TErSLA: The Evolutionary and Reinforcement based Self-Learning vehicle Approach

Course: Introduction to Intelligent Vehicles

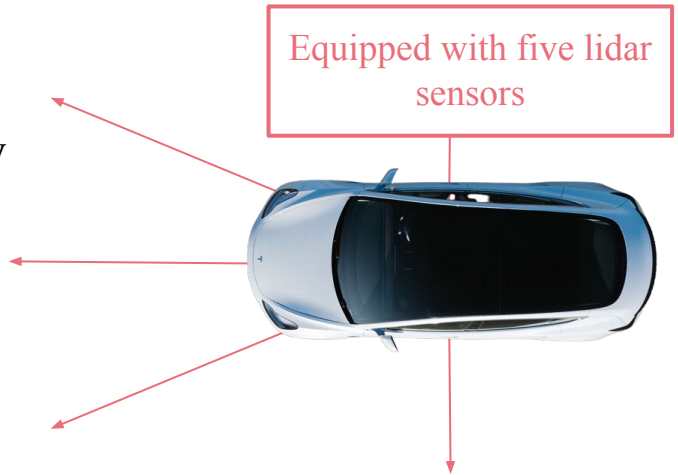
楊敦捷 (R10922147)

李勝維 (R11944004)

廖金億 (R11944021)

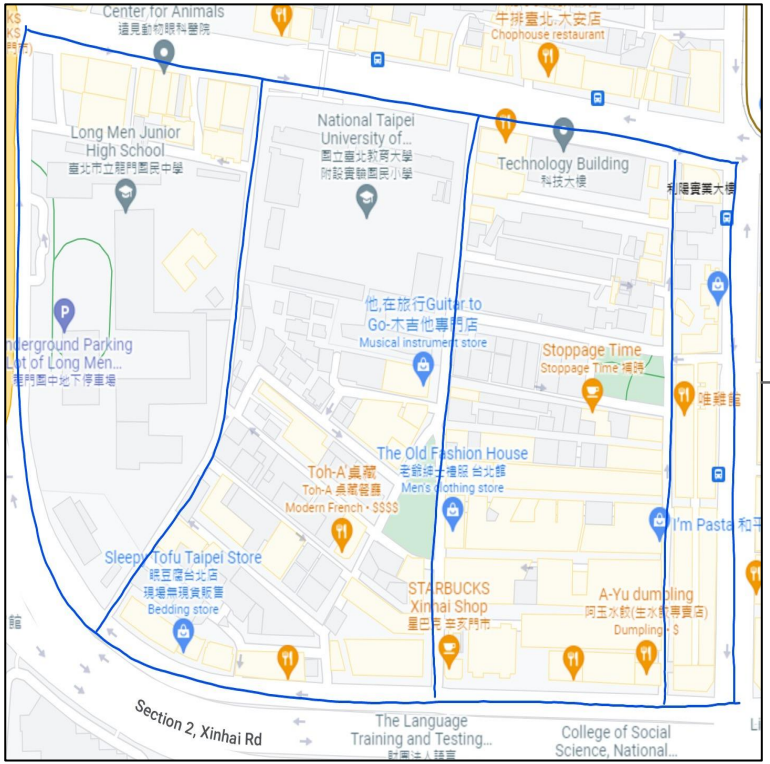
Abstracts

- Motivation:
  - 台大校園罕見重大車禍 「限速20公里」博士生被撞進加護病房 | 社會新聞
- Goals:
  - Employs a deep neural-network-based autonomous car with a focus on obstacle and pedestrian avoidance.
- Approach:
  - Deploys a self-learning framework empowered by
    - Reinforcement Learning(DQN)
    - Genetic Algorithm

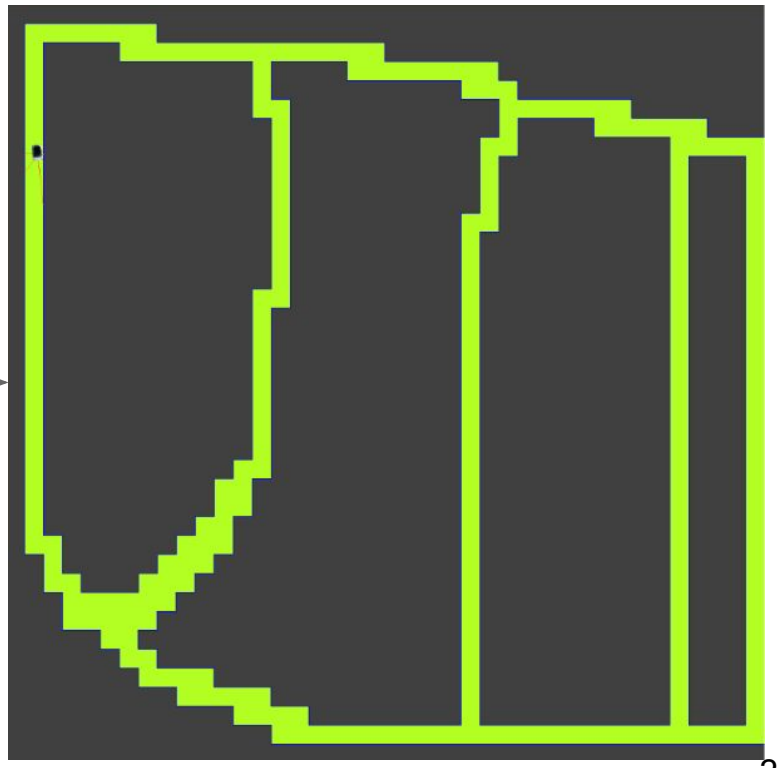


Scenario

- Playground: the beloved lane 118(118 巷)



Mapping Function



Reinforcement Learning: Deep Q Network(DQN)

- **A deep learning model learns to control policies directly from sensory input using reinforcement learning.** [Playing Atari with Deep Reinforcement Learning]
  - Similar to Q-learning[Christopher JCH Watkins and Peter Dayan. Q-learning. 1992]
    - *State*: A agent will observes its current state S_n from the environment
    - *Action*: Selects and performs an action A_n
    - *Reward*: Receive an rewards R depends on the sequent state S_{n+1} after action A_n
- **Key components of DQN**
  - How to compute the *Reward* ?
    - With a discount parameter γ that ensures the reward sum converges, we can compute a discounted, cumulative reward which is less important from the uncertain far future but having more impact in the near future.

$$R_{t0} = sum(\gamma^t . R_t) \text{ for t from } t_0 \text{ to } \infty$$

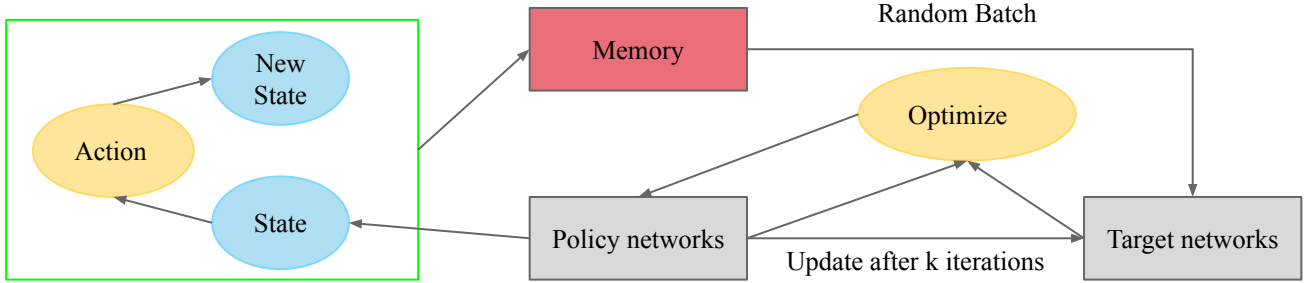
  - How to select an *Action*?
    - Training a function Q that could tell us what our return would be, if we were to take an action in a given state, then we could construct a policy that maximizes our rewards.

$$Q: State \times Action \rightarrow Reward$$

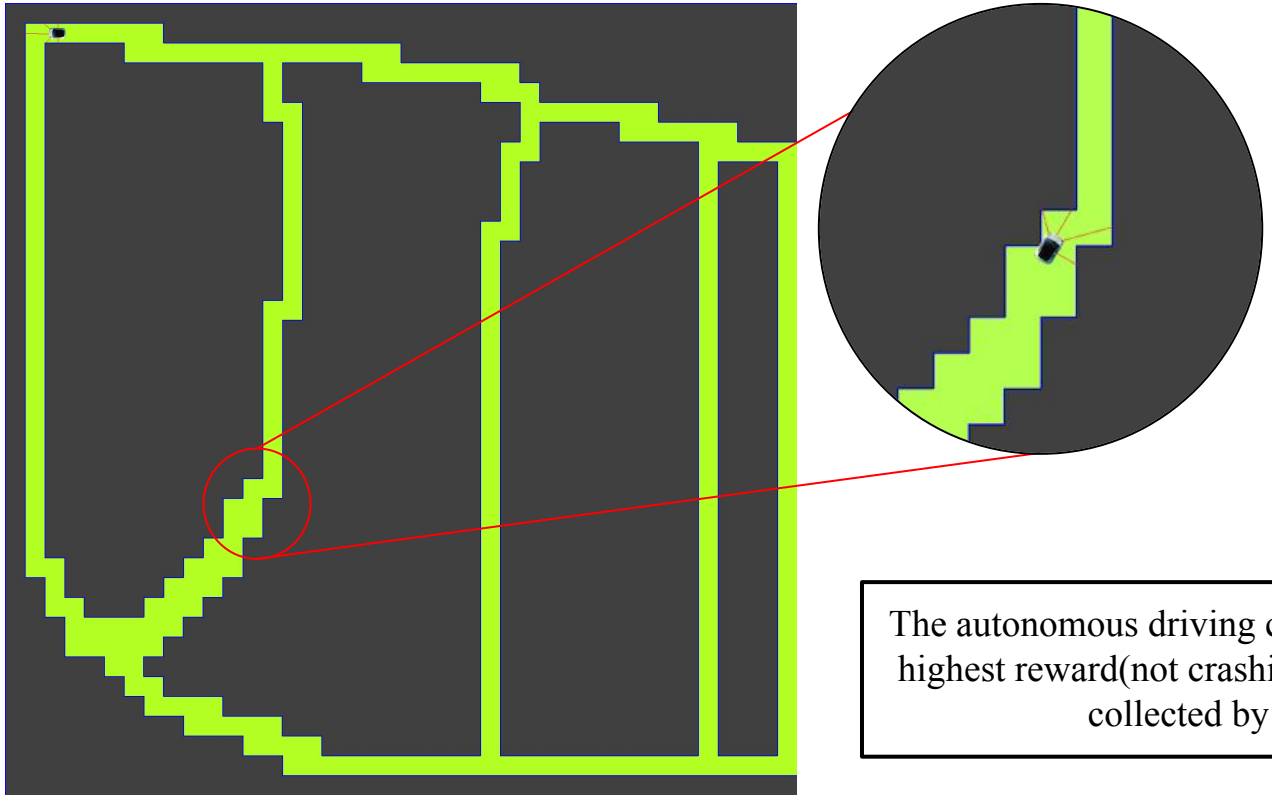
$$Action = \pi^*(state) = argmax_{action} Q(state, action)$$

Training a Deep Q Network(DQN)

- **Q-network**
  - Q function obeys the Bellman equation[Bellman & Dreyfus, 1962; Ross, 1983)]
    - *Bellman equation*
      - It assures us that there is at least one optimal stationary policy π^* which is such that:
        $Q(s_n, a) = reward_n + \gamma . Q(s_{n+1}, \pi^*(s_{n+1}))$
    - *Difference error*
      - $\delta = Q(s_n, a) - reward_n + \gamma . Q(s_{n+1}, \pi^*(s_{n+1}))$
    - *Model*:
      - We will construct a neural network(Q network) as ***Q*** by minimizing the *difference error* δ, which regard as mean square loss(MSE) in our case
- **Training a Q-network**
  - In order to improve stability, we will add an additional *target network* to serve as *Q network* itself and update it after every *k* iterations.



Deep Q Network(DQN) in our case

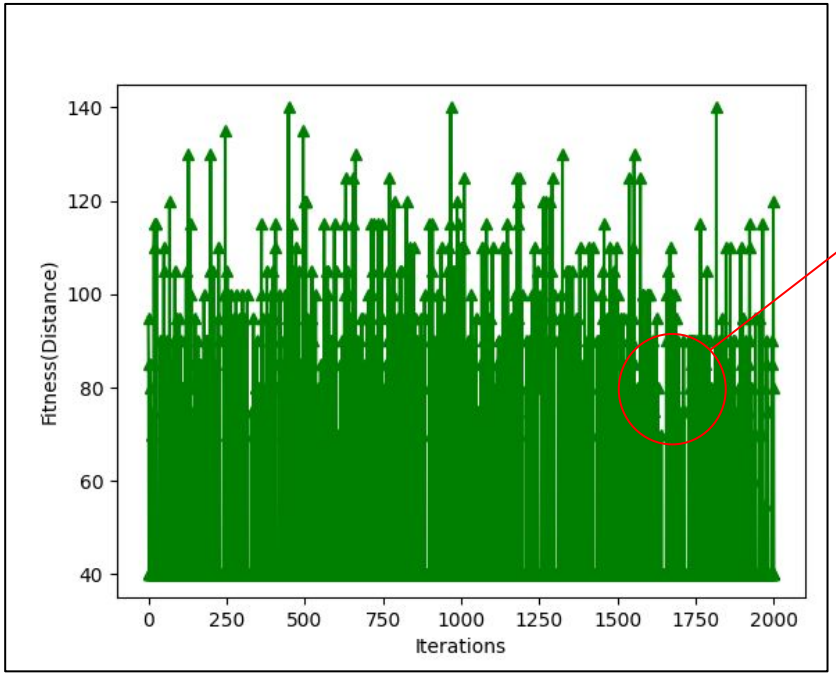


Choosing the direction which does not collide with the obstacles(walls) will have the highest reward so far.

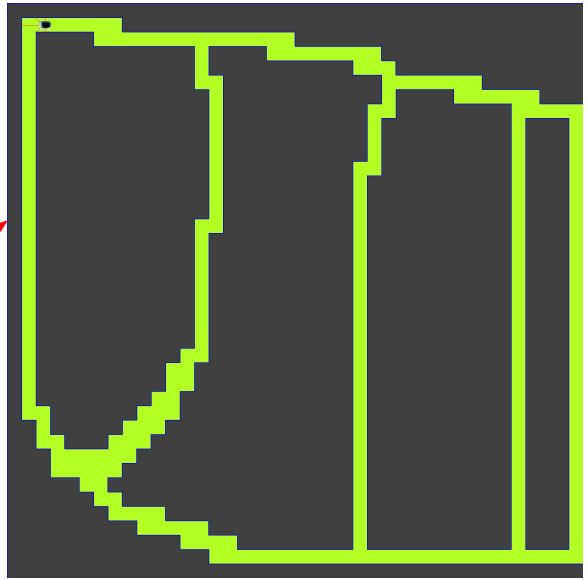
The autonomous driving car will learn how to getting the highest reward(not crashing) from the environment data collected by five lidar sensors.

Problems in our case

The model is hard to converge(time-consuming) while training, the figure shows how bumpy it is.



Died young

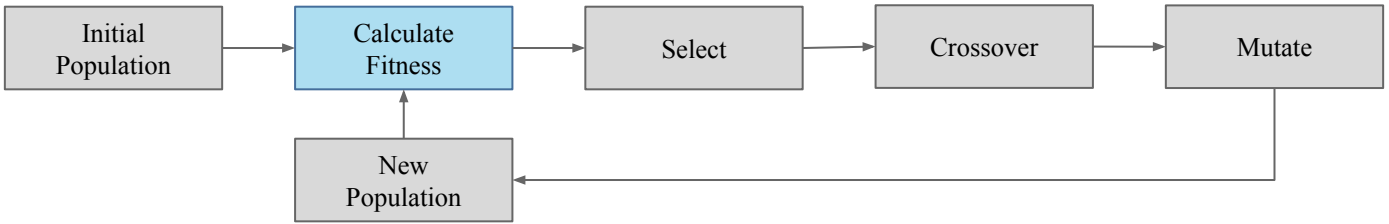


Feasible solutions:

- Loss → Huber Loss (Q-Model)
- Redefine the reward function (distance → living time)

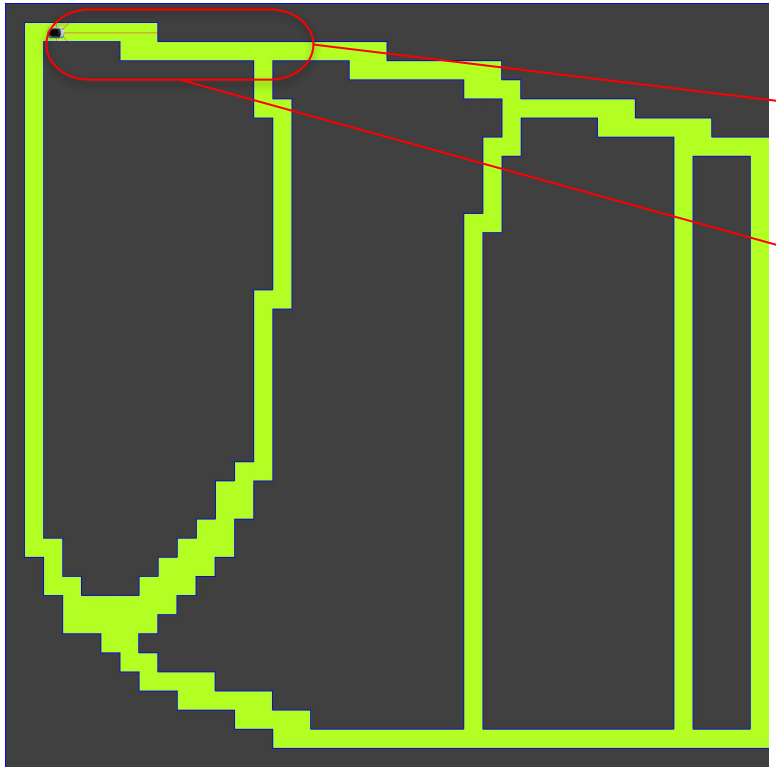
Evolutionary Learning: Genetic Algorithm

- **Genetic algorithm was first proposed by John Holland in 1975.** [Adaptation in Natural and Artificial Systems.]
  - Concept is similar to simulated annealing
- **Genetic algorithm is a search metaheuristic algorithm inspired by theory of natural evolution.**
  - Genetic algorithm
    - *Population*: A collection of solutions to a problem.
    - *Fitness*: Measure of how well a given solution to a problem performs.
- **Key point of Genetic Algorithm**

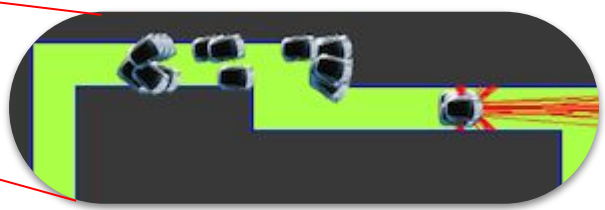


Genetic Algorithm

- **In our case**



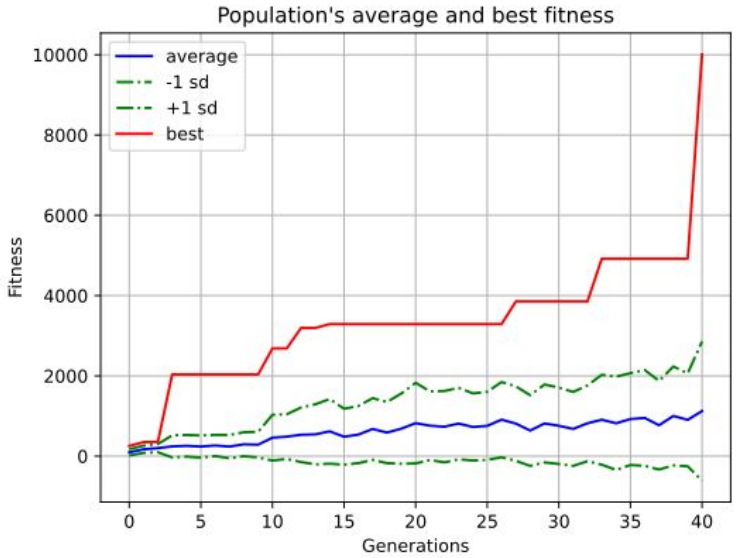
Initial a larger of population size

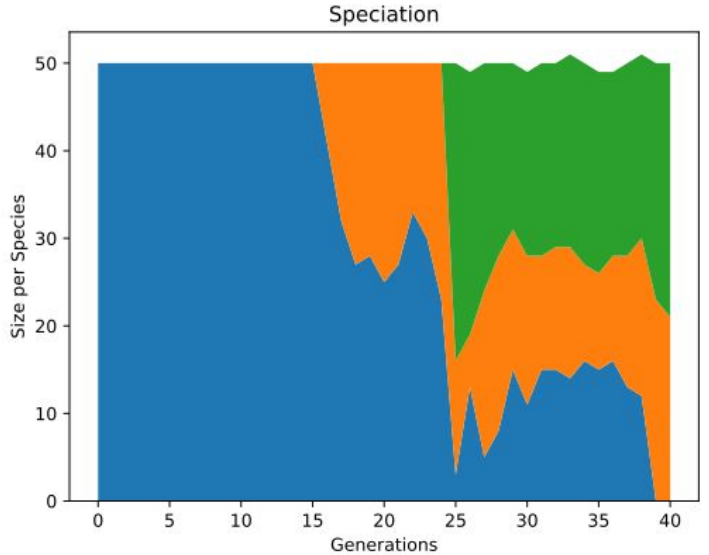


If the car is able to travel a **long distance**, it would have a **high fitness** and be more likely to survive and reproduce.

Genetic Algorithm

- **In our case (performance)**

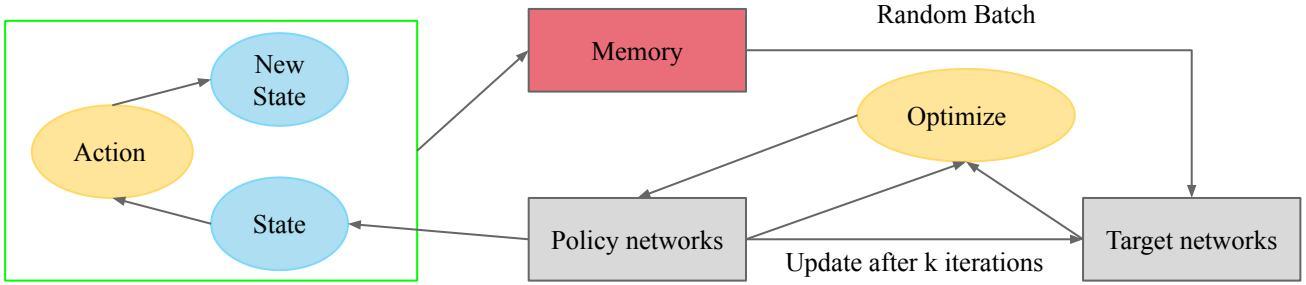




Future Development

- **More hyperparameter tuning / more advanced methods**
  - The RL algorithm suffers from unstable learning curve
  - GA is often trapped at a local minimum

- **Add pedestrians that interact with the environment and our vehicle**
  - Redesigning reward/fitness function
  - Changing the map layout
  - More precise action control


New
State
Action
State
Memory
Random Batch
Optimize
Policy networks
Target networks
Update after k iterations