# turtlebot3_machine_learning

Rough overview of turtlebot3_dqn → turtlebot3_d3qn

# Understanding the code : commented with link provided

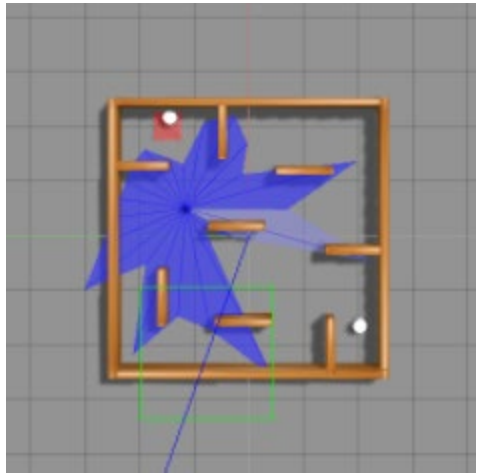
https://github.com/laphisboy/RL_fall/blob/master/fall_week_7/environment_stage_%23.py

https://github.com/laphisboy/RL_fall/blob/master/fall_week_7/turtlebot3_dqn_stage_%23.py

# Basic overview

## Burger and its environment

2. Red box is goal destination



1. Burger with 24 laser scanners to “see”

3. Complicated walls where if the burger crashes than has to start over (including the moving white obstacles)

4. Burger actions: rotate +-1.5rad/s whilst always moving forward with 0.15m/s

| Action | Angular velocity(rad/s) |
|---|---|
| 0 | -1.5 |
| 1 | -0.75 |
| 2 | 0 |
| 3 | 0.75 |
| 4 | 1.5 |

# Basic overview

## Quick peek at burger's environment code

1. Burger crash/goal

```
if min_range > min(scan_range) > 0:
    done = True

# so current distance would mean how much further to go for the GOAL
current_distance = round(math.hypot(self.goal_x - self.position.x, self.goal_y - self.position.y),2)

# if close enough, let it be a GOAL
if current_distance < 0.2:
    self.get_goalbox = True
```

# Basic overview

## Quick peek at burger's environment code

2. Burger reward

```
# is it going in the right direction?
heading = goal_angle - yaw

angle = -pi / 4 + heading + (pi / 8 * i) + pi / 2

tr = 1 - 4 * math.fabs(0.5 - math.modf(0.25 + 0.5 * angle % (2 * math.pi) / math.pi)[0])
yaw_reward.append(tr)
```

# Basic overview

## Quick peek at burger's environment code

2. Burger reward

```
# if too close to obstacle, negative reward
if obstacle_min_range < 0.5:
    ob_reward = -5
else:
    ob_reward = 0
```

# Basic overview

## Quick peek at DQN code

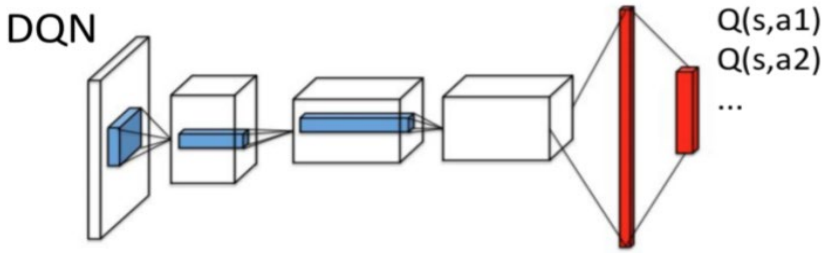
1. DQN

```
model = Sequential()
dropout = 0.2

model.add(Dense(64, input_shape=(self.state_size,), activation='relu', kernel_initializer='lecun_uniform'))

model.add(Dense(64, activation='relu', kernel_initializer='lecun_uniform'))
model.add(Dropout(dropout))

model.add(Dense(self.action_size, kernel_initializer='lecun_uniform'))
model.add(Activation('linear'))
model.compile(loss='mse', optimizer=RMSprop(lr=self.learning_rate, rho=0.9, epsilon=1e-06))
model.summary()
```



# Basic overview

# Quick peek at D3QN code

1. D3QN

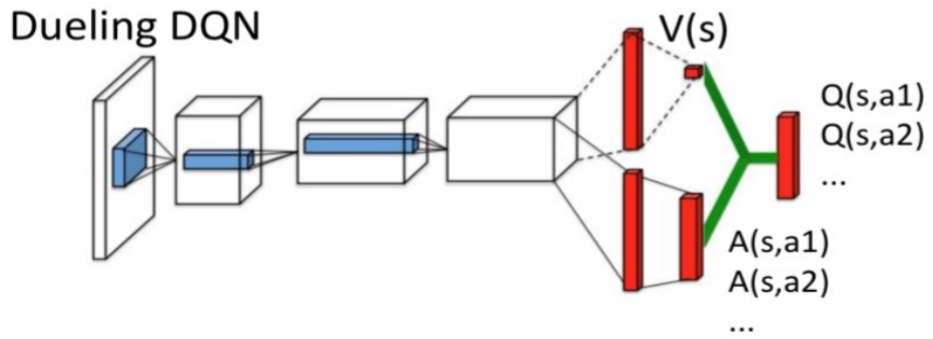
```
model = Sequential()
dropout = 0.2

model.add(Dense(64, input_shape=(self.state_size,), activation='relu', kernel_initializer='he_uniform'))

model.add(Dense(64, activation='relu', kernel_initializer='he_uniform'))
model.add(Dropout(dropout))

# (edit3) self.action_size --> self.action_size + 1 to incorporate V + A
# (edit3) uses mean of advantaged to calculate advantage : do you get it? hehe
model.add(Dense(self.action_size + 1, kernel_initializer='he_uniform'))
model.add(Activation('linear'))
model.add(Lambda(lambda x: K.expand_dims(x[:, 0], -1) + x[:, 1:] - K.mean(x[:, 1:], axis=1, keepdims=True), output_shape=(self.action_size,)))
model.compile(loss='mse', optimizer=RMSprop(lr=self.learning_rate, rho=0.9, epsilon=1e-06))
model.summary()
```

Only had to implement 1 more layer



# Code link

For step by step code edit refer to link ☺

edit1 → edit4

https://github.com/laphisboy/RL_fall/tree/master/fall_week_7