```
%load_ext autoreload
%autoreload 2

#connecting to drive

from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/MyDrive/RL-part12/'
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/MyDrive/RL-part12
```

```
%load_ext autoreload
%autoreload 2



!pip install --upgrade pip setuptools

!apt-get install swig
!apt-get install swig
!pip install pygame==2.1.0
!pip install swig==4.*
!pip install -U gym[box2d]

!pip install --upgrade pip setuptools
!pip install -U gym[box2d]
!pip install numpy torch wandb swig gymnasium[box2d] matplotlib
termcolor
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-
packages (23.1.2)
Collecting pip
  Downloading pip-24.0-py3-none-any.whl (2.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 29.8 MB/s eta
0:00:00
ent already satisfied: setuptools in /usr/local/lib/python3.10/dist-
packages (67.7.2)
Collecting setuptools
  Using cached setuptools-70.0.0-py3-none-any.whl (863 kB)
Installing collected packages: setuptools, pip
  Attempting uninstall: setuptools
    Found existing installation: setuptools 67.7.2
    Uninstalling setuptools-67.7.2:
      Successfully uninstalled setuptools-67.7.2
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
```

```
      Successfully uninstalled pip-23.1.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
ipython 7.34.0 requires jedi>=0.16, which is not installed.
Successfully installed pip-24.0 setuptools-70.0.0
```

{"id":"d4dd302b04a7403a8b11e66482b3ab93","pip_warning":{"packages":
["_distutils_hack","pkg_resources","setuptools"]}}

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  swig4.0
Suggested packages:
  swig-doc swig-examples swig4.0-examples swig4.0-doc
The following NEW packages will be installed:
  swig swig4.0
0 upgraded, 2 newly installed, 0 to remove and 45 not upgraded.
Need to get 1,116 kB of archives.
After this operation, 5,542 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig4.0 amd64 4.0.2-1ubuntu1 [1,110 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 swig all 4.0.2-1ubuntu1 [5,632 B]
Fetched 1,116 kB in 2s (466 kB/s)
Selecting previously unselected package swig4.0.
(Reading database ... 121913 files and directories currently installed.)
Preparing to unpack .../swig4.0_4.0.2-1ubuntu1_amd64.deb ...
Unpacking swig4.0 (4.0.2-1ubuntu1) ...
Selecting previously unselected package swig.
Preparing to unpack .../swig_4.0.2-1ubuntu1_all.deb ...
Unpacking swig (4.0.2-1ubuntu1) ...
Setting up swig4.0 (4.0.2-1ubuntu1) ...
Setting up swig (4.0.2-1ubuntu1) ...
Processing triggers for man-db (2.10.2-1) ...
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
swig is already the newest version (4.0.2-1ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Collecting pygame==2.1.0
  Downloading pygame-2.1.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (9.5 kB)
Downloading pygame-2.1.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.3/18.3 MB 81.3 MB/s eta 0:00:00
```

```
e
  Attempting uninstall: pygame
    Found existing installation: pygame 2.5.2
    Uninstalling pygame-2.5.2:
      Successfully uninstalled pygame-2.5.2
Successfully installed pygame-2.1.0
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
Collecting swig==4.*
  Downloading swig-4.2.1-py2.py3-none-
manylinux_2_5_x86_64.manylinux1_x86_64.whl.metadata (3.6 kB)
Downloading swig-4.2.1-py2.py3-none-
manylinux_2_5_x86_64.manylinux1_x86_64.whl (1.9 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 29.3 MB/s eta
0:00:00
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
Requirement already satisfied: gym[box2d] in
/usr/local/lib/python3.10/dist-packages (0.25.2)
Collecting gym[box2d]
  Downloading gym-0.26.2.tar.gz (721 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 721.7/721.7 kB 12.8 MB/s eta
0:00:00
ents to build wheel ... etadata (pyproject.toml) ... ent already
satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages
(from gym[box2d]) (1.25.2)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (0.0.8)
Collecting box2d-py==2.3.5 (from gym[box2d])
  Downloading box2d-py-2.3.5.tar.gz (374 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 374.4/374.4 kB 28.2 MB/s eta
0:00:00
etadata (setup.py) ... ent already satisfied: pygame==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (2.1.0)
Requirement already satisfied: swig==4.* in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (4.2.1)
Building wheels for collected packages: box2d-py, gym
  Building wheel for box2d-py (setup.py) ... e=box2d_py-2.3.5-cp310-
cp310-linux_x86_64.whl size=2376100
sha256=2e0c0161c04fb959b3e72c4f7d53bbd8c8668445404e0dc1f955b8eb2873722
3
  Stored in directory:
/root/.cache/pip/wheels/db/8f/6a/eaaadf056fba10a98d986f6dce954e6201ba3
```

```
126926fc5ad9e
  Building wheel for gym (pyproject.toml) ... : filename=gym-0.26.2-
py3-none-any.whl size=827626
sha256=69a2eb9bcaad225ffaca6a2de752bd4017a02c635949d7e8c68bbef668e89ca
3
  Stored in directory:
/root/.cache/pip/wheels/b9/22/6d/3e7b32d98451b4cd9d12417052affbeeeea01
2955d437da1da
Successfully built box2d-py gym
Installing collected packages: box2d-py, gym
  Attempting uninstall: gym
    Found existing installation: gym 0.25.2
    Uninstalling gym-0.25.2:
      Successfully uninstalled gym-0.25.2
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
dopamine-rl 4.0.9 requires gym<=0.25.2, but you have gym 0.26.2 which
is incompatible.
Successfully installed box2d-py-2.3.5 gym-0.26.2
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-
packages (24.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (70.0.0)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
Requirement already satisfied: gym[box2d] in
/usr/local/lib/python3.10/dist-packages (0.26.2)
Requirement already satisfied: numpy>=1.18.0 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (1.25.2)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (0.0.8)
Requirement already satisfied: box2d-py==2.3.5 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (2.3.5)
Requirement already satisfied: pygame==2.1.0 in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (2.1.0)
Requirement already satisfied: swig==4.* in
/usr/local/lib/python3.10/dist-packages (from gym[box2d]) (4.2.1)
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
```

```
https://pip.pypa.io/warnings/venv
Requirement already satisfied: numpy in
/usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: torch in
/usr/local/lib/python3.10/dist-packages (2.3.0+cu121)
Collecting wandb
  Downloading wandb-0.17.1-py3-none-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata (10 kB)
Requirement already satisfied: swig in /usr/local/lib/python3.10/dist-
packages (4.2.1)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: termcolor in
/usr/local/lib/python3.10/dist-packages (2.4.0)
Collecting gymnasium[box2d]
  Downloading gymnasium-0.29.1-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: filelock in
/usr/local/lib/python3.10/dist-packages (from torch) (3.14.0)
Requirement already satisfied: typing-extensions>=4.8.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (4.12.1)
Requirement already satisfied: sympy in
/usr/local/lib/python3.10/dist-packages (from torch) (1.12.1)
Requirement already satisfied: networkx in
/usr/local/lib/python3.10/dist-packages (from torch) (3.3)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.10/dist-packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in
/usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-
manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-
manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-
manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-
manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-
manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-
manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.2.106 (from torch)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-
```

```
manylinux1_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-
manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-
manylinux1_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-nccl-cu12==2.20.5 (from torch)
  Downloading nvidia_nccl_cu12-2.20.5-py3-none-
manylinux2014_x86_64.whl.metadata (1.8 kB)
Collecting nvidia-nvtx-cu12==12.1.105 (from torch)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-
manylinux1_x86_64.whl.metadata (1.7 kB)
Requirement already satisfied: triton==2.3.0 in
/usr/local/lib/python3.10/dist-packages (from torch) (2.3.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-
cu12==11.4.5.107->torch)
  Downloading nvidia_nvjitlink_cu12-12.5.40-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Requirement already satisfied: click!=8.0.0,>=7.1 in
/usr/local/lib/python3.10/dist-packages (from wandb) (8.1.7)
Collecting docker-pycreds>=0.4.0 (from wandb)
  Downloading docker_pycreds-0.4.0-py2.py3-none-any.whl.metadata (1.8
kB)
Collecting gitpython!=3.1.29,>=1.0.0 (from wandb)
  Downloading GitPython-3.1.43-py3-none-any.whl.metadata (13 kB)
Requirement already satisfied: platformdirs in
/usr/local/lib/python3.10/dist-packages (from wandb) (4.2.2)
Requirement already satisfied: protobuf!=4.21.0,<6,>=3.19.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (3.20.3)
Requirement already satisfied: psutil>=5.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (5.9.5)
Requirement already satisfied: pyyaml in
/usr/local/lib/python3.10/dist-packages (from wandb) (6.0.1)
Requirement already satisfied: requests<3,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from wandb) (2.31.0)
Collecting sentry-sdk>=1.0.0 (from wandb)
  Downloading sentry_sdk-2.5.1-py2.py3-none-any.whl.metadata (10 kB)
Collecting setproctitle (from wandb)
  Downloading setproctitle-1.3.3-cp310-cp310-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux
2014_x86_64.whl.metadata (9.9 kB)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from wandb) (70.0.0)
Requirement already satisfied: cloudpickle>=1.2.0 in
/usr/local/lib/python3.10/dist-packages (from gymnasium[box2d])
(2.2.1)
Collecting farama-notifications>=0.0.1 (from gymnasium[box2d])
  Downloading Farama_Notifications-0.0.4-py3-none-any.whl.metadata
```

```
(558 bytes)
Requirement already satisfied: box2d-py==2.3.5 in
/usr/local/lib/python3.10/dist-packages (from gymnasium[box2d])
(2.3.5)
Collecting pygame>=2.1.3 (from gymnasium[box2d])
  Downloading pygame-2.5.2-cp310-cp310-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (13 kB)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.2.1)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (24.0)
Requirement already satisfied: pillow>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.4.0 in
/usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0-
>wandb) (1.16.0)
Collecting gitdb<5,>=4.0.1 (from gitpython!=3.1.29,>=1.0.0->wandb)
  Downloading gitdb-4.0.11-py3-none-any.whl.metadata (1.2 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.0.0-
>wandb) (2024.6.2)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath<1.4.0,>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython!
=3.1.29,>=1.0.0->wandb)
  Downloading smmap-5.0.1-py3-none-any.whl.metadata (4.3 kB)
Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl
(410.6 MB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 3.0 MB/s eta
0:00:00
anylinux1_x86_64.whl (14.1 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 110.9 MB/s eta
0:00:00
anylinux1_x86_64.whl (23.7 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 87.3 MB/s eta
0:00:00
e_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 42.8 MB/s eta
0:00:00
anylinux1_x86_64.whl (731.7 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 1.5 MB/s eta
0:00:00
anylinux1_x86_64.whl (121.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 17.7 MB/s eta
0:00:00
anylinux1_x86_64.whl (56.5 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 38.9 MB/s eta
0:00:00
anylinux1_x86_64.whl (124.2 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 18.2 MB/s eta
0:00:00
anylinux1_x86_64.whl (196.0 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 5.7 MB/s eta
0:00:00
anylinux2014_x86_64.whl (176.2 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 176.2/176.2 MB 12.9 MB/s eta
0:00:00
anylinux1_x86_64.whl (99 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 8.4 MB/s eta
0:00:00
anylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (6.8 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6.8/6.8 MB 114.1 MB/s eta
0:00:00
a_Notifications-0.0.4-py3-none-any.whl (2.5 kB)
Downloading GitPython-3.1.43-py3-none-any.whl (207 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.3/207.3 kB 17.2 MB/s eta
0:00:00
e-2.5.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(14.0 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.0/14.0 MB 104.8 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 289.6/289.6 kB 23.1 MB/s eta
0:00:00
nasium-0.29.1-py3-none-any.whl (953 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 953.9/953.9 kB 49.6 MB/s eta
0:00:00
```

```
anylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2
014_x86_64.whl (30 kB)
Downloading gitdb-4.0.11-py3-none-any.whl (62 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 5.6 MB/s eta
0:00:00
anylinux2014_x86_64.whl (21.3 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.3/21.3 MB 94.9 MB/s eta
0:00:00
map-5.0.1-py3-none-any.whl (24 kB)
Installing collected packages: farama-notifications, smmap,
setproctitle, sentry-sdk, pygame, nvidia-nvtx-cu12, nvidia-nvjitlink-
cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-
cuda-runtime-cu12, nvidia-cuda-nvrtc-cu12, nvidia-cuda-cupti-cu12,
nvidia-cublas-cu12, gymnasium, docker-pycreds, nvidia-cusparse-cu12,
nvidia-cudnn-cu12, gitdb, nvidia-cusolver-cu12, gitpython, wandb
  Attempting uninstall: pygame
    Found existing installation: pygame 2.1.0
    Uninstalling pygame-2.1.0:
      Successfully uninstalled pygame-2.1.0
ERROR: pip's dependency resolver does not currently take into account
all the packages that are installed. This behaviour is the source of
the following dependency conflicts.
dopamine-rl 4.0.9 requires gym<=0.25.2, but you have gym 0.26.2 which
is incompatible.
Successfully installed docker-pycreds-0.4.0 farama-notifications-0.0.4
gitdb-4.0.11 gitpython-3.1.43 gymnasium-0.29.1 nvidia-cublas-cu12-
12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-
12.1.105 nvidia-cuda-runtime-cu12-12.1.105 nvidia-cudnn-cu12-8.9.2.26
nvidia-cufft-cu12-11.0.2.54 nvidia-curand-cu12-10.3.2.106 nvidia-
cusolver-cu12-11.4.5.107 nvidia-cusparse-cu12-12.1.0.106 nvidia-nccl-
cu12-2.20.5 nvidia-nvjitlink-cu12-12.5.40 nvidia-nvtx-cu12-12.1.105
pygame-2.5.2 sentry-sdk-2.5.1 setproctitle-1.3.3 smmap-5.0.1 wandb-
0.17.1
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package manager.
It is recommended to use a virtual environment instead:
https://pip.pypa.io/warnings/venv
```

```python
import test1
from utils import *
```

Reinforcement Learning Part 1: DQN

By Lawrence Liu and Tonmoy Monsoor

Some General Instructions

- As before, please keep the names of the layer consistent with what is requested in model.py. Otherwise the test functions will not work

- You will need to fill in the model.py, the DQN.py file, the buffer.py file, and the env_wrapper.py

DO NOT use Windows for this project, gymnasium does is not supported for windows and installing it will be a pain.

Introduction to the Enviroment

We will be training a DQN agent to play the game of CarRacing. The agent will be trained to play the game using the pixels of the game as an input. The reward structure is as follows for each frame:

- -0.1 for each frame
- +1000/N where N is the number of tiles visited by the car in the episode

The overall goal of this game is to design a agent that is able to play the game with a average test score of above 600. In discrete mode the actions can take 5 actions,

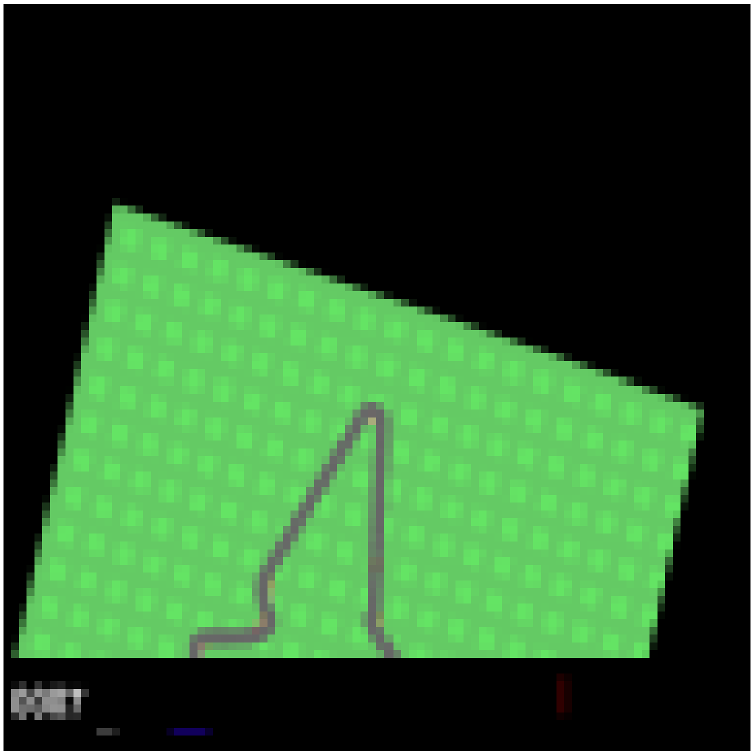
- 0: Do Nothing
- 1: Turn Left
- 2: Turn Right
- 3: Accelerate
- 4: Brake

First let us visualize the game and understand the environment.

```python
import gymnasium as gym
import numpy as np
env = gym.make('CarRacing-v2', continuous=False,
render_mode='rgb_array')
env.np_random = np.random.RandomState(42)


from IPython.display import HTML

frames = []
s, _ = env.reset()

while True:
    a = env.action_space.sample()
    s, r, terminated, truncated, _ = env.step(a)
    frames.append(s)
    if terminated or truncated:
        break
```

```
anim = animate(frames)
HTML(anim.to_jshtml())
```

```
<IPython.core.display.HTML object>
```



So a couple things we can note:

- at the beginning of the game, we have 50 frames of the game slowly zooming into the car, we should ignore this period, ie no-op during this period.
- there is a black bar at the bottom of the screen, we should crop this out of the observation.

In addition, another thing to note is that the current frame doesn't give much information about the velocity and acceleration of the car, and that the car does not move much for each frame.

Environment Wrapper (5 points)

As a result, you will need to complete `EnvWrapper` in `env_wrapper.py`. You can find more information in the docstring for the wrapper, however the main idea is that it is a wrapper to the environment that does the following:

- skips the first 50 frames of the game
- crops out the black bar and reshapes the observation to a 84x84 image, as well as turning the resulting image to grayscale
- performs the actions for `skip_frames` frames

- stacks the last `num_frames` frames together to give the agent some information about the velocity and acceleration of the car.

```
from env_wrapper import EnvWrapper

test1.test_wrapper(EnvWrapper)

Passed reset
Passed step
```

CNN Model (5 points)

Now we are ready to build the model. Our architecture of the CNN model is the one proposed by Mnih et al in "Human-level control through deep reinforcement learning". Specifically this consists of the following layers:

- A convolutional layer with 32 filters of size 8x8 with stride 4 and relu activation
- A convolutional layer with 64 filters of size 4x4 with stride 2 and relu activation
- A convolutional layer with 64 filters of size 3x3 with stride 1 and relu activation
- A fully connected layer with 512 units and relu activation
- A fully connected layer with the number of outputs of the environment

Please implement this model `Nature_Paper_Conv` in `model.py` as well as the helper `MLP` class.

```
import model
test1.test_model_DQN(model.Nature_Paper_Conv)

Passed
```

DQN (40 points)

Now we are ready to implement the DQN algorithm.

title

Replay Buffer (5 points)

First start by implementing the DQN replay buffer `ReplayBufferDQN` in `buffer.py`. This buffer will store the transitions of the agent and sample them for training.

```
from replay_buffer import ReplayBufferDQN

test1.test_DQN_replay_buffer(ReplayBufferDQN)

Passed
```

DQN (15 points)

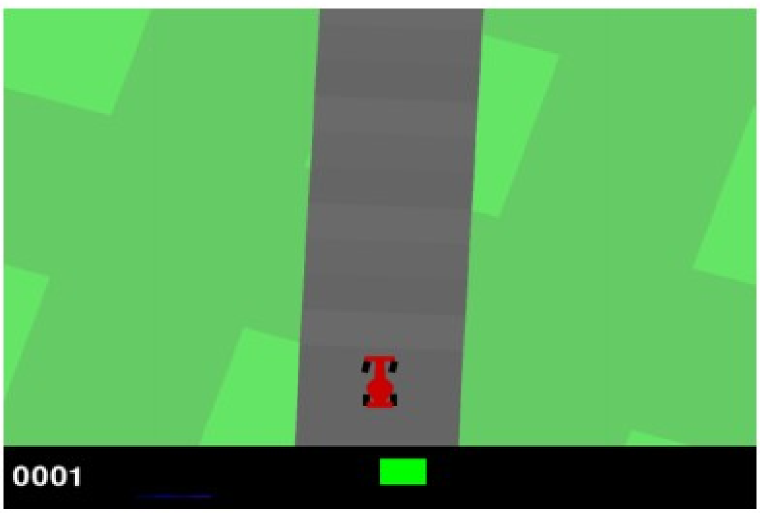
Now implement the `_optimize_model` and `sample_action` functions in `DQN` in `DQN.py`. The `_optimize_model` function will sample a batch of transitions from the replay buffer and update the model. The `sample_action` function will sample an action from the model given the current state. Train the model over 200 episdoes, validating every 50 episodes for 30 episodes, before testing the model for 50 episodes at the end.

```
import DQN
import utils
import torch


trainerDQN = DQN.DQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cpu',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
700,0.1),
                save_path = utils.get_save_path("DQN","./runs/"))

trainerDQN.train(200,50,30,50,50)


----------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
<ipython-input-4-bead0dfb3872> in <cell line: 1>()
----> 1 import DQN
      2 import utils
      3 import torch
      4
      5

ModuleNotFoundError: No module named 'DQN'

----------------------------------------------------------------------
-----
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
----------------------------------------------------------------------
-----
```

Please include a plot of the training and validation rewards over the episodes in the report. An additional question to answer is does the loss matter in DQN? Why or why not?

We can also draw a animation of the car in one game, the code is provided below

```
eval_env = gym.make('CarRacing-v2', continuous=True,
render_mode='rgb_array')
eval_env = EnvWrapper(eval_env)

total_rewards, frames = trainerDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())

<IPython.core.display.HTML object>
```



Double DQN

In the original paper, where the algorithim is shown above, the estimated target Q value was computed using the current Q network's weights. However, this can lead to overestimation of the Q values. To mitigate this, we can use the target network to compute the target Q value. This is known as Double DQN.

Hard updating Target Network (5 points)

Original implementations for this involved hard updates, where the model weights were copied to the target network every C steps. This is known as hard updating. This was what was used in the Nature Paper by Mnih et al 2015 "Human-level control through deep reinforcement learning"

Please implement this by implementing the `_optimize_model` and `_update_model` classes in `HardUpdateDQN` in `DQN.py`.

```python
import DQN
import utils
import torch

trainerHardUpdateDQN = DQN.HardUpdateDQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                update_freq = 100,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cuda',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
1000,0.1),
                save_path =
utils.get_save_path("DoubleDQN_HardUpdates/","./runs/"))

trainerHardUpdateDQN.train(100,50,30,50,50)
```

```
saving to ./runs/DoubleDQN_HardUpdates/run3
```

```
/content/drive/MyDrive/RL-part12/DQN.py:316: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  states = torch.tensor(states, device=self.device,
dtype=torch.float32)
/content/drive/MyDrive/RL-part12/DQN.py:317: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  actions = torch.tensor(actions, device=self.device,
dtype=torch.int64).unsqueeze(1)
/content/drive/MyDrive/RL-part12/DQN.py:318: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  rewards = torch.tensor(rewards, device=self.device,
dtype=torch.float32).unsqueeze(1)
/content/drive/MyDrive/RL-part12/DQN.py:319: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
```

```
  next_states = torch.tensor(next_states, device=self.device,
dtype=torch.float32)
/content/drive/MyDrive/RL-part12/DQN.py:320: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  dones = torch.tensor(dones, device=self.device,
dtype=torch.float32).unsqueeze(1)
/usr/local/lib/python3.10/dist-packages/torch/autograd/graph.py:744:
UserWarning: Plan failed with a cudnnException:
CUDNN_BACKEND_EXECUTION_PLAN_DESCRIPTOR: cudnnFinalize Descriptor
Failed cudnn_status: CUDNN_STATUS_NOT_SUPPORTED (Triggered internally
at ../aten/src/ATen/native/cudnn/Conv_v8.cpp:919.)
  return Variable._execution_engine.run_backward(  # Calls into the C+
+ engine to run the backward pass

Episode: 1: Time: 12.53152084350586 Total Reward: -62.15328467153341
Avg_Loss: 0.6992917292674256
Episode: 2: Time: 11.742269277572632 Total Reward: -41.428571428572155
Avg_Loss: 0.595020932317594
Episode: 3: Time: 12.232085704803467 Total Reward: -64.69696969697057
Avg_Loss: 0.6009603425500398
Episode: 4: Time: 12.192229509353638 Total Reward: -59.78873239436707
Avg_Loss: 0.5602807334444228
Episode: 5: Time: 11.55905556678772 Total Reward: -60.64885496183287
Avg_Loss: 0.5863901792141069
Episode: 6: Time: 11.876625061035156 Total Reward: -52.03125000000054
Avg_Loss: 0.5489930003302312
Episode: 7: Time: 12.009033203125 Total Reward: -66.0144927536238
Avg_Loss: 0.5121881691347651
Episode: 8: Time: 11.616037845611572 Total Reward: -76.0606060606062
Avg_Loss: 0.5259098462885669
Episode: 9: Time: 11.929719924926758 Total Reward: -68.8562091503272
Avg_Loss: 0.5038694544937932
Episode: 10: Time: 12.21669316291809 Total Reward: -51.678700361011174
Avg_Loss: 0.5787985477627826
Episode: 11: Time: 12.50355076789856 Total Reward: -76.87311178247725
Avg_Loss: 0.710008535909803
Episode: 12: Time: 11.759137153625488 Total Reward: -47.21843003413028
Avg_Loss: 0.6875895914316428
Episode: 13: Time: 11.849242448806763 Total Reward: -74.66101694915265
Avg_Loss: 0.823110145367995
Episode: 14: Time: 11.009843111038208 Total Reward: -
138.13762057877875 Avg_Loss: 0.8495799407938519
Episode: 15: Time: 11.602227449417114 Total Reward: -14.29824561403532
Avg_Loss: 3.6153195305970036
Episode: 16: Time: 11.72810626029968З Total Reward: -
23.813559322034553 Avg_Loss: 2.210540623286692
Episode: 17: Time: 11.939311265945435 Total Reward: -32.69470404984455
```

```
Avg_Loss: 2.2715076897449853
Episode: 18: Time: 11.861148357391357 Total Reward: 5.323624595468333
Avg_Loss: 2.7451429026467458
Episode: 19: Time: 11.38285517692566 Total Reward: 31.760563380283358
Avg_Loss: 1.1751332340621148
Episode: 20: Time: 11.798308372497559 Total Reward: 53.6486486486513
Avg_Loss: 1.449909035272959
Episode: 21: Time: 11.85446310043335 Total Reward: 199.32624113475597
Avg_Loss: 1.8837501559437824
Episode: 22: Time: 12.012518644332886 Total Reward: 51.953405017921845
Avg_Loss: 2.0032235575573787
Episode: 23: Time: 12.165983438491821 Total Reward: -8.461538461538364
Avg_Loss: 2.3404047383981594
Episode: 24: Time: 11.898322582244873 Total Reward: -65.6959706959714
Avg_Loss: 2.4070041192178966
Episode: 25: Time: 11.281588554382324 Total Reward: -11.08391608391678
Avg_Loss: 2.0801786633850146
Episode: 26: Time: 11.50476360321045 Total Reward: 93.81118881119218
Avg_Loss: 2.620824527464995
Episode: 27: Time: 12.08803415298462 Total Reward: 99.35736677116185
Avg_Loss: 2.4216282111005625
Episode: 28: Time: 11.912178993225098 Total Reward: 81.05633802817074
Avg_Loss: 2.4464187325925386
Episode: 29: Time: 11.798665046691895 Total Reward: 87.79569892473341
Avg_Loss: 2.429009882467134
Episode: 30: Time: 11.868998289108276 Total Reward: -20.00000000000049
Avg_Loss: 2.6417624594784583
Episode: 31: Time: 11.710969686508179 Total Reward: 45.68441064638732
Avg_Loss: 2.6535132101603915
Episode: 32: Time: 12.062574625015259 Total Reward: 107.61437908496879
Avg_Loss: 2.833824892254437
Episode: 33: Time: 12.143572568893433 Total Reward: 107.5316455696245
Avg_Loss: 2.4112580575111533
Episode: 34: Time: 12.329484939575195 Total Reward: 228.89937106918697
Avg_Loss: 2.5308823815914763
Episode: 35: Time: 12.26021432876587 Total Reward: 97.69102990033643
Avg_Loss: 2.4885283560812974
Episode: 36: Time: 12.225733518600464 Total Reward: 66.99376947040788
Avg_Loss: 2.821686149144373
Episode: 37: Time: 11.653918981552124 Total Reward: 121.41791044776478
Avg_Loss: 3.210357710468669
Episode: 38: Time: 12.314339876174927 Total Reward: 262.5949367088582
Avg_Loss: 3.0631140405390442
Episode: 39: Time: 11.85021185874939 Total Reward: 141.36363636364
Avg_Loss: 3.1117773046012687
Episode: 40: Time: 11.646278858184814 Total Reward: -33.93129770992427
Avg_Loss: 3.4069880597731648
Episode: 41: Time: 11.963972806930542 Total Reward: -
2.8947368421058144 Avg_Loss: 3.444867387038319
```

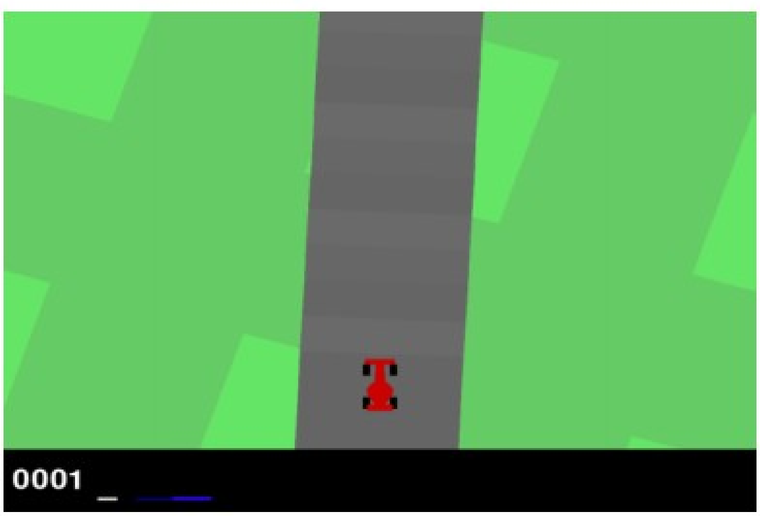
```
Episode: 42: Time: 11.957952499389648 Total Reward: 294.0909090909072
Avg_Loss: 3.3691224691246737
Episode: 43: Time: 12.073222637176514 Total Reward: 274.49152542373145
Avg_Loss: 3.0487865450001563
Episode: 44: Time: 12.193732261657715 Total Reward: 185.82191780822308
Avg_Loss: 2.7532901062684902
Episode: 45: Time: 11.804687976837158 Total Reward: 312.1428571428569
Avg_Loss: 2.8518605157106864
Episode: 46: Time: 12.270053148269653 Total Reward: 119.05750798722167
Avg_Loss: 2.8879797844325794
Episode: 47: Time: 12.015372037887573 Total Reward: 41.05442176870956
Avg_Loss: 3.224301057452915
Episode: 48: Time: 11.995104312896729 Total Reward: 347.17687074829416
Avg_Loss: 2.963618974725739
Episode: 49: Time: 12.014806747436523 Total Reward: 254.0909090909129
Avg_Loss: 3.711046317294866
Validation Mean Reward: 327.114810465886 Validation Std Reward:
141.84452449355294
Episode: 50: Time: 12.477738380432129 Total Reward: -61.36085626911386
Avg_Loss: 3.6189422870383545
Episode: 51: Time: 12.350550413131714 Total Reward: 309.7619047618995
Avg_Loss: 3.45121672999959
Episode: 52: Time: 12.451483726501465 Total Reward: 345.86021505376175
Avg_Loss: 3.713229477405548
Episode: 53: Time: 11.858627796173096 Total Reward: 364.55882352940876
Avg_Loss: 3.830740837740297
Episode: 54: Time: 11.97512173652649 Total Reward: 343.84892086330785
Avg_Loss: 3.6837762784557184
Episode: 55: Time: 11.941289901733398 Total Reward: 184.35222672065217
Avg_Loss: 3.243297742194488
Episode: 56: Time: 12.110434293746948 Total Reward: 339.6289752650149
Avg_Loss: 3.5335010415365717
Episode: 57: Time: 12.527174472808838 Total Reward: 204.63898916967648
Avg_Loss: 3.8626845756999586
Episode: 58: Time: 12.076569557189941 Total Reward: 197.9687500000013
Avg_Loss: 3.8040203424561927
Episode: 59: Time: 11.910866737365723 Total Reward: 487.41758241757657
Avg_Loss: 3.6591576057321884
Episode: 60: Time: 11.95696473121643 Total Reward: 163.06451612903595
Avg_Loss: 3.4247672442628554
Episode: 61: Time: 12.159114122390747 Total Reward: 155.00000000000338
Avg_Loss: 3.530189103939954
Episode: 62: Time: 11.965272426605225 Total Reward: 120.54770318021599
Avg_Loss: 3.9334776646950664
Episode: 63: Time: 11.987609148025513 Total Reward: 324.46308724832176
Avg_Loss: 4.31837462827939
Episode: 64: Time: 12.273281335830688 Total Reward: 287.9113924050582
Avg_Loss: 3.9611133337020874
Episode: 65: Time: 12.46273207664497 Total Reward: 235.484330484335
```

```
Avg_Loss: 3.8265929407432298
Episode: 66: Time: 12.000812530517578 Total Reward: 366.016949152541
Avg_Loss: 4.297037656567678
Episode: 67: Time: 11.836947679519653 Total Reward: 267.98932384340975
Avg_Loss: 3.8182560541048773
Episode: 68: Time: 11.592691659927368 Total Reward: 488.6734693877443
Avg_Loss: 4.418390542018313
Episode: 69: Time: 11.695602655410767 Total Reward: 357.4714828897321
Avg_Loss: 4.2950701943966525
Episode: 70: Time: 11.801357984542847 Total Reward: 459.34782608695
Avg_Loss: 4.177855560759537
Episode: 71: Time: 11.977038145065308 Total Reward: 27.25705329153744
Avg_Loss: 4.575971016362936
Episode: 72: Time: 12.227049589157104 Total Reward: 284.4212218649522
Avg_Loss: 4.866785826302376
Episode: 73: Time: 12.0078866481781 Total Reward: 363.33333333333064
Avg_Loss: 5.336714987995244
Episode: 74: Time: 12.362622022628784 Total Reward: 379.04844290656945
Avg_Loss: 5.346491337323389
Episode: 75: Time: 12.475548505783081 Total Reward: 295.62500000000006
Avg_Loss: 5.528011252900131
Episode: 76: Time: 12.214735269546509 Total Reward: 242.70491803279162
Avg_Loss: 4.673720028720984
Episode: 77: Time: 11.939579963684082 Total Reward: 191.20689655172743
Avg_Loss: 4.770161946781543
Episode: 78: Time: 12.220985412597656 Total Reward: 348.3333333333288
Avg_Loss: 5.558071137476368
Episode: 79: Time: 12.07649302482605 Total Reward: 117.30769230769621
Avg_Loss: 4.906452148902316
Episode: 80: Time: 12.24681305885315 Total Reward: 225.51282051282462
Avg_Loss: 4.592162496903363
Episode: 81: Time: 11.924290418624878 Total Reward: 338.0985915492963
Avg_Loss: 4.017545136583953
Episode: 82: Time: 12.032289028167725 Total Reward: 191.6666666666686
Avg_Loss: 4.308270046190054
Episode: 83: Time: 12.443589210510254 Total Reward: 180.96439169139796
Avg_Loss: 4.696126081863372
Episode: 84: Time: 11.857173204421997 Total Reward: 404.99999999999307
Avg_Loss: 4.829787048972955
Episode: 85: Time: 12.155384540557861 Total Reward: 328.4527687296425
Avg_Loss: 4.432432207239776
Episode: 86: Time: 11.764258623123169 Total Reward: 199.7368421052667
Avg_Loss: 4.570977178942256
Episode: 87: Time: 12.109614133834839 Total Reward: 232.92207792208285
Avg_Loss: 4.289638028425329
Episode: 88: Time: 12.24470591545105 Total Reward: 295.9774436090248
Avg_Loss: 4.390419376998389
Episode: 89: Time: 12.211179733276367 Total Reward: 237.2884012539231
Avg_Loss: 4.821797059363678
```

```
Episode: 90: Time: 11.793045997619629 Total Reward: 401.1240310077431
Avg_Loss: 4.559586583065386
Episode: 91: Time: 11.682868719100952 Total Reward: 337.23443223442723
Avg_Loss: 4.464701604943316
Episode: 92: Time: 11.935480833053589 Total Reward: 435.8219178082146
Avg_Loss: 4.192759582475454
Episode: 93: Time: 12.126565217971802 Total Reward: 460.92105263157043
Avg_Loss: 4.399263517696316
Episode: 94: Time: 11.718321084976196 Total Reward: 334.0780141843894
Avg_Loss: 4.312920090030222
Episode: 95: Time: 12.529597282409668 Total Reward: 114.8765432098802
Avg_Loss: 4.617140087760797
Episode: 96: Time: 11.81791353225708 Total Reward: 383.8732394366155
Avg_Loss: 4.592132741162757
Episode: 97: Time: 12.634928703308105 Total Reward: 243.36858006042712
Avg_Loss: 4.309277185371944
Episode: 98: Time: 11.950563907623291 Total Reward: 305.0000000000002
Avg_Loss: 4.959136028249724
Episode: 99: Time: 11.891663074493408 Total Reward: 307.09790209789645
Avg_Loss: 5.6263150708014225
Validation Mean Reward: 291.04586093796365 Validation Std Reward:
128.62639972851264
Test Mean Reward: 282.15487055973006 Test Std Reward: 154.494797631247
```

```
total_rewards, frames = trainerHardUpdateDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())
```

```
<IPython.core.display.HTML object>
```



Soft Updates (5 points)

A more recent improvement is to use soft updates, also known as Polyak averaging, where the target network is updated with a small fraction of the current model weights every step. In other words:

$$\theta_{target} = \tau \theta_{model} + (1 - \tau) \theta_{target}$$

for some $\tau < ¿ 1$ Please implement this by implementing the `_update_model` class in `SoftUpdateDQN` in `DQN.py`.

```
traineSoftUpdateDQN = DQN.SoftUpdateDQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                tau = 0.01,
                update_freq = 1,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cuda',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
1000,0.1),
                save_path =
utils.get_save_path("DoubleDQN_SoftUpdates","./runs/"))

traineSoftUpdateDQN.train(50,50,10,10,30)


#could not display the results as google colab ran out of ram, and
previous outputs got cleared. The answers to the questions below were
given based on the previous outputs that got wiped out due to runtime
disconnection
```

```
saving to ./runs/DoubleDQN_SoftUpdates/run0

/content/drive/MyDrive/RL-part12/DQN.py:316: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  states = torch.tensor(states, device=self.device,
dtype=torch.float32)
/content/drive/MyDrive/RL-part12/DQN.py:317: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  actions = torch.tensor(actions, device=self.device,
```

```
dtype=torch.int64).unsqueeze(1)
/content/drive/MyDrive/RL-part12/DQN.py:318: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  rewards = torch.tensor(rewards, device=self.device,
dtype=torch.float32).unsqueeze(1)
/content/drive/MyDrive/RL-part12/DQN.py:319: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  next_states = torch.tensor(next_states, device=self.device,
dtype=torch.float32)
/content/drive/MyDrive/RL-part12/DQN.py:320: UserWarning: To copy
construct from a tensor, it is recommended to use
sourceTensor.clone().detach() or
sourceTensor.clone().detach().requires_grad_(True), rather than
torch.tensor(sourceTensor).
  dones = torch.tensor(dones, device=self.device,
dtype=torch.float32).unsqueeze(1)

Episode: 1: Time: 11.81830358505249 Total Reward: 6.010101010101124
Avg_Loss: 0.9277429278868778
Episode: 2: Time: 11.876135110855103 Total Reward: -66.92982456140406
Avg_Loss: 0.8862019642060545
Episode: 3: Time: 12.710886240005493 Total Reward: -76.0126582278481
Avg_Loss: 0.6582750984068427
Episode: 4: Time: 12.67055058479309 Total Reward: -30.59322033898335
Avg_Loss: 0.6692482691665157
Episode: 5: Time: 11.904571771621704 Total Reward: -57.02531645569691
Avg_Loss: 0.6645211355150247
Episode: 6: Time: 12.859395027160645 Total Reward: -48.55191256830632
Avg_Loss: 0.6339102279739219
Episode: 7: Time: 12.53765606880188 Total Reward: -37.942942942943105
Avg_Loss: 0.6654018079157636
Episode: 8: Time: 12.217175722122192 Total Reward: -36.605839416059126
Avg_Loss: 0.6665439669888059
Episode: 9: Time: 12.147297620773315 Total Reward: -30.06493506493579
Avg_Loss: 0.7128296946214527
Episode: 10: Time: 11.811034440994263 Total Reward: -26.6546762589935
Avg_Loss: 0.7159173822014773
Episode: 11: Time: 12.186279773712158 Total Reward: -
39.805194805195455 Avg_Loss: 0.7447198288781303
Episode: 12: Time: 13.151418209075928 Total Reward: 59.92957746479032
Avg_Loss: 0.9935924203581169
Episode: 13: Time: 12.768341302871704 Total Reward: 148.72759856631245
Avg_Loss: 1.3026809762505924
```

```
total_rewards, frames = traineSoftUpdateDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())

----------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
<ipython-input-1-e2bf74bb33b2> in <cell line: 1>()
----> 1 total_rewards, frames =
traineSoftUpdateDQN.play_episode(0,True,42)
      2 anim = animate(frames)
      3 HTML(anim.to_jshtml())

NameError: name 'traineSoftUpdateDQN' is not defined
```

Questions:

- Which method performed better? (5 points)
- If we modify the τ for soft updates or the C for the hard updates, how does this affect the performance of the model, come up with a intuition for this, then experimentally verify this. (5 points)

Intuitively, increasing tau in soft updates means that more of the current model's weights will be copied over to the target network at each step. This can lead to overestimation of the target Q-values, resulting in unstable loss and degraded performance. Similarly, if we increase the update frequency for hard updates, the target model will more frequently adopt the current model's weights. This, too, can result in overestimation of Q-values and instability in the learning process, causing worse performance.

Experimental verification involves varying tau and the update frequency in controlled experiments to observe their effects on model stability and performance metrics. By systematically adjusting these parameters and monitoring the resulting loss and performance, we can validate the intuition that both high tau and frequent updates lead to instability and reduced performance.

```
%load_ext autoreload
%autoreload 2

The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload

!pip install numpy torch wandb swig gymnasium[box2d] matplotlib
termcolor

zsh:1: no matches found: gymnasium[box2d]

import test1
from utils import *
```

Reinforcement Learning Part 1: DQN

By Lawrence Liu and Tonmoy Monsoor

Some General Instructions

- As before, please keep the names of the layer consistent with what is requested in model.py. Otherwise the test functions will not work
- You will need to fill in the model.py, the DQN.py file, the buffer.py file, and the env_wrapper.py

DO NOT use Windows for this project, gymnasium does is not supported for windows and installing it will be a pain.

Introduction to the Enviroment

We will be training a DQN agent to play the game of CarRacing. The agent will be trained to play the game using the pixels of the game as an input. The reward structure is as follows for each frame:

- -0.1 for each frame
- +1000/N where N is the number of tiles visited by the car in the episode

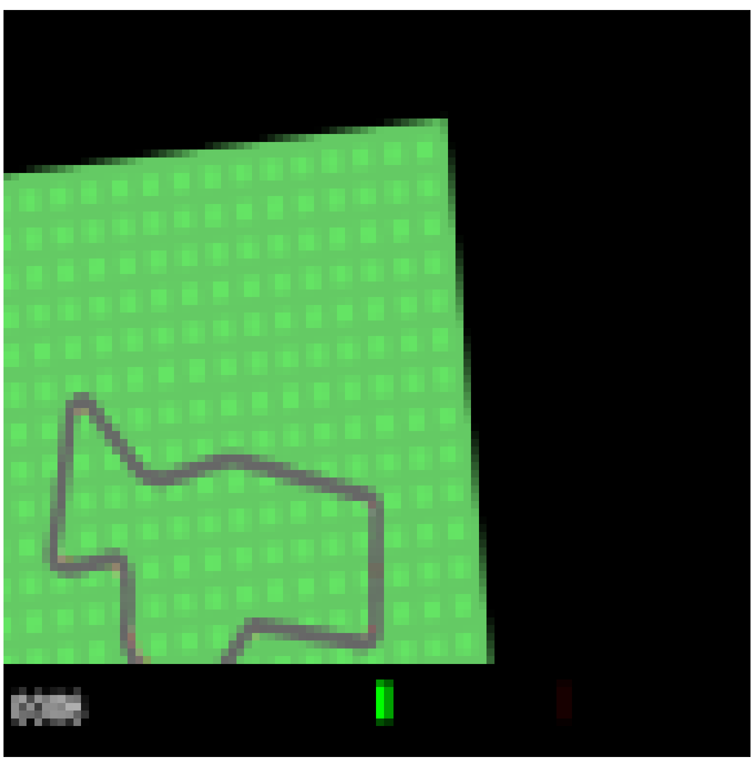
The overall goal of this game is to design a agent that is able to play the game with a average test score of above 600. In discrete mode the actions can take 5 actions,

- 0: Do Nothing
- 1: Turn Left
- 2: Turn Right
- 3: Accelerate
- 4: Brake

First let us visualize the game and understand the environment.

```python
import gymnasium as gym
import numpy as np
env = gym.make('CarRacing-v2', continuous=False,
render_mode='rgb_array')
env.np_random = np.random.RandomState(42)


from IPython.display import HTML

frames = []
s, _ = env.reset()

while True:
    a = env.action_space.sample()
    s, r, terminated, truncated, _ = env.step(a)
    frames.append(s)
    if terminated or truncated:
        break


anim = animate(frames)
HTML(anim.to_jshtml())
```

```
<IPython.core.display.HTML object>
```



So a couple things we can note:

- at the beginning of the game, we have 50 frames of the game slowly zooming into the car, we should ignore this period, ie no-op during this period.
- there is a black bar at the bottom of the screen, we should crop this out of the observation.

In addition, another thing to note is that the current frame doesn't give much information about the velocity and acceleration of the car, and that the car does not move much for each frame.

Environment Wrapper (5 points)

As a result, you will need to complete `EnvWrapper` in `env_wrapper.py`. You can find more information in the docstring for the wrapper, however the main idea is that it is a wrapper to the environment that does the following:

- skips the first 50 frames of the game
- crops out the black bar and reshapes the observation to a 84x84 image, as well as turning the resulting image to grayscale
- performs the actions for `skip_frames` frames
- stacks the last `num_frames` frames together to give the agent some information about the velocity and acceleration of the car.

```
from env_wrapper import EnvWrapper
test1.test_wrapper(EnvWrapper)

Passed reset
Passed step
```

CNN Model (5 points)

Now we are ready to build the model. Our architecture of the CNN model is the one proposed by Mnih et al in "Human-level control through deep reinforcement learning". Specifically this consists of the following layers:

- A convolutional layer with 32 filters of size 8x8 with stride 4 and relu activation
- A convolutional layer with 64 filters of size 4x4 with stride 2 and relu activation
- A convolutional layer with 64 filters of size 3x3 with stride 1 and relu activation
- A fully connected layer with 512 units and relu activation
- A fully connected layer with the number of outputs of the environment

Please implement this model `Nature_Paper_Conv` in `model.py` as well as the helper `MLP` class.

```
import model
test1.test_model_DQN(model.Nature_Paper_Conv)

Passed
```

DQN (40 points)

Now we are ready to implement the DQN algorithm.

title

Replay Buffer (5 points)

First start by implementing the DQN replay buffer `ReplayBufferDQN` in `buffer.py`. This buffer will store the transitions of the agent and sample them for training.

```
from replay_buffer import ReplayBufferDQN

test1.test_DQN_replay_buffer(ReplayBufferDQN)
```

```
Passed
```

DQN (15 points)

Now implement the `_optimize_model` and `sample_action` functions in `DQN` in `DQN.py`. The `_optimize_model` function will sample a batch of transitions from the replay buffer and update the model. The `sample_action` function will sample an action from the model given the current state. Train the model over 200 episdoes, validating every 50 episodes for 30 episodes, before testing the model for 50 episodes at the end.

```
import DQN
import utils
import torch


trainerDQN = DQN.DQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cpu',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
700,0.1),
                save_path = utils.get_save_path("DQN","./runs/"))

trainerDQN.train(200,50,30,50,50)
```

```
saving to ./runs/DQN/run5
Episode: 1: Time: 33.64623475074768 Total Reward: -46.45631067961198
Avg_Loss: 0.621379971195748
Episode: 2: Time: 49.94780421257019 Total Reward: -49.22535211267659
Avg_Loss: 0.7479008084491772
Episode: 3: Time: 50.54845309257507 Total Reward: -37.56756756756773
```

```
Avg_Loss: 0.834273373254207
Episode: 4: Time: 49.68352818489075 Total Reward: -73.72340425531954
Avg_Loss: 0.6650738081886989
Episode: 5: Time: 50.79087996482849 Total Reward: -66.32616487455289
Avg_Loss: 0.6004117653869531
Episode: 6: Time: 51.46130609512329 Total Reward: -53.33333333333401
Avg_Loss: 0.5451677008023282
Episode: 7: Time: 53.26769781112671 Total Reward: -10.750915750916104
Avg_Loss: 0.598176203652465
Episode: 8: Time: 52.90106701850891 Total Reward: -28.566433566433965
Avg_Loss: 0.6221847685539171
Episode: 9: Time: 52.767723083496094 Total Reward: -18.0769230769237
Avg_Loss: 0.8168262183008825
Episode: 10: Time: 53.60030007362366 Total Reward: 86.81818181818642
Avg_Loss: 1.04799678937352
Episode: 11: Time: 50.190484046936035 Total Reward: 28.23943661972038
Avg_Loss: 1.3439788921665745
Episode: 12: Time: 50.82638621330261 Total Reward: 30.4480286738371
Avg_Loss: 1.3861980509607732
Episode: 13: Time: 51.64816498756409 Total Reward: 258.12499999998954
Avg_Loss: 1.9806916128937937
Episode: 14: Time: 54.55801177024841 Total Reward: 425.91254752850443
Avg_Loss: 3.5661716409841504
Episode: 15: Time: 51.78815817832947 Total Reward: 35.71895424836572
Avg_Loss: 3.079725874071362
Episode: 16: Time: 52.11056184768677 Total Reward: 117.02531645570079
Avg_Loss: 2.9803095098052705
Episode: 17: Time: 53.69854784011841 Total Reward: 304.37106918237856
Avg_Loss: 3.0869274104342743
Episode: 18: Time: 52.72233438491821 Total Reward: 277.0930232558041
Avg_Loss: 3.5243931845957492
Episode: 19: Time: 52.645715951919556 Total Reward: 51.41744548286924
Avg_Loss: 3.9933031474341867
Episode: 20: Time: 54.29171419143677 Total Reward: 684.8507462686431
Avg_Loss: 4.215122670436106
Episode: 21: Time: 54.68485379219055 Total Reward: 82.21518987342205
Avg_Loss: 4.851664407413547
Episode: 22: Time: 52.806293964385986 Total Reward: 130.45454545454913
Avg_Loss: 5.445202106938643
Episode: 23: Time: 53.46469521522522 Total Reward: 187.44274809160788
Avg_Loss: 4.729979395866394
Episode: 24: Time: 58.42589282989502 Total Reward: 56.31578947368739
Avg_Loss: 5.309162298170459
Episode: 25: Time: 53.06667995452881 Total Reward: 683.1818181818039
Avg_Loss: 5.294066311431532
Episode: 26: Time: 53.0435791015625 Total Reward: 169.40677966102106
Avg_Loss: 5.860033095383844
Episode: 27: Time: 52.147809743881226 Total Reward: 223.493150684931
Avg_Loss: 5.210964104708503
```

```
Episode: 28: Time: 53.11340308189392 Total Reward: 540.7142857142768
Avg_Loss: 5.50252356599359
Episode: 29: Time: 56.79640316963196 Total Reward: 96.69329073482851
Avg_Loss: 6.523201285540557
Episode: 30: Time: 54.681421995162964 Total Reward: 170.3061224489833
Avg_Loss: 6.38453203089097
Episode: 31: Time: 53.700287103652954 Total Reward: 41.0544217687102
Avg_Loss: 6.367567416499643
Episode: 32: Time: 56.46209001541138 Total Reward: 435.90909090908576
Avg_Loss: 6.447676725748207
Episode: 33: Time: 53.82415795326233 Total Reward: -16.875000000000387
Avg_Loss: 6.432568666814756
Episode: 34: Time: 54.0392529964447 Total Reward: 88.73493975903952
Avg_Loss: 6.256359584191266
Episode: 35: Time: 54.00353169441223 Total Reward: 155.00000000000406
Avg_Loss: 6.5316011079219205
Episode: 36: Time: 55.09529519081116 Total Reward: 258.146853146854
Avg_Loss: 6.041156560433011
Episode: 37: Time: 53.61705994606018 Total Reward: 192.23404255319608
Avg_Loss: 6.578339706949827
Episode: 38: Time: 55.72029709815979 Total Reward: 387.01438848919867
Avg_Loss: 6.314972243389161
Episode: 39: Time: 57.086889028549194 Total Reward: 15.726643598617102
Avg_Loss: 5.627142270072167
Episode: 40: Time: 55.74293613433838 Total Reward: 364.07473309608355
Avg_Loss: 6.054150554813257
Episode: 41: Time: 55.525856733322144 Total Reward: 223.4931506849328
Avg_Loss: 6.14043435880116
Episode: 42: Time: 61.85062289237976 Total Reward: 158.73134328358563
Avg_Loss: 5.613795484815325
Episode: 43: Time: 60.33282780647278 Total Reward: -33.906752411576264
Avg_Loss: 6.178658878602901
Episode: 44: Time: 57.814558029174805 Total Reward: 475.4697986577085
Avg_Loss: 6.142874357580137
Episode: 45: Time: 56.246787786483765 Total Reward: 499.6843853820505
Avg_Loss: 6.496735743113926
Episode: 46: Time: 64.1799910068512 Total Reward: 703.3870967741833
Avg_Loss: 5.942577422166071
Episode: 47: Time: 57.02860689163208 Total Reward: 369.2857142857139
Avg_Loss: 6.443236067014582
Episode: 48: Time: 58.42327880859375 Total Reward: 499.20289855071746
Avg_Loss: 6.3071842058366085
Episode: 49: Time: 58.35699391365051 Total Reward: 375.58823529411467
Avg_Loss: 6.102885396540666
Validation Mean Reward: 489.33650102589104 Validation Std Reward:
225.11574617679648
Episode: 50: Time: 61.259177923202515 Total Reward: 645.2135231316628
Avg_Loss: 6.064129487807009
Episode: 51: Time: 57.49976897239685 Total Reward: 378.46938775509227
```

```
Avg_Loss: 6.76241214085026
Episode: 52: Time: 56.12747287750244 Total Reward: 574.2015209125376
Avg_Loss: 6.450969653971055
Episode: 53: Time: 55.936400175094604 Total Reward: 491.9565217391273
Avg_Loss: 7.051125503387771
Episode: 54: Time: 59.46299886703491 Total Reward: 428.51097178682835
Avg_Loss: 7.625298823128228
Episode: 55: Time: 60.04116988182068 Total Reward: 332.65273311896226
Avg_Loss: 7.7188250998489
Episode: 56: Time: 59.03689384460449 Total Reward: 384.1666666666015
Avg_Loss: 6.575207440292134
Episode: 57: Time: 69.49121427536011 Total Reward: 161.05536332180347
Avg_Loss: 6.677355480795147
Episode: 58: Time: 65.27638602256775 Total Reward: 345.6250000000002
Avg_Loss: 7.082136759237081
Episode: 59: Time: 61.8988778591156 Total Reward: 278.7704918032703
Avg_Loss: 7.4480417716402965
Episode: 60: Time: 59.03009581565857 Total Reward: 487.758620689644
Avg_Loss: 6.897337224804053
Episode: 61: Time: 60.85618591308594 Total Reward: 424.99999999999415
Avg_Loss: 6.944099206884368
Episode: 62: Time: 59.46905303001404 Total Reward: 234.23076923077278
Avg_Loss: 6.726859815481331
Episode: 63: Time: 61.01921105384827 Total Reward: 347.30769230768664
Avg_Loss: 7.014593502553571
Episode: 64: Time: 61.4706130027771 Total Reward: 250.0704225352152
Avg_Loss: 7.083203234091527
Episode: 65: Time: 58.27303099632263 Total Reward: 411.66666666665463
Avg_Loss: 6.527263447016227
Episode: 66: Time: 58.728004932403564 Total Reward: 376.8100890207636
Avg_Loss: 6.686196259590758
Episode: 67: Time: 59.80058217048645 Total Reward: 360.88235294116765
Avg_Loss: 6.781131441853628
Episode: 68: Time: 62.2064208984375 Total Reward: 263.3061889250846
Avg_Loss: 6.988901278551887
Episode: 69: Time: 66.20619010925293 Total Reward: 104.99999999999967
Avg_Loss: 6.862469565968554
Episode: 70: Time: 68.87205505371094 Total Reward: 440.7142857142813
Avg_Loss: 6.348082040538307
Episode: 71: Time: 73.3116500377655 Total Reward: 115.52631578947813
Avg_Loss: 6.193681653307266
Episode: 72: Time: 66.82011985778809 Total Reward: 256.0971786833897
Avg_Loss: 6.572448198534861
Episode: 73: Time: 67.07410407066345 Total Reward: 300.3488372093046
Avg_Loss: 6.286181738396652
Episode: 74: Time: 68.32024717330933 Total Reward: 663.2417582417484
Avg_Loss: 6.561651724727214
Episode: 75: Time: 68.93181800842285 Total Reward: 120.7534246575378
Avg_Loss: 6.971023908182352
```

```
Episode: 76: Time: 67.61667704582214 Total Reward: 381.97368421052346
Avg_Loss: 6.878874406093309
Episode: 77: Time: 65.25357699394226 Total Reward: 291.5248226950358
Avg_Loss: 6.751811821921533
Episode: 78: Time: 62.17144298553467 Total Reward: 43.888888888891984
Avg_Loss: 6.72028810537162
Episode: 79: Time: 64.2475860118866 Total Reward: 464.8591549295706
Avg_Loss: 6.679002023544632
Episode: 80: Time: 69.67176795005798 Total Reward: 267.53776435045427
Avg_Loss: 6.45061404865329
Episode: 81: Time: 63.353769063949585 Total Reward: 196.22807017544164
Avg_Loss: 7.541419429939334
Episode: 82: Time: 63.25091791152954 Total Reward: 370.0349650349621
Avg_Loss: 6.338344218851137
Episode: 83: Time: 63.65936303138733 Total Reward: 550.4849498327656
Avg_Loss: 6.8350174777648025
Episode: 84: Time: 62.64472508430481 Total Reward: 368.02250803857777
Avg_Loss: 6.355866239852264
Episode: 85: Time: 61.353074073791504 Total Reward: 183.38827838828186
Avg_Loss: 7.150538857243642
Episode: 86: Time: 66.87700009346008 Total Reward: 379.0259740259704
Avg_Loss: 6.766335490370999
Episode: 87: Time: 68.94229888916016 Total Reward: 194.65517241379655
Avg_Loss: 6.875291861405893
Episode: 88: Time: 68.01729488372803 Total Reward: 445.6504065040557
Avg_Loss: 7.262423921032112
Episode: 89: Time: 66.76441383361816 Total Reward: 408.4722222222163
Avg_Loss: 6.9876094024722315
Episode: 90: Time: 65.48671197891235 Total Reward: 356.17845117844365
Avg_Loss: 7.047443471035035
Episode: 91: Time: 67.8172619342804 Total Reward: 87.58426966292232
Avg_Loss: 7.466605549099064
Episode: 92: Time: 64.95865893363953 Total Reward: 399.545454545448
Avg_Loss: 7.199181750041096
Episode: 93: Time: 64.8789541721344 Total Reward: 355.35460992907434
Avg_Loss: 6.717558355892406
Episode: 94: Time: 71.82892322540283 Total Reward: 449.71544715446714
Avg_Loss: 6.818872549453704
Episode: 95: Time: 66.78087210655212 Total Reward: 222.30769230769442
Avg_Loss: 6.915401888494732
Episode: 96: Time: 73.58318901062012 Total Reward: 437.679738562083
Avg_Loss: 7.386136882445392
Episode: 97: Time: 68.11665487289429 Total Reward: 40.76158940397267
Avg_Loss: 7.262045689991543
Episode: 98: Time: 66.49566316604614 Total Reward: -18.344947735192314
Avg_Loss: 6.798180919735372
Episode: 99: Time: 64.9186019897461 Total Reward: -50.10204081632726
Avg_Loss: 6.823781408181711
Validation Mean Reward: 141.69282624338342 Validation Std Reward:
```

```
221.00208002518363
Episode: 100: Time: 71.8241069316864 Total Reward: 209.79452054794882
Avg_Loss: 7.240328903959579
Episode: 101: Time: 77.15736484527588 Total Reward: 258.5714285714202
Avg_Loss: 7.464505012295827
Episode: 102: Time: 74.4230899810791 Total Reward: 362.2368421052581
Avg_Loss: 7.5919334457701995
Episode: 103: Time: 68.99015927314758 Total Reward: 140.29411764706168
Avg_Loss: 7.4646740841264485
Episode: 104: Time: 69.27991461753845 Total Reward: 76.97452229299802
Avg_Loss: 7.1311458319175145
Episode: 105: Time: 74.05055212974548 Total Reward: 197.92929292929708
Avg_Loss: 7.7816496977285174
Episode: 106: Time: 76.51363682746887 Total Reward: 272.92452830188944
Avg_Loss: 9.07080288995214
Episode: 107: Time: 82.45590877532959 Total Reward: -
31.02693602693692 Avg_Loss: 8.386322747759458
Episode: 108: Time: 76.02178406715393 Total Reward: 226.10091743119554
Avg_Loss: 7.399962234396894
Episode: 109: Time: 73.47811603546143 Total Reward: 229.84076433121436
Avg_Loss: 7.623166126363418
Episode: 110: Time: 75.75035881996155 Total Reward: 198.72937293729757
Avg_Loss: 8.217188180995588
Episode: 111: Time: 68.99765586853027 Total Reward: 280.4385964912285
Avg_Loss: 6.728052752358573
Episode: 112: Time: 77.47173976898193 Total Reward: 470.3710247349747
Avg_Loss: 7.573547732930224
Episode: 113: Time: 76.03002214431763 Total Reward: 85.32786885246068
Avg_Loss: 7.289745062339206
Episode: 114: Time: 76.53569293022156 Total Reward: 366.78343949044216
Avg_Loss: 7.625339350780519
Episode: 115: Time: 74.22380495071411 Total Reward: 558.7102473498185
Avg_Loss: 7.762692403392632
Episode: 116: Time: 78.05453276634216 Total Reward: 448.252595155702
Avg_Loss: 6.892880026031943
Episode: 117: Time: 80.07034420967102 Total Reward: 373.16479400748824
Avg_Loss: 8.251333960965901
Episode: 118: Time: 79.90969491004944 Total Reward: 546.7322834645574
Avg_Loss: 7.442910329634402
Episode: 119: Time: 78.96516394615173 Total Reward: 510.5776892430181
Avg_Loss: 7.490799329861873
Episode: 120: Time: 81.22752285003662 Total Reward: 295.2439024390228
Avg_Loss: 7.405261684866512
Episode: 121: Time: 77.25574898719788 Total Reward: 319.1104294478489
Avg_Loss: 6.852371616523807
Episode: 122: Time: 81.75012993812561 Total Reward: 343.88888888888556
Avg_Loss: 7.377834673689193
Episode: 123: Time: 84.49005007743835 Total Reward: 408.4482758620624
Avg_Loss: 8.00041523400475
```

```
Episode: 124: Time: 81.69341731071472 Total Reward: 341.6666666666646
Avg_Loss: 7.669552850122211
Episode: 125: Time: 82.16062808036804 Total Reward: 325.89552238805817
Avg_Loss: 6.926144568359151
Episode: 126: Time: 83.68137502670288 Total Reward: 399.9152542372795
Avg_Loss: 8.838991395565643
Episode: 127: Time: 78.8473379611969 Total Reward: 462.3770491803208
Avg_Loss: 7.5701461218986195
Episode: 128: Time: 77.66732883453369 Total Reward: 169.70588235294218
Avg_Loss: 7.306439012038608
Episode: 129: Time: 73.17180800437927 Total Reward: 286.4102564102592
Avg_Loss: 7.1799589836296915
Episode: 130: Time: 74.52871131896973 Total Reward: 434.999999999992
Avg_Loss: 7.223027390592239
Episode: 131: Time: 74.96689081192017 Total Reward: 194.7526501766811
Avg_Loss: 8.666567024062662
Episode: 132: Time: 76.98279619216919 Total Reward: 520.9999999999939
Avg_Loss: 7.634790927422147
Episode: 133: Time: 69.8701810836792 Total Reward: 177.46376811594382
Avg_Loss: 7.784738930333562
Episode: 134: Time: 47.01002025604248 Total Reward: -
16.576056338026206 Avg_Loss: 7.59377086074264
Episode: 135: Time: 76.55198907852173 Total Reward: 453.8958990536218
Avg_Loss: 7.1285843303223615
Episode: 136: Time: 78.05792808532715 Total Reward: 472.27272727271816
Avg_Loss: 7.948981841071313
Episode: 137: Time: 85.72963571548462 Total Reward: 367.5407166123756
Avg_Loss: 9.282568515849714
Episode: 138: Time: 75.57768511772156 Total Reward: 184.56989247311964
Avg_Loss: 7.565983527848701
Episode: 139: Time: 77.61507105827332 Total Reward: 458.5714285714209
Avg_Loss: 8.172599740389014
Episode: 140: Time: 76.05552697181702 Total Reward: 153.27586206896962
Avg_Loss: 7.71006051231833
Episode: 141: Time: 73.91905570030212 Total Reward: 508.4482758620626
Avg_Loss: 7.302748458726065
Episode: 142: Time: 73.9156539440155 Total Reward: 666.9047619047493
Avg_Loss: 7.3280482953336055
Episode: 143: Time: 83.91173982620239 Total Reward: 607.7027027026879
Avg_Loss: 7.822093986663498
Episode: 144: Time: 83.35806584358215 Total Reward: 252.985347985349
Avg_Loss: 10.432934545669236
Episode: 145: Time: 81.25071597099304 Total Reward: 180.74750830565262
Avg_Loss: 7.337890401607802
Episode: 146: Time: 80.3260018825531 Total Reward: 356.72413793103044
Avg_Loss: 7.091957631231356
Episode: 147: Time: 83.27202296257019 Total Reward: 502.97297297296404
Avg_Loss: 6.868378148359411
Episode: 148: Time: 78.32472109794617 Total Reward: 241.9565217391259
```

```
Avg_Loss: 7.041921863034994
Episode: 149: Time: 84.8521728515625 Total Reward: 607.1276595744594
Avg_Loss: 7.1250444981230405
Validation Mean Reward: 560.0880399847724 Validation Std Reward:
217.7028981356559
Episode: 150: Time: 77.69760632514954 Total Reward: 433.4280936454777
Avg_Loss: 7.389433575277569
Episode: 151: Time: 77.35571479797363 Total Reward: 415.2739726027329
Avg_Loss: 7.6518665361805125
Episode: 152: Time: 81.08129692077637 Total Reward: 368.41463414633097
Avg_Loss: 8.178523160830265
Episode: 153: Time: 79.62837409973145 Total Reward: 359.0059347180979
Avg_Loss: 7.216892825455225
Episode: 154: Time: 72.45755791664124 Total Reward: 544.2857142857032
Avg_Loss: 6.7612808111335045
Episode: 155: Time: 84.58133888244629 Total Reward: 325.6642066420644
Avg_Loss: 8.035128827856369
Episode: 156: Time: 80.53923463821411 Total Reward: 476.4285714285688
Avg_Loss: 7.776280762768593
Episode: 157: Time: 108.10552024841309 Total Reward:
335.71161048688043 Avg_Loss: 8.259226254054479
Episode: 158: Time: 85.30701994895935 Total Reward: 431.6903914590707
Avg_Loss: 7.480665193886316
Episode: 159: Time: 88.2815010547638 Total Reward: 432.67527675276165
Avg_Loss: 7.074238134031536
Episode: 160: Time: 86.08399796485901 Total Reward: 323.06020066888937
Avg_Loss: 6.967288572247289
Episode: 161: Time: 78.71313905715942 Total Reward: 423.15181518151235
Avg_Loss: 7.404480775865186
Episode: 162: Time: 86.85184097290039 Total Reward: 290.5799373040741
Avg_Loss: 8.070461848202873
Episode: 163: Time: 91.75559997558594 Total Reward: 387.8767123287645
Avg_Loss: 7.437357658097724
Episode: 164: Time: 101.74043273925781 Total Reward:
465.13745704466714 Avg_Loss: 8.080201364364944
Episode: 165: Time: 84.93437385559082 Total Reward: 149.25287356322235
Avg_Loss: 9.156547015454588
Episode: 166: Time: 81.22871899604797 Total Reward: 600.4732510287989
Avg_Loss: 7.427548683991953
Episode: 167: Time: 92.59352612495422 Total Reward: 569.206642066415
Avg_Loss: 6.9820642110680335
Episode: 168: Time: 81.97988700866699 Total Reward: 197.99363057325155
Avg_Loss: 8.32588833019513
Episode: 169: Time: 85.28010988235474 Total Reward: 256.5151515151525
Avg_Loss: 8.692153489890218
Episode: 170: Time: 83.00338816642761 Total Reward: 495.2777777777674
Avg_Loss: 7.401759772741494
Episode: 171: Time: 78.25646996498108 Total Reward: 574.0140845070332
Avg_Loss: 7.926327085294643
```

```
Episode: 172: Time: 82.7880539894104 Total Reward: 572.8700361010716
Avg_Loss: 7.281446593148368
Episode: 173: Time: 83.59668397903442 Total Reward: 394.7260273972574
Avg_Loss: 7.528526702848803
Episode: 174: Time: 96.13386607170105 Total Reward: 398.7106918238943
Avg_Loss: 7.258989763860943
Episode: 175: Time: 95.2544891834259 Total Reward: 247.1052631578978
Avg_Loss: 8.50469161782946
Episode: 176: Time: 92.52732276916504 Total Reward: 313.66873065015545
Avg_Loss: 7.340523414251183
Episode: 177: Time: 81.07248616218567 Total Reward: 142.9310344827628
Avg_Loss: 8.459865051157335
Episode: 178: Time: 83.56491088867188 Total Reward: 496.24087591240186
Avg_Loss: 7.5564393656594415
Episode: 179: Time: 76.50907897949219 Total Reward: 318.1054131054103
Avg_Loss: 7.547323698757076
Episode: 180: Time: 72.93215012550354 Total Reward: 560.4307116104765
Avg_Loss: 7.826677541772859
Episode: 181: Time: 73.83921003341675 Total Reward: 328.5668789808908
Avg_Loss: 8.360199584680444
Episode: 182: Time: 72.59744310379028 Total Reward: 454.3421052631496
Avg_Loss: 7.333664076668875
Episode: 183: Time: 80.8677020072937 Total Reward: 490.36585365853
Avg_Loss: 8.58086670947676
Episode: 184: Time: 78.92857122421265 Total Reward: 519.8148148148052
Avg_Loss: 7.927594110745342
Episode: 185: Time: 71.30113506317139 Total Reward: 423.6335403726637
Avg_Loss: 6.995670780414293
Episode: 186: Time: 71.57406687736511 Total Reward: 502.0149253731261
Avg_Loss: 8.083983775948276
Episode: 187: Time: 76.87101006507874 Total Reward: 333.57142857142514
Avg_Loss: 7.723429567673627
Episode: 188: Time: 74.07410407066345 Total Reward: 498.10344827584936
Avg_Loss: 7.461824718643637
Episode: 189: Time: 73.33454704284668 Total Reward: -9.473684210527104
Avg_Loss: 7.512720678032947
Episode: 190: Time: 74.05295991897583 Total Reward: 283.67647058823826
Avg_Loss: 7.804621593291018
Episode: 191: Time: 77.3699939250946 Total Reward: 354.43820224717865
Avg_Loss: 7.435303150104875
Episode: 192: Time: 77.00978112220764 Total Reward: 404.99999999999386
Avg_Loss: 9.533743613908271
Episode: 193: Time: 76.73225116729736 Total Reward: 509.5627376425788
Avg_Loss: 7.847385360413239
Episode: 194: Time: 74.43513703346252 Total Reward: 441.42384105959525
Avg_Loss: 7.173261998080406
Episode: 195: Time: 40.473479986190796 Total Reward: 42.62727272727463
Avg_Loss: 7.394262275998554
Episode: 196: Time: 73.74002528190613 Total Reward: 278.376623376623
```

```
Avg_Loss: 8.074731802239137
Episode: 197: Time: 77.2547378540039 Total Reward: 411.8493150684892
Avg_Loss: 8.068089658472719
Episode: 198: Time: 74.18098521232605 Total Reward: 514.7560975609681
Avg_Loss: 7.846896720533615
Episode: 199: Time: 76.00750279426575 Total Reward: 694.8832684824782
Avg_Loss: 7.718907142887597
Validation Mean Reward: 585.0353719021165 Validation Std Reward:
148.9263359347255
Test Mean Reward: 587.8407838539056 Test Std Reward:
134.53964499395124
```

Please include a plot of the training and validation rewards over the episodes in the report. An additional question to answer is does the loss matter in DQN? Why or why not?

In Deep Q-Networks (DQN), the loss function is crucial as it measures the discrepancy between predicted Q-values and target Q-values, guiding the neural network's weight updates. Minimizing the loss ensures the model's Q-value predictions align closely with expected rewards, promoting stability and convergence. A stable loss indicates that the model is learning effectively and avoiding issues like overestimation, which can lead to suboptimal policies. However, while the loss is important for training, the ultimate goal of DQN is to maximize cumulative rewards. Therefore, a low loss does not always directly translate to high performance, as the model must also navigate exploration-exploitation trade-offs and the environment's stochastic nature.

We can also draw a animation of the car in one game, the code is provided below

```
#plotting the training rewards and the validation rewards
import matplotlib.pyplot as plt

def plot_metrics():
    episodes = list(range(1, 200))  # Assuming you want to plot for
199 episodes

    times = [
        33.64623475074768, 49.94780421257019, 50.54845309257507,
49.68352818489075, 50.79087996482849,
        51.46130609512329, 53.26769781112671, 52.90106701850891,
52.767723083496094, 53.60030007362366,
        50.190484046936035, 50.82638621330261, 51.64816498756409,
54.55801177024841, 51.78815817832947,
        52.11056184768677, 53.69854784011841, 52.72233438491821,
52.645715951919556, 54.29171419143677,
        54.68485379219055, 52.806293964385986, 53.46469521522522,
58.42589282989502, 53.06667995452881,
        53.0435791015625, 52.147809743881226, 53.11340308189392,
56.79640316963196, 54.681421995162964,
        53.700287103652954, 56.46209001541138, 53.82415795326233,
54.0392529964447, 54.00353169441223,
```

```
        55.09529519081116, 53.61705994606018, 55.72029709815979,
57.086889028549194, 55.74293613433838,
        55.525856733322144, 61.85062289237976, 60.33282780647278,
57.814558029174805, 56.246787786483765,
        64.1799910068512, 57.02860689163208, 58.42327880859375,
58.35699391365051, 61.259177923202515,
        57.49976897239685, 56.12747287750244, 55.936400175094604,
59.46299886703491, 60.04116988182068,
        59.03689384460449, 69.49121427536011, 65.27638602256775,
61.8988778591156, 59.03009581565857,
        60.85618591308594, 59.46905303001404, 61.01921105384827,
61.4706130027771, 58.27303099632263,
        58.728004932403564, 59.80058217048645, 62.2064208984375,
66.20619010925293, 68.87205505371094,
        73.3116500377655, 66.82011985778809, 67.07410407066345,
68.32024717330933, 68.93181800842285,
        67.61667704582214, 65.25357699394226, 62.17144298553467,
64.2475860118866, 69.67176795005798,
        63.353769063949585, 63.25091791152954, 63.65936303138733,
62.64472508430481, 61.353074073791504,
        66.87700009346008, 68.94229888916016, 68.01729488372803,
66.76441383361816, 65.48671197891235,
        67.8172619342804, 64.95865893363953, 64.8789541721344,
71.82892322540283, 66.78087210655212,
        73.58318901062012, 68.11665487289429, 66.49566316604614,
64.9186019897461, 71.8241069316864,
        77.15736484527588, 74.4230899810791, 68.99015927314758,
69.27991461753845, 74.05055212974548,
        76.51363682746887, 82.45590877532959, 76.02178406715393,
73.47811603546143, 75.75035881996155,
        68.99765586853027, 77.47173976898193, 76.03002214431763,
76.53569293022156, 74.22380495071411,
        78.05453276634216, 80.07034420967102, 79.90969491004944,
78.96516394615173, 81.22752285003662,
        77.25574898719788, 81.75012993812561, 84.49005007743835,
81.69341731071472, 82.16062808036804,
        83.68137502670288, 78.8473379611969, 77.66732883453369,
73.17180800437927, 74.52871131896973,
        74.96689081192017, 76.98279619216919, 69.8701810836792,
47.01002025604248, 76.55198907852173,
        78.05792808532715, 85.72963571548462, 75.57768511772156,
77.61507105827332, 76.05552697181702,
        73.91905570030212, 73.9156539440155, 83.91173982620239,
83.35806584358215, 81.25071597099304,
        80.3260018825531, 83.27202296257019, 78.32472109794617,
84.8521728515625, 77.69760632514954,
        77.35571479797363, 81.08129692077637, 79.62837409973145,
72.45755791664124, 84.58133888244629,
        80.53923463821411, 108.10552024841309, 85.30701994895935,
```

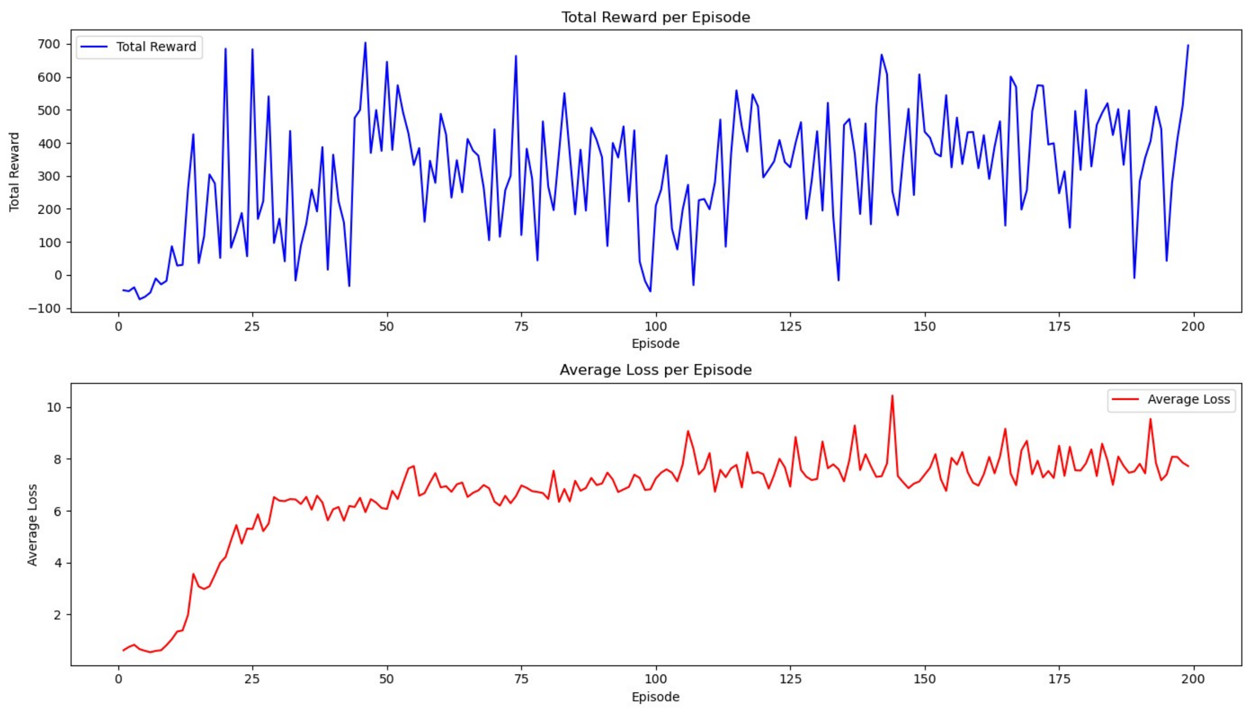
```
88.2815010547638, 86.08399796485901,
        78.71313905715942, 86.85184097290039, 91.75559997558594,
101.74043273925781, 84.93437385559082,
        81.22871899604797, 92.59352612495422, 81.97988700866699,
85.28010988235474, 83.00338816642761,
        78.25646996498108, 82.7880539894104, 83.59668397903442,
96.13386607170105, 95.2544891834259,
        92.52732276916504, 81.07248616218567, 83.56491088867188,
76.50907897949219, 72.93215012550354,
        73.83921003341675, 72.59744310379028, 80.8677020072937,
78.92857122421265, 71.30113506317139,
        71.57406687736511, 76.87101006507874, 74.07410407066345,
73.33454704284668, 74.05295991897583,
        77.3699939250946, 77.00978112220764, 76.73225116729736,
74.43513703346252, 40.473479986190796,
        73.74002528190613, 77.2547378540039, 74.18098521232605,
76.00750279426575
    ]

    total_rewards = [
        -46.45631067961198, -49.22535211267659, -37.56756756756773, -
73.72340425531954, -66.32616487455289,
        -53.33333333333401, -10.750915750916104, -28.566433566433965,
-18.0769230769237, 86.81818181818642,
        28.23943661972038, 30.4480286738371, 258.12499999998954,
425.91254752850443, 35.71895424836572,
        117.02531645570079, 304.37106918237856, 277.0930232558041,
51.41744548286924, 684.8507462686431,
        82.21518987342205, 130.45454545454913, 187.44274809160788,
56.31578947368739, 683.1818181818039,
        169.40677966102106, 223.493150684931, 540.7142857142768,
96.69329073482851, 170.3061224489833,
        41.0544217687102, 435.90909090908576, -16.875000000000387,
88.73493975903952, 155.00000000000406,
        258.146853146854, 192.23404255319608, 387.01438848919867,
15.726643598617102, 364.07473309608355,
        223.4931506849328, 158.73134328358563, -33.906752411576264,
475.4697986577085, 499.6843853820505,
        703.3870967741833, 369.2857142857139, 499.20289855071746,
375.58823529411467, 645.2135231316628,
        378.46938775509227, 574.2015209125376, 491.9565217391273,
428.51097178682835, 332.65273311896226,
        384.16666666666015, 161.05536332180347, 345.6250000000002,
278.7704918032703, 487.758620689644,
        424.99999999999415, 234.23076923077278, 347.30769230768664,
250.0704225352152, 411.66666666665463,
        376.8100890207636, 360.88235294116765, 263.3061889250846,
104.99999999999967, 440.7142857142813,
        115.52631578947813, 256.0971786833897, 300.3488372093046,
```

```
663.2417582417484, 120.7534246575378,
        381.97368421052346, 291.5248226950358, 43.888888888891984,
464.8591549295706, 267.53776435045427,
        196.22807017544164, 370.0349650349621, 550.4849498327656,
368.02250803857777, 183.38827838828186,
        379.0259740259704, 194.65517241379655, 445.6504065040557,
408.4722222222163, 356.17845117844365,
        87.5842696629232, 399.545454545448, 355.35460992907434,
449.71544715446714, 222.30769230769442,
        437.679738562083, 40.76158940397267, -18.344947735192314, -
50.10204081632726, 209.79452054794882,
        258.5714285714202, 362.2368421052581, 140.29411764706168,
76.97452229299802, 197.92929292929708,
        272.92452830188944, -31.026936026936692, 226.10091743119554,
229.84076433121436, 198.72937293729757,
        280.4385964912285, 470.3710247349747, 85.32786885246068,
366.78343949044216, 558.7102473498185,
        448.252595155702, 373.16479400748824, 546.7322834645574,
510.5776892430181, 295.2439024390228,
        319.1104294478489, 343.88888888888556, 408.4482758620624,
341.6666666666646, 325.89552238805817,
        399.9152542372795, 462.3770491803208, 169.70588235294218,
286.4102564102592, 434.999999999992,
        194.7526501766811, 520.9999999999939, 177.46376811594382, -
16.576056338026206, 453.8958990536218,
        472.27272727271816, 367.5407166123756, 184.56989247311964,
458.5714285714209, 153.27586206896962,
        508.4482758620626, 666.9047619047493, 607.7027027026879,
252.985347985349, 180.74750830565262,
        356.72413793103044, 502.97297297296404, 241.9565217391259,
607.1276595744594, 433.4280936454777,
        415.2739726027329, 368.41463414633097, 359.0059347180979,
544.2857142857032, 325.6642066420644,
        476.4285714285688, 335.71161048688043, 431.6903914590707,
432.67527675276165, 323.06020066888937,
        423.15181518151235, 290.5799373040741, 387.8767123287645,
465.13745704466714, 149.25287356322235,
        600.4732510287989, 569.206642066415, 197.99363057325155,
256.5151515151525, 495.2777777777674,
        574.0140845070332, 572.8700361010716, 394.7260273972574,
398.7106918238943, 247.1052631578978,
        313.66873065015545, 142.9310344827628, 496.24087591240186,
318.1054131054103, 560.4307116104765,
        328.5668789808908, 454.3421052631496, 490.36585365853,
519.8148148148052, 423.6335403726637,
        502.0149253731261, 333.57142857142514, 498.10344827584936, -
9.473684210527104, 283.67647058823826,
        354.43820224717865, 404.99999999999386, 509.5627376425788,
441.42384105959525, 42.62727272727463,
```

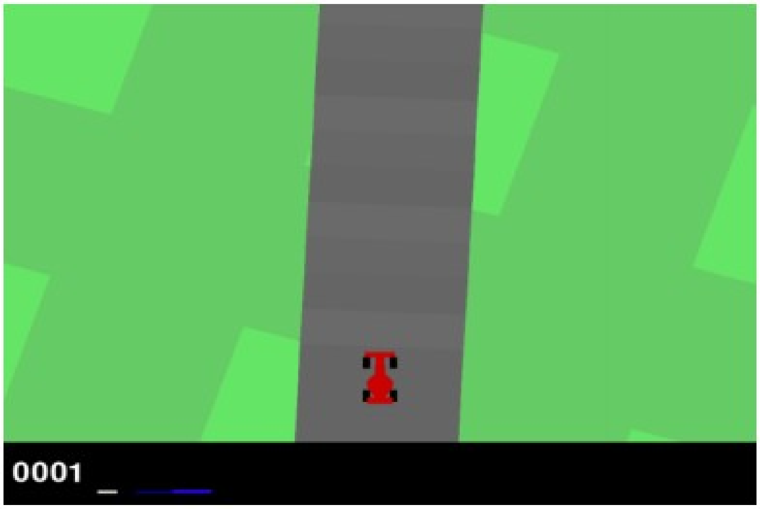
```
        278.376623376623, 411.8493150684892, 514.7560975609681,
694.8832684824782
    ]

    avg_losses = [
        0.621379971195748, 0.7479008084491772, 0.834273373254207,
0.6650738081886989, 0.6004117653869531,
        0.5451677008023282, 0.598176203652465, 0.6221847685539171,
0.8168262183008825, 1.04799678937352,
        1.3439788921665745, 1.3861980509607732, 1.9806916128937937,
3.5661716409841504, 3.079725874071362,
        2.9803095098052705, 3.0869274104342743, 3.5243931845957492,
3.9933031474341867, 4.215122670436106,
        4.851664407413547, 5.445202106938643, 4.729979395866394,
5.309162298170459, 5.294066311431532,
        5.860033095383844, 5.210964104708503, 5.50252356599359,
6.523201285540557, 6.38453203089097,
        6.367567416499643, 6.447676725748207, 6.432568666814756,
6.256359584191266, 6.5316011079219205,
        6.041156560433011, 6.578339706949827, 6.314972243389161,
5.627142270072167, 6.054150554813257,
        6.14043435880116, 5.613795484815325, 6.178658878602901,
6.142874357580137, 6.496735743113926,
        5.942577422166071, 6.443236067014582, 6.3071842058366085,
6.102885396540666, 6.064129487807009,
        6.76241214085026, 6.450969653971055, 7.051125503387771,
7.625298823128228, 7.7188250998489,
        6.575207440292134, 6.677355480795147, 7.082136759237081,
7.4480417716402965, 6.897337224804053,
        6.944099206884368, 6.726859815481331, 7.014593502553571,
7.083203234091527, 6.527263447016227,
        6.686196259590758, 6.781131441853628, 6.988901278551887,
6.862469565968554, 6.348082040538307,
        6.193681653307266, 6.572448198534861, 6.286181738396652,
6.561651724727214, 6.971023908182352,
        6.878874406093309, 6.751811821921533, 6.72028810537162,
6.679002023544632, 6.45061404865329,
        7.541419429939334, 6.338344218851137, 6.8350174777648025,
6.355866239852264, 7.150538857243642,
        6.766335490370999, 6.875291861405893, 7.262423921032112,
6.9876094024722315, 7.047443471035035,
        7.466605549099064, 7.199181750041096, 6.717558355892406,
6.818872549453704, 6.915401888494732,
        7.386136882445392, 7.262045689991543, 6.798180919735372,
6.823781408181711, 7.240328903959579,
        7.464505012295827, 7.5919334457701995, 7.4646740841264485,
7.1311458319175145, 7.7816496977285174,
        9.07080288995214, 8.386322747759458, 7.399962234396894,
7.623166126363418, 8.217188180995588,
```

```
        6.728052752358573, 7.573547732930224, 7.289745062339206,
7.625339350780519, 7.762692403392632,
        6.892880026031943, 8.251333960965901, 7.442910329634402,
7.490799329861873, 7.405261684866512,
        6.852371616523807, 7.377834673689193, 8.00041523400475,
7.669552850122211, 6.926144568359151,
        8.838991395565643, 7.5701461218986195, 7.306439012038608,
7.1799589836296915, 7.223027390592239,
        8.666567024062662, 7.634790927422147, 7.784738930333562,
7.59377086074264, 7.1285843303223615,
        7.948981841071313, 9.282568515849714, 7.565983527848701,
8.172599740389014, 7.71006051231833,
        7.302748458726065, 7.3280482953336055, 7.822093986663498,
10.432934545669236, 7.337890401607802,
        7.091957631231356, 6.868378148359411, 7.041921863034994,
7.1250444981230405, 7.389433575277569,
        7.6518665361805125, 8.178523160830265, 7.216892825455225,
6.7612808111335045, 8.035128827856369,
        7.776280762768593, 8.259226254054479, 7.480665193886316,
7.074238134031536, 6.967288572247289,
        7.404480775865186, 8.070461848202873, 7.437357658097724,
8.080201364364944, 9.156547015454588,
        7.427548683991953, 6.9820642110680335, 8.32588833019513,
8.692153489890218, 7.401759772741494,
        7.926327085294643, 7.281446593148368, 7.528526702848803,
7.258989763860943, 8.50469161782946,
        7.340523414251183, 8.459865051157335, 7.5564393656594415,
7.547323698757076, 7.826677541772859,
        8.360199584680444, 7.333664076668875, 8.58086670947676,
7.927594110745342, 6.995670780414293,
        8.083983775948276, 7.723429567673627, 7.461824718643637,
7.512720678032947, 7.804621593291018,
        7.435303150104875, 9.533743613908271, 7.847385360413239,
7.173261998080406, 7.394262275998554,
        8.074731802239137, 8.068089658472719, 7.8468967205336115,
7.718907142887597
    ]

    plt.figure(figsize=(14, 8))

    # Plot Total Reward
    plt.subplot(2, 1, 1)
    plt.plot(episodes, total_rewards, label='Total Reward', color='b')
    plt.xlabel('Episode')
    plt.ylabel('Total Reward')
    plt.title('Total Reward per Episode')
    plt.legend()

    # Plot Avg Loss
```

```python
    plt.subplot(2, 1, 2)
    plt.plot(episodes, avg_losses, label='Average Loss', color='r')
    plt.xlabel('Episode')
    plt.ylabel('Average Loss')
    plt.title('Average Loss per Episode')
    plt.legend()

    plt.tight_layout()
    plt.savefig('./runs/DQN/run5/metrics_plot.png')
    plt.show()

# Call the function to plot the metrics
plot_metrics()

#playing an episode
eval_env = gym.make('CarRacing-v2', continuous=True,
render_mode='rgb_array')
eval_env = EnvWrapper(eval_env)

total_rewards, frames = trainerDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())
```




```
<IPython.core.display.HTML object>
```



Double DQN

In the original paper, where the algorithim is shown above, the estimated target Q value was computed using the current Q network's weights. However, this can lead to overestimation of the Q values. To mitigate this, we can use the target network to compute the target Q value. This is known as Double DQN.

Hard updating Target Network (5 points)

Original implementations for this involved hard updates, where the model weights were copied to the target network every C steps. This is known as hard updating. This was what was used in the Nature Paper by Mnih et al 2015 "Human-level control through deep reinforcement learning"

Please implement this by implementing the `_optimize_model` and `_update_model` classes in `HardUpdateDQN` in `DQN.py`.

```
trainerHardUpdateDQN = DQN.HardUpdateDQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                update_freq = 100,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cuda',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
1000,0.1),
                save_path =
utils.get_save_path("DoubleDQN_HardUpdates/","./runs/"))
```

```
trainerHardUpdateDQN.train(200,50,30,50,50)

total_rewards, frames = trainerHardUpdateDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())
```

Soft Updates (5 points)

A more recent improvement is to use soft updates, also known as Polyak averaging, where the target network is updated with a small fraction of the current model weights every step. In other words:

$$\theta_{target} = \tau \theta_{model} + (1 - \tau) \theta_{target}$$

for some $\tau <¿1$ Please implement this by implementing the `_update_model` class in `SoftUpdateDQN` in `DQN.py`.

```
traineSoftUpdateDQN = DQN.SoftUpdateDQN(EnvWrapper(env),
                model.Nature_Paper_Conv,
                tau = 0.01,
                update_freq = 1,
                lr = 0.00025,
                gamma = 0.95,
                buffer_size=100000,
                batch_size=32,
                loss_fn = "mse_loss",
                use_wandb = False,
                device = 'cuda',
                seed = 42,
                epsilon_scheduler = utils.exponential_decay(1,
1000,0.1),
                save_path =
utils.get_save_path("DoubleDQN_SoftUpdates","./runs/"))

traineSoftUpdateDQN.train(200,50,30,50,50)

----------------------------------------------------------------------
-----
FileNotFoundError                         Traceback (most recent call
last)
Input In [26], in <cell line: 1>()
      1 traineSoftUpdateDQN = DQN.SoftUpdateDQN(EnvWrapper(env),
      2                 model.Nature_Paper_Conv,
      3                 tau = 0.01,
      4                 update_freq = 1,
      5                 lr = 0.00025,
      6                 gamma = 0.99,
      7                 buffer_size=100000,
      8                 batch_size=32,
```

```
      9                    loss_fn = "mse_loss",
     10                    use_wandb = True,
     11                    device = 'cuda',
     12                    seed = 42,
     13                    epsilon_scheduler = utils.exponential_decay(1,
1000,0.1),
---> 14                    save_path =
utils.get_save_path("DoubleDQN_SoftUpdates","./runs/"))
     16 traineSoftUpdateDQN.train(200,50,30,50,50)

File /mnt/SSD3/lawrence/ECE239-Project4/utils.py:44, in
get_save_path(suffix, directory)
     42 save_path = os.path.join(directory,suffix)
     43 #find the number of run directories in the directory
---> 44 runs = [d for d in os.listdir(save_path) if "run" in d]
     45 runs = sorted(runs,key = lambda x: int(x.split("run")[1]))
     46 if len(runs) == 0:

FileNotFoundError: [Errno 2] No such file or directory:
'./runs/DoubleDQN_SoftUpdates'

total_rewards, frames = traineSoftUpdateDQN.play_episode(0,True,42)
anim = animate(frames)
HTML(anim.to_jshtml())
```

Questions:

- Which method performed better? (5 points)
- If we modify the τ for soft updates or the C for the hard updates, how does this affect the performance of the model, come up with a intuition for this, then experimentally verify this. (5 points)

```python
import cv2
import numpy as np
import gymnasium as gym
import matplotlib.pyplot as plt
from utils import preprocess #this is a helper function that may be useful to grayscale and
crop the image


class EnvWrapper(gym.Wrapper):
    def __init__(
        self,
        env:gym.Env,
        skip_frames:int=4,
        stack_frames:int=4,
        initial_no_op:int=50,
        do_nothing_action:int=0,
        **kwargs
    ):
        """the environment wrapper for CarRacing-v2

        Args:
            env (gym.Env): the original environment
            skip_frames (int, optional): the number of frames to skip, in other words we will
            repeat the same action for `skip_frames` steps. Defaults to 4.
            stack_frames (int, optional): the number of frames to stack, we stack
            `stack_frames` frames to form the state and allow agent understand the motion of
the car. Defaults to 4.
            initial_no_op (int, optional): the initial number of no-op steps to do nothing at
the beginning of the episode. Defaults to 50.
            do_nothing_action (int, optional): the action index for doing nothing. Defaults to
0, which should be correct unless you have modified the
            discretization of the action space.
        """
        super(EnvWrapper, self).__init__(env, **kwargs)
        self.initial_no_op = initial_no_op
        self.skip_frames = skip_frames
        self.stack_frames = stack_frames
        self.observation_space = gym.spaces.Box(
            low=0,
            high=1,
            shape=(stack_frames, 84, 84),
            dtype=np.float32
        )
        self.do_nothing_action = do_nothing_action



    def reset(self, **kwargs):
        # call the enviroment reset
        s, info = self.env.reset(**kwargs)

        # Do nothing for the next self.initial_no_op` steps
        for i in range(self.initial_no_op):
            s, r, terminated, truncated, info = self.env.step(self.do_nothing_action)

        # Crop and resize the frame
        s = preprocess(s)

        # stack the frames to form the initial state
        self.stacked_state = np.tile(s, (self.stack_frames, 1, 1))  # [
        return self.stacked_state, info

    def step(self, action):
        reward = 0
        for _ in range(self.skip_frames):
            s, r, terminated, truncated, info = self.env.step(action)
            reward += r
```

```
            if terminated or truncated:
                break

        s = preprocess(s)
        self.stacked_state = np.concatenate((self.stacked_state[1:], s[np.newaxis]), axis=0)

        return self.stacked_state, reward, terminated, truncated, info
```

```python
import torch as torch
import torch.nn as nn


import torch
import torch.nn as nn
import numpy as np


class MLP(nn.Module):
    def __init__(self, input_size:int, action_size:int,
hidden_size:int=256,non_linear:nn.Module=nn.ReLU):
        """
        input: tuple[int]
            The input size of the image, of shape (channels, height, width)
        action_size: int
            The number of possible actions
        hidden_size: int
            The number of neurons in the hidden layer

        This is a seperate class because it may be useful for the bonus questions
        """
        super(MLP, self).__init__()
        #====== TODO: ======
        # self.linear1 =
        # self.output = #output layer
        # self.non_linear = non_linear()
        self.linear1 = nn.Linear(input_size, hidden_size)
        self.output = nn.Linear(hidden_size, action_size)
        self.non_linear = non_linear()




    def forward(self, x:torch.Tensor)->torch.Tensor:
        x_ = self.non_linear(self.linear1(x))
        return self.output(x_)


class Nature_Paper_Conv(nn.Module):
    """
    A class that defines a neural network with the following architecture:
    - 1 convolutional layer with 32 8x8 kernels with a stride of 4x4 w/ ReLU activation
    - 1 convolutional layer with 64 4x4 kernels with a stride of 2x2 w/ ReLU activation
    - 1 convolutional layer with 64 3x3 kernels with a stride of 1x1 w/ ReLU activation
    - 1 fully connected layer with 512 neurons and ReLU activation.
    Based on 2015 paper 'Human-level control through deep reinforcement learning' by Mnih et al
    """
    def __init__(self, input_size:tuple[int], action_size:int,**kwargs):
        """
        input: tuple[int]
            The input size of the image, of shape (channels, height, width)
        action_size: int
            The number of possible actions
        **kwargs: dict
            additional kwargs to pass for stuff like dropout, etc if you would want to
implement it
        """
        super(Nature_Paper_Conv, self).__init__()
        #===== TODO: ======
        # self.CNN = nn.Sequential(*[
        #
        # ])
        # self.MLP =
        self.CNN = nn.Sequential(
            nn.Conv2d(input_size[0], 32, 8, stride=4),
            nn.ReLU(),
```

```
            nn.Conv2d(32, 64, 4, stride=2),
            nn.ReLU(),
            nn.Conv2d(64, 64, 3, stride=1),
            nn.ReLU()


        )
        self.MLP = MLP(64*7*7, action_size, hidden_size=512)



    def forward(self, x:torch.Tensor)->torch.Tensor:
        x_ = self.CNN(x)
        x_ = x_.view(x_.size(0), -1)
        return self.MLP(x_)
```

```python
import random
import torch
import numpy as np

class ReplayBufferDQN:
    def __init__(self, buffer_size:int, seed:int = 42):
        self.buffer_size = buffer_size
        self.seed = seed
        self.buffer = []
        random.seed(self.seed)

    def add(self, state:np.ndarray, action:int, reward:float, next_state:np.ndarray
            , done:bool):
        """Add a new experience to the buffer

        Args:
            state (np.ndarray): the current state of shape [n_c,h,w]
            action (int): the action taken
            reward (float): the reward received
            next_state (np.ndarray): the next state of shape [n_c,h,w]
            done (bool): whether the episode is done
        """
        self.buffer.append((state,action,reward,next_state,done))

        # forget earliest memory
        if len(self.buffer) > self.buffer_size:
            self.buffer.pop(0)


    def sample(self,batch_size:int,device = 'cpu'):
        """Sample a batch of experiences from the buffer

        Args:
            batch_size (int): the number of samples to take

        Returns:
            states (torch.Tensor): a np.ndarray of shape [batch_size,n_c,h,w] of dtype float32
            actions (torch.Tensor): a np.ndarray of shape [batch_size] of dtype int64
            rewards (torch.Tensor): a np.ndarray of shape [batch_size] of dtype float32
            next_states (torch.Tensor): a np.ndarray of shape [batch_size,n_c,h,w] of dtype
float32
            dones (torch.Tensor): a np.ndarray of shape [batch_size] of dtype bool
        """
        idx = random.sample(range(len(self.buffer)),batch_size)

        states,actions,rewards,next_states,dones = [],[],[],[],[]

        for i in idx:
            state,action,reward,next_state,done = self.buffer[i]
            states.append(torch.from_numpy(state))
            actions.append(action)
            rewards.append(reward)
            next_states.append(torch.from_numpy(next_state))
            dones.append(done)

        # each state is [n_c, h, w]
        # stack them to get shape [batch_size,n_c,h,w]
        states = torch.stack(states).to(device).float()
        actions = torch.tensor(actions).to(device).long()
        rewards = torch.tensor(rewards).to(device).float()
        next_states = torch.stack(next_states).to(device).float()
        dones = torch.tensor(dones).to(device).bool()
        return states,actions,rewards,next_states,dones


    def __len__(self):
        return len(self.buffer)
```

```python
import torch
import torch.nn as nn
import traceback
from termcolor import colored
import gymnasium as gym
import numpy as np
import sys

#get the operating system
if sys.platform.startswith('darwin'):
    # Mac OS X
    suffix = "_mac"
else:
    suffix = ""

def test_model_DQN(model):
    try:
        model = model((4,84,84),5)

        model.load_state_dict(torch.load("test_weights.pt", map_location="cpu"))
        # Test the forward function
        test_outputs =  torch.load(f"test_outputs{suffix}.pt",map_location=torch.device('cpu'))
        test_inputs = test_outputs["S"]
        test_outputs = test_outputs["outputs"]
        model.eval()
        with torch.no_grad():
            for i in range(len(test_inputs)):
                # print(torch.tensor(test_inputs[i]).float().shape)
                assert
torch.allclose(model(torch.tensor(test_inputs[i]).float().unsqueeze(0)),torch.tensor(test_outputs[i])),
f"expected {test_outputs[i]} but got {model(torch.tensor(test_inputs[i]).float().unsqueeze(0))}"

        print(colored("Passed","green"))
    except Exception as e:
        print(e)
        print(colored("Failed","red"))
        traceback.print_exc()
        return

def test_model_DDPG(model):
    try:
        model = model((3,96,96),5)

        model.load_state_dict(torch.load("test_weights.pt", map_location="cpu"))
        # Test the forward function
        test_outputs =  torch.load(f"test_outputs{suffix}.pt",map_location=torch.device('cpu'))
        test_inputs = test_outputs["S"]
        test_outputs = test_outputs["outputs"]
        model.eval()
        with torch.no_grad():
            for i in range(len(test_inputs)):
                # print(torch.tensor(test_inputs[i]).float().shape)
                assert
torch.allclose(model(torch.tensor(test_inputs[i]).float().unsqueeze(0)),torch.tensor(test_outputs[i])),
f"expected {test_outputs[i]} but got {model(torch.tensor(test_inputs[i]).float().unsqueeze(0))}"

        print(colored("Passed","green"))
    except Exception as e:
        print(e)
        print(colored("Failed","red"))
        traceback.print_exc()
        return

def test_wrapper(wrapper):
    try:
        env = gym.make('CarRacing-v2', continuous=False, render_mode='rgb_array')
        wrapper = wrapper(env)

        # Test the reset function
        test_outputs = torch.load(f"test_outputs{suffix}.pt",map_location=torch.device('cpu'))
        test_inputs = test_outputs["outputs"]
        test_outputs = test_outputs["S"]
```

```
        s,_ = wrapper.reset(seed = 42)
        # print(s.shape)
        # print(test_outputs[0].shape)
        wrong_indexs = ~np.isclose(test_outputs[0],s)
        assert np.allclose(test_outputs[0],s), f"at {np.where(wrong_indexs)} expected 
{test_outputs[0][wrong_indexs]} but got {s[wrong_indexs]}"
        print(colored("Passed reset","green"))

        for i  in range(len(test_outputs)-1):
            # Test the step function
            s,r,terminated, truncated, info = wrapper.step(np.argmax(test_inputs[i]))
            assert np.allclose(test_outputs[i+1],s), f"expected {test_outputs[i+1]} but got {s}"

        print(colored("Passed step","green"))
    except Exception as e:
        print(e)
        print(colored("Failed","red"))
        traceback.print_exc()
        return

import pickle

def check_same_torch(a,b):
    #first check shape
    if a.shape != b.shape:
        return False
    #then check values
    return torch.allclose(a,b)





def test_DQN_replay_buffer(buffer_class):
    with open(f"test_replay_buffer_inputs{suffix}.pkl","rb") as f:
        buffer_inputs = pickle.load(f)
    buffer_samples = torch.load(f"test_replay_buffer_samples{suffix}.pth")
    try:
        buffer = buffer_class(40,seed = 42)
        j = 0
        for i in range(100):
            buffer.add(buffer_inputs["states"][i],buffer_inputs["actions"][i],buffer_inputs["rewards"]
[i],buffer_inputs["next_states"][i],buffer_inputs["dones"][i])
            if i % 30 == 29:
                # print(i)
                target_outputs = buffer_samples[j]
                actual_outputs = buffer.sample(5)
                for k in range(len(target_outputs)):
                    # print(target_outputs[k],actual_outputs[k])
                    assert check_same_torch(target_outputs[k],actual_outputs[k]), f"expected 
{target_outputs[k][0]} but got {buffer.sample(1)[k][0]}"
                # assert np.all(buffer_samples[j] == buffer.sample(40)), f"expected 
{buffer_samples[j]} but got {buffer.sample(40)}"
                j += 1

        print(colored("Passed","green"))
    except:
        print(colored("Failed","red"))
        traceback.print_exc()
        return





if __name__ == "__main__":
    from replay_buffer import ReplayBufferDQN

    test_DQN_replay_buffer(ReplayBufferDQN)


    from env_wrapper import EnvWrapper
    test_wrapper(EnvWrapper)
```

```python
from model import Nature_Paper_Conv
test_model_DQN(Nature_Paper_Conv)
```

```python
import torch
import torch.optim as optim
import torch.nn.functional as F
import torch.nn
import gymnasium as gym
from replay_buffer import ReplayBufferDQN
import wandb
import random
import numpy as np
import os
import time
from utils import exponential_decay
import typing


class DQN:
    def __init__(self, env: typing.Union[gym.Env, gym.Wrapper],
                 model: torch.nn.Module,
                 model_kwargs: dict = {},
                 lr: float = 0.001, gamma: float = 0.99,
                 buffer_size: int = 10000, batch_size: int = 32,
                 loss_fn: str = 'mse_loss',
                 use_wandb: bool = False, device: str = 'cpu',
                 seed: int = 42,
                 epsilon_scheduler=exponential_decay(1, 700, 0.1),
                 save_path: str = None):
        self.env = env
        self._set_seed(seed)

        self.observation_space = self.env.observation_space.shape
        self.model = model(
            self.observation_space,
            self.env.action_space.n, **model_kwargs
        ).to(device)
        self.model.train()
        self.optimizer = optim.Adam(self.model.parameters(), lr=lr)
        self.gamma = gamma

        self.replay_buffer = ReplayBufferDQN(buffer_size)
        self.batch_size = batch_size
        self.i_update = 0
        self.device = device
        self.epsilon_decay = epsilon_scheduler
        self.save_path = save_path if save_path is not None else "./"

        if loss_fn == 'smooth_l1_loss':
            self.loss_fn = F.smooth_l1_loss
        elif loss_fn == 'mse_loss':
            self.loss_fn = F.mse_loss
        else:
            raise ValueError('loss_fn must be either smooth_l1_loss or mse_loss')

        self.wandb = use_wandb
        if self.wandb:
            wandb.init(project='racing-car-dqn')
            wandb.config.update({
                'lr': lr,
                'gamma': gamma,
                'buffer_size': buffer_size,
                'batch_size': batch_size,
                'loss_fn': loss_fn,
                'device': device,
                'seed': seed,
                'save_path': save_path
            })

    def train(self, n_episodes: int = 1000, validate_every: int = 100, n_validation_episodes:
```

```python
int = 10, n_test_episodes: int = 10, save_every: int = 100):
        os.makedirs(self.save_path, exist_ok=True)
        best_val_reward = -np.inf

        for episode in range(n_episodes):
            state, _ = self.env.reset()
            done = False
            truncated = False
            total_reward = 0
            i = 0
            loss = 0
            start_time = time.time()
            epsilon = self.epsilon_decay()
            while (not done) and (not truncated):
                action = self._sample_action(state, epsilon)
                next_state, reward, done, truncated, _ = self.env.step(action)
                self.replay_buffer.add(state, action, reward, next_state, done)
                total_reward += reward
                state = next_state

                not_warm_starting, l = self._optimize_model()
                if not_warm_starting:
                    loss += l
                    epsilon = self.epsilon_decay()
                    i += 1
            if i != 0:
                if self.wandb:
                    wandb.log({'total_reward': total_reward, 'loss': loss / i})
                print(f"Episode: {episode}: Time: {time.time() - start_time} Total Reward:
{total_reward} Avg_Loss: {loss / i}")
            if episode % validate_every == validate_every - 1:
                mean_reward, std_reward = self.validate(n_validation_episodes)
                if self.wandb:
                    wandb.log({'mean_reward': mean_reward, 'std_reward': std_reward})
                print("Validation Mean Reward: {} Validation Std Reward:
{}".format(mean_reward, std_reward))
                if mean_reward > best_val_reward:
                    best_val_reward = mean_reward
                    self._save('best')

            if episode % save_every == save_every - 1:
                self._save(str(episode))

        self._save('final')
        self.load_model('best')
        mean_reward, std_reward = self.validate(n_test_episodes)
        if self.wandb:
            wandb.log({'mean_test_reward': mean_reward, 'std_test_reward': std_reward})
        print("Test Mean Reward: {} Test Std Reward: {}".format(mean_reward, std_reward))

    def _optimize_model(self):
        if len(self.replay_buffer) < 10 * self.batch_size:
            return False, 0

        states, actions, rewards, next_states, dones =
self.replay_buffer.sample(self.batch_size, self.device)

        with torch.no_grad():
            dones = dones.float()  # Convert dones to float
            target_q = rewards + self.gamma * (1 - dones) * self.model(next_states).max(1)[0]

        q_values = self.model(states).gather(1, actions.unsqueeze(1)).squeeze(1)

        loss = self.loss_fn(q_values, target_q)

        self.optimizer.zero_grad()
        loss.backward()
```

```python
        self.optimizer.step()

        return True, loss.item()

    def _sample_action(self, state: np.ndarray, epsilon: float = 0.1) -> int:
        sample = random.random()
        if sample < epsilon:
            return random.randint(0, self.env.action_space.n - 1)
        else:
            with torch.no_grad():
                state = torch.tensor(state, dtype=torch.float32).unsqueeze(0).to(self.device)
                return self.model(state).argmax().item()

    def _set_seed(self, seed: int):
        random.seed(seed)
        np.random.seed(seed)
        self.seed = seed
        torch.manual_seed(seed)
        torch.cuda.manual_seed(seed)
        torch.backends.cudnn.deterministic = True
        gym.utils.seeding.np_random(seed)

    def _validate_once(self):
        state, _ = self.env.reset()
        done = False
        truncated = False
        total_reward = 0
        while (not done) and (not truncated):
            action = self._sample_action(state, 0)
            next_state, reward, done, truncated, _ = self.env.step(action)
            total_reward += reward
            state = next_state
        return total_reward

    def validate(self, n_episodes: int = 10):
        rewards_per_episode = []
        for _ in range(n_episodes):
            rewards_per_episode.append(self._validate_once())
        return np.mean(rewards_per_episode), np.std(rewards_per_episode)

    def load_model(self, suffix: str = ''):
        self.model.load_state_dict(torch.load(os.path.join(self.save_path,
f'model_{suffix}.pt')))

    def _save(self, suffix: str = ''):
        torch.save(self.model.state_dict(), os.path.join(self.save_path,
f'model_{suffix}.pt'))

    def play_episode(self, epsilon: float = 0, return_frames: bool = True, seed: int = None):
        if seed is not None:
            state, _ = self.env.reset(seed=seed)
        else:
            state, _ = self.env.reset()

        done = False
        total_reward = 0
        if return_frames:
            frames = []

        with torch.no_grad():
            while not done:
                action = self._sample_action(state, epsilon)
                next_state, reward, terminated, truncated, _ = self.env.step(action)
                total_reward += reward
                done = terminated or truncated
                if return_frames:
                    frames.append(self.env.render())
```

```
                state = next_state

        if return_frames:
            return total_reward, frames

        return total_reward
class HardUpdateDQN(DQN):
    def __init__(self, env, model, model_kwargs: dict = {}, update_freq: int = 5, *args,
**kwargs):
        super().__init__(env, model, model_kwargs, *args, **kwargs)
        init_state_dict = self.model.state_dict()
        self.target_model = model(self.observation_space, self.env.action_space.n,
**model_kwargs).to(self.device)
        self.target_model.load_state_dict(init_state_dict)
        self.update_freq = update_freq

    def _optimize_model(self):

        #====== TODO: ======
        if len(self.replay_buffer) < 10 * self.batch_size:
            return False, 0

        states, actions, rewards, next_states, dones =
self.replay_buffer.sample(self.batch_size, self.device)

        with torch.no_grad():
            dones = dones.float()
            target_q = rewards + self.gamma * (1 - dones) *
self.target_model(next_states).max(1)[0]

        q_values = self.model(states).gather(1, actions.unsqueeze(1)).squeeze(1)

        loss = self.loss_fn(q_values, target_q)

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

        self._update_model()
        return True, loss.item()

    def _update_model(self):
        #====== TODO: ======
        self.i_update += 1
        if self.i_update % self.update_freq == 0:
            self.target_model.load_state_dict(self.model.state_dict())

    def _save(self, suffix: str = ''):
        torch.save(self.model.state_dict(), os.path.join(self.save_path,
f'model_{suffix}.pt'))
        torch.save(self.target_model.state_dict(), os.path.join(self.save_path,
f'target_model_{suffix}.pt'))

    def load_model(self, suffix: str = ''):
        self.model.load_state_dict(torch.load(os.path.join(self.save_path,
f'model_{suffix}.pt')))
        self.target_model.load_state_dict(torch.load(os.path.join(self.save_path,
f'target_model_{suffix}.pt')))

class SoftUpdateDQN(HardUpdateDQN):
    def __init__(self, env, model, model_kwargs: dict = {}, tau: float = 0.01, *args,
**kwargs):
        super().__init__(env, model, model_kwargs, *args, **kwargs)
        self.tau = tau

    def _update_model(self):
         #====== TODO: ======
```

```
        for target_param, param in zip(self.target_model.parameters(),
self.model.parameters()):
            target_param.data.copy_(self.tau * param.data+(1 - self.tau) * target_param.data)
```