UDC 004.031.2

EVOLUTION STRATEGY FOR POLICY SEARCH IN ROBOTICS

Borys Tymchenko
Svitlana Antoshchuk, Sc. D
Odessa National Polytechnic University, UKRAINE

ABSTRACT. At the introduced article, we provide an enhancement to evolution strategy to generate more robust sub-optimal policies in reinforcement learning tasks. Our solution is applicable to the autonomous vehicles researches, where is a need of designing or detecting complex patterns in visual features and control policies, that cannot be described by human researcher.

Introduction. Generally, in robotics applications, precise algorithms and task decomposition is used e.g., autonomous vehicles detecting lanes, planning path and executing control tasks. A recent trend in various researches is to apply machine learning algorithms. Bojarsky et al. [3] have shown, that deep convolutional neural network (CNN) can successfully act as a driver for an autonomous vehicle. However, learned reaction-based drivers cannot serve in real conditions due to their unpredictability. It limits possible use cases for machine learning usage in safety-critical applications.

Nevertheless, results, achieved with usage of reinforcement learning applied to neural networks, can unveil new important visual features and/or control policies.

The aim of this work is to develop a learning method for end-to-end car controller in a simulated environment for future use as salient feature patterns detector.

Recent discoveries [1, 2] show, that DNNs can be trained with evolutional strategies (ES), that can be viewed as a black-box optimization instead of SGD-like methods. ES is a more broad term than just genetic algorithms (GA). Structure of agent's neural network (fig. 2) was chosen as reduced version of [3] because our environment is less sophisticated (smaller input size, fixed lighting, no alien cars, pedestrians, etc.)

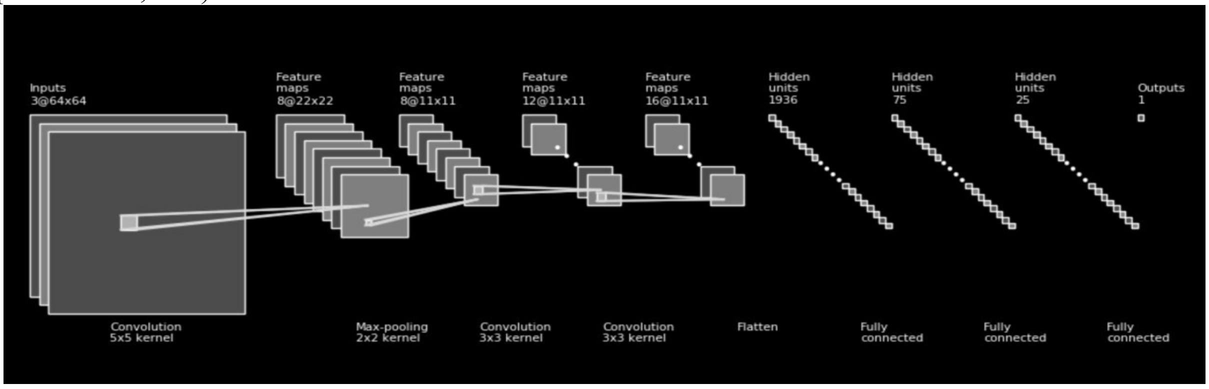

Figure 1. Agent's neural network structure

For genome representation, we reshaped every neural network layer's weights into a linear array and concatenated them together. We store genes as real values. We use standard GA scheme with tournament selection and safe mutations through rescaling (SM-R)[4]. In addition, we used *elitism* to preserve best agents and not lose optimal solution during evolution process.

Main idea of SM-R is to make offspring neural network behave similarly to it's parent. To achieve this, mutation is represented as noise vector and magnitude scalar. Noise vector is an unimodal normal Gaussian noise. Magnitude scalar is then found with line search to desired neural network behaviour divergence. Divergence is represented as euclidian distance between parent's and offspring's responds to a recorded parent's experience.

As a simulated environment, we use TORCS [5] racing simulator (Fig. 2). We use high-dimensional input space, namely an array of RGB pixels from front-facing camera on agent. As outputs, we consider steering angle normalized to range [-1, 1]. Agents have no prior knowledge about car physics or environment.Each agent's fitness is represented directly as sparse reward:

$$R = \sum v_x cos(\theta) - v_x sin(\theta) - k|\alpha|$$

where R is sparse reward that is given in the end of episode, θ is deviation angle and v_x is linear velocity, α is steering angle and *k* is a balancing constant bound to simulation. Episode is considered as ended, when car collides with track edge, turns around, or finishes a lap successfully.

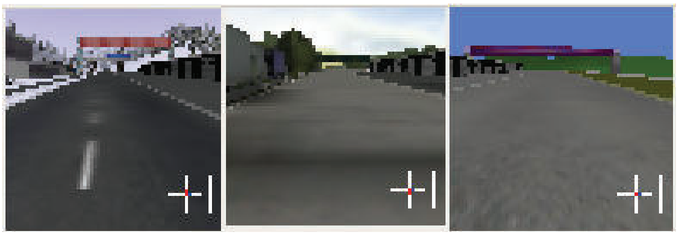
Figure 2. Examples of TORCS tracks as seen by agent

To train agents, we evolved them for 250 generations in populations of size 200 on different tracks. To check the robustness of trained agents and verify, that learned policies are not overfit to a particular track, we tested policies on different tracks. Tests show, that 58% of agents can finish a lap in 4 of 5 tracks. The most difficult track was “Alpine 2”, because there are winter conditions and every rash movement can lead to skidding. It was completed only by the agent evolved on this track.

Verified agents robustness shows, that agents learn general concepts about driving and car dynamics. This allows us to use parts of agents’ neural networks as detectors for specific road features, such as lanes.

In addition, we can train neural networks with fixed convolution part in specific environment in order to learn specific movement model. It is considered as future research.

Conclusion. In a present article, an approach of evolving neural network policies in simulated environment was presented. Tests show robustness and high fitness of agents. The fact, that agents can work in different environments shows that learned salient features and patterns can be used as parts of another computer vision and control algorithms.

BIBLIOGRAPHY

1. ES Is More Than Just a Traditional Finite-Difference Approximator [Electronic source]. – Access mode: URL: *https://arxiv.org/abs/1712.06568 – Name from screen*

2. Deep Neuroevolution: Genetic Algorithms are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning [Electronic source]. – Access mode: URL: *https://goo.gl/7kf9z1 – Name from screen*

3. *End to End Learning for Self-Driving Cars* [Electronic source]. – Access mode: URL: *https://arxiv.org/abs/1604.07316 – Name from screen*

4. *Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients* [Electronic source]. – Access mode: URL: *https://arxiv.org/abs/1712.06563 – Name from screen*

5. *Gym TORCS.* [Electronic source]. – Access mode: URL: *https://goo.gl/Ha3xWg – Name from screen*