Title: Training a Reinforcement Learning Model Using Visual Input in CarRacing-v3 Environment

Daniel López
Institut TIC de Barcelona
Thu May 22nd, 2025

Abstract

This paper explores the training of a reinforcement learning model using image input in the CarRacing-v3 environment, employing the PPO algorithm and convolutional neural network (CNN)-based policies.

---

1. Introduction and State of the Art

Reinforcement Learning (RL) has proven effective in solving sequential decision-making problems, where agents learn to act by interacting with their environment. With advances in computational power and algorithm design, RL has achieved significant milestones in fields such as video games, robotics, and autonomous driving. Integrating visual perception into RL tasks poses an additional challenge, as the agent must process high-dimensional pixel input in real time.

Among the various RL algorithms, Proximal Policy Optimization (PPO) has emerged as a robust and scalable method for both continuous and discrete action spaces. PPO balances simplicity and performance, making it suitable for environments with complex dynamics and high-dimensional observations.

In the context of car racing simulations, the CarRacing-v3 environment is a benchmark task involving continuous control, partial observability, and a dynamically generated track. Effectively handling visual input in this environment requires convolutional policies capable of learning spatial hierarchies from visual data.

---

2. Methodology and Development

2.1. Environment and Preprocessing

The CarRacing-v3 environment was chosen for its visual complexity and continuous action space. This environment returns RGB images as observations, requiring the use of CNN-based policies. To improve sample efficiency, we employed a wrapper called `SubprocVecEnv` to parallelize experience collection across four environment instances.

The following preprocessing steps were applied:

- Use of `VecTransposeImage` to rearrange image dimensions to be compatible with PyTorch-based models.

- Monitoring each environment instance using `Monitor` for logging purposes.

2.2. Model Configuration

The core of the training pipeline is a PPO model with a CNN-based policy (`CnnPolicy`). Hyperparameters were carefully selected based on prior benchmarks and empirical tuning:

- **Learning rate:** 2.5e-4
- **Steps per update:** 516
- **Batch size:** 64
- **Epochs per update:** 10
- **Discount factor (gamma):** 0.99
- **GAE lambda:** 0.95
- **Clip range:** 0.2

The PPO model was configured to log training data using TensorBoard for real-time monitoring.

2.3. Evaluation Strategy

To systematically evaluate model performance, an `EvalCallback` was used. This callback periodically evaluates the model using the same environment and saves the best-performing policy in a separate directory. This approach allows for effective tracking of improvements and helps prevent overfitting.

The total training time was set to 1 million timesteps—a "timestep" being one interaction between the agent and the environment (i.e., receiving an observation, taking an action, and obtaining a reward)—which provided a balance between performance and computational cost.

2.4. Training Process

During training, the agent receives image-based observations and learns a control policy that maximizes cumulative reward. The PPO algorithm updates policy parameters using minibatches of data collected from all parallel environments.

At the end of the training, the model was saved locally for further testing or deployment. Evaluation logs and saved models allow for thorough post-training analysis.

---

3. Results and Conclusions

After 1 million timesteps of training, the model showed significant improvement in navigating the tracks. The use of PPO with CNNs enabled the agent to learn effective driving strategies from visual input. Average rewards increased steadily over time.

Key observations include:

- PPO with `CnnPolicy` is effective for visual-based RL tasks.
- Parallel environment execution substantially reduces training time.
- Evaluation callbacks provide a reliable mechanism for performance tracking.

While the model performed well, some limitations remain:

- Generalization to unseen track configurations was limited.
- The policy occasionally failed under sharp curves or track anomalies.

Future work could involve augmenting the training data, incorporating domain randomization, or using attention mechanisms to improve perception.

References

1. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. *arXiv preprint arXiv:1707.06347*.
2. Brockman, G., et al. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.
3. Raffin, A., et al. (2021). Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*.
4. Gymnasium Documentation. Available at: https://gymnasium.farama.org/
5. OpenAI ChatGPT. Available at: https://chat.openai.com/