No. FA8750-18-C-0103. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. We thank Mads Sønderstrup for his help during the start of this project.

REFERENCES

[1] Ha, D., Dai, A., and Le, Q. V. Hypernetworks. *arXiv preprint arXiv:1609.09106* (2016).
[2] Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence* (2018).
[3] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature 518*, 7540 (2015), 529–533.
[4] Risi, S., and Stanley, K. O. An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons. *Artificial life 18*, 4 (2012), 331–363.
[5] Risi, S., and Stanley, K. O. Deep neuroevolution of recurrent and discrete world models. In *Proceedings of the Genetic and Evolutionary Computation Conference* (2019), pp. 456–462.
[6] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).
[7] Stanley, K. O., D'Ambrosio, D. B., and Gauci, J. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life 15*, 2 (2009), 185–212.
[8] Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567* (2017).