NeuroEvolution of a Temporal Policy for Autonomous Driving in CARLA

Manuel Di Lullo
Sapienza University of Rome
dilullo.2047615@studenti.uniroma1.it

November 22, 2025

Abstract

We present a NeuroEvolution-based framework for training an autonomous driving agent in the CARLA simulator. As part of the Automated Verification of Intelligent Systems (AVIS) curriculum, this project explores gradient-free optimization as an alternative to traditional reinforcement learning. We propose a lightweight "Simple Temporal Model" that integrates short-term memory to handle state estimation without the computational overhead of complex Recurrent Neural Networks (RNNs). To accelerate the evolutionary process, we implement a batch training architecture allowing the parallel evaluation of 4 vehicles simultaneously. We report experimental results showing that Natural Evolution Strategies (NES) can successfully evolve a policy capable of navigation and obstacle avoidance, achieving a peak fitness score of 16,486 at generation 64.

1 Introduction

Autonomous driving (AD) represents a critical challenge in intelligent systems verification. The high-dimensional state space and the severe consequences of failure require robust control policies. While Deep Reinforcement Learning (DRL) methods such as PPO or DQN are standard, they often suffer from sample inefficiency and difficult hyperparameter tuning due to the reliance on gradient backpropagation through complex value function approximators.

In this work, we explore Natural Evolution Strategies (NES), a black-box optimization method that evolves a population of parameters towards higher rewards without calculating gradients. This approach adheres to the "Keep It Simple, Stupid" (KISS) philosophy, favoring lightweight architectures over deep, opaque networks.

The contribution of this paper is twofold:

1. The design of a **Simple Temporal Model** that captures sequential dependencies in driving dynamics using a memory buffer rather than gated recurrent units.

2. The implementation of a **Batch Training System** in CARLA 0.9.15, which evaluates multiple agents (4 vehicles) in parallel, significantly reducing the wall-clock time required for evolutionary convergence.

2 Related Work

Reinforcement Learning in CARLA. The CARLA simulator [?] is the de facto standard for open-source AD research. Typical approaches involve end-to-end learning where Convolutional Neural Networks (CNNs) map pixels to control commands. However, these methods are computationally expensive and opaque to verification.

NeuroEvolution. Evolutionary strategies have shown competitive performance against DRL in continuous control tasks. Salimans et al. (OpenAI) demonstrated that NES could rival DRL in Atari games and MuJoCo tasks while offering better parallelization properties. Unlike Genetic Algorithms (GA) which rely on crossover, NES estimates a search gradient using a population of perturbed parameters, making it mathematically more grounded for continuous parameter spaces.

3 Method

3.1 The Simple Temporal Model

To navigate effectively, an agent must understand not just its current position, but its momentum and recent history. Standard linear models fail to capture this. Instead of using computationally heavy LSTMs, we propose a custom *Simple Temporal Model*.

The model maintains a memory buffer M_t of the past k actions. The input state vector S_t is concatenated with this memory. The update rule uses a decay factor λ to smooth temporal transitions:

$$M_t = (1-\lambda)\cdot A_{t-1} + \lambda \cdot M_{t-1} \tag{1}$$

where A_{t-1} represents the previous control actions. This architecture allows the agent to "remember" a turn initiated in the previous timestep, preventing erratic steering oscillation.

The input state consists of 10 high-level features extracted from the simulator: speed, distance/angle to destination, distance/angle to next waypoint, collision flag, lane invasion status, and obstacle distance.

3.2 Natural Evolution Strategies (NES)

We optimize the model parameters θ using NES. At each generation g, we generate a population of N candidates by adding Gaussian noise ϵ to the current parameters:

$$\theta_i = \theta_g + \sigma\epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, I) \tag{2}$$

where σ is the exploration noise (sigma). Each candidate is evaluated to obtain a fitness score F_i. The parameters are updated via:

$$\theta_{g+1} = \theta_g + \alpha \frac{1}{N\sigma} \sum_{i=1}^{N} F_i \epsilon_i \tag{3}$$

where α is the learning rate. This effectively moves the policy in the direction of higher fitness.

4 Training Procedure

4.1 Environment Setup

The training is conducted in CARLA 0.9.15. To manage the computational load, we implemented a **Batch Vehicle Trainer**. This system spawns 4 vehicles simultaneously in the same synchronous simulation world. The NES optimizer distributes the population of parameters across these vehicles. This parallelization provides a theoretical $4\times$ speedup in data collection, bounded only by the rendering capabilities of the host machine.

4.2 Reward Function

To guide the evolution, we designed a composite reward function:

$$R_{total} = R_{wp} + R_{move} - P_{coll} - P_{offroad} - P_{stuck} \tag{4}$$

- R_{wp}: Large bonus (+100) for reaching a new waypoint.
- R_{move}: Small continuous reward for velocity maintenance.
- P_{coll}: Heavy penalty (-50) for collisions.
- P_{stuck}: Penalty for failing to advance > 3 meters in 50 steps.

5 Results

The training was conducted using the configuration described in Table 1. The system was initialized with a population size of 50 individuals.

5.1 Experimental Setup

The critical configuration for this experiment was the **Batch Size of 4**. By evaluating four vehicles concurrently, we maximized the utility of the NES algorithm, which naturally supports parallelization.

Table 1: Training Configuration and Best Model Statistics

Parameter	Value
Environment	
Simulator	CARLA 0.9.15
Batch Size	4 Vehicles
NPC Vehicles	0
NES Hyperparameters	
Population Size	50
Sigma (σ)	0.1
Learning Rate (α)	0.01
Sigma Decay	0.995
Performance Results	
Total Generations Logged	103
Best Generation	**64**
Peak Fitness	**16,486.00**
Completion Rate (Peak)	24.0%

5.2 Analysis

The training logs (visualized in summary Table 1) reveal the learning trajectory.

- **Early Phase (Gen 1-20):** The agent exhibited high variance, with fitness scores fluctuating significantly as the population explored the parameter space.
- **Convergence (Gen 60-70):** The system achieved its global maximum performance at **Generation 64** with a fitness of 16,486. This corresponds to the model successfully navigating complex route segments without collisions.
- **Stability:** Post-convergence, the sigma decay mechanism reduced the exploration noise, allowing the model to fine-tune the driving behavior, though no higher peak was found in the subsequent 40 generations logged.

The completion rate peaked at 24%, indicating that while the best individuals could complete the route, a significant portion of the mutated population (the offspring) still engaged in exploration that led to failure, which is characteristic of the NES search strategy.

6 Conclusions

We have presented an automated verification approach for intelligent systems using evolutionary algorithms. By employing a batch training strategy with 4 concurrent vehicles, we demonstrated that it is possible to train a temporal-aware driving policy in CARLA without gradients. The results show that a simple linear model with memory (Simple Temporal Model) is sufficient to solve basic navigation tasks, achieving high fitness scores and route completions. Future work will focus on increasing the batch size further and introducing dynamic traffic scenarios to verify the robustness of the evolved policy.