Deep Q-Learning

Install dependencies for AI gym to run properly (shouldn't take more than a minute). If running on google cloud or running locally, only need to run once. Colab may require installing everytime the vm shuts down.

In [1]:
```
# !pip3 install gym pyvirtualdisplay
# !sudo apt-get install -y xvfb python-opengl ffmpeg
```

In [2]:
```
# !pip3 install --upgrade setuptools --user
# !pip3 install ez_setup
# !pip3 install gym[atari]
```

For this assignment we will implement the Deep Q-Learning algorithm with Experience Replay as described in breakthrough paper **"Playing Atari with Deep Reinforcement Learning"**. We will train an agent to play the famous game of **Breakout**.

In [1]:
```
%matplotlib inline

import sys
import gym
import torch
import pylab
import random
import numpy as np
from collections import deque
from datetime import datetime
from copy import deepcopy
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
from torch.autograd import Variable
from utils import find_max_lives, check_live, get_frame, get_init_state
from model import DQN
from config import *

import matplotlib.pyplot as plt
# %load_ext autoreload
# %autoreload 2
```

In [4]:
```
# !pip3 install -q gym[atari]
# !pip install -q autorom[accept-rom-license]
```

In [5]:
```
env = gym.make('ALE/Breakout-v5')
state = env.reset()
#print(state)
```

```
A.L.E: Arcade Learning Environment (version 0.8.1+53f58b7)
[Powered by Stella]
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/core.py
:317: DeprecationWarning: WARN: Initializing wrapper in old step API which r
eturns one bool instead of two. It is recommended to set `new_step_api=True`
to use new step API. This will be the default behaviour in future.
  deprecation(
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/wrapper
s/step_api_compatibility.py:39: DeprecationWarning: WARN: Initializing envir
onment in old step API which returns one bool instead of two. It is recommen
ded to set `new_step_api=True` to use new step API. This will be the default
behaviour in future.
  deprecation(
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/utils/p
assive_env_checker.py:190: UserWarning: WARN: Future gym versions will requi
re that `Env.reset` can be passed `return_info` to return information from t
he environment resetting.
  logger.warn(
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/utils/p
assive_env_checker.py:137: UserWarning: WARN: The obs returned by the `reset
()` method was expecting a numpy array, actual type: <class 'tuple'>
  logger.warn(
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/spaces/
box.py:226: UserWarning: WARN: Casting input x to numpy array.
  logger.warn("Casting input x to numpy array.")
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/utils/p
assive_env_checker.py:167: UserWarning: WARN: The obs returned by the `reset
()` method is not within the observation space with exception: setting an ar
ray element with a sequence. The requested array has an inhomogeneous shape
after 1 dimensions. The detected shape was (2,) + inhomogeneous part.
  logger.warn(f"{pre} is not within the observation space with exception: {e
}")
```

Understanding the environment

In the following cell, we initialize our game of **Breakout** and you can see how the environment looks like. For further documentation of the of the environment refer to https://gym.openai.com/envs.

In breakout, we will use 3 actions "fire", "left", and "right". "fire" is only used to reset the game when a life is lost, "left" moves the agent left and "right" moves the agent right.

In [6]:
```
# env = gym.make('BreakoutDeterministic-v4')
# state = env.reset()
```

In [7]:
```
number_lives = find_max_lives(env)
state_size = env.observation_space.shape
action_size = 3 #fire, left, and right
```

```
/home/aalkhami/miniconda3/envs/myenv/lib/python3.9/site-packages/gym/utils/p
assive_env_checker.py:241: DeprecationWarning: `np.bool8` is a deprecated al
ias for `np.bool_`.  (Deprecated NumPy 1.24)
  if not isinstance(terminated, (bool, np.bool8)):
```

Creating a DQN Agent

Here we create a DQN Agent. This agent is defined in the **agent.py**. The corresponding neural network is defined in the **model.py**. Once you've created a working DQN agent, use the code in agent.py to create a double DQN agent in **agent_double.py**. Set the flag "double_dqn" to True to train the double DQN agent.

Evaluation Reward : The average reward received in the past 100 episodes/games.

Frame : Number of frames processed in total.

Memory Size : The current size of the replay memory.

In [8]:
```
double_dqn = False # set to True if using double DQN agent

if double_dqn:
    from agent_double import Agent
else:
    from agent import Agent

agent = Agent(action_size)
evaluation_reward = deque(maxlen=evaluation_reward_length)
frame = 0
memory_size = 0
```

Main Training Loop

In this training loop, we do not render the screen because it slows down training signficantly. To watch the agent play the game, run the code in next section "Visualize Agent Performance"

In [9]:
```
rewards, episodes = [], []
best_eval_reward = 0
for e in range(EPISODES):
    done = False
    score = 0
```

```
    history = np.zeros([5, 84, 84], dtype=np.uint8)
    step = 0
    d = False
    state = env.reset()
    next_state = state
    life = number_lives

    get_init_state(history, state)

    while not done:
        step += 1
        frame += 1

        # Perform a fire action if ball is no longer on screen to continue c
        if step > 1 and len(np.unique(next_state[:189] == state[:189])) < 2:
            action = 0
        else:
            action = agent.get_action(np.float32(history[:4, :, :]) / 255.)
        state = next_state
        next_state, reward, done, info = env.step(action + 1)

        frame_next_state = get_frame(next_state)
        history[4, :, :] = frame_next_state
        terminal_state = check_live(life, info['lives'])

        life = info['lives']
        r = np.clip(reward, -1, 1)
        r = reward

        # Store the transition in memory
        agent.memory.push(deepcopy(frame_next_state), action, r, terminal_st
        # Start training after random sample generation
        if(frame >= train_frame):
            agent.train_policy_net(frame)
            # Update the target network only for Double DQN only
            if double_dqn and (frame % update_target_network_frequency)== 0:
                agent.update_target_net()
        score += reward
        history[:4, :, :] = history[1:, :, :]

        if done:
            evaluation_reward.append(score)
            rewards.append(np.mean(evaluation_reward))
            episodes.append(e)
            pylab.plot(episodes, rewards, 'b')
            pylab.xlabel('Episodes')
            pylab.ylabel('Rewards')
            pylab.title('Episodes vs Reward')
            pylab.savefig("./save_graph/breakout_dqn.png") # save graph for

            # every episode, plot the play time
            print("episode:", e, "  score:", score, "  memory length:",
                  len(agent.memory), "  epsilon:", agent.epsilon, "   steps:
```

```
                      "    lr:", agent.optimizer.param_groups[0]['lr'], "    eval

            # if the mean of scores of last 100 episode is bigger than 8 sav
            ### Change this save condition to whatever you prefer ###
            if np.mean(evaluation_reward) >= 8 and np.mean(evaluation_reward
                torch.save(agent.policy_net, "./save_model/breakout_dqn.pth"
                best_eval_reward = np.mean(evaluation_reward)
```

```
/tmp/ipykernel_975387/1450953243.py:21: DeprecationWarning: elementwise comp
arison failed; this will raise an error in the future.
  if step > 1 and len(np.unique(next_state[:189] == state[:189])) < 2:
episode: 0   score: 2.0   memory length: 198   epsilon: 1.0    steps: 198
lr: 0.0001     evaluation reward: 2.0
episode: 1   score: 3.0   memory length: 445   epsilon: 1.0    steps: 247
lr: 0.0001     evaluation reward: 2.5
episode: 2   score: 0.0   memory length: 568   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.6666666666666667
episode: 3   score: 3.0   memory length: 815   epsilon: 1.0    steps: 247
lr: 0.0001     evaluation reward: 2.0
episode: 4   score: 2.0   memory length: 1012   epsilon: 1.0    steps: 197
lr: 0.0001     evaluation reward: 2.0
episode: 5   score: 2.0   memory length: 1210   epsilon: 1.0    steps: 198
lr: 0.0001     evaluation reward: 2.0
episode: 6   score: 3.0   memory length: 1438   epsilon: 1.0    steps: 228
lr: 0.0001     evaluation reward: 2.142857142857143
episode: 7   score: 1.0   memory length: 1606   epsilon: 1.0    steps: 168
lr: 0.0001     evaluation reward: 2.0
episode: 8   score: 5.0   memory length: 1913   epsilon: 1.0    steps: 307
lr: 0.0001     evaluation reward: 2.3333333333333335
episode: 9   score: 2.0   memory length: 2111   epsilon: 1.0    steps: 198
lr: 0.0001     evaluation reward: 2.3
episode: 10   score: 3.0   memory length: 2339   epsilon: 1.0    steps: 228
lr: 0.0001     evaluation reward: 2.3636363636363638
episode: 11   score: 1.0   memory length: 2511   epsilon: 1.0    steps: 172
lr: 0.0001     evaluation reward: 2.25
episode: 12   score: 3.0   memory length: 2757   epsilon: 1.0    steps: 246
lr: 0.0001     evaluation reward: 2.3076923076923075
episode: 13   score: 4.0   memory length: 3016   epsilon: 1.0    steps: 259
lr: 0.0001     evaluation reward: 2.4285714285714284
episode: 14   score: 1.0   memory length: 3188   epsilon: 1.0    steps: 172
lr: 0.0001     evaluation reward: 2.3333333333333335
episode: 15   score: 0.0   memory length: 3311   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 2.1875
episode: 16   score: 1.0   memory length: 3480   epsilon: 1.0    steps: 169
lr: 0.0001     evaluation reward: 2.1176470588235294
episode: 17   score: 1.0   memory length: 3649   epsilon: 1.0    steps: 169
lr: 0.0001     evaluation reward: 2.0555555555555554
episode: 18   score: 1.0   memory length: 3800   epsilon: 1.0    steps: 151
lr: 0.0001     evaluation reward: 2.0
episode: 19   score: 2.0   memory length: 4018   epsilon: 1.0    steps: 218
lr: 0.0001     evaluation reward: 2.0
episode: 20   score: 3.0   memory length: 4265   epsilon: 1.0    steps: 247
lr: 0.0001     evaluation reward: 2.0476190476190474
```

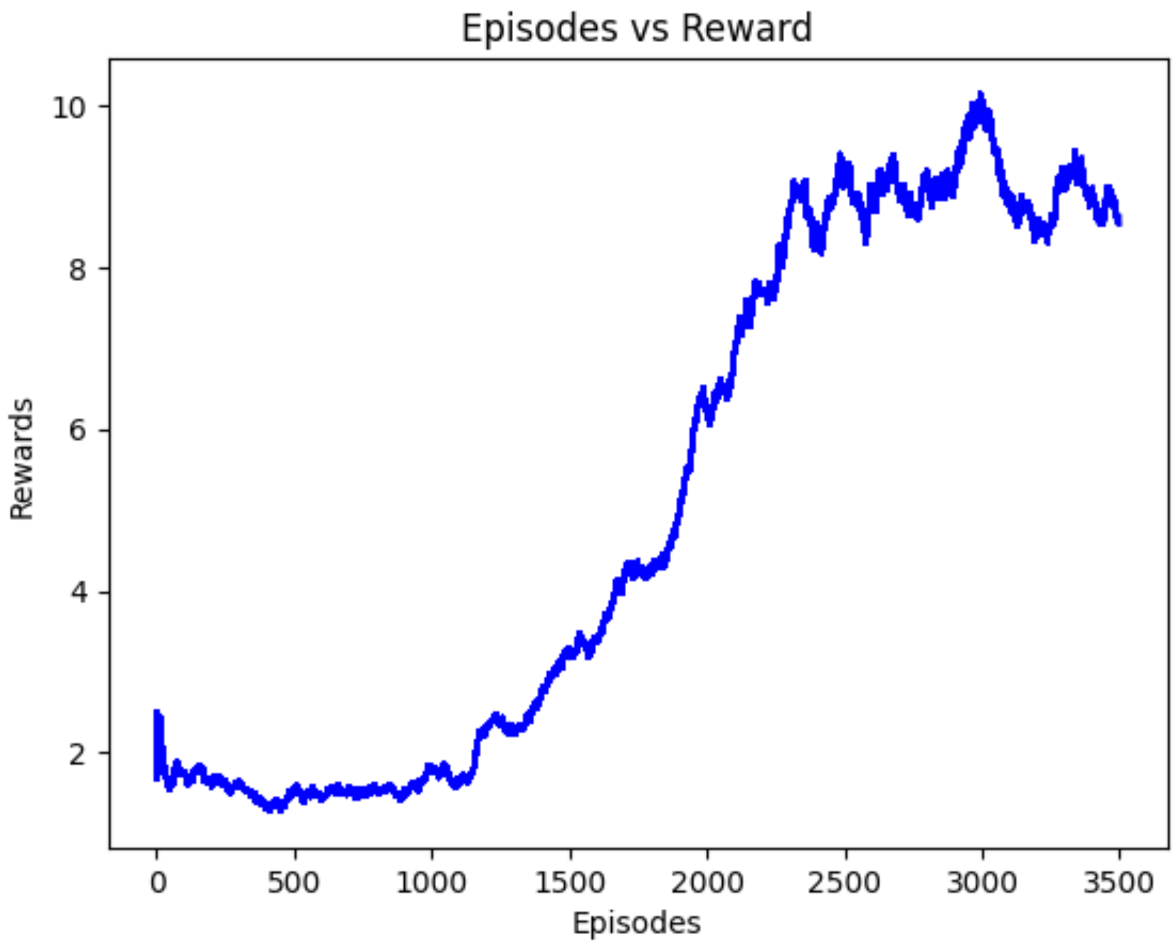
```
episode: 21   score: 2.0   memory length: 4452   epsilon: 1.0    steps: 187
lr: 0.0001     evaluation reward: 2.0454545454545454
episode: 22   score: 0.0   memory length: 4574   epsilon: 1.0    steps: 122
lr: 0.0001     evaluation reward: 1.9565217391304348
episode: 23   score: 2.0   memory length: 4792   epsilon: 1.0    steps: 218
lr: 0.0001     evaluation reward: 1.9583333333333333
episode: 24   score: 0.0   memory length: 4914   epsilon: 1.0    steps: 122
lr: 0.0001     evaluation reward: 1.88
episode: 25   score: 1.0   memory length: 5085   epsilon: 1.0    steps: 171
lr: 0.0001     evaluation reward: 1.8461538461538463
episode: 26   score: 0.0   memory length: 5208   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.7777777777777777
episode: 27   score: 1.0   memory length: 5377   epsilon: 1.0    steps: 169
lr: 0.0001     evaluation reward: 1.75
episode: 28   score: 3.0   memory length: 5602   epsilon: 1.0    steps: 225
lr: 0.0001     evaluation reward: 1.793103448275862
episode: 29   score: 2.0   memory length: 5821   epsilon: 1.0    steps: 219
lr: 0.0001     evaluation reward: 1.8
episode: 30   score: 0.0   memory length: 5943   epsilon: 1.0    steps: 122
lr: 0.0001     evaluation reward: 1.7419354838709677
episode: 31   score: 4.0   memory length: 6257   epsilon: 1.0    steps: 314
lr: 0.0001     evaluation reward: 1.8125
episode: 32   score: 0.0   memory length: 6379   epsilon: 1.0    steps: 122
lr: 0.0001     evaluation reward: 1.7575757575757576
episode: 33   score: 0.0   memory length: 6502   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.7058823529411764
episode: 34   score: 1.0   memory length: 6673   epsilon: 1.0    steps: 171
lr: 0.0001     evaluation reward: 1.6857142857142857
episode: 35   score: 2.0   memory length: 6873   epsilon: 1.0    steps: 200
lr: 0.0001     evaluation reward: 1.6944444444444444
episode: 36   score: 0.0   memory length: 6996   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.6486486486486487
episode: 37   score: 0.0   memory length: 7119   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.605263157894737
episode: 38   score: 1.0   memory length: 7270   epsilon: 1.0    steps: 151
lr: 0.0001     evaluation reward: 1.5897435897435896
episode: 39   score: 3.0   memory length: 7539   epsilon: 1.0    steps: 269
lr: 0.0001     evaluation reward: 1.625
episode: 40   score: 2.0   memory length: 7737   epsilon: 1.0    steps: 198
lr: 0.0001     evaluation reward: 1.6341463414634145
episode: 41   score: 2.0   memory length: 7935   epsilon: 1.0    steps: 198
lr: 0.0001     evaluation reward: 1.6428571428571428
episode: 42   score: 0.0   memory length: 8058   epsilon: 1.0    steps: 123
lr: 0.0001     evaluation reward: 1.6046511627906976
episode: 43   score: 1.0   memory length: 8227   epsilon: 1.0    steps: 169
lr: 0.0001     evaluation reward: 1.5909090909090908
episode: 44   score: 0.0   memory length: 8349   epsilon: 1.0    steps: 122
lr: 0.0001     evaluation reward: 1.5555555555555556
episode: 45   score: 1.0   memory length: 8517   epsilon: 1.0    steps: 168
lr: 0.0001     evaluation reward: 1.5434782608695652
episode: 46   score: 2.0   memory length: 8720   epsilon: 1.0    steps: 203
lr: 0.0001     evaluation reward: 1.553191489361702
episode: 47   score: 3.0   memory length: 8968   epsilon: 1.0    steps: 248
```

```
episode: 3435   score: 10.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 466    lr: 2.6214400000000017e-08     evaluation reward:
8.64
episode: 3436   score: 14.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 597    lr: 2.6214400000000017e-08     evaluation reward:
8.69
episode: 3437   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 424    lr: 2.6214400000000017e-08     evaluation reward: 8
.62
episode: 3438   score: 6.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 359    lr: 2.6214400000000017e-08     evaluation reward: 8
.55
episode: 3439   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 497    lr: 2.6214400000000017e-08     evaluation reward: 8
.54
episode: 3440   score: 11.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 502    lr: 2.6214400000000017e-08     evaluation reward:
8.58
episode: 3441   score: 6.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 340    lr: 2.6214400000000017e-08     evaluation reward: 8
.54
episode: 3442   score: 13.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 633    lr: 2.6214400000000017e-08     evaluation reward:
8.61
episode: 3443   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 454    lr: 2.6214400000000017e-08     evaluation reward: 8
.63
episode: 3444   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 507    lr: 2.6214400000000017e-08     evaluation reward: 8
.64
episode: 3445   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 477    lr: 2.6214400000000017e-08     evaluation reward: 8
.7
episode: 3446   score: 16.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 685    lr: 2.6214400000000017e-08     evaluation reward:
8.8
episode: 3447   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 434    lr: 2.6214400000000017e-08     evaluation reward: 8
.78
episode: 3448   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 386    lr: 2.6214400000000017e-08     evaluation reward: 8
.8
episode: 3449   score: 12.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 563    lr: 2.6214400000000017e-08     evaluation reward:
8.84
episode: 3450   score: 12.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 630    lr: 2.6214400000000017e-08     evaluation reward:
8.76
episode: 3451   score: 14.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 588    lr: 2.6214400000000017e-08     evaluation reward:
8.83
episode: 3452   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 425    lr: 2.6214400000000017e-08     evaluation reward: 8
```

```
.86
episode: 3453   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 455    lr: 2.6214400000000017e-08    evaluation reward: 8
.88
episode: 3454   score: 14.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 562    lr: 2.6214400000000017e-08    evaluation reward:
8.91
episode: 3455   score: 18.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 678    lr: 2.6214400000000017e-08    evaluation reward:
8.99
episode: 3456   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 407    lr: 2.6214400000000017e-08    evaluation reward: 8
.97
episode: 3457   score: 12.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 435    lr: 2.6214400000000017e-08    evaluation reward:
8.94
episode: 3458   score: 16.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 625    lr: 2.6214400000000017e-08    evaluation reward:
9.0
episode: 3459   score: 6.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 345    lr: 2.6214400000000017e-08    evaluation reward: 8
.94
episode: 3460   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 451    lr: 2.6214400000000017e-08    evaluation reward: 8
.92
episode: 3461   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 456    lr: 2.6214400000000017e-08    evaluation reward: 8
.94
episode: 3462   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 439    lr: 2.6214400000000017e-08    evaluation reward: 8
.99
episode: 3463   score: 7.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 407    lr: 2.6214400000000017e-08    evaluation reward: 8
.98
episode: 3464   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 421    lr: 2.6214400000000017e-08    evaluation reward: 9
.0
episode: 3465   score: 5.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 303    lr: 2.6214400000000017e-08    evaluation reward: 8
.97
episode: 3466   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 407    lr: 2.6214400000000017e-08    evaluation reward: 8
.94
episode: 3467   score: 6.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 354    lr: 2.6214400000000017e-08    evaluation reward: 8
.95
episode: 3468   score: 4.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 279    lr: 2.6214400000000017e-08    evaluation reward: 8
.93
episode: 3469   score: 11.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 572    lr: 2.6214400000000017e-08    evaluation reward:
8.89
episode: 3470   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
```

8555413 steps: 502 lr: 2.6214400000000017e-08 evaluation reward: 8
.9
episode: 3471 score: 12.0 memory length: 1000000 epsilon: 0.0099980200
08555413 steps: 483 lr: 2.6214400000000017e-08 evaluation reward:
8.93
episode: 3472 score: 5.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 283 lr: 2.6214400000000017e-08 evaluation reward: 8
.84
episode: 3473 score: 6.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 338 lr: 2.6214400000000017e-08 evaluation reward: 8
.84
episode: 3474 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 391 lr: 2.6214400000000017e-08 evaluation reward: 8
.88
episode: 3475 score: 5.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 289 lr: 2.6214400000000017e-08 evaluation reward: 8
.85
episode: 3476 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 440 lr: 2.6214400000000017e-08 evaluation reward: 8
.79
episode: 3477 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 404 lr: 2.6214400000000017e-08 evaluation reward: 8
.84
episode: 3478 score: 5.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 290 lr: 2.6214400000000017e-08 evaluation reward: 8
.83
episode: 3479 score: 2.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 198 lr: 2.6214400000000017e-08 evaluation reward: 8
.74
episode: 3480 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 375 lr: 2.6214400000000017e-08 evaluation reward: 8
.75
episode: 3481 score: 6.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 371 lr: 2.6214400000000017e-08 evaluation reward: 8
.75
episode: 3482 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 407 lr: 2.6214400000000017e-08 evaluation reward: 8
.79
episode: 3483 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 406 lr: 2.6214400000000017e-08 evaluation reward: 8
.8
episode: 3484 score: 7.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 387 lr: 2.6214400000000017e-08 evaluation reward: 8
.81
episode: 3485 score: 8.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 426 lr: 2.6214400000000017e-08 evaluation reward: 8
.75
episode: 3486 score: 3.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 213 lr: 2.6214400000000017e-08 evaluation reward: 8
.64
episode: 3487 score: 7.0 memory length: 1000000 epsilon: 0.00999802000
8555413 steps: 367 lr: 2.6214400000000017e-08 evaluation reward: 8
.62

```
episode: 3488   score: 7.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 366    lr: 2.6214400000000017e-08     evaluation reward: 8
.63
episode: 3489   score: 3.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 226    lr: 2.6214400000000017e-08     evaluation reward: 8
.59
episode: 3490   score: 7.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 405    lr: 2.6214400000000017e-08     evaluation reward: 8
.6
episode: 3491   score: 11.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 540    lr: 2.6214400000000017e-08     evaluation reward:
8.63
episode: 3492   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 492    lr: 2.6214400000000017e-08     evaluation reward: 8
.64
episode: 3493   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 484    lr: 2.6214400000000017e-08     evaluation reward: 8
.55
episode: 3494   score: 13.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 584    lr: 2.6214400000000017e-08     evaluation reward:
8.57
episode: 3495   score: 9.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 418    lr: 2.6214400000000017e-08     evaluation reward: 8
.59
episode: 3496   score: 10.0   memory length: 1000000   epsilon: 0.0099980200
08555413    steps: 513    lr: 2.6214400000000017e-08     evaluation reward:
8.62
episode: 3497   score: 6.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 324    lr: 2.6214400000000017e-08     evaluation reward: 8
.64
episode: 3498   score: 7.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 383    lr: 2.6214400000000017e-08     evaluation reward: 8
.55
episode: 3499   score: 8.0   memory length: 1000000   epsilon: 0.00999802000
8555413    steps: 421    lr: 2.6214400000000017e-08     evaluation reward: 8
.53
```



Visualize Agent Performance

BE AWARE THIS CODE BELOW MAY CRASH THE KERNEL IF YOU RUN THE SAME CELL TWICE.

Please save your model before running this portion of the code.

In [10]:
```
torch.save(agent.policy_net, "./save_model/breakout_dqn_latest.pth")
```

In [ ]:
```
from gym.wrappers import Monitor
#from gym.wrappers.monitor import Monitor
import glob
import io
import base64

from IPython.display import HTML
from IPython import display as ipythondisplay

from pyvirtualdisplay import Display

# Displaying the game live
def show_state(env, step=0, info=""):
    plt.figure(3)
    plt.clf()
    plt.imshow(env.render(mode='rgb_array'))
    plt.title("%s | Step: %d %s" % ("Agent Playing",step, info))
    plt.axis('off')

    ipythondisplay.clear_output(wait=True)
    ipythondisplay.display(plt.gcf())

# Recording the game and replaying the game afterwards
def show_video():
    mp4list = glob.glob('video/*.mp4')
    if len(mp4list) > 0:
        mp4 = mp4list[0]
        video = io.open(mp4, 'r+b').read()
        encoded = base64.b64encode(video)
        ipythondisplay.display(HTML(data='''<video alt="test" autoplay
                loop controls style="height: 400px;">
                <source src="data:video/mp4;base64,{0}" type="video/mp4" />
             </video>'''.format(encoded.decode('ascii'))))
    else:
        print("Could not find video")


def wrap_env(env):
    env = Monitor(env, './video', force=True)
    return env
```

In [ ]:
```
display = Display(visible=0, size=(300, 200))
display.start()

# Load agent
# agent.load_policy_net("./save_model/breakout_dqn.pth")
agent.epsilon = 0.0 # Set agent to only exploit the best action

env = gym.make('BreakoutDeterministic-v5')
env = wrap_env(env)

done = False
```

```
score = 0
step = 0
state = env.reset()
next_state = state
life = number_lives
history = np.zeros([5, 84, 84], dtype=np.uint8)
get_init_state(history, state)

while not done:

    # Render breakout
    env.render()
#     show_state(env,step) # uncommenting this provides another way to visua

    step += 1
    frame += 1

    # Perform a fire action if ball is no longer on screen
    if step > 1 and len(np.unique(next_state[:189] == state[:189])) < 2:
        action = 0
    else:
        action = agent.get_action(np.float32(history[:4, :, :]) / 255.)
    state = next_state

    next_state, reward, done, info = env.step(action + 1)

    frame_next_state = get_frame(next_state)
    history[4, :, :] = frame_next_state
    terminal_state = check_live(life, info['lives'])

    life = info['lives']
    r = np.clip(reward, -1, 1)
    r = reward

    # Store the transition in memory
    agent.memory.push(deepcopy(frame_next_state), action, r, terminal_state)
    # Start training after random sample generation
    score += reward

    history[:4, :, :] = history[1:, :, :]
env.close()
show_video()
display.stop()
```

```
In [ ]:
```