Ashwin Ramaswamy
Last Updated: April 2023

Autonomous Car FoodFinder

1. Introduction and Description

This document is a comprehensive guide to the FoodFinder Autonomous Car project. It is a simple reinforcement learning project entirely written in Python and its associated frameworks and libraries. Within 30-45 minutes, a user will be able to train a car to drive to every fool pellet while staying within the road infrastructure.

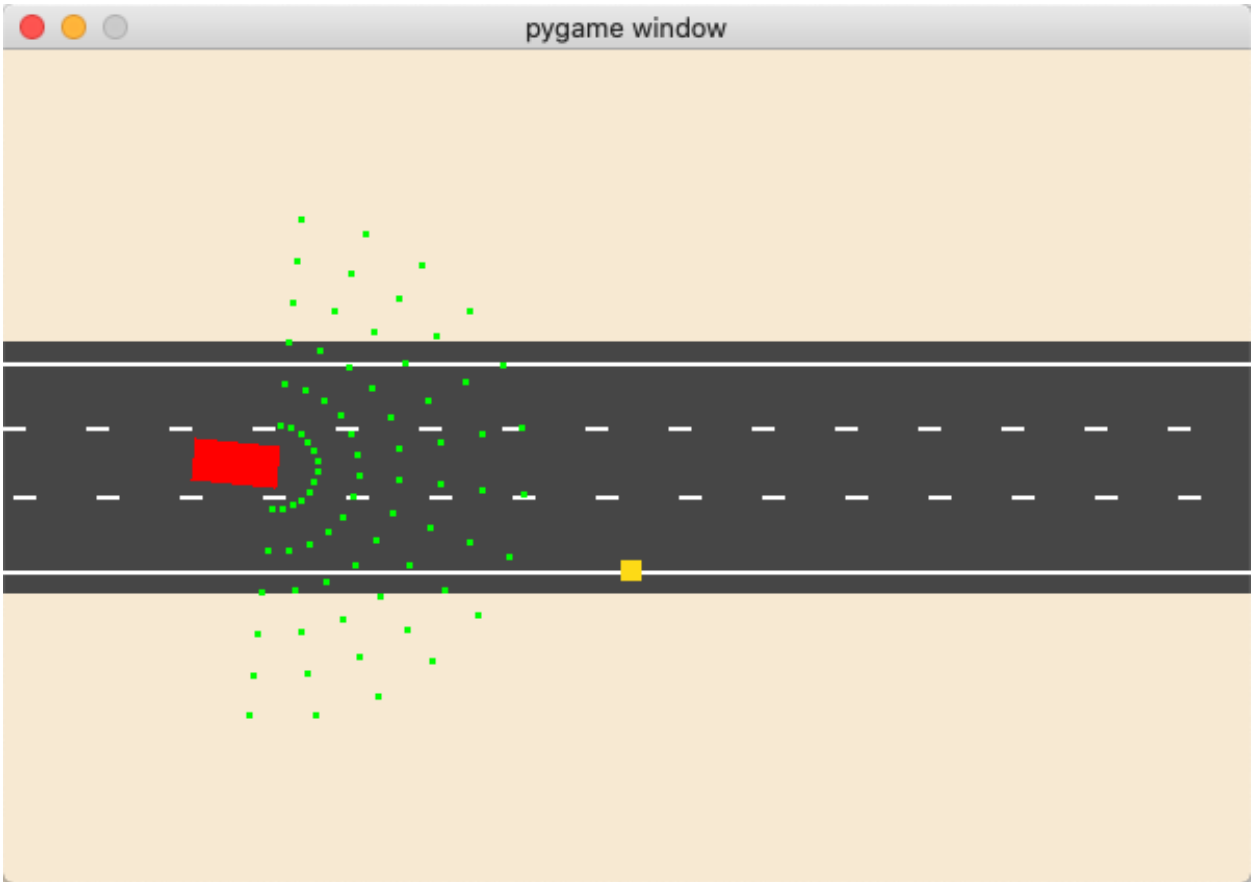

Figure 1. The Pygame GUI of the car object, its perception sensor, the road, and the food pellet.

2. Required Packages and Software

This project should run on any platform that is able to support the following packages:

- Python version >= 3.6
- NumPy version >= 1.19.5
- PyTorch version >= 1.12.1
- Matplotlib version >= 3.3.6

A display must be available if you want to enable the GUI as shown in Figure 1. The GUI can be disabled by commenting out the following line in the `play_step()` function of game.py:

```
self.update_ui()
```

3. Running the Code

After all the necessary packages have been installed, the Zip file containing the code can be extracted. This should contain 6 Python files: agent.py, calc.py, car.py, game.py, model.py, and train.py, related as such:

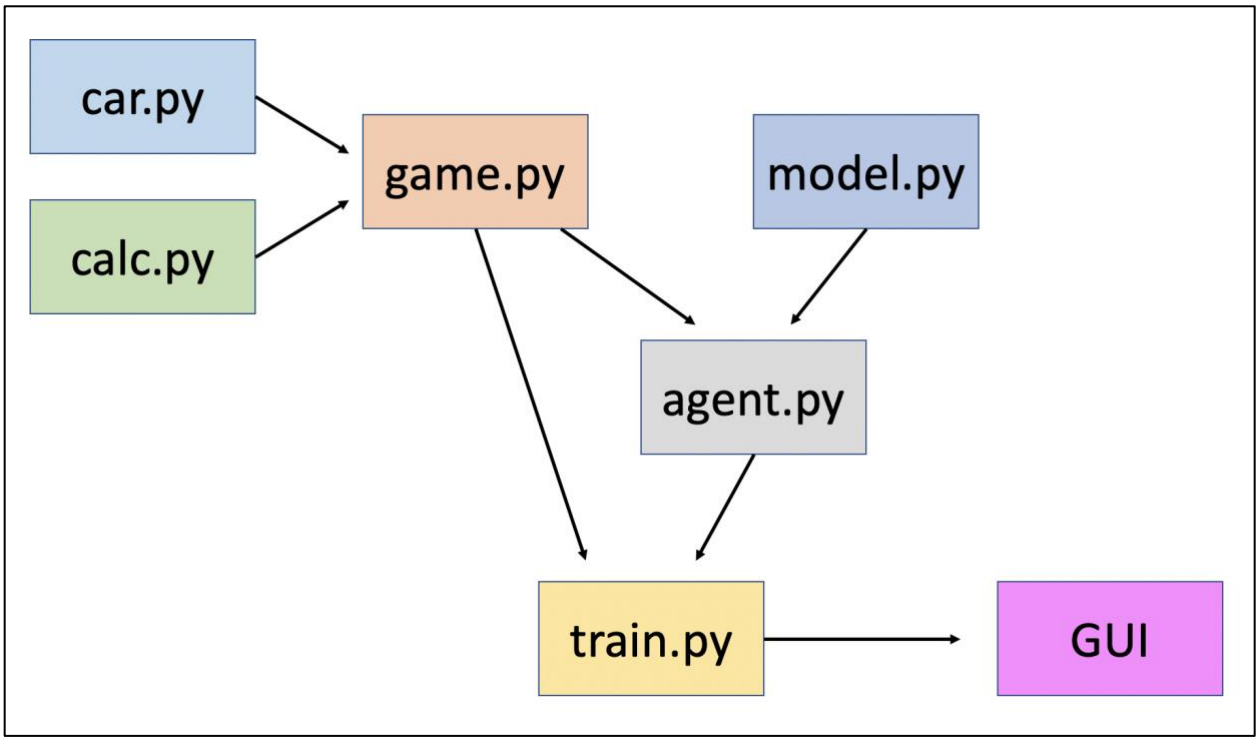

Figure 2. A dependency relationship structure of each code file.

Run the following command in a Linux (or Mac) terminal to start training the car:

```
python3 train.py
```

Running the above command will produce a GUI as shown in Figure 1 accompanied with a text output in the terminal that contains all of the statistics for each training iteration:

```
Iter: 22        Reward -771     Max_Seen: -194          Last25Avg: -1590.60     Last25Max: -169.76
Iter: 23        Reward -1026    Max_Seen: -194          Last25Avg: -1631.64     Last25Max: -169.76
Iter: 24        Reward -526     Max_Seen: -194          Last25Avg: -1652.68     Last25Max: -169.76
Iter: 25        Reward -1383    Max_Seen: -194          Last25Avg: -1708.00     Last25Max: -169.76
Iter: 26        Reward -657     Max_Seen: -194          Last25Avg: -1564.52     Last25Max: -169.76
Iter: 27        Reward -1744    Max_Seen: -194          Last25Avg: -1599.88     Last25Max: -169.76
```

Figure 3. The text output of the training statistics.

This contains information on the training iteration, reward accrued after each episode, the highest reward seen across all training, the average of the last 25 rewards, and the highest average reward seen across all sliding windows of 25 consecutive episodes.

Usually, after 30 – 45 minutes of training, the car is able to consistently maneuver around and keep collecting each food pellet without colliding with the frame or going off road.

4. Adjustable Code

You can adjust the hyperparameters of the model in agent.py, such as the **learning rate**, **discount rate**, **memory size**, and **number of neurons per layer**:

```
MAX_MEMORY = 100_000
BATCH_SIZE = 1000
LR = 0.0001

class Agent:
    def __init__(self):
        self.n_games = 0
        self.gamma = 0.85 # discount rate
        self.memory = deque(maxlen=MAX_MEMORY)
        self.outputs = 6
        self.model = Linear_QNet(85, 256, 256, self.outputs)
        self.trainer = QTrainer(self.model, LR, self.gamma)
```

You can also adjust the reward heuristics in game.py, for information such as **collisions**, whether the **car ate the food pellet**, and whether the **car goes towards or away from the food pellet**.

```
    if self.is_collision() or self.frame_iteration > 1500:
            game_over = True
            reward = -15
            return reward, game_over

    if self.ate_food():
            reward = 15
            self.place_food()
    else:
            dst = distance(self.car.front, self.food)
            if dst < self.min_dist:
                self.min_dist = dst
                reward = 2
            else:
                reward = -5
```

Additionally, the architecture of the neural network can be adjusted in the model.py file. It is made using PyTorch and by default is set to 2 fully-connected layers with a ReLU non-linearity modifying each layer.

5. Credits and Acknowledgements

First and foremost, this work would not have been possible without the decades of AI research and open-source projects that are available for students to learn.

[1] This project was inspired by the Snake Game Reinforcement Learning tutorial produced by Patrick Loeber:
https://www.youtube.com/playlist?list=PLqnslRFeH2UrDh7vUmJ60YrmWd64mTTKV
https://github.com/patrickloeber/snake-ai-pytorch

[2] Additionally, the Q-learning video form the YouTube channel “The Computer Scientist” was extremely formative in my understanding of Q-learning and Reinforcement Learning frameworks such as OpenAI gym:
https://www.youtube.com/watch?v=wN3rxIKmMgE

[3] Finally, a general overview of Markov decision processes and search heuristics was derived from the Fall 2018 iteration of CS 188 at UC Berkeley lecture videos and projects:
https://inst.eecs.berkeley.edu/~cs188/fa18/

This project is also open-sourced for education purposes.