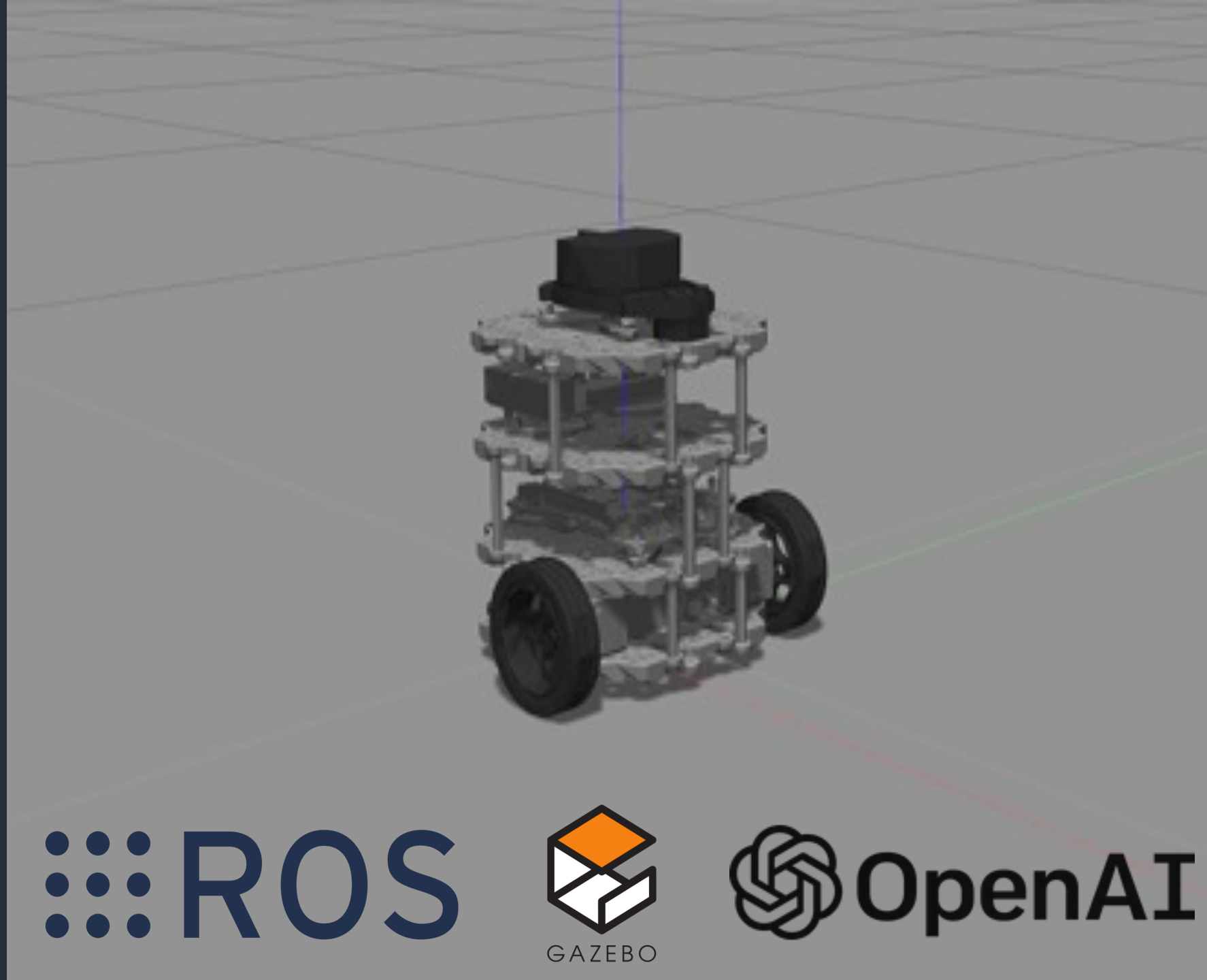
ALPEN-ADRIA UNIVERSITÄT
KLAGENFURT | WIEN GRAZ

Training a DQN for Robot Control

Gabriele Voltan

work done with Gianluca Zavan



ROS

GAZEBO

OpenAI

OUTLINE

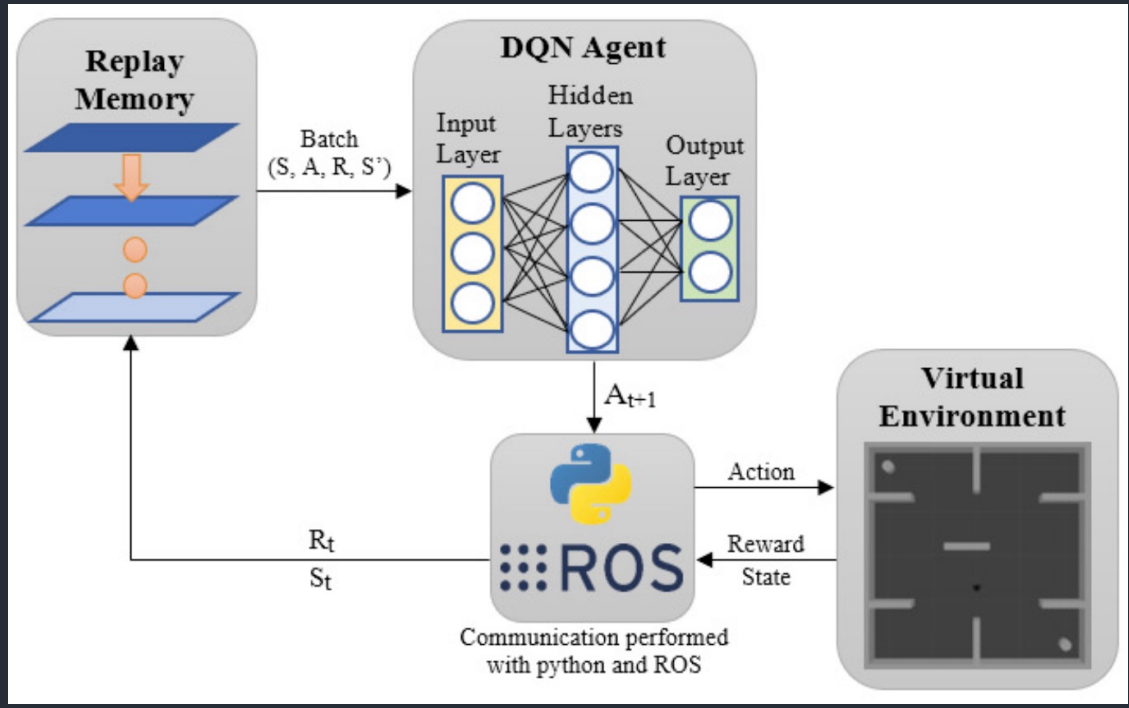
- Goal of the project
- Used approach
- Relevant parameters
- Metrics
- General changes
- Results
- Conclusions

GOAL OF THE PROJECT

- The goal is to train a DQN architecture for controlling a robot avoiding collisions with the surrounding virtual environment

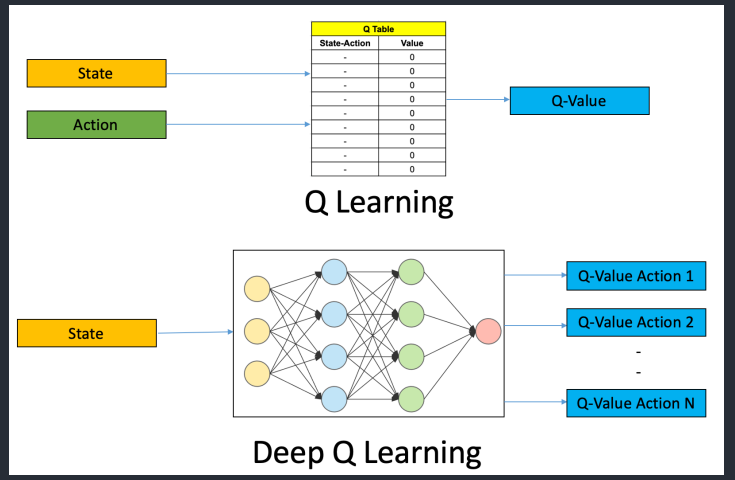
- INPUT: measurements obtained from lasers mounted on board the Turtlebot3 robot

- Use the OpenAI Reinforcement Learning library with ROS and Gazebo



DQN APPROACH

- Use of the "Deep Learning" version of the Q-learning algorithm
- Deep Q-Network (DQN) uses two nets:
  - **Policy Net**: learns the Q-values for each action in a given state
    - Is updated using RMSProp
  - **Target Net**: used to calculated target Q-values during training.
    - To provide greater stability to the training, its weights are updated periodically



RELEVANT HYPER-PARAMETERS

DQN PARAMETERS

- **Gamma γ**: value in [0,1] that determines the importance of future rewards
  - A higher discount factor gives more weight to future rewards, encouraging the agent to consider long-term consequences
- **Epsilon ε**: agent chooses a random action with probability ε and chooses the action with the highest Q-value (exploitation) with probability 1-ε
  - The value of ε typically starts high (encouraging exploration) and decays over time to favor exploitation as training progresses
- **Target update:** how often the target network is updated with the weights of the policy network
- **Learning rate α**: controls the step size of updates to the policy network's weights during training
  - A higher learning rate leads to larger updates, which can cause instability or overshooting of optimal values, while a lower learning rate results in slower but potentially more stable learning
- **Batch size:** number of experiences sampled from the replay buffer to update the policy network in each training iteration
  - Larger batch sizes can provide more stable updates but may require more computation

RELEVANT HYPER-PARAMETERS

TURTLEBOT3 PARAMETERS

- **New ranges**: how many laser readings we jump in each observation reading
  - The bigger the less laser resolution
- **Max Linear Acceleration**: linear acceleration value in which we consider Turtlebot 3 has crashed into something
  - If set to a low value (such as the standard 5.0) even normal accelerations will be considered as collisions
- **Min Range**: minimum meters below which we consider the robot to have crashed
  - If set too high, the system may detect a collision even when one did not occur
- **Action reward**: points awarded for the action performed
  - If unbalanced, some actions may be performed fewer times than others
- **End episode reward**: points given when ending an episode
  - It must be significantly greater than the action rewards

METRICS

- To measure the performance obtained by the agent, we use two metrics:
  - **Time per epoch**: time elapsed before the robot had a collision
  - **Reward per epoch**: reward obtained by the robot for each epoch
- Both of these metrics will be represented with a plot, where on the x-axis there will be the epochs and on the y-axis the time/reward
- To consider the performances as good, both plots must have an average increasing trend

- Note that all the plots are printed after a Standard scaling of the results and the average every 20 episodes is printed in red

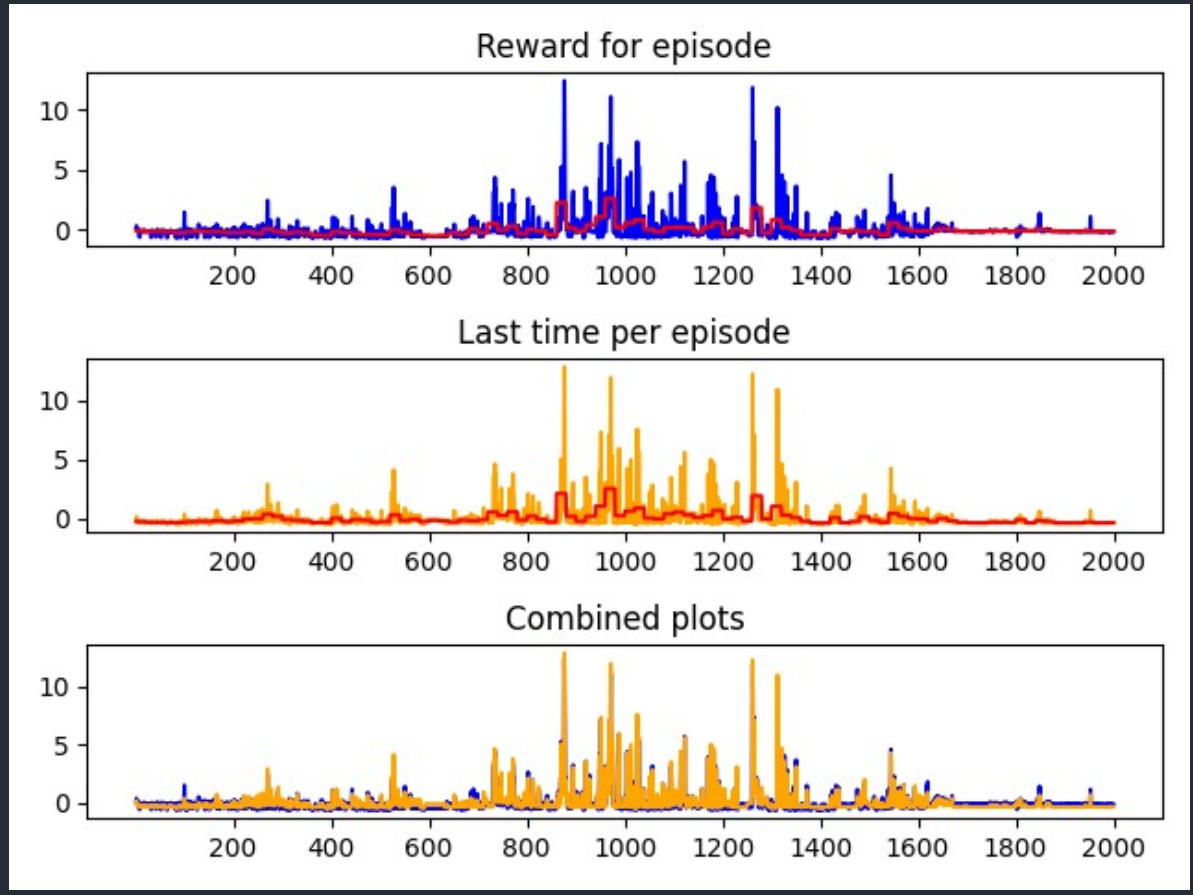
GENERAL CHANGES

- To avoid having cases in which the system detected a collision that did not actually occur, we decided to increase the *max_linear_acceleration* parameter to 50.0
- To have more laser detections, we increased the *new_ranges* parameter to 120
- We changed the robot's spawn point, as in the default one the robot fell on an obstacle and sometimes generated a fake crash
- The rewards of actions, after trying to modify their values and observing their changes, were set to the original values, to reduce the size of the search domain of the parameters

INITIAL RESULTS

- The model provided initially had the following parameters:
  - Architecture: (Input, 64, 128, 64, Output)
  - Optimizer: RMSprop with l.r. = 0.01
  - Epsilod decay = 200
  - Batch size = 128
  - Target update = 10

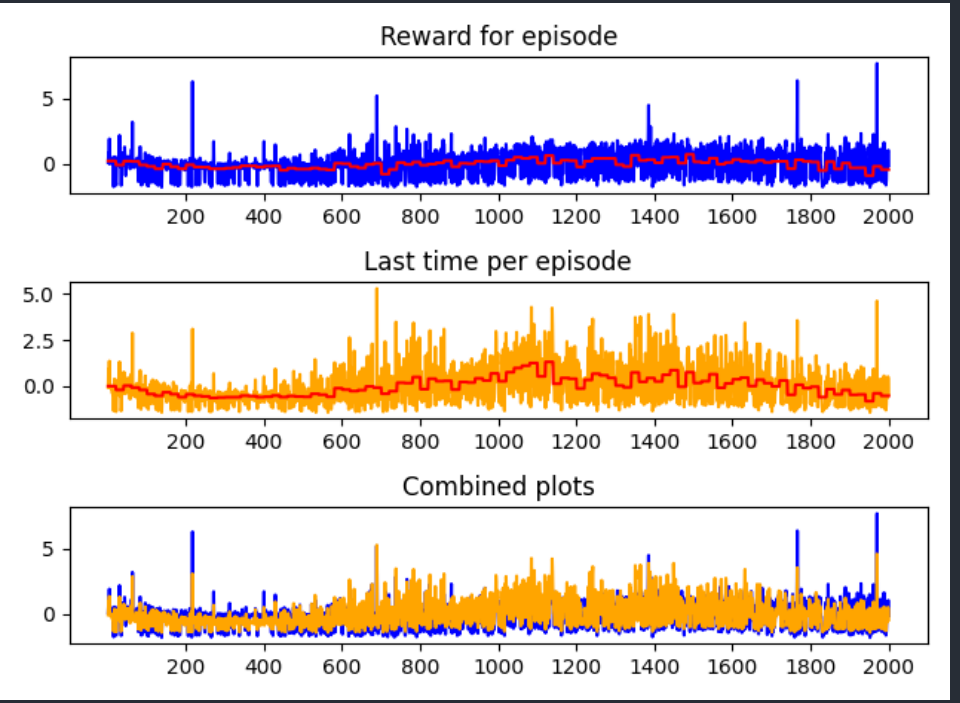
➢ As we can see from the plots, the performance are not good



OTHER RESULTS

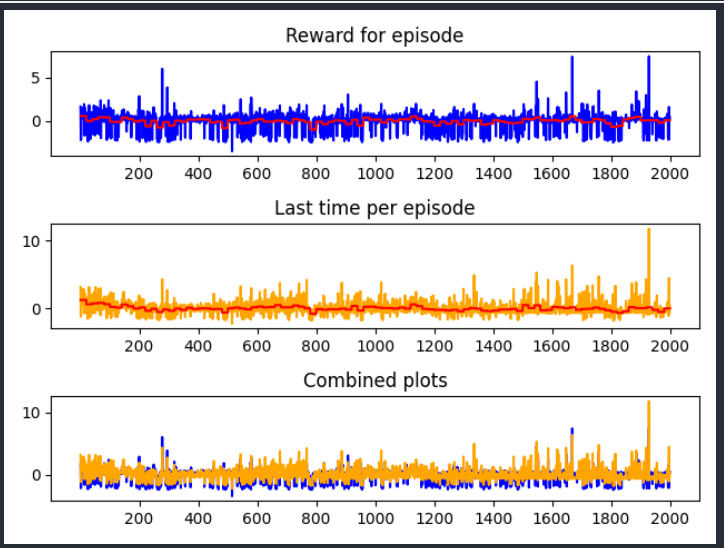
- Starting from the initial results, we tried several experiments done with different architectures and parameters
- Although all the results, together with the code, are on GitHub at https://github.com/GabryV00/DQN_ROS, some with different parameters are reported

  - Architecture: (Input, 64, 128, 256, 128, 64, Output)
  - Optimizer: RMSprop with l.r. = 0.0001
  - Epsilod decay = *25000*
  - Batch size = 32
  - Target update = 50

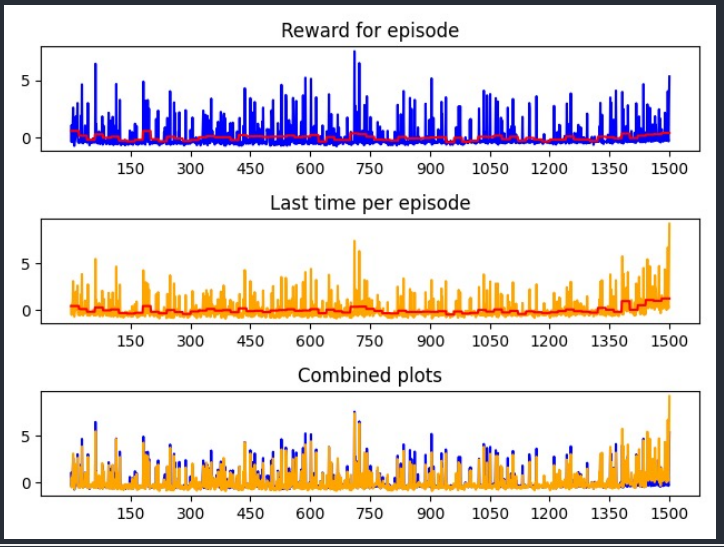


OTHER RESULTS - 2

- Architecture: (Input, 64, 128, 256, 128, 64, Output)
- Optimizer: RMSprop with l.r. = 0.0001
- Epsilod decay = 20000
- Batch size = 32
- Target update = *100*



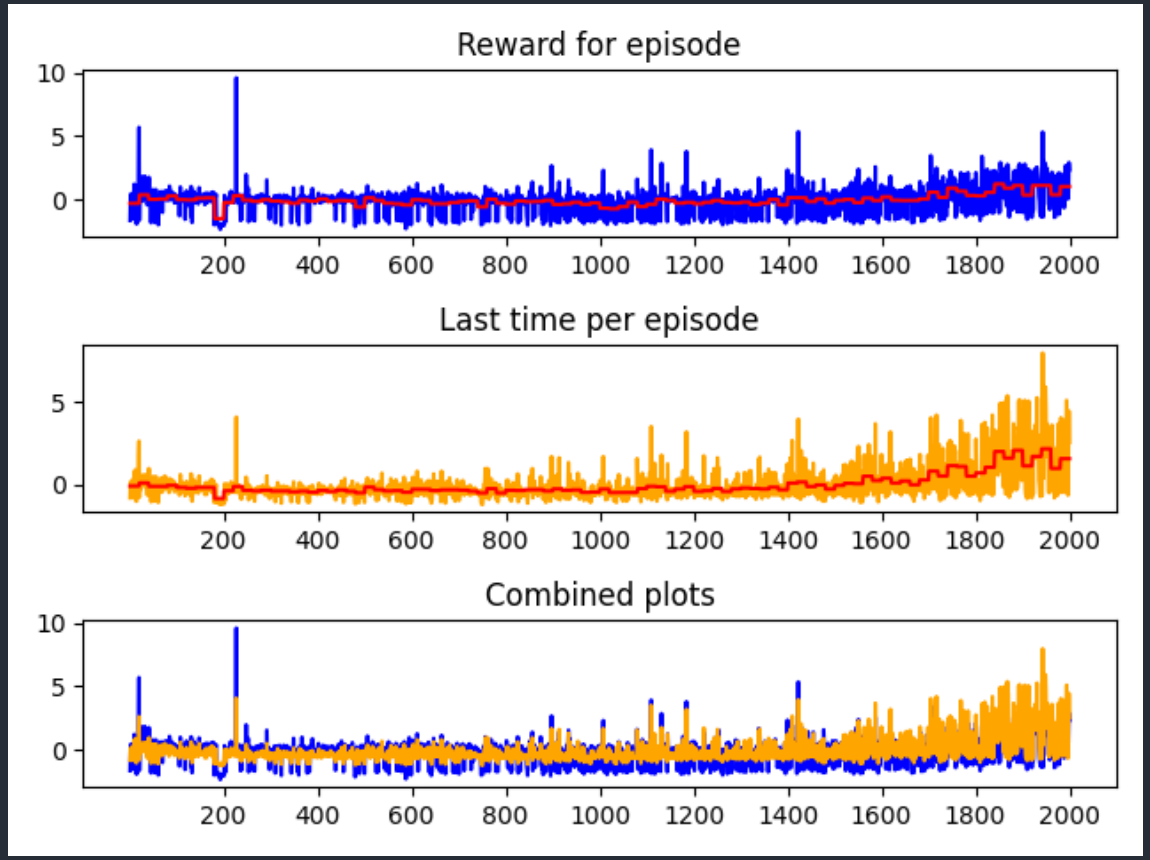
- Architecture: *(Input, 128, 6 x 256, 128, 64, 32, Output)*
- Optimizer: RMSprop with l.r. = *0.00025*
- Epsilod decay = *500*
- Batch size = 32
- Target update = *30*



BEST RESULT

- The best result was achieved with this architecture and parameters
  - Architecture: (Input, 64, 128, 256, 128, 64, Output)
  - Optimizer: RMSprop with l.r. = 0.0001
  - Epsilod decay = 20000
  - Batch size = 32
  - Target update = 50

❖ This graph represents a good result as it is increasing



CONCLUSIONS

- In this work, the DQN approach was used to solve the problem of robot navigation, using laser measurements, in a virtual environment
- All significant parameters, both of the DQN network and of the environment, were analyzed and explained
- Starting from the initial configuration, several changes were made to the architecture and parameters, which, after various experiments, led to acceptable results
- Although the experiments were only conducted for only 2000 episodes, due to the long running time, it is possible to see how there is an increasing trend in the latest reported result
- We are quite certain that, by increasing the number of episodes, with the configuration proposed as the best we would be able to obtain a much more accentuated growing graph

THANKS FOR THE ATTENTION

GABRIELE VOLTAN