Problem 1. We will implement a basic deep Q learning (DQL) to solve the `CartPole-v1` problem shown in Fig. 1

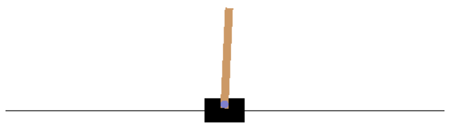


Fig. 1. The `CartPole-v1` environment

A pole is attached by an un-actuated joint to a cart. The system is controlled by applying a force of +1 or -1 to the cart. The pendulum starts upright, and the goal is to prevent it from falling over.

- State: 4-D vector with [Cart position, Cart velocity, Pole angle, Pole angular velocity]
- Action: 2 discrete values 0: push cart to the left; 1: push cart to the right
- Reward: +1 for every step taken
- Terminal: Pole angle is more than 12 degrees or; cart falls outside of the screen or; episode last for longer than 500.

We consider the `CartPole-v1` problem solved if the RL algorithm can last 501 steps without falling for 5 consecutive episodes. We do not restrict how many episodes it experience to get to solving, but for DQL it should be no more than 200.

In the attachment of this assignment, we have provided with you the base code that includes the agent-environment interaction and deep Q network initialization (in Python and PyTorch framework). However, the updates of the neural networks are missing and are left out for you to fill. You will find the code snippet in the `dqn.py` marked as TODO:

```
def update(self, buffer):
    t = buffer.sample(self.batch_size)

    s = t.obs
    a = t.action
    r = t.reward
    sp = t.next_obs
    done = t.done

    # TODO: perform a single Q-network update step. Also, update the target Q-network
     every C Q-network update steps
```

Please submit your Python script `dqn.py` with the filled `update(self, buffer)` function. In addition, please attach the plot showing the episodic return over the course of training. You can use any of the plotting libraries to generate the figure. One example is the Tensorboard toolkit which we shall discuss.

1 Setup

This section will prepare all the prerequisites for running the homework program. If you are familiar with virtual environments and Python for deep learning please feel free to skip this section.

We will create a virtual environment to organize all the dependencies for our project. Then we will install the PyTorch deep learning framework, Gym reinforcement learning environment, and the Tensorboard visualization toolkit.

1.1 Create a conda Environment

1.1.1 Install Anaconda or Miniconda

We will be using conda to create and manage our virtual environments. To use conda please install Anaconda on your system first:

`https://docs.anaconda.com/anaconda/install/#installation`

Anaconda contains many data science tools and will take 3GB. If your system has limited disk space, you can alternatively install the Miniconda instead of the full Anaconda:

`https://docs.conda.io/en/latest/miniconda.html`

1.1.2 Create and open a conda Environment

Step 1: After installing Anaconda, for Windows machines, please search and open "Anaconda Prompt" from the start menu. For Linux/Mac machines, just open the terminal.

Step 2: Create a conda environment by typing

```
$ conda create -n myenv python=3.7
```

in the Anaconda prompt or the terminal. You can replace `myenv` with your preferred name. The `$` symbol indicates shell command.

More details about creating and managing environment can be found at

`https://docs.conda.io/projects/conda/en/latest/user-guide/`
`tasks/manage-environments.html#`

Step 3: Activate the conda environment by typing the following in the Anaconda prompt or the terminal:

```
$ conda activate myenv
```

Now you should see something like `(myenv)$` which means the environment is activated in this terminal.

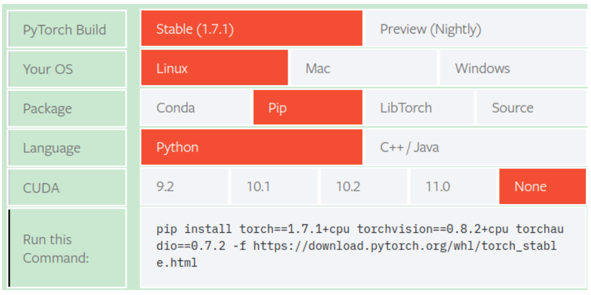
We will install all the following packages in the environment we've just created. So please keep the environment activated and use the same terminal/prompt for all the following installations.

1.2 Install PyTorch

Go to

`https://pytorch.org/get-started/locally/`

and select your appropriate setup. For example mine looks like the following:



PyTorch Build	[✓ Stable (1.7.1)]	Preview (Nightly)			
Your OS	[✓ Linux]	Mac	Windows		
Package	Conda	[✓ Pip]	LibTorch	Source	
Language	[✓ Python]	C++ / Java			
CUDA	9.2	10.1	10.2	11.0	[✓ None]
Run this Command:	`pip install torch==1.7.1+cpu torchvision==0.8.2+cpu torchaudio==0.7.2 -f https://download.pytorch.org/whl/torch_stable.html`				

Please copy and paste the Run this Command in your terminal/prompt and run it. By selecting CUDA=None, the neural network training will be done on CPU. If your machine has a CUDA supporting GPU please feel free to select the appropriate CUDA version and modify the code to run the training process on GPU.

1.3 Install OpenAI Gym

Run the following in your terminal/prompt:

```
(myenv)$ pip install gym
```

More details about Gym can be found at

`https://gym.openai.com/docs/#installation`

1.4 Install Tensorboard

Run the following in your terminal/prompt:

```
(myenv)$ pip install tensorboard==1.15
```

More details can be found at

`https://pytorch.org/tutorials/recipes/recipes/tensorboard_with_pytorch.html`

1.5 Install Stable-Baselines (Optional)

Stable-Baselines is a set of high quality reinforcement learning algorithm implementations. In our homework we will not use them as we will develop our own. However you can compare your results with these implementations. To install Stable-Baselines run the following in your terminal/prompt:

```
(myenv)$ pip install stable-baselines
```

More details can be found at

`https://stable-baselines.readthedocs.io/en/master/`

2 Run the Codes

In the opened terminal, navigate to the directory containing the homework files. Then run the code:

```
(myenv)$ python run_dqn.py
```

Or run the Stable-Baselines implementation:

```
(myenv)$ python run_dqn_baseline.py
```

3 Visualize the Results

To visualize the training curves we could use Tensorboard. Please open a new terminal and activate the conda environment:

```
$ conda activate myenv
```

Then navigate to the homework directory containing a folder called `tensorboard` and run

```
(myenv)$ tensorboard --logdir=tensorboard
```

The terminal will display an HTTP path. Copy and paste that path to your browser. Then Tensorboard should lively show and update the training logs we've declared in the `run_dqn.py` file.