```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
%cd '/content/drive/MyDrive/rld/assignment5_materials/assignment5_materials'
```

```
/content/drive/MyDrive/rld/assignment5_materials/assignment5_materials
```

```
!ls -ltr
```

```
total 7585
-rw------- 1 root root     549 May  4 20:50  utils.py
-rw------- 1 root root     844 May  4 20:50  model.py
-rw------- 1 root root    2352 May  4 20:50  memory.py
-rw------- 1 root root    2919 May  4 20:50  agent.py
-rw------- 1 root root   20465 May  5 14:38  breakout_dqn.png
-rw------- 1 root root   21243 May  5 18:28  breakout_dqn_taaha_da_1.png
-rw------- 1 root root  468897 May  5 21:06 'Copy of MP5.ipynb'
-rw------- 1 root root   19595 May  5 21:06  breakout_dqn_n2.png
-rw------- 1 root root     376 May  5 22:17  config.py
-rw------- 1 root root  328299 May  6 02:27  Taaha_9pm_1.ipynb
-rw------- 1 root root   18751 May  6 02:27  breakout_dqn_Taaha_9pm.png
-rw------- 1 root root    7590 May  6 02:47 'CS 444 MP5 Report.docx'
-rw------- 1 root root   18239 May  6 03:07  breakout_dqn_n1.png
-rw------- 1 root root   44628 May  6 03:12  MP5.ipynb
drwx------ 2 root root    4096 May  6 05:13  __pycache__
drwx------ 2 root root    4096 May  6 05:13  old
-rw------- 1 root root    4294 May  6 05:58  agent_double.py
-rw------- 1 root root   21391 May  6 06:04  breakout_dqn_taaha_jon.png
-rw------- 1 root root 6755816 May  6 14:05  1_am_breakout_dqn.pth
-rw------- 1 root root   18379 May  6 14:54  1_am.png
```

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.

```
!pip3 install gym==0.25.2 pyvirtualdisplay
!sudo apt-get install -y xvfb python-opengl ffmpeg
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gym==0.25.2 in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: pyvirtualdisplay in /usr/local/lib/python3.10/dist-packages (3.0)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym==0.25.2) (1.22.4)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym==0.25.2) (0.0.8)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym==0.25.2) (2.2.1)
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-opengl is already the newest version (3.1.0+dfsg-2build1).
ffmpeg is already the newest version (7:4.2.7-0ubuntu0.1).
xvfb is already the newest version (2:1.20.13-1ubuntu1~20.04.8).
0 upgraded, 0 newly installed, 0 to remove and 24 not upgraded.
```

```
!pip3 install --upgrade setuptools --user
!pip3 install ez_setup
!pip3 install gym[atari]
!pip3 install gym[accept-rom-license]
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (67.7.2)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting ez_setup
  Downloading ez_setup-0.9.tar.gz (6.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: ez_setup
  Building wheel for ez_setup (setup.py) ... done
  Created wheel for ez_setup: filename=ez_setup-0.9-py3-none-any.whl size=11012 sha256=3b926d5073ca5ca2432bd25315c437258f4587875e9569f47
  Stored in directory: /root/.cache/pip/wheels/7a/d6/77/8f495e85fb7df23d41c328b9ea3cf0d9e83631b20bba479293
Successfully built ez_setup
Installing collected packages: ez_setup
Successfully installed ez_setup-0.9
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
```

```
Requirement already satisfied: gym[atari] in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym[atari]) (0.0.8)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym[atari]) (2.2.1)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym[atari]) (1.22.4)
Collecting ale-py~=0.7.5
  Downloading ale_py-0.7.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 19.5 MB/s eta 0:00:00
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/dist-packages (from ale-py~=0.7.5->gym[atari]) (5.12.0)
Installing collected packages: ale-py
Successfully installed ale-py-0.7.5
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gym[accept-rom-license] in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym[accept-rom-license]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym[accept-rom-license]) (0.0.8)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym[accept-rom-license]) (1.22.4)
Collecting autorom[accept-rom-license]~=0.4.2
  Downloading AutoROM-0.4.2-py3-none-any.whl (16 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from autorom[accept-rom-license]~=0.4.2->gym[accept-rom
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from autorom[accept-rom-license]~=0.4.2->gym[accept-
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from autorom[accept-rom-license]~=0.4.2->gym[accept-rom-
Collecting AutoROM.accept-rom-license
  Downloading AutoROM.accept-rom-license-0.6.1.tar.gz (434 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 434.7/434.7 kB 9.2 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-l
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-licer
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]~=0.4.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->autorom[accept-rom-license]
Building wheels for collected packages: AutoROM.accept-rom-license
  Building wheel for AutoROM.accept-rom-license (pyproject.toml) ... done
  Created wheel for AutoROM.accept-rom-license: filename=AutoROM.accept_rom_license-0.6.1-py3-none-any.whl size=446676 sha256=4f6a008ea9
  Stored in directory: /root/.cache/pip/wheels/6b/1b/ef/a43ff1a2f1736d5711faa1ba4c1f61be1131b8899e6a057811
Successfully built AutoROM.accept-rom-license
Installing collected packages: AutoROM.accept-rom-license, autorom
Successfully installed AutoROM.accept-rom-license-0.6.1 autorom-0.4.2
```

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**.

```
%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
```

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://www.gymlibrary.dev/environments/atari/breakout/.

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.

```
env = gym.make('BreakoutDeterministic-v4')
state = env.reset()
```

```
/usr/local/lib/python3.10/dist-packages/gym/core.py:317: DeprecationWarning: WARN: Initializing wrapper in old step API which returns or
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/wrappers/step_api_compatibility.py:39: DeprecationWarning: WARN: Initializing environment ir
  deprecation(
```

```
number_lives = find_max_lives(env)
state_size = env.observation_space.shape
action_size = 3 #fire, left, and right
```

```
/usr/local/lib/python3.10/dist-packages/gym/utils/passive_env_checker.py:227: DeprecationWarning: WARN: Core environment is written in c
  logger.deprecation(
```

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.

```
#double_dqn = False # set to True if using double DQN agent
double_dqn = True
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
```

```
!cp /content/drive/MyDrive/rld/assignment5_materials/assignment5_materials/1_am_breakout_dqn.pth /content/
```

```
agent.load_policy_net('/content/1_am_breakout_dqn.pth')
```

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"

```
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
    state = env.reset()
    next_state = state
    life = number_lives

    get_init_state(history, state, HISTORY_SIZE)

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

        # Perform a fire action if ball is no longer on screen to continue onto next life
```

```
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 = reward

# Store the transition in memory
agent.memory.push(deepcopy(frame_next_state), action, r, terminal_state)
# 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("1_am.png") # save graph for training visualization

    # every episode, plot the play time
    print("episode:", e, "  score:", score, "  memory length:",
          len(agent.memory), "  epsilon:", agent.epsilon, "   steps:", step,
          "   lr:", agent.optimizer.param_groups[0]['lr'], "    evaluation reward:", np.mean(evaluation_reward))

    # if the mean of scores of last 100 episode is bigger than 5 save model
    ### Change this save condition to whatever you prefer ###
    if np.mean(evaluation_reward) > 5 and np.mean(evaluation_reward) > best_eval_reward:
        torch.save(agent.policy_net, "1_am_breakout_dqn.pth")
        best_eval_reward = np.mean(evaluation_reward)
```

```
episode: 2584   score: 12.0   memory length: 868440   epsilon: 0.009998020008555413    steps: 458    lr: 1.638400000000001e-07     eva
episode: 2585   score: 20.0   memory length: 869174   epsilon: 0.009998020008555413    steps: 734    lr: 1.638400000000001e-07     eva
episode: 2586   score: 20.0   memory length: 869958   epsilon: 0.009998020008555413    steps: 784    lr: 1.638400000000001e-07     eva
episode: 2587   score: 10.0   memory length: 870405   epsilon: 0.009998020008555413    steps: 447    lr: 1.638400000000001e-07     eva
episode: 2588   score: 13.0   memory length: 871001   epsilon: 0.009998020008555413    steps: 596    lr: 1.638400000000001e-07     eva
episode: 2589   score: 16.0   memory length: 871595   epsilon: 0.009998020008555413    steps: 594    lr: 1.638400000000001e-07     eva
episode: 2590   score: 17.0   memory length: 872201   epsilon: 0.009998020008555413    steps: 606    lr: 1.638400000000001e-07     eva
episode: 2591   score: 8.0   memory length: 872577   epsilon: 0.009998020008555413    steps: 376    lr: 1.638400000000001e-07     eval
episode: 2592   score: 13.0   memory length: 873148   epsilon: 0.009998020008555413    steps: 571    lr: 1.638400000000001e-07     eva
episode: 2593   score: 9.0   memory length: 873593   epsilon: 0.009998020008555413    steps: 445    lr: 1.638400000000001e-07     eval
episode: 2594   score: 15.0   memory length: 874283   epsilon: 0.009998020008555413    steps: 690    lr: 1.638400000000001e-07     eva
episode: 2595   score: 6.0   memory length: 874625   epsilon: 0.009998020008555413    steps: 342    lr: 1.638400000000001e-07     eval
episode: 2596   score: 14.0   memory length: 875306   epsilon: 0.009998020008555413    steps: 681    lr: 1.638400000000001e-07     eva
episode: 2597   score: 8.0   memory length: 875728   epsilon: 0.009998020008555413    steps: 422    lr: 1.638400000000001e-07     eval
episode: 2598   score: 20.0   memory length: 876346   epsilon: 0.009998020008555413    steps: 618    lr: 1.638400000000001e-07     eva
episode: 2599   score: 16.0   memory length: 876915   epsilon: 0.009998020008555413    steps: 569    lr: 1.638400000000001e-07     eva
episode: 2600   score: 7.0   memory length: 877323   epsilon: 0.009998020008555413    steps: 408    lr: 1.638400000000001e-07     eval
episode: 2601   score: 14.0   memory length: 878009   epsilon: 0.009998020008555413    steps: 686    lr: 1.638400000000001e-07     eva
episode: 2602   score: 12.0   memory length: 878577   epsilon: 0.009998020008555413    steps: 568    lr: 1.638400000000001e-07     eva
episode: 2603   score: 16.0   memory length: 879254   epsilon: 0.009998020008555413    steps: 677    lr: 1.638400000000001e-07     eva
episode: 2604   score: 16.0   memory length: 879853   epsilon: 0.009998020008555413    steps: 599    lr: 1.638400000000001e-07     eva
episode: 2605   score: 16.0   memory length: 880487   epsilon: 0.009998020008555413    steps: 634    lr: 1.638400000000001e-07     eva
episode: 2606   score: 18.0   memory length: 881276   epsilon: 0.009998020008555413    steps: 789    lr: 1.638400000000001e-07     eva
episode: 2607   score: 17.0   memory length: 881978   epsilon: 0.009998020008555413    steps: 702    lr: 1.638400000000001e-07     eva
```

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.

```
#from gym.wrappers import Monitor # If importing monitor raises issues, try using `from gym.wrappers import RecordVideo`
from gym.wrappers.record_video import RecordVideo

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 = RecordVideo(env, './video')
    return env
```

```
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-v4')
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, history.shape[0])

while not done:

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

    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
    #print(info)
    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()
```

```
/usr/local/lib/python3.10/dist-packages/gym/core.py:317: DeprecationWarning: WARN: Init
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/wrappers/step_api_compatibility.py:39: Depre
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/wrappers/record_video.py:78: UserWarning: W
  logger.warn(
/usr/local/lib/python3.10/dist-packages/gym/wrappers/monitoring/video_recorder.py:78: De
  logger.deprecation(
/usr/local/lib/python3.10/dist-packages/gym/core.py:43: DeprecationWarning: WARN: The ar
See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/utils/passive_env_checker.py:297: UserWarnin
  logger.warn(
/usr/local/lib/python3.10/dist-packages/gym/core.py:49: DeprecationWarning: WARN: You ar
If you want to render in human mode, initialize the environment in this way: gym.make('E
See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(
/usr/local/lib/python3.10/dist-packages/gym/utils/passive_env_checker.py:227: Deprecatio
  logger.deprecation(
/usr/local/lib/python3.10/dist-packages/gym/core.py:43: DeprecationWarning: WARN: The ar
See here for more information: https://www.gymlibrary.ml/content/api/
  deprecation(
```

```
<pyvirtualdisplay.display.Display at 0x7f12ee9c3a90>
```