Evo-NAS: Evolutionary-Neural Hybrid Agent for Architecture Search

Maziarz et al. [2], reviewed by Keyvan Beroukhim, Nicolas Buton and Laura Nguyen

Sorbonne Université Sciences, M2 DAC

Introduction

Neural Networks have yielded success in many supervised and unsupervised learning problems. However, designing state-of-the-art deep learning models – especially neural architectures – requires a considerable amount of human time and expertise.
To tackle the challenge of automating neural networks design, several approaches have been developed. Deep Reinforcement Learning based agents have shown promising results in learning complex architectural patterns and exploring vast and compositional search spaces. On the other hand, evolutionary algorithms have proven to be more sample efficient, which is a significant asset for such a resource intensive application. Both methods generate architectures that rival the best human-invented ones. Maziarz et al. [2] introduce a class of Evolutionary-Neural hybrid agents designed to retain the best qualities of both methods.

Baselines

Random Search (RS). Generates a new model by sampling every architectural choice from a uniform distribution over the available values.

Neural Architecture Search (RL Agent). Zoph and Le [5] use an RNN as a controller to compose a variable-length sequence of architectural choices that defines a generated neural network. The resulting model is then trained on a downstream task, and its accuracy on a validation set serves as the reward for training the agent using policy gradient.

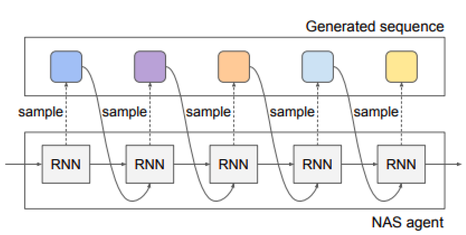


Figure: How the RL agent samples an architecture.

Aging Evolution Architecture Search (Evolutionary Agent). Real et al. [3] modify the tournament selection algorithm by introducing an age property to favor the younger genotypes. A population of P generated models is improved in iterations. At each iteration, a sample of S models is selected at random and the best one is randomly mutated to produce a new model. The resulting model is trained and replaces the oldest one in the population.

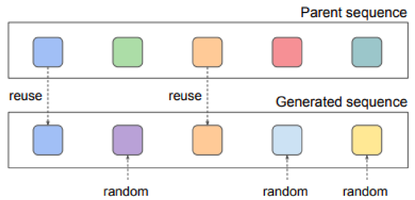


Figure: How the Evolutionary agent samples an architecture.

Evolutionary-Neural Architecture Search (Evo-NAS)

Using an RNN, the Evo-NAS agent generates new architectures by mutating a parent model. At each iteration, the parent model is chosen by picking the model with the highest reward among S samples randomly drawn from the population of the P most recent models. Each architectural choice in the mutated sequence defining the child model is either reused from the parent sequence with probability $1-p$, or sampled from the policy learned with probability p.

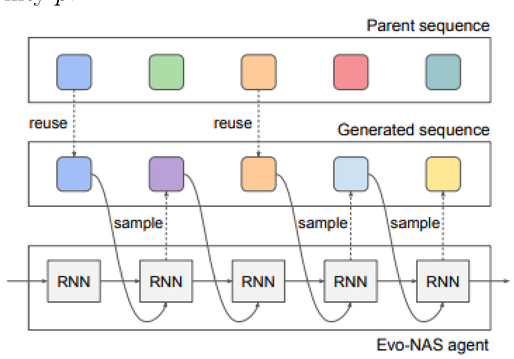


Figure: How the Evo-NAS samples an architecture.

The RNN is trained with policy gradient to maximize the expected quality metric achieved by the generated models on the downstream task.
The authors conjecture that Evo-NAS retains the sample-efficiency of Evolutionary agents (biases the search toward the area around the best samples found so far), and the ability to learn complex patterns of RL agents (learns to restrict its local search to the manifold learned by the RNN).

Training algorithms

Reinforce [4]. Reinforce is a standard policy-gradient training algorithm, often considered the default choice for applications where an agent needs to be trained to explore a complex search space to find the optimal solution.

Priority Queue Training (PQT) [1]. With PQT, the gradients are generated to directly maximize the log likelihood of the best samples produced so far. PQT has higher sample efficiency than Reinforce, as good models generate multiple updates.

Results

We compare the 4 agents on a synthetic task, *Learn To Count*, and on NASBench, where the agents search for network architectures for CIFAR-10 classification task. Additionally, we consider one version per training algorithm for RL and Evo-NAS agents.

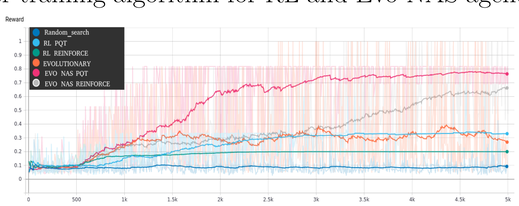


Figure: Reward on Learning To Count task.

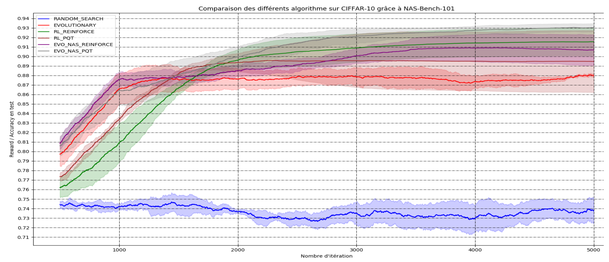
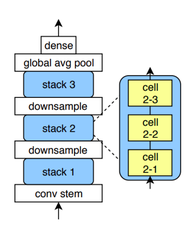
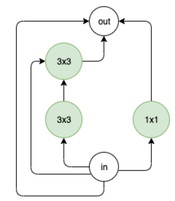


Figure: Reward on NASBench task averaged over 5 replicas.



(a) Outer skeleton of each model on NASBench.



(b) Best cell generated by Evo-NAS on NASBench (green : conv + batchNorm + ReLU).

References

[1] Daniel A Abolafia et al. "Neural program synthesis with priority queue training". In: *arXiv preprint arXiv:1801.03526* (2018).

[2] Krzysztof Maziarz et al. "Evolutionary-neural hybrid agents for architecture search". In: *arXiv preprint arXiv:1811.09828* (2018).

[3] Esteban Real et al. "Regularized evolution for image classifier architecture search". In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 33. 2019, pp. 4780–4789.

[4] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[5] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (2016).