Multi Agent Project

In this project i used the MADDPG algorithm to solve the problem, the code is base in https://github.com/shariqiqbal2810/maddpg-pytorch/ with some modification to various files like the utils functions, the replay_buffer and memory etc.. my first approach was to sample not randomly snapshots of the game if not continuous samples like 2 or more frames for that reason i created two ddpg python classes (ddpg.py and ddpgv2.py) which allows to do that

```
def step(self, agent_id, state, action, reward, next_state, done,t_step):


    self.states.append(state)
    self.actions.append(action)
    self.rewards.append(reward)
    self.next_states.append(next_state)
    self.dones.append(done)

    #self.replay_buffer.add(state, action, reward, next_state, done)
    if t_step % self.num_history == 0:
        # Save experience / reward

        self.replay_buffer.add(self.states, self.actions, self.rewards, self.next_states, self.dones)
        self.states = []
        self.actions = []
        self.rewards = []
        self.next_states = []
        self.dones = []

    # Learn, if enough samples are available in memory
    if len(self.replay_buffer) > self.batch_size:

        obs, acs, rews, next_obs, don = self.replay_buffer.sample()
        self.update(agent_id ,obs,  acs, rews, next_obs, don,t_step)
```

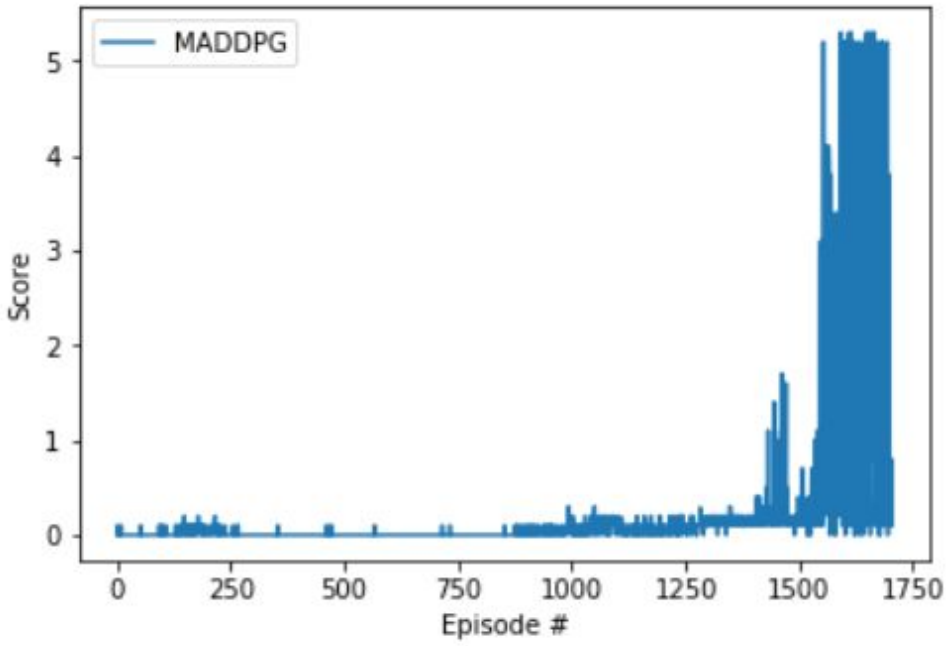
Image 1. the self.num_history controls how much continuous frames to use

but this approach did not work out because the agents got stocked in the same value for thousands of iterations i changed the network configuration (layer, learning rate, tau, noise reduction, no linear activation functions etc..) but the problem continued.

```
Mean Score of both Agents: -0.005
Mean Score of both Agents: 0.006
, Mean Score of both Agents: -0.005
Mean Score of both Agents: 0.013
, Mean Score of both Agents: -0.004
, Mean Score of both Agents: -0.005
, Mean Score of both Agents: -0.004
```

so i decided to eliminated that option as you can see in ddpgv2.py which has the last version with the performance desire but uses the same algorithm (except the continuous frame option) the lines are almost equal.

```
Episode 0       Mean Score of both Agents: 0.0000, replay memory0 15.000, repleay mem1 15.000
Episode 200     Mean Score of both Agents: 0.0330, replay memory0 3604.000, repleay mem1 3604.000
Episode 400     Mean Score of both Agents: 0.0010, replay memory0 6782.000, repleay mem1 6782.000
Episode 600     Mean Score of both Agents: 0.0010, replay memory0 9738.000, repleay mem1 9738.000
Episode 800     Mean Score of both Agents: 0.0020, replay memory0 12633.000, repleay mem1 12633.000
Episode 1000    Mean Score of both Agents: 0.0380, replay memory0 16348.000, repleay mem1 16348.000
Episode 1200    Mean Score of both Agents: 0.0618, replay memory0 21833.000, repleay mem1 21833.000
Episode 1400    Mean Score of both Agents: 0.1279, replay memory0 28502.000, repleay mem1 28502.000
Episode 1600    Mean Score of both Agents: 1.0039, replay memory0 56166.000, repleay mem1 56166.000
Episode 1673    Mean Score of both Agents: 2.1570, replay memory0 89280.000, repleay mem1 89280.000
```



Next Steps

i could use another algorithm or try other configurations like recurrent neural networks in the actor and critic