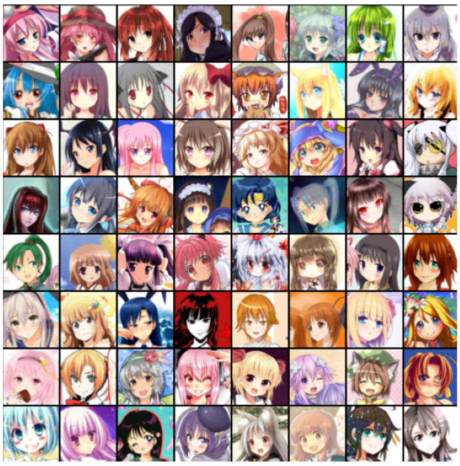
Deep Learning (Homework 3)

Due date : 2022/6/3 23:55:00 (Hard Deadline)

1 Generative adversarial network (50%)

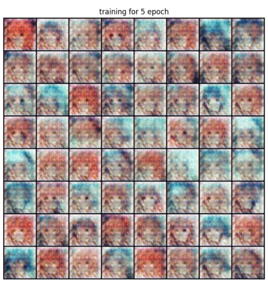
In this exercise, you will implement a Deep Convolutional Generative Network (DCGAN) [1] to synthesis images by using the provided **anime faces dataset**.

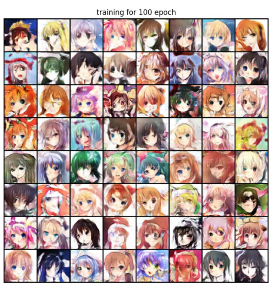


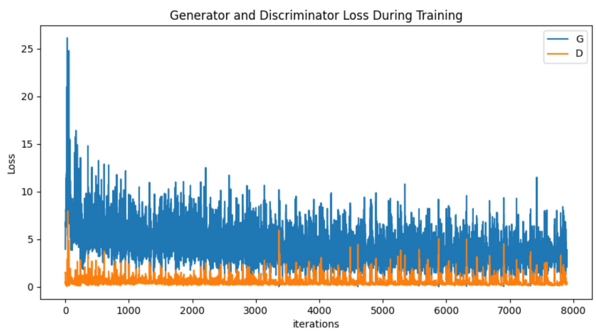
1. Construct a DCGAN with GAN objective, you can refer to the tutorial website provided by PyTorch for implementation.

$$\max_D \mathcal{L}(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{z\sim p_{\boldsymbol{z}}} \log(1 - D(G(\boldsymbol{z})))$$
$$\min_G \mathcal{L}(G) = \mathbb{E}_{z\sim p_x} \log(1 - D(G(\boldsymbol{z})))$$

   (a) **Describe** how you preprocess the dataset (such as resize, crop, rotate and flip) and **explain** why. (5%)
   (b) **Plot** the learning curves for both generator and discriminator. (15%)
   (c) **Draw** some samples generated from your generator at different training stages. For example, you may show the results when running at 5^{th} and final epoch. (10%)



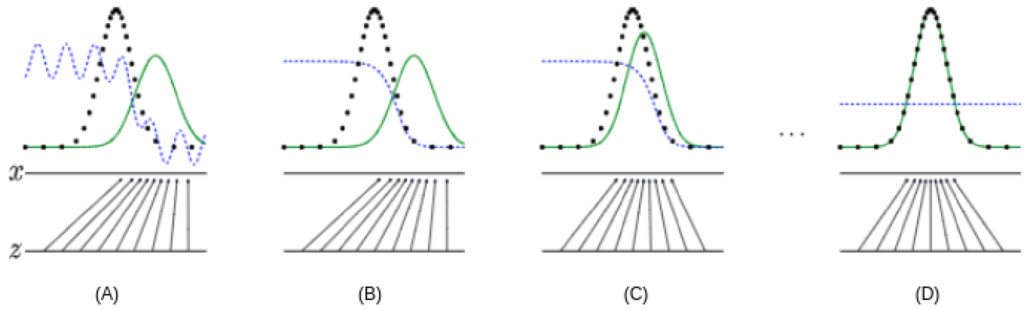




2. Please **answer the following questions** in your submission report, you can refer to the paper to answer these questions. (**Note**: If your answer is more complete and precise, you will receive a higher score.)

   (a) Please **describe** the meaning of the following four pictures during training of GAN, where blue dashed line indicates the discriminator, green solid line indicates the generator. The answer should include the following: (**Note**: Each step should be discussed.)(10%)

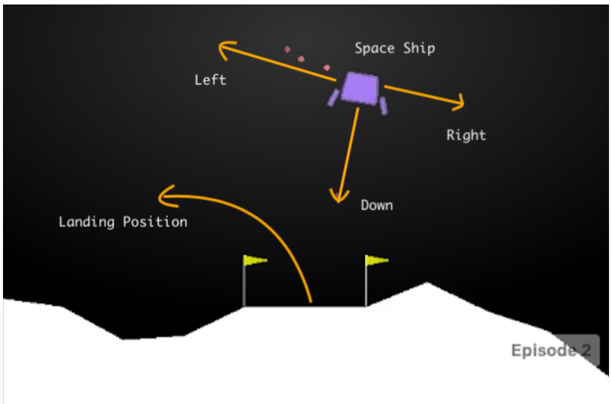
   - what is the meaning of black dashed line, x and z
   - which step is to train the generator or discriminator and show the corresponding objective function
   - why $D(x)$ equals to $\frac{1}{2}$ in ideal case when the training is finished



   (b) The Helvetica Scenario often happens during training procedure of GAN. Please **explain** why this problem occurs and how to avoid it. (5%)

   (c) Both VAE and GAN are generative models. The following figures are random generated results by using VAE (left) and GAN (right). Please compare two results and **describe** the pros and cons of two models. (5%) (**Hint**: You can compare the loss function and training method using these two models.)

2 Reinforcement Learning (50%)



In first section, you need to clearly know the meaning of state value $V_\pi(s)$ and state-action value $Q_\pi(s,a)$ in reinforcement learning. Then you will implement Deep Q Learning (DQN) [2] algorithm to approximate the actual Q value. In second section, you will design a stochastic policy and implement Proximal Policy Optimization (PPO) [3] algorithm to learn the RL agent. In addition, you also need to know how to collect and use trajectories from RL environment.

$$V_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s\right] = \sum_a \pi(a \mid s)\sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V_\pi\left(s'\right)\right]$$

$$Q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}\middle| S_t = s, A_t = a\right] = \sum_{s'}\sum_{r} p\left(s', r \mid s, a\right)\left[r + \gamma\sum_{a'}\pi\left(a' \mid s'\right)Q_\pi\left(s', a'\right)\right]$$

1. In this part, you need to implement the Deep Q Network algorithm to estimate the Q value on the openAI gym environment LunarLander-v2. After training, you can use the DQN agent to interact with the environment (choose the action which has the highest Q value).

   (a) Please follow the algorithm shown below and use ϵ-greedy policy to implement the DQN. (show the result movie) (35%)

   (b) Please choose some hyper-parameters about collecting and using trajectories and analyze how these hyper-parameters affect the training result or training time. (15%)

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1}\end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

$$
\begin{aligned}
\nabla_\theta J(\theta) &= E_{\tau \sim p_{\theta_{old}}(a|s)} \left[\frac{p_\theta(a \mid s)}{p_{\theta_{old}}(a \mid s)} \, r(s,a) \nabla_\theta \log p_\theta(a \mid s) \right] \\
&= E_{\tau \sim p_{\theta_{old}}(a|s)} \left[IS(a \mid s) A(s,a) \nabla_\theta \log p_\theta(a \mid s) \right] \\
where, \ &\textbf{Importance Sampling}: IS(a \mid s) = \frac{p_\theta(a \mid s)}{p_{\theta_{old}}(a \mid s)}, \ \textbf{Advantage}: A(s,a) = r(s,a) \\
&= E_{\tau \sim p_{\theta_{old}}(a|s)} \left[PPO(a,s) \nabla_\theta \log p_\theta(a \mid s) \right] \\
where, \ &PPO(a,s) = min \left\{ [IS(a \mid s) A(s,a)], [\textbf{clamp}(IS(a \mid s), 1-\beta, 1+\beta) A(s,a)] \right\} \\
&\beta = 0.2
\end{aligned}
$$

In Pytorch, you can simplify to use:

$$\textbf{Loss} = -J(\theta) = -PPO(a,s)$$

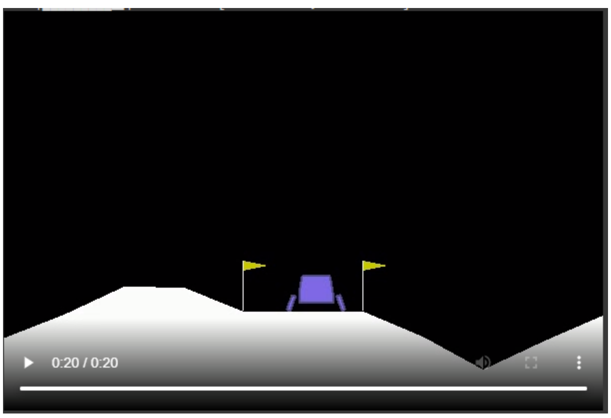
and use Loss.backward()

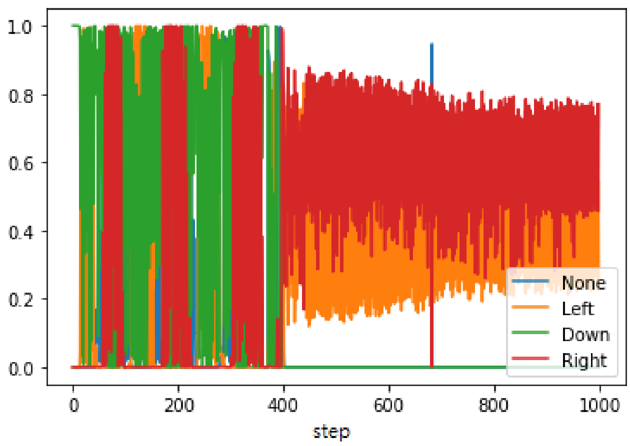
2. (Bonus) In this part, you need to implement the PPO algorithm in actor-critic style. In order to simplify the implementation steps, you can use fixed trained DQN model instead of training value function to calculate advantage $A(s,a)$ by follow algorithm:

   1. Collect $\{\mathbf{s}_i, \mathbf{a}_i\}$ from $\pi_\theta(\mathbf{a} \mid \mathbf{s})$ to buffer.
   2. Sample $\{\mathbf{s}_i, \mathbf{a}_i\}$ from the buffer.
   3. Evaluate $\hat{A}^\pi(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \hat{V}_\phi^\pi(\mathbf{s}_i') - \hat{V}_\phi^\pi(\mathbf{s}_i) = Q(a_i \mid s_i) - mean\{Q(a \mid s_i)\}$
   4. Calculate $PPO(a,s)$
   5. $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$ (by setting $\textbf{Loss} = -PPO(a,s)$)

   (a) Please use stochastic policy (categorical distribution)[1] and trained DQN model from previous part to train an agent in LunarLander-v2 environment.(show the result movie) (10%)

   (b) After training, please draw the probability-step picture for each action during testing and show below the result movie. (10%)





[1]Pytorch distributions module: https://pytorch.org/docs/stable/distributions.html

3 Rule

- In your submission, you need to submit two files. And only the following file format is accepted:
  - **hw3_<ProblemNumber>_<StudentID>.ipynb** file which need to contain all the results, codes and reports for each exercise (e.g. **hw3_1_0123456.ipynb**).
- Implementation will be graded by
  - Completeness
  - Algorithm correctness
  - Description of model design
  - Discussion and analysis
- Only Python implementation is acceptable.
- DO NOT PLAGIARISM. (We will check program similarity score.)

References

[1] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. of International Conference on Learning Representations*, 2016.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller, "Playing Atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.