Gurhar Khalsa's Final Project

- github url: https://github.com/gurhar1133/deepQAgent

Problem Statement and Goal

- This project focuses on using reinforcement learning techniques to train and agent to play Space Invaders
- The goal is to get the agent to learn strategies that are better than just taking random action choices

The Environment

- The environment is a 210 by 160 pixel frame with rgb colors. Thus each frame is a 210x160x3 tensor. The agent has 6 action choices to choose from given an environment frame. The actions are move right, move left, move right and shoot, move left and shoot, shoot without moving, and no-op.

Approach

- I created my own deep q learning agent. You can see from the code below that the training function has options for experience replay and non-experience replay. (see code comments)
- The agent is a deep q learning agent. Such an agent is like a q learning agent, but for state spaces in which a q table would be too small for the large state space.
- The agent uses a neural network model to approximate the q table, refitting the model based on experiences interacting with the environment
- I also included epsilon decay so that the agent is more exploratory at first and settles on a more fixed strategy later in training
- Experience replay is a deep reinforcement learning technique in which the agent stores state, action, newstate, reward tuples it experiences while interacting with the environment in a memory buffer. It then uses randomly sampled minibatches from the experience buffer to train the model. This is helpful because a requirement of SGD optimization is that training data is independent and identically distributed. But training experience sequences are often highly correlated. Experience replay with random minibatch sampling avoids the negative effects of this correlation.

In [ ]:
```
!pip install tensorflow==2.3.1 gym keras-rl2 gym[atari]
```

In [1]:
```
!pip install PyVirtualDisplay
!sudo apt-get install xvfb
```

```
Collecting PyVirtualDisplay
  Downloading PyVirtualDisplay-2.2-py3-none-any.whl (15 kB)
```

```
Collecting EasyProcess
  Downloading EasyProcess-0.3-py2.py3-none-any.whl (7.9 kB)
Installing collected packages: EasyProcess, PyVirtualDisplay
Successfully installed EasyProcess-0.3 PyVirtualDisplay-2.2
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  xvfb
0 upgraded, 1 newly installed, 0 to remove and 37 not upgraded.
Need to get 784 kB of archives.
After this operation, 2,270 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic-updates/universe amd64 xvfb amd64
2:1.19.6-1ubuntu4.9 [784 kB]
Fetched 784 kB in 1s (729 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based fronte
nd cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76, <> li
ne 1.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package xvfb.
(Reading database ... 155222 files and directories currently installed.)
Preparing to unpack .../xvfb_2%3a1.19.6-1ubuntu4.9_amd64.deb ...
Unpacking xvfb (2:1.19.6-1ubuntu4.9) ...
Setting up xvfb (2:1.19.6-1ubuntu4.9) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
```

In [2]:
```
import os
os.getcwd()
os.listdir()
```

Out[2]:
```
['.config', 'sample_data']
```

In [3]:
```
os.mkdir('videos')
```

In [4]:
```
import urllib.request
urllib.request.urlretrieve('http://www.atarimania.com/roms/Roms.rar','Roms.rar')
!pip install unrar
!unrar x Roms.rar
!mkdir rars
!mv HC\ ROMS.zip   rars
!mv ROMS.zip  rars
!python -m atari_py.import_roms rars
```

```
Collecting unrar
  Downloading unrar-0.4-py3-none-any.whl (25 kB)
Installing collected packages: unrar
Successfully installed unrar-0.4

UNRAR 5.50 freeware      Copyright (c) 1993-2017 Alexander Roshal


Extracting from Roms.rar

Extracting  HC ROMS.zip                                               36  OK
Extracting  ROMS.zip                                                  7 99  OK
```

```
All OK
copying adventure.bin from ROMS/Adventure (1980) (Atari, Warren Robinett) (CX261
3, CX2613P) (PAL).bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_r
oms/adventure.bin
copying air_raid.bin from ROMS/Air Raid (Men-A-Vision) (PAL) ~.bin to /usr/loca
l/lib/python3.7/dist-packages/atari_py/atari_roms/air_raid.bin
copying alien.bin from ROMS/Alien (1982) (20th Century Fox Video Games, Douglas
'Dallas North' Neubauer) (11006) ~.bin to /usr/local/lib/python3.7/dist-package
s/atari_py/atari_roms/alien.bin
copying amidar.bin from ROMS/Amidar (1982) (Parker Brothers, Ed Temple) (PB5310)
~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/amidar.bin
copying assault.bin from ROMS/Assault (AKA Sky Alien) (1983) (Bomb - Onbase) (CA
281).bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/assault.b
in
copying asterix.bin from ROMS/Asterix (AKA Taz) (07-27-1983) (Atari, Jerome Domu
rat, Steve Woita) (CX2696) (Prototype).bin to /usr/local/lib/python3.7/dist-pack
ages/atari_py/atari_roms/asterix.bin
copying asteroids.bin from ROMS/Asteroids (1981) (Atari, Brad Stewart - Sears)
(CX2649 - 49-75163) [no copyright] ~.bin to /usr/local/lib/python3.7/dist-packag
es/atari_py/atari_roms/asteroids.bin
copying atlantis.bin from ROMS/Atlantis (Lost City of Atlantis) (1982) (Imagic,
Dennis Koble) (720103-1A, 720103-1B, IA3203, IX-010-04) ~.bin to /usr/local/lib/
python3.7/dist-packages/atari_py/atari_roms/atlantis.bin
copying bank_heist.bin from ROMS/Bank Heist (Bonnie & Clyde, Cops 'n' Robbers, H
old-Up, Roaring 20's) (1983) (20th Century Fox Video Games, Bill Aspromonte) (11
012) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/bank_he
ist.bin
copying battle_zone.bin from ROMS/Battlezone (1983) (Atari - GCC, Mike Feinstei
n, Brad Rice) (CX2681) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/
atari_roms/battle_zone.bin
copying beam_rider.bin from ROMS/Beamrider (1984) (Activision - Cheshire Enginee
ring, David Rolfe, Larry Zwick) (AZ-037-04) ~.bin to /usr/local/lib/python3.7/di
st-packages/atari_py/atari_roms/beam_rider.bin
copying berzerk.bin from ROMS/Berzerk (1982) (Atari, Dan Hitchens - Sears) (CX26
50 - 49-75168) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_ro
ms/berzerk.bin
copying bowling.bin from ROMS/Bowling (1979) (Atari, Larry Kaplan - Sears) (CX26
28 - 6-99842, 49-75117) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_p
y/atari_roms/bowling.bin
copying boxing.bin from ROMS/Boxing - La Boxe (1980) (Activision, Bob Whitehead)
(AG-002, CAG-002, AG-002-04) ~.bin to /usr/local/lib/python3.7/dist-packages/ata
ri_py/atari_roms/boxing.bin
copying breakout.bin from ROMS/Breakout - Breakaway IV (Paddle) (1978) (Atari, B
rad Stewart - Sears) (CX2622 - 6-99813, 49-75107) ~.bin to /usr/local/lib/python
3.7/dist-packages/atari_py/atari_roms/breakout.bin
copying carnival.bin from ROMS/Carnival (1982) (Coleco - Woodside Design Associa
tes, Steve 'Jessica Stevens' Kitchen) (2468) ~.bin to /usr/local/lib/python3.7/d
ist-packages/atari_py/atari_roms/carnival.bin
copying centipede.bin from ROMS/Centipede (1983) (Atari - GCC) (CX2676) ~.bin to
/usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/centipede.bin
copying chopper_command.bin from ROMS/Chopper Command (1982) (Activision, Bob Wh
itehead) (AX-015, AX-015-04) ~.bin to /usr/local/lib/python3.7/dist-packages/ata
ri_py/atari_roms/chopper_command.bin
copying crazy_climber.bin from ROMS/Crazy Climber (1983) (Atari - Roklan, Joe Ga
ucher, Alex Leavens) (CX2683) ~.bin to /usr/local/lib/python3.7/dist-packages/at
ari_py/atari_roms/crazy_climber.bin
copying defender.bin from ROMS/Defender (1982) (Atari, Robert C. Polaro, Alan J.
Murphy - Sears) (CX2609 - 49-75186) ~.bin to /usr/local/lib/python3.7/dist-packa
ges/atari_py/atari_roms/defender.bin
copying demon_attack.bin from ROMS/Demon Attack (Death from Above) (1982) (Imagi
c, Rob Fulop) (720000-200, 720101-1B, 720101-1C, IA3200, IA3200C, IX-006-04) ~.b
in to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/demon_attack.bi
n
copying donkey_kong.bin from ROMS/Donkey Kong (1982) (Coleco - Woodside Design A
ssociates - Imaginative Systems Software, Garry Kitchen) (2451) ~.bin to /usr/lo
```

```
cal/lib/python3.7/dist-packages/atari_py/atari_roms/donkey_kong.bin
copying double_dunk.bin from ROMS/Double Dunk (Super Basketball) (1989) (Atari,
Matthew L. Hubbard) (CX26159) ~.bin to /usr/local/lib/python3.7/dist-packages/at
ari_py/atari_roms/double_dunk.bin
copying elevator_action.bin from ROMS/Elevator Action (1983) (Atari, Dan Hitchen
s) (CX26126) (Prototype) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_p
y/atari_roms/elevator_action.bin
copying enduro.bin from ROMS/Enduro (1983) (Activision, Larry Miller) (AX-026, A
X-026-04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/en
duro.bin
copying fishing_derby.bin from ROMS/Fishing Derby (1980) (Activision, David Cran
e) (AG-004) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/
fishing_derby.bin
copying freeway.bin from ROMS/Freeway (1981) (Activision, David Crane) (AG-009,
AG-009-04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/f
reeway.bin
copying frogger.bin from ROMS/Frogger (1982) (Parker Brothers, Ed English, David
Lamkins) (PB5300) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari
_roms/frogger.bin
copying frostbite.bin from ROMS/Frostbite (1983) (Activision, Steve Cartwright)
(AX-031) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/fro
stbite.bin
copying galaxian.bin from ROMS/Galaxian (1983) (Atari - GCC, Mark Ackerman, Tom
Calderwood, Glenn Parker) (CX2684) ~.bin to /usr/local/lib/python3.7/dist-packag
es/atari_py/atari_roms/galaxian.bin
copying gopher.bin from ROMS/Gopher (Gopher Attack) (1982) (U.S. Games Corporati
on - JWDA, Sylvia Day, Todd Marshall, Robin McDaniel, Henry Will IV) (VC2001) ~.
bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/gopher.bin
copying gravitar.bin from ROMS/Gravitar (1983) (Atari, Dan Hitchens, Mimi Nyden)
(CX2685) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/gra
vitar.bin
copying hero.bin from ROMS/H.E.R.O. (1984) (Activision, John Van Ryzin) (AZ-036-
04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/hero.bin
copying ice_hockey.bin from ROMS/Ice Hockey - Le Hockey Sur Glace (1981) (Activi
sion, Alan Miller) (AX-012, CAX-012, AX-012-04) ~.bin to /usr/local/lib/python3.
7/dist-packages/atari_py/atari_roms/ice_hockey.bin
copying jamesbond.bin from ROMS/James Bond 007 (James Bond Agent 007) (1984) (Pa
rker Brothers - On-Time Software, Joe Gaucher, Louis Marbel) (PB5110) ~.bin to /
usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/jamesbond.bin
copying journey_escape.bin from ROMS/Journey Escape (1983) (Data Age, J. Ray Det
tling) (112-006) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_
roms/journey_escape.bin
copying kaboom.bin from ROMS/Kaboom! (Paddle) (1981) (Activision, Larry Kaplan,
David Crane) (AG-010, AG-010-04) ~.bin to /usr/local/lib/python3.7/dist-package
s/atari_py/atari_roms/kaboom.bin
copying kangaroo.bin from ROMS/Kangaroo (1983) (Atari - GCC, Kevin Osborn) (CX26
89) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/kangaro
o.bin
copying keystone_kapers.bin from ROMS/Keystone Kapers - Raueber und Gendarm (198
3) (Activision, Garry Kitchen - Ariola) (EAX-025, EAX-025-04I - 711 025-725) (PA
L).bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/keystone_ka
pers.bin
copying king_kong.bin from ROMS/King Kong (1982) (Tigervision - Software Electro
nics Corporation, Karl T. Olinger - Teldec) (7-001 - 3.60001 VE) (PAL).bin to /u
sr/local/lib/python3.7/dist-packages/atari_py/atari_roms/king_kong.bin
copying koolaid.bin from ROMS/Kool-Aid Man (Kool Aid Pitcher Man) (1983) (M Netw
ork, Stephen Tatsumi, Jane Terjung - Kool Aid) (MT4648) ~.bin to /usr/local/lib/
python3.7/dist-packages/atari_py/atari_roms/koolaid.bin
copying krull.bin from ROMS/Krull (1983) (Atari, Jerome Domurat, Dave Staugas)
(CX2682) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/kru
ll.bin
copying kung_fu_master.bin from ROMS/Kung-Fu Master (1987) (Activision - Imagine
ering, Dan Kitchen, Garry Kitchen) (AG-039-04) ~.bin to /usr/local/lib/python3.
7/dist-packages/atari_py/atari_roms/kung_fu_master.bin
copying laser_gates.bin from ROMS/Laser Gates (AKA Innerspace) (1983) (Imagic, D
```

```
an Oliver) (720118-2A, 13208, EIX-007-04I) (PAL).bin to /usr/local/lib/python3.
7/dist-packages/atari_py/atari_roms/laser_gates.bin
copying lost_luggage.bin from ROMS/Lost Luggage (Airport Mayhem) (1982) (Apollo
 - Games by Apollo, Larry Minor, Ernie Runyon, Ed Salvo) (AP-2004) [no opening sc
ene] ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/lost_lu
ggage.bin
copying montezuma_revenge.bin from ROMS/Montezuma's Revenge - Featuring Panama J
oe (1984) (Parker Brothers - JWDA, Henry Will IV) (PB5760) ~.bin to /usr/local/l
ib/python3.7/dist-packages/atari_py/atari_roms/montezuma_revenge.bin
copying mr_do.bin from ROMS/Mr. Do! (1983) (CBS Electronics, Ed English) (4L447
8) (PAL).bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/mr_d
o.bin
copying ms_pacman.bin from ROMS/Ms. Pac-Man (1983) (Atari - GCC, Mark Ackerman,
Glenn Parker) (CX2675) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/
atari_roms/ms_pacman.bin
copying name_this_game.bin from ROMS/Name This Game (Guardians of Treasure) (198
3) (U.S. Games Corporation - JWDA, Roger Booth, Sylvia Day, Ron Dubren, Todd Mar
shall, Robin McDaniel, Wes Trager, Henry Will IV) (VC1007) ~.bin to /usr/local/l
ib/python3.7/dist-packages/atari_py/atari_roms/name_this_game.bin
copying pacman.bin from ROMS/Pac-Man (1982) (Atari, Tod Frye) (CX2646) (PAL).bin
to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/pacman.bin
copying phoenix.bin from ROMS/Phoenix (1983) (Atari - GCC, Mike Feinstein, John
Mracek) (CX2673) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_
roms/phoenix.bin
copying video_pinball.bin from ROMS/Pinball (AKA Video Pinball) (Zellers).bin to
/usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/video_pinball.bin
copying pitfall.bin from ROMS/Pitfall! - Pitfall Harry's Jungle Adventure (Jungl
e Runner) (1982) (Activision, David Crane) (AX-018, AX-018-04) ~.bin to /usr/loc
al/lib/python3.7/dist-packages/atari_py/atari_roms/pitfall.bin
copying pooyan.bin from ROMS/Pooyan (1983) (Konami) (RC 100-X 02) ~.bin to /usr/
local/lib/python3.7/dist-packages/atari_py/atari_roms/pooyan.bin
copying private_eye.bin from ROMS/Private Eye (1984) (Activision, Bob Whitehead)
(AG-034-04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/
private_eye.bin
copying qbert.bin from ROMS/Q-bert (1983) (Parker Brothers - Western Technologie
s, Dave Hampton, Tom Sloper) (PB5360) ~.bin to /usr/local/lib/python3.7/dist-pac
kages/atari_py/atari_roms/qbert.bin
copying riverraid.bin from ROMS/River Raid (1982) (Activision, Carol Shaw) (AX-0
20, AX-020-04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_ro
ms/riverraid.bin
copying road_runner.bin from patched version of ROMS/Road Runner (1989) (Atari -
Bobco, Robert C. Polaro) (CX2663) ~.bin to /usr/local/lib/python3.7/dist-package
s/atari_py/atari_roms/road_runner.bin
copying robotank.bin from ROMS/Robot Tank (Robotank) (1983) (Activision, Alan Mi
ller) (AZ-028, AG-028-04) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_
py/atari_roms/robotank.bin
copying seaquest.bin from ROMS/Seaquest (1983) (Activision, Steve Cartwright) (A
X-022) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/seaqu
est.bin
copying sir_lancelot.bin from ROMS/Sir Lancelot (1983) (Xonox - K-Tel Software -
Product Guild, Anthony R. Henderson) (99006, 6220) (PAL).bin to /usr/local/lib/p
ython3.7/dist-packages/atari_py/atari_roms/sir_lancelot.bin
copying skiing.bin from ROMS/Skiing - Le Ski (1980) (Activision, Bob Whitehead)
(AG-005, CAG-005, AG-005-04) ~.bin to /usr/local/lib/python3.7/dist-packages/ata
ri_py/atari_roms/skiing.bin
copying solaris.bin from ROMS/Solaris (The Last Starfighter, Star Raiders II, Un
iverse) (1986) (Atari, Douglas Neubauer, Mimi Nyden) (CX26136) ~.bin to /usr/loc
al/lib/python3.7/dist-packages/atari_py/atari_roms/solaris.bin
copying space_invaders.bin from ROMS/Space Invaders (1980) (Atari, Richard Maure
r - Sears) (CX2632 - 49-75153) ~.bin to /usr/local/lib/python3.7/dist-packages/a
tari_py/atari_roms/space_invaders.bin
copying star_gunner.bin from ROMS/Stargunner (1983) (Telesys, Alex Leavens) (100
5) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/star_gunn
er.bin
copying surround.bin from ROMS/Surround (32 in 1) (Bit Corporation) (R320).bin t
```

```
o /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/surround.bin
copying tennis.bin from ROMS/Tennis - Le Tennis (1981) (Activision, Alan Miller)
(AG-007, CAG-007) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari
_roms/tennis.bin
copying time_pilot.bin from ROMS/Time Pilot (1983) (Coleco - Woodside Design Ass
ociates, Harley H. Puthuff Jr.) (2663) ~.bin to /usr/local/lib/python3.7/dist-pa
ckages/atari_py/atari_roms/time_pilot.bin
copying trondead.bin from ROMS/TRON - Deadly Discs (TRON Joystick) (1983) (M Net
work - INTV - APh Technological Consulting, Jeff Ronne, Brett Stutz) (MT5662) ~.
bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/trondead.bin
copying tutankham.bin from ROMS/Tutankham (1983) (Parker Brothers, Dave Engman,
Dawn Stockbridge) (PB5340) ~.bin to /usr/local/lib/python3.7/dist-packages/atari
_py/atari_roms/tutankham.bin
copying up_n_down.bin from ROMS/Up 'n Down (1984) (SEGA - Beck-Tech, Steve Beck,
Phat Ho) (009-01) ~.bin to /usr/local/lib/python3.7/dist-packages/atari_py/atari
_roms/up_n_down.bin
copying venture.bin from ROMS/Venture (1982) (Coleco, Joseph Biel) (2457) ~.bin
to /usr/local/lib/python3.7/dist-packages/atari_py/atari_roms/venture.bin
copying pong.bin from ROMS/Video Olympics - Pong Sports (Paddle) (1977) (Atari,
Joe Decuir - Sears) (CX2621 - 99806, 6-99806, 49-75104) ~.bin to /usr/local/lib/
python3.7/dist-packages/atari_py/atari_roms/pong.bin
copying wizard_of_wor.bin from ROMS/Wizard of Wor (1982) (CBS Electronics - Rokl
an, Joe Hellesen, Joe Wagner) (M8774, M8794) ~.bin to /usr/local/lib/python3.7/d
ist-packages/atari_py/atari_roms/wizard_of_wor.bin
copying yars_revenge.bin from ROMS/Yars' Revenge (Time Freeze) (1982) (Atari, Ho
ward Scott Warshaw - Sears) (CX2655 - 49-75167) ~.bin to /usr/local/lib/python3.
7/dist-packages/atari_py/atari_roms/yars_revenge.bin
copying zaxxon.bin from ROMS/Zaxxon (1983) (Coleco) (2454) ~.bin to /usr/local/l
ib/python3.7/dist-packages/atari_py/atari_roms/zaxxon.bin
```

In [5]:

```
import pickle
import gym
from gym import wrappers
import numpy as np
#import pybullet_envs
import time
import tensorflow as tf
import io
import glob
import base64
from IPython.display import HTML
from IPython import display as ipythondisplay
from pyvirtualdisplay import Display
import os
import datetime
import time
import matplotlib.pyplot as plt
%matplotlib inline
```

In [6]:

```
!apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
!pip install -U colabgymrender
```

```
Collecting colabgymrender
  Downloading colabgymrender-1.0.9-py3-none-any.whl (3.1 kB)
Requirement already satisfied: moviepy in /usr/local/lib/python3.7/dist-packages
(from colabgymrender) (0.2.3.5)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
(from moviepy->colabgymrender) (1.18.5)
Requirement already satisfied: imageio<3.0,>=2.1.2 in /usr/local/lib/python3.7/d
ist-packages (from moviepy->colabgymrender) (2.4.1)
Requirement already satisfied: tqdm<5.0,>=4.11.2 in /usr/local/lib/python3.7/dis
```

```
t-packages (from moviepy->colabgymrender) (4.62.3)
Requirement already satisfied: decorator<5.0,>=4.0.2 in /usr/local/lib/python3.
7/dist-packages (from moviepy->colabgymrender) (4.4.2)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages
(from imageio<3.0,>=2.1.2->moviepy->colabgymrender) (7.1.2)
Installing collected packages: colabgymrender
Successfully installed colabgymrender-1.0.9
```

In [ ]:

```
import numpy as np
#import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Convolution2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import BatchNormalization, Dropout
from colabgymrender.recorder import Recorder
from collections import deque
import random
```

In [8]:

```
class myDQAgent():

    def __init__(self, model, env, epsilon=0.2, epsilon_lim=0.05,
                 gamma=0.5, decay=0.9, batch_size=32, memory_limit=2000, replay=
        # Class for deep q agent. Takes a model, an environment and optional hyp
        # to it's constructor
        self.model = model
        self.env = env
        self.epsilon = epsilon # 1-epsilon chance of random action choices durin
        self.epsilon_limit = epsilon_lim # a minimum bound on epsilon that decay
        self.discount = gamma # discount factor
        self.decay = decay # decay is applied to epsilon after each training rou
        self.batch_size = batch_size
        self.replay=replay # Boolean determines if training is done with or with
        self.memory_limit = memory_limit # a limit on the size of the memory buf
        self.memory = self.init_memory(self.memory_limit, self.batch_size, self.

    def init_memory(self, limit, batch_size, env):
        mem = deque(maxlen=limit)
        return mem

    def getActionValue(self, state):
        return np.max(self.model.predict(np.expand_dims(state, axis=0)))

    def getAction(self, state, rand=True):
        if rand == False: # Non randomized action choices
            pred = self.model.predict(np.expand_dims(state, axis=0))
            action = np.argmax(pred)
            return action
        else:
            if np.random.random() < self.epsilon: # action choices in accordance
                return np.random.randint(0, self.env.action_space.n)
            else:
                pred = self.model.predict(np.expand_dims(state, axis=0))

                action = np.argmax(pred)

                return action

    def play_and_remember(self, episode):
        # This function is used with memory replay to play a round of
```

```
        # space invaders and save all state, action, reward, newstate and done t
        # to be sampled randomly during training later
        done = False
        ep_reward = 0
        state = self.env.reset()
        while not done:
            action = self.getAction(state)
            new_state, reward, done, info = self.env.step(action)
            ep_reward += reward
            self.memory.append({"state":state, "action":action, "reward":reward,
            state = new_state
        print(f"Episode {episode + 1} results: Reward after terminal =  {ep_rewa


    def train(self, n_episodes=5000):
        # Training function
        if self.replay:
            # If replay is true, then we play and remember an episode
            for i in range(n_episodes):

                self.play_and_remember(i)

                if len(self.memory) > self.batch_size:
                    # if the memory is sufficiently large, we sample it and
                    # train a batch
                    samples = random.sample(self.memory, self.batch_size)
                    X = []
                    ys = []
                    for sample in samples:
                        _action, _state, _new_state, _reward, _done = sample

                        if _done:
                            y = _reward
                        else:
                            y = _reward + self.discount * self.getActionValu

                        targ_vec = self.model.predict(np.expand_dims(_state,
                        targ_vec[0][_action] = y
                        X.append(_state)
                        ys.append(targ_vec[0])

                    X = np.array(X)
                    ys = np.array(ys)
                    self.model.fit(X, ys, batch_size=self.batch_size, epochs

                if self.epsilon > self.epsilon_limit:
                    self.epsilon *= self.decay

        else:
            # this is the training loop for non memory replay
            for i in range(n_episodes):

                state = np.asarray(env.reset())
                done = False
                ep_reward = 0
                while not done:

                    action = self.getAction(state)
                    new_state, reward, done, info = self.env.step(action)
                    new_state = np.asarray(new_state)
```

```
                        ep_reward += reward
                        if done:
                            print(f"Episode {i + 1}, Reward after terminal: {ep_
                        else:
                            y = reward + self.discount * self.getActionValue(new

                        targ_vec = self.model.predict(np.expand_dims(state, axis
                        targ_vec[0][action] = y
                        self.model.fit(np.expand_dims(state, axis=0), targ_vec,
                        state = new_state
                    self.epsilon *= self.decay

class Qnet():
    # The Qnet is a convolutional neural net that has a light and not light opti
    # this just means there is a choice for a larger or smaller model option
    def __init__(self,in_dim, actions, lr=1e-3, light=False):
        if light:
            self.model = self.build_model_light(in_dim, actions, lr)
        else:
            self.model = self.build_model(in_dim, actions, lr)

    def build_model(self, in_dim, actions, lr):
        model = Sequential()
        model.add(Convolution2D(128, (3, 3), activation='relu', input_shape=( 21
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(Convolution2D(32, (3, 3), activation='relu'))

        model.add(Flatten())

        model.add(Dense(512, activation='relu'))
        model.add(Dense(actions, activation="linear"))
        optimizer = tf.keras.optimizers.Adam(
        learning_rate=lr)
        model.compile(optimizer, loss="mse")
        model.summary()
        return model

    def build_model_light(self, in_dim, actions, lr):
        model = Sequential()
        model.add(Convolution2D(32, (3, 3), activation='relu', input_shape=( 210
        model.add(Convolution2D(64, (3, 3), activation='relu'))
        model.add(Convolution2D(32, (3, 3), activation='relu'))

        model.add(Flatten())

        model.add(Dense(32, activation='relu'))
        model.add(Dense(actions, activation="linear"))
        optimizer = tf.keras.optimizers.Adam(
        learning_rate=lr)
        model.compile(optimizer, loss="mse")
        model.summary()
        return model
```

Initial training with a smaller model and no experience replay:

In [14]:
```
env = gym.make("SpaceInvaders-v0")
env = Recorder(env, './video')
state = env.reset()
BATCH_SIZE = 64
actions = env.action_space.n
in_dim = env.observation_space.shape
print(in_dim)
```

```
(210, 160, 3)
```

In [15]:
```
qnet = Qnet(in_dim, actions, light=True).model
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 208, 158, 32)      896
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 206, 156, 64)      18496
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 204, 154, 32)      18464
_________________________________________________________________
flatten_2 (Flatten)          (None, 1005312)           0
_________________________________________________________________
dense_4 (Dense)              (None, 32)                32170016
_________________________________________________________________
dense_5 (Dense)              (None, 6)                 198
=================================================================
Total params: 32,208,070
Trainable params: 32,208,070
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
agent = myDQAgent(model=qnet, env=env, epsilon=0.2,gamma=0.5, batch_size=BATCH_S
agent.train(n_episodes=30)
```

```
Episode 1, Reward after terminal: 210.0, epsilon = 0.2
Episode 2, Reward after terminal: 590.0, epsilon = 0.18000000000000002
Episode 3, Reward after terminal: 460.0, epsilon = 0.16200000000000003
Episode 4, Reward after terminal: 210.0, epsilon = 0.14580000000000004
Episode 5, Reward after terminal: 105.0, epsilon = 0.13122000000000003
Episode 6, Reward after terminal: 135.0, epsilon = 0.11809800000000004
Episode 7, Reward after terminal: 460.0, epsilon = 0.10628820000000004
Episode 8, Reward after terminal: 180.0, epsilon = 0.09565938000000004
Episode 9, Reward after terminal: 155.0, epsilon = 0.08609344200000005
Episode 10, Reward after terminal: 105.0, epsilon = 0.07748409780000004
Episode 11, Reward after terminal: 155.0, epsilon = 0.06973568802000003
Episode 12, Reward after terminal: 260.0, epsilon = 0.06276211921800003
Episode 13, Reward after terminal: 245.0, epsilon = 0.056485907296200025
Episode 14, Reward after terminal: 260.0, epsilon = 0.050837316566580026
Episode 15, Reward after terminal: 180.0, epsilon = 0.04575358490992203
Episode 16, Reward after terminal: 180.0, epsilon = 0.04117822641892983
Episode 17, Reward after terminal: 180.0, epsilon = 0.03706040377703684
Episode 18, Reward after terminal: 280.0, epsilon = 0.03335436339933316
Episode 19, Reward after terminal: 240.0, epsilon = 0.030018927059399847
Episode 20, Reward after terminal: 285.0, epsilon = 0.027017034353459864
Episode 21, Reward after terminal: 570.0, epsilon = 0.02431533091811388
Episode 22, Reward after terminal: 490.0, epsilon = 0.02188379782630249
Episode 23, Reward after terminal: 315.0, epsilon = 0.019695418043672242
Episode 24, Reward after terminal: 180.0, epsilon = 0.01772587623930502
Episode 25, Reward after terminal: 180.0, epsilon = 0.015953288615374518
Episode 26, Reward after terminal: 240.0, epsilon = 0.014357959753837067
```

```
Episode 27, Reward after terminal: 210.0, epsilon = 0.012922163778453361
Episode 28, Reward after terminal: 210.0, epsilon = 0.011629947400608026
Episode 29, Reward after terminal: 285.0, epsilon = 0.010466952660547223
Episode 30, Reward after terminal: 210.0, epsilon = 0.0094202573944925
```

In [17]:

```
env = gym.make("SpaceInvaders-v0")
# env = Recorder(env, './video')
all_r = 0
random_rs = []
for i in range(100):
    observation = env.reset()
    done = False
    total_reward = 0
    while not done:
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        total_reward += reward
    # env.play()
    # print("Total reward:",total_reward)
    random_rs.append(total_reward)
    all_r += total_reward
print("avg_r", all_r/100)
```

```
avg_r 156.0
```

In [20]:

```
env = gym.make("SpaceInvaders-v0")
# 100 episode score averages or histogram comparison
all_r = 0
trained_rs = []
for i in range(100):

    observation = env.reset()
    done = False
    total_reward = 0
    while not done:

        action = agent.getAction(observation, rand=False)
        # print(action)
        observation, reward, done, info = env.step(action)
        total_reward += reward
    # env.play()
    # print("reward:",total_reward)
    trained_rs.append(total_reward)
    all_r += total_reward
print("avg_r", all_r/100)
```

```
avg_r 285.0
```

In [21]:

```
trained_rs
```

Out[21]:

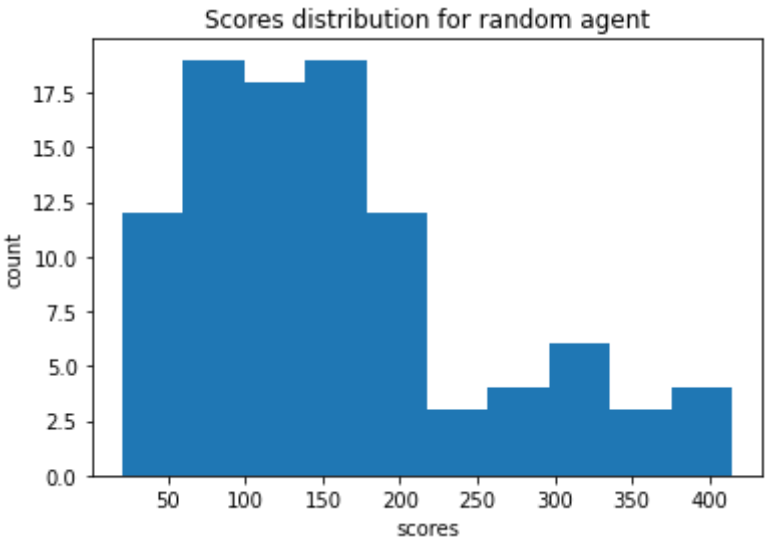
```
[285.0,
 285.0,
 285.0,
 285.0,
 285.0,
 285.0,
 285.0,
 285.0,
 285.0,
```

```
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
```

```
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0,
285.0]
```

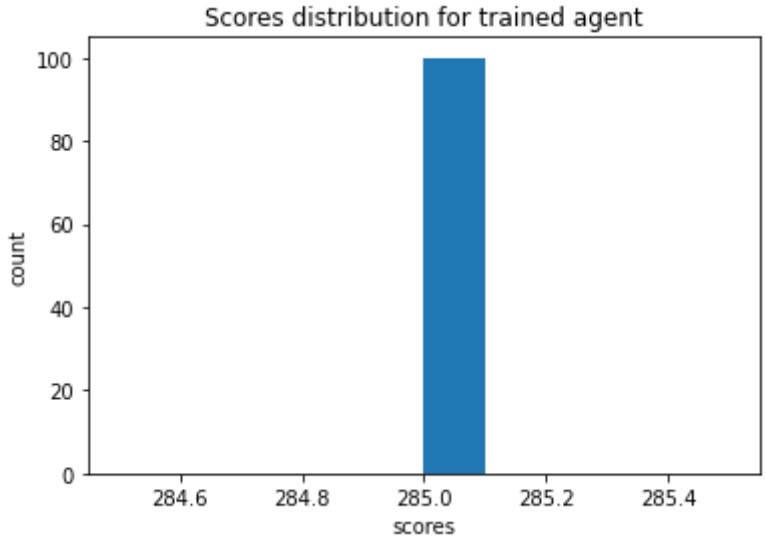
In [22]:

```
plt.title("Scores distribution for random agent")
plt.xlabel("scores")
plt.ylabel("count")
plt.hist(random_rs);
```



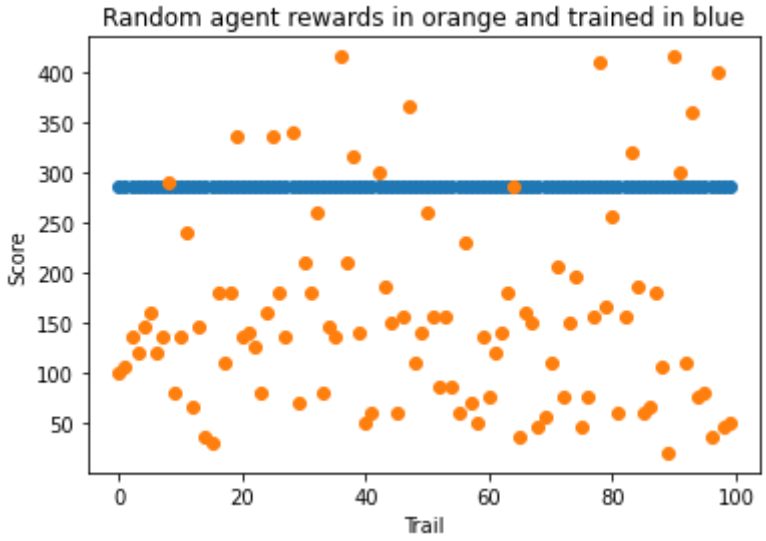
In [23]:

```
plt.title("Scores distribution for trained agent")
plt.xlabel("scores")
plt.ylabel("count")
plt.hist(trained_rs);
```



In [24]:

```
plt.scatter(range(len(trained_rs)),trained_rs);
plt.scatter(range(len(random_rs)),random_rs);
plt.ylabel("Score");
plt.xlabel("Trail")
plt.title("Random agent rewards in orange and trained in blue");
```



The above comparison shows that on 100 trials, the average reward of the trained agent is better than that of a random agent.

- The random agent had a few exceptions where it got a decent score, but most of the time it got a bad score.
- The trained agent learned a fixed strategy

In [25]:

```
env = gym.make("SpaceInvaders-v0")
env = Recorder(env, './video')

observation = env.reset()
done = False
total_reward = 0
while not done:
```

```
    action = env.action_space.sample()
    observation, reward, done, info = env.step(action)
    total_reward += reward
print("Total reward of a random agent:", total_reward)
env.play()
```

```
Total reward of a random agent: 105.0
```

See untrained.mp4 for above video

In [26]:

```
# agent.epsilon = .00
observation = env.reset()
done = False
total_reward = 0
while not done:
    action = agent.getAction(observation, rand=False)
    observation, reward, done, info = env.step(action)
    total_reward += reward
print("Total reward of a deep q agent:", total_reward)
env.play()
```

```
Total reward of a deep q agent: 285.0
```

See noReplay285.mp4 for above video

- also, see noReplay270.mp4 for another example of a simple strategy learned by the agent without experience replay

The agent learned a very basic strategy. But one that works better than just making random moves on average

- It seems like the agent has found and stuck to a local minimum in the strategy space.
- Ive included another example in noReplay270.mp4 of a simple strategy learned by the agent without experience replay
- This result also makes sense given the hyperparameters. The somewhat low epsilon favors known good strategies over exploration and the gamma parameter is not very high, which favors early rewards
- It also makes sense that the agent gets the same score every game since "The original Atari 2600 console had no feature for generating random numbers. As a consequence, the ALE is also fully deterministic." (https://towardsdatascience.com/are-the-space-invaders-deterministic-or-stochastic-595a30becae2#:~:text=The%20original%20Atari%202600%20console,learning%20to%20mal

- **Note**: the agent trained with experience replay (later in this noteboook) did not always get the same score. To my understanding, open AI's implementation does introduce a small amount of stochasticity. It therefore makes sense that some strategies might always yeild the same score, and some might interact with the stochasticity of the environment in such a way as to not have deterministic outcomes

Now lets use a larger model and experience replay:

In [5]:
```
qnet = Qnet(in_dim, actions).model
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 208, 158, 128)     3584
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 206, 156, 64)      73792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 204, 154, 32)      18464
_________________________________________________________________
flatten (Flatten)            (None, 1005312)           0
_________________________________________________________________
dense (Dense)                (None, 512)               514720256
_________________________________________________________________
dense_1 (Dense)              (None, 6)                 3078
=================================================================
Total params: 514,819,174
Trainable params: 514,819,174
Non-trainable params: 0
_________________________________________________________________
```

In [7]:
```
agent = myDQAgent(model=qnet, env=env, epsilon=.2, gamma=0.6, decay=0.99, memory
                  batch_size=BATCH_SIZE, replay=True)
agent.train(n_episodes=20) # <-- 16 episodes might be better
```

```
Episode 1 results: Reward after terminal =  225.0, epsilon: 0.2
Episode 2 results: Reward after terminal =  180.0, epsilon: 0.198
Episode 3 results: Reward after terminal =  85.0, epsilon: 0.19602
Episode 4 results: Reward after terminal =  410.0, epsilon: 0.1940598
Episode 5 results: Reward after terminal =  105.0, epsilon: 0.192119202
Episode 6 results: Reward after terminal =  90.0, epsilon: 0.19019800998
Episode 7 results: Reward after terminal =  305.0, epsilon: 0.1882960298802
```

```
Episode 8 results: Reward after terminal =  105.0, epsilon: 0.186413069581398
Episode 9 results: Reward after terminal =  420.0, epsilon: 0.18454893888558402
Episode 10 results: Reward after terminal =  345.0, epsilon: 0.18270344949672818
Episode 11 results: Reward after terminal =  215.0, epsilon: 0.18087641500176088
Episode 12 results: Reward after terminal =  485.0, epsilon: 0.17906765085174328
Episode 13 results: Reward after terminal =  135.0, epsilon: 0.17727697434322584
Episode 14 results: Reward after terminal =  155.0, epsilon: 0.17550420459979357
Episode 15 results: Reward after terminal =  245.0, epsilon: 0.17374916255379563
Episode 16 results: Reward after terminal =  395.0, epsilon: 0.17201167092825767
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-7-45b68fe965e6> in <module>()
      5 agent = myDQAgent(model=qnet, env=env, epsilon=.2, gamma=0.6, decay=0.9
9, memory_limit=100000, epsilon_lim=0.05,
      6                   batch_size=BATCH_SIZE, replay=True)
----> 7 agent.train(n_episodes=1500)
      8

<ipython-input-3-dfa5112e5260> in train(self, n_episodes)
     56                 for i in range(n_episodes):
     57
---> 58                     self.play_and_remember(i)
     59
     60                     if len(self.memory) > self.batch_size:

<ipython-input-3-dfa5112e5260> in play_and_remember(self, episode)
     43         state = self.env.reset()
     44         while not done:
---> 45             action = self.getAction(state)
     46             new_state, reward, done, info = self.env.step(action)
     47             ep_reward += reward

<ipython-input-3-dfa5112e5260> in getAction(self, state, rand)
     32                 return np.random.randint(0, self.env.action_space.n)
     33             else:
---> 34                 pred = self.model.predict(np.expand_dims(state, axis=0))
     35                 # print(pred)
     36                 action = np.argmax(pred)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.p
y in _method_wrapper(self, *args, **kwargs)
    128       raise ValueError('{} is not supported in multi-worker mode.'.forma
t(
    129           method.__name__))
--> 130     return method(self, *args, **kwargs)
    131
    132   return tf_decorator.make_decorator(

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.p
y in predict(self, x, batch_size, verbose, steps, callbacks, max_queue_size, wor
kers, use_multiprocessing)
   1577           use_multiprocessing=use_multiprocessing,
   1578           model=self,
-> 1579           steps_per_execution=self._steps_per_execution)
   1580
   1581       # Container that configures and calls `tf.keras.Callback`s.

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/data_adapt
er.py in __init__(self, x, y, sample_weight, batch_size, steps_per_epoch, initia
l_epoch, epochs, shuffle, class_weight, max_queue_size, workers, use_multiproces
sing, model, steps_per_execution)
   1115         use_multiprocessing=use_multiprocessing,
   1116         distribution_strategy=ds_context.get_strategy(),
-> 1117         model=model)
   1118
```

```
   1119     strategy = ds_context.get_strategy()

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/data_adapt
er.py in __init__(self, x, y, sample_weights, sample_weight_modes, batch_size, e
pochs, steps, shuffle, **kwargs)
    362     indices_dataset = indices_dataset.flat_map(slice_batch_indices)
    363
--> 364     dataset = self.slice_inputs(indices_dataset, inputs)
    365
    366     if shuffle == "batch":

/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/data_adapt
er.py in slice_inputs(self, indices_dataset, inputs)
    395
    396     dataset = dataset.map(
--> 397         grab_batch, num_parallel_calls=dataset_ops.AUTOTUNE)
    398
    399     # Default optimizations are disabled to avoid the overhead of (unnec
essary)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops/dataset_ops.py
in map(self, map_func, num_parallel_calls, deterministic)
   1700           num_parallel_calls,
   1701           deterministic,
-> 1702           preserve_cardinality=True)
   1703
   1704   def flat_map(self, map_func):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops/dataset_ops.py
in __init__(self, input_dataset, map_func, num_parallel_calls, deterministic, us
e_inter_op_parallelism, preserve_cardinality, use_legacy_function)
   4082         self._transformation_name(),
   4083         dataset=input_dataset,
-> 4084         use_legacy_function=use_legacy_function)
   4085     if deterministic is None:
   4086       self._deterministic = "default"

/usr/local/lib/python3.7/dist-packages/tensorflow/python/data/ops/dataset_ops.py
in __init__(self, func, transformation_name, dataset, input_classes, input_shape
s, input_types, input_structure, add_to_graph, use_legacy_function, defun_kwarg
s)
   3369       with tracking.resource_tracker_scope(resource_tracker):
   3370         # TODO(b/141462134): Switch to using garbage collection.
-> 3371         self._function = wrapper_fn.get_concrete_function()
   3372         if add_to_graph:
   3373           self._function.add_to_graph(ops.get_default_graph())

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in ge
t_concrete_function(self, *args, **kwargs)
   2937     """
   2938     graph_function = self._get_concrete_function_garbage_collected(
-> 2939         *args, **kwargs)
   2940     graph_function._garbage_collector.release()  # pylint: disable=prote
cted-access
   2941     return graph_function

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in _g
et_concrete_function_garbage_collected(self, *args, **kwargs)
   2904       args, kwargs = None, None
   2905     with self._lock:
-> 2906       graph_function, args, kwargs = self._maybe_define_function(args, k
wargs)
   2907       seen_names = set()
   2908       captured = object_identity.ObjectIdentitySet(
```

```
/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in _m
aybe_define_function(self, args, kwargs)
   3211
   3212       self._function_cache.missed.add(call_context_key)
-> 3213       graph_function = self._create_graph_function(args, kwargs)
   3214       self._function_cache.primary[cache_key] = graph_function
   3215       return graph_function, args, kwargs

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/function.py in _c
reate_graph_function(self, args, kwargs, override_flat_arg_shapes)
   3073             arg_names=arg_names,
   3074             override_flat_arg_shapes=override_flat_arg_shapes,
-> 3075             capture_by_value=self._capture_by_value),
   3076         self._function_attributes,
   3077         function_spec=self.function_spec,

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/func_graph.py
in func_graph_from_py_func(name, python_func, args, kwargs, signature, func_grap
h, autograph, autograph_options, add_control_dependencies, arg_names, op_return_
value, collections, capture_by_value, override_flat_arg_shapes)
    901       kwarg_shapes = None
    902     func_args = _get_defun_inputs_from_args(
--> 903         args, arg_names, flat_shapes=arg_shapes)
    904     func_kwargs = _get_defun_inputs_from_kwargs(
    905         kwargs, flat_shapes=kwarg_shapes)

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/func_graph.py
in _get_defun_inputs_from_args(args, names, flat_shapes)
   1137   """Maps Python function positional args to graph-construction input
s."""
   1138   return _get_defun_inputs(
-> 1139       args, names, structure=args, flat_shapes=flat_shapes)
   1140
   1141

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/func_graph.py
in _get_defun_inputs(args, names, structure, flat_shapes)
   1210           placeholder = graph_placeholder(
   1211               arg.dtype, placeholder_shape,
-> 1212               name=requested_name)
   1213         except ValueError:
   1214           # Sometimes parameter names are not valid op names, so fall ba
ck to

/usr/local/lib/python3.7/dist-packages/tensorflow/python/eager/graph_only_ops.py
in graph_placeholder(dtype, shape, name)
     38   op = g._create_op_internal(  # pylint: disable=protected-access
     39       "Placeholder", [], [dtype], input_types=[],
---> 40       attrs=attrs, name=name)
     41   result, = op.outputs
     42   if op_callbacks.should_invoke_op_callbacks():

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/func_graph.py
in _create_op_internal(self, op_type, inputs, dtypes, input_types, name, attrs,
 op_def, compute_device)
    591     return super(FuncGraph, self)._create_op_internal(  # pylint: disabl
e=protected-access
    592         op_type, inputs, dtypes, input_types, name, attrs, op_def,
--> 593         compute_device)
    594
    595   def capture(self, tensor, name=None, shape=None):

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in _cr
eate_op_internal(self, op_type, inputs, dtypes, input_types, name, attrs, op_de
f, compute_device)
```

```
   3483           input_types=input_types,
   3484           original_op=self._default_original_op,
-> 3485           op_def=op_def)
   3486       self._create_op_helper(ret, compute_device=compute_device)
   3487     return ret

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in __i
nit__(self, node_def, g, inputs, output_types, control_inputs, input_types, orig
inal_op, op_def)
   1973         op_def = self._graph._get_op_def(node_def.op)
   1974       self._c_op = _create_c_op(self._graph, node_def, inputs,
-> 1975                                 control_input_ops, op_def)
   1976       name = compat.as_str(node_def.name)
   1977     # pylint: enable=protected-access

/usr/local/lib/python3.7/dist-packages/tensorflow/python/framework/ops.py in _cr
eate_c_op(graph, node_def, inputs, control_inputs, op_def)
   1810
   1811   try:
-> 1812     c_op = pywrap_tf_session.TF_FinishOperation(op_desc)
   1813   except errors.InvalidArgumentError as e:
   1814     # Convert to ValueError for backwards compatibility.

KeyboardInterrupt:
```

I stopped the training above a little early in the interest of time

In [29]:

```
env = gym.make("SpaceInvaders-v0")
env = Recorder(env, './video')

all_r = 0
# trained_rs = []
for i in range(15):
    observation = env.reset()
    done = False
    total_reward = 0
    while not done:
        action = agent.getAction(observation, rand=False)
        observation, reward, done, info = env.step(action)
        total_reward += reward

    print("reward:",total_reward)
    # trained_rs.append(total_reward)
    all_r += total_reward
print("avg_r:", all_r/15)
```

```
reward: 270.0
reward: 440.0
reward: 170.0
reward: 175.0
reward: 250.0
reward: 445.0
reward: 265.0
reward: 205.0
reward: 565.0
reward: 435.0
reward: 450.0
reward: 175.0
reward: 345.0
reward: 175.0
reward: 400.0
avg_r: 317.6666666666667
```

In [31]:
```
env = gym.make("SpaceInvaders-v0")
env = Recorder(env, './video')
observation = env.reset()
done = False
total_reward = 0
while not done:
    action = agent.getAction(observation, rand=False)
    observation, reward, done, info = env.step(action)
    total_reward += reward
print("Total reward of a deep q agent with memory replay:", total_reward)
env.play()
```

```
Total reward of a deep q agent with memory replay: 450.0
```

See wExperienceReplay.mp4 for above video

Above are results from an experience replay agent

- Notice the slight improvement on the agent without experience replay.
- The agent with memory replay now gets an average score of 315 (on 15 trials)
- More training and hyperparameter tuning would likely result in even better scores

Conclusions and Suggestions for Further Work

- Ideally I would have been able to get even better performance out of the experience replay agent. However, I am pleased that my agent still performed better on average than a random agent and that my replay agent seems to have been a little better than my agent without experience replay.
- Also improvement of the replay agent is still promising. Especially when considering that training was not very long. It just takes a long time to train the agent, and there are many different hyperparameter configurations to consider.
- I think that with more time to tune the hyperparameters I could have gotten better performance out of the experience replay agent and this is a suggestion for further work.
- Additionally, experimenting with different neural network architectures could improve results.

References:

- https://towardsdatascience.com/deep-q-network-dqn-ii-b6bf911b6b2c
- https://towardsdatascience.com/are-the-space-invaders-deterministic-or-stochastic-595a30becae2#:~:text=The%20original%20Atari%202600%20console,learning%20to%20mal
- https://towardsdatascience.com/deep-q-learning-tutorial-mindqn-2a4c855abffc
- https://towardsdatascience.com/introduction-to-various-reinforcement-learning-algorithms-i-q-learning-sarsa-dqn-ddpg-72a5e0cb6287
- https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

In [ ]: