# How to optimize fuel consumption by using RL?

Hugo Jiménez

Leibniz Universität Hannover

# Overview

- I want an RL agent that learns to control a car (acceleration, deceleration, and gears) and optimize de fuel consumption.
- We have to see the behavior of a car in the real world and try to model it into an RL environment

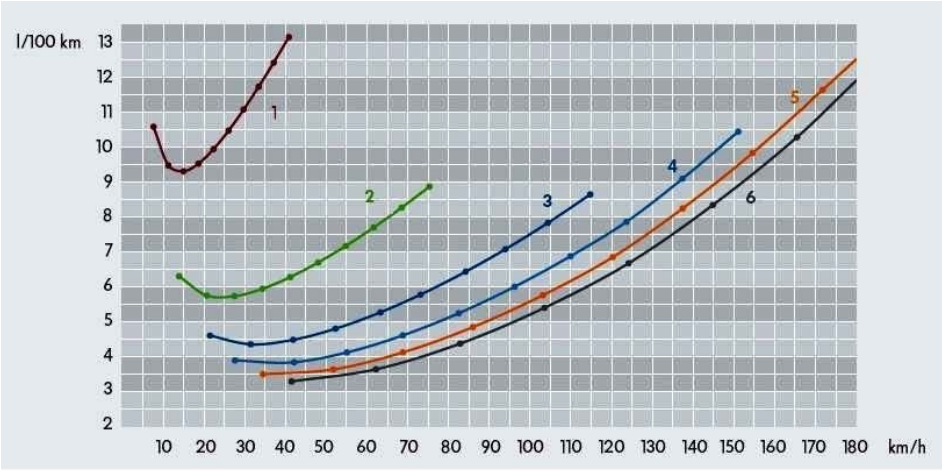




Leibniz Universität Hannover

# Modelling

- The agent have to choose between accelerate, don’t accelerate, decelerate, up a gear, or down gear.
- Constant acceleration and deceleration
- Real functions to approximate the behavior of a car as much as possible




VW Golf TSI 90 kW Température extérieure 20°C

Revolutions / Gear

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1000 | 9,5 | 16,9 | 26,3 | 33,8 | 43,0 | 52,6 |
| 2000 | 18,9 | 33,8 | 52,6 | 67,6 | 86,1 | 105,2 |
| 3000 | 28,4 | 50,7 | 78,9 | 101,4 | 129,1 | 157,8 |
| 4000 | 37,9 | 67,6 | 105,2 | 135,2 | 172,1 | 210,3 |
| 5000 | 47,3 | 84,5 | 131,5 | 169,0 | 215,1 | 262,9 |
| 6000 | 56,8 | 101,4 | 157,8 | 202,8 | 258,2 | 315,5 |

$$Speed = \frac{C_r \cdot 60 \cdot RPM/1000}{R_m \cdot RDF}$$

Leibniz Universität Hannover

# Implementation

- States: Position, speed, gear, revolutions and speed limit
- Actions: Accelerate, don't accelerate, decelerate, gear up and gear down.
- Reward: $$\begin{cases} -speed - limit & if \ \ speed > limit \\ -\frac{revolutions}{\beta_1} - speed \cdot \beta_2 & if \ revolutions < 2000 \ or \ \ revolutions > 3000 \\ \alpha \cdot speed - consume & otherwise \end{cases}$$
- Update function: **Position:** position += speed*0.2778

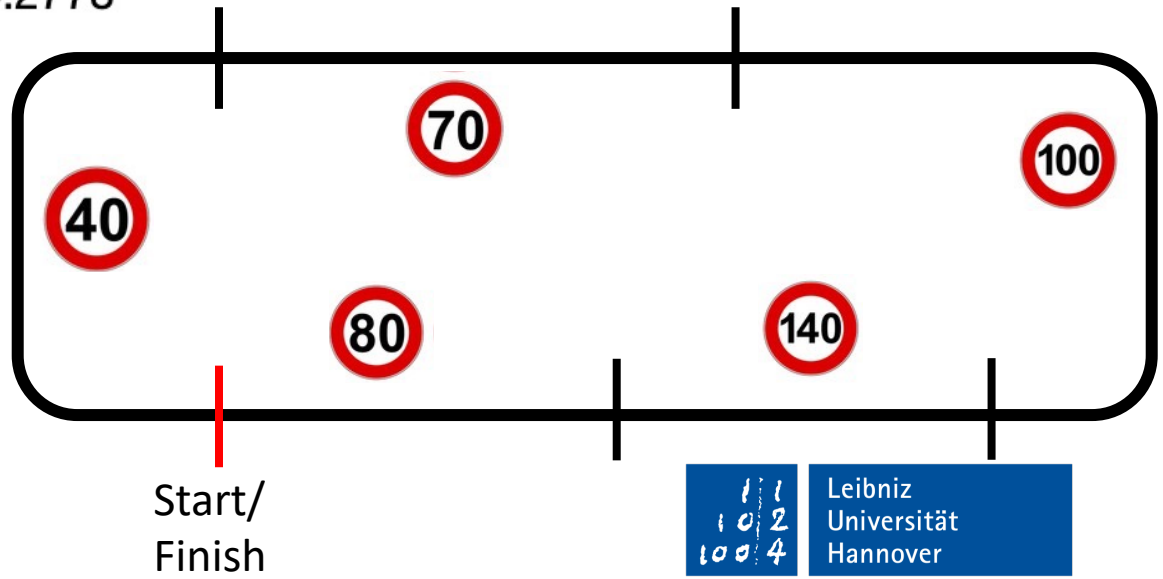
**Speed:** speed += (action - 1) * (6)

**Revolutions:** revolutions = (self.gear_r[int(gear-1)] * 4.64 * speed * 1000)/(1.83*60)

Obtained from the formula: $Speed = \frac{C_r \cdot 60 \cdot RPM/1000}{R_m \cdot RDF}$

**Consume:**

```
if gear == 1:
    consume = 0.0059*speed**2-0.1857*speed+11.0892
elif gear == 2:
    consume = 0.0012*speed**2-0.0574*speed+6.6088
elif gear == 3:
    consume = 0.0005*speed**2-0.0207*speed+4.5582
elif gear == 4:
    consume = 0.0004*speed**2-0.0216*speed+4.1160
elif gear == 5:
    consume = 0.0004*speed**2-0.0183*speed+3.6267
elif gear == 6:
    consume = 0.0004*speed**2-0.0198*speed+3.4347
```

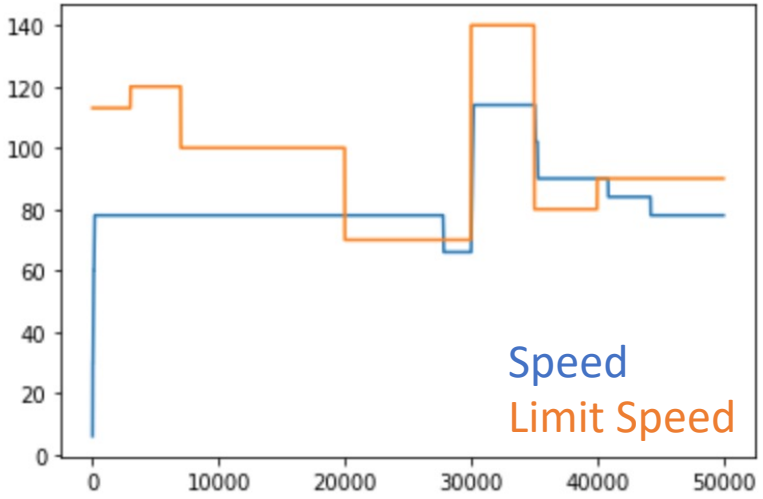


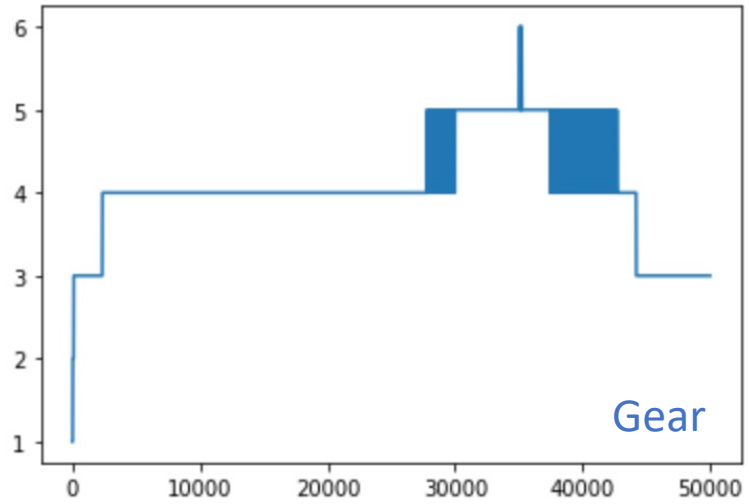

Leibniz Universität Hannover

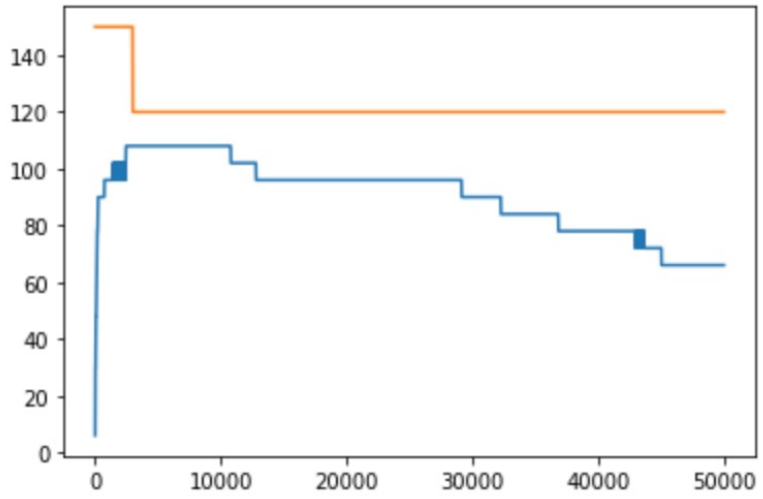
# Problems

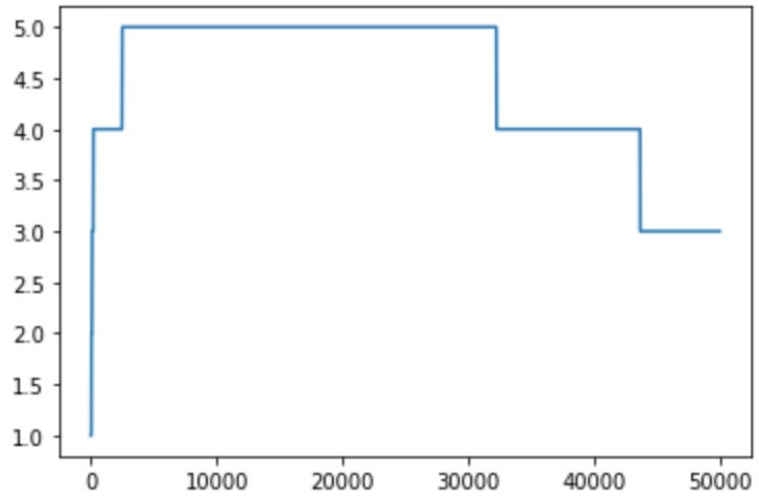
- Tuning reward function to obtain a consistent result
- Choosing an agent
- Lots of problems with the packages’ compatibility
- Very Long execution time

# Results





```
mean_reward_before_train
```

```
{'12345': -44356.800000000454,
 '54321': -44356.800000000454,
 '12345321': -44356.800000000454}
```

```
mean_reward_after_train
```

```
{'12345': 12128.109116994576,
 '54321': -15145.955977070702,
 '12345321': -13743.970899999305}
```

Leibniz Universität Hannover

# Conclusions

- I have managed to create an environment that approximates the behavior of a car in real life
- The results have great variability, hence the importance of reproducibility.
- Next steps: make an environment where you can decide the acceleration that is provided and that is not fixed