

**Machine Teaching:  
AI Agent for Car Racing Game**

Version: 1.0  
Author: José Henrique Nóbrega Pereira  
Fontys ICT  
Date: June 9, 2025

|  |
| --- |
| A black background with purple text  AI-generated content may be incorrect. |

**Introduction**

This document will go through some of the code and functionality of the same, explaining why certain things were added and their purpose in the final code. This includes functions like convergence checking, final evaluation and monitoring. These serve to ensure proper model training and so that the performance can be objectively checked after training.

**Image Processing and Stacking**

The original screenshot of the game has a size of 96x96 pixels. Before inputting this into the AI agent, some processing is done. First the image is resized to 84x84 pixels and then a grayscale is added.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 1. Code snippet of the resizing and grayscale function

Afterwards in another function some frames are skipped, since not all frames need to be inputted into the AI agent, and the frames are also stacked, meaning the AI receives multiple game frames at the same time in order, this helps give some temporal context.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 2. Skipping and Stacking frames code snippet.

This image processing is important to be done beforehand because it reduces the computational resources needed for the training, leading to a faster and more energy efficient training process.

**Separated Actor and Critic Networks**

Since the reinforcement learning method used is PPO, this means that there are two distinct neural networks working together. These are the Actor and the Critic. This allows for independent optimization of the policy and value estimation.

**A screen shot of a computer program

AI-generated content may be incorrect.**

Figure 3. Code snippet of the Actor Neural Network.  
A similar code is used for the Critic Neural Network

**Convergence Criteria**

To avoid having to set a certain number of episodes that the AI is going to train before the training even starts, a convergence criteria is added. This allows the training to stop only when convergence is hit, meaning that the AI is not showing significant performance improvement anymore. This avoids the training to stop before the AI reaches maximum performance and the training needing to be restarted.

Each 50 episodes the moving average is calculated and compared to the last 50-episode moving average. If there is a change of less than 1% then a counter is incremented. If the counter reaches 10 then the training stops, and convergence has been hit. If there is a change bigger than 1%, then improvements are still being made, and the counter is reset back to 0.

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 4. Code snippet of the Convergence Checking logic

**Model Saving and Statistic Logging**

During training the model is saved periodically, this ensures that if the hardware runs out of battery or some similar problem and the training is stopped early, there is some models saved midway. In this case each 500 episodes the critic and actor models are saved.

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure 5. Code snippet of the periodic saving of the AI model

The best model is also saved every time that the moving average hits a new all time high value. This ensure that the best version of the AI is always saved. Even if the performance starts dropping midway during training, there is always the best model saved.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 6. Code snippet of the Best Model Saving

The model is also saved at the end of the training when convergence is hit.

A computer code with white text

AI-generated content may be incorrect.

Figure 7. Code Snippet of the Final Model Saving

When training is done and we have the model saved, the history of reward values is store in a pickle file. This is important information about the training process that can be used to plot some graphs to better understand how the AI agent improved overtime and draw conclusions for future trainings.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 8. Code Snippet of the Training Statistics Saving in a Pickle File

**Evaluation**

To evaluate the AI performance and see it play the game, a separate python script is created (code in appendix). This creates a Car Racing environment and loads the Critic and Actor models into the environment. This allows humans to visualize the results of the training with gameplay instead of just looking at the reward values it got during training.

**Conclusion**

Overall, the training of the AI ran smoothly once all the features in this document were implemented. The use of a convergence check made so that the training could run without being supervised and the image processing made it accessible to less powerful hardware. The goal was reached with success since an AI agent was trained and could play better than a human player.

On a personal level this project helped me learn how to train an AI using PPO and the use or neural networks. My knowledge about reinforcement learning is deeper than before and it will certainly be used in future personal projects and I will be looking to implement it into my future professional career as well.

**Appendix**

A screen shot of a computer program

AI-generated content may be incorrect.

Figure 10. Python Script to Visualize the Best Model Playing (part 1 out of 2)

A screen shot of a computer screen

AI-generated content may be incorrect.

Figure 11. Python Script to Visualize the Best Model Playing (part 2 out of 2)