Deep Learning (Homework 3)

Due date : 2021/6/4 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 celebrities' face dataset. You can download dataset **img_align_celeba.zip** which was collected from the origin website CelebFaces.

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

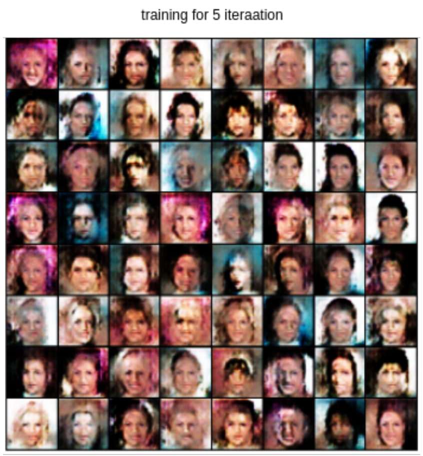
$$\max_D \mathcal{L}(D) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\boldsymbol{x}) + \mathbb{E}_{z\sim p_{\boldsymbol{z}}} \log(1 - D(G(\boldsymbol{z})))$$

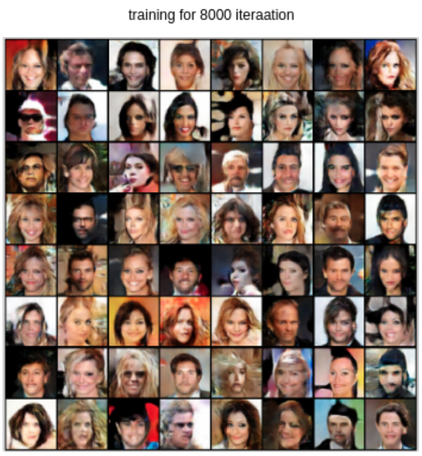
$$\min_G \mathcal{L}(G) = \mathbb{E}_{z\sim p_x} \log(1 - D(G(\boldsymbol{z})))$$

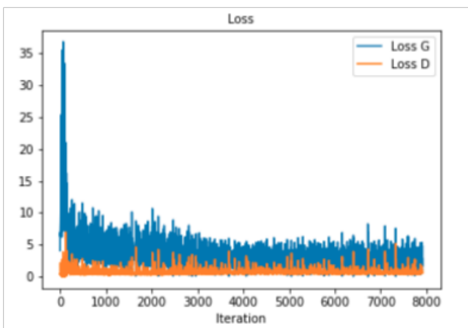
   (a) **Describe** how you preprocess the dataset (such as resize, crop, rotate and flip) and **explain** why. (5%)

   (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 learning iteration. (5%)



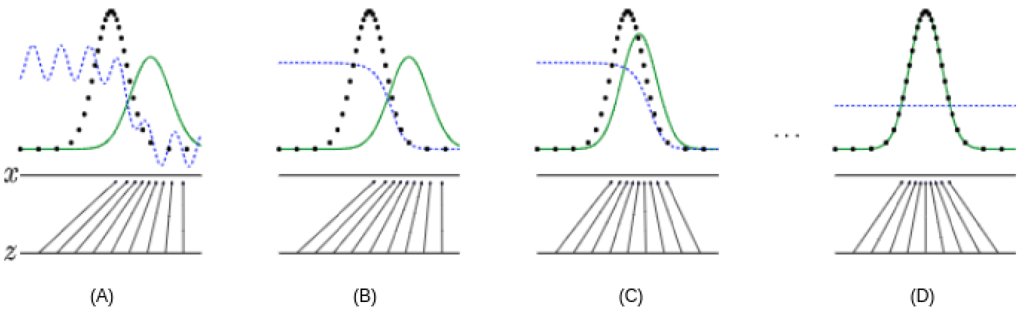




2. Please **answer** the following questions in your submission report, you can refer to the paper to answer these questions. (**Note**: If your answer is more complete and precise, you will receive a higher score.)

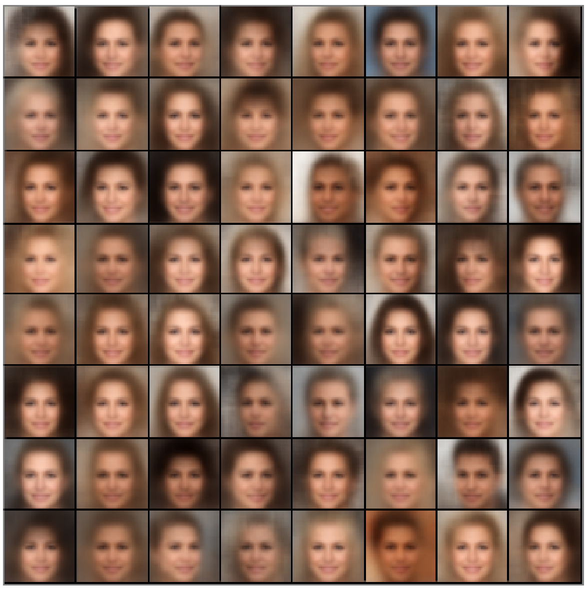
   (a) Please **describe** the meaning of the following four pictures during training of GAN, where blue dashed line indicates the discriminator, green solid line indicates the generator. The answer should include the following: (**Note**: Each step should be discussed.)

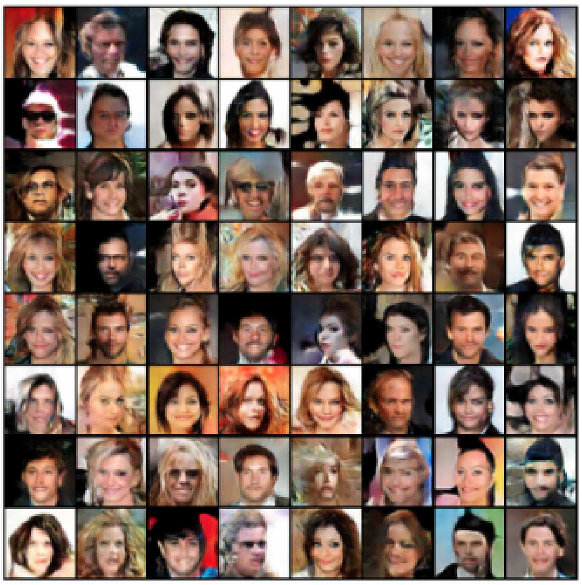
   - what is the meaning of black dashed line, x and z
   - which step is to train the generator or discriminator and show the corresponding objective function
   - why $D(x)$ equals to $\frac{1}{2}$ in ideal case when the training is finished



   (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. (10%) (**Hint**: You can compare the loss function and training method using these two models.)





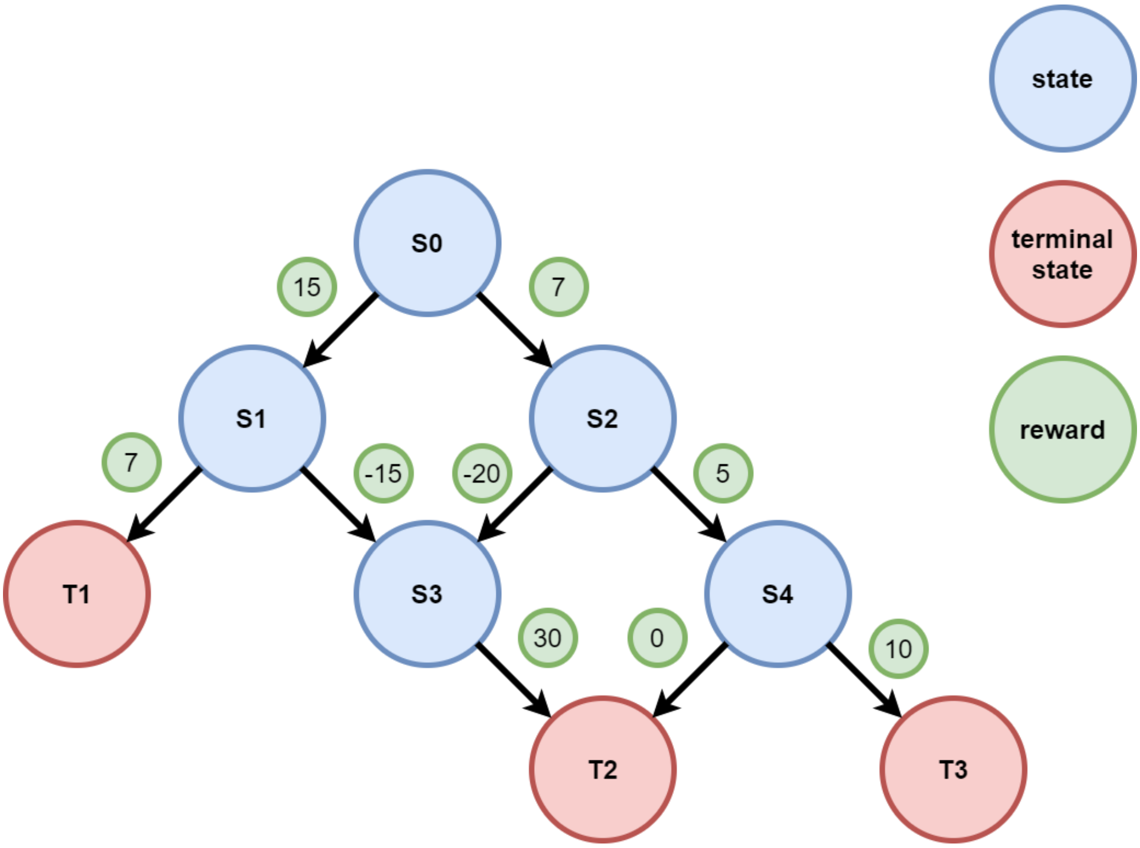
2 Deep Q Network (50%)

In this section, you need to clearly know the meaning of value function in reinforcement learning. You will calculate the actual state value $V_\pi(s)$ and state-action value $Q_\pi(s,a)$ under different policies π for each state-action pair from a given finite Markov Decision Process (MDP). Then you will implement Deep Q Learning (DQN) [2] algorithm to approximate the actual Q value. Finally, you will need to compare the difference between different methods and make some discussions.

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

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

1. Please calculate the state value $V_\pi(s)$ and the state-action value $Q_\pi(s,a)$ for state and state-action pair from the following finite MDP under the given policy π and the discount factor $\gamma = 0.9$:



   (a) Agent will use the uniform policy, which means that a_1 and a_2 will be evenly taken by the agent (8%)

   (b) Agent will use the specific policy which is denoted by $\pi = [P_{S0}, P_{S1}, P_{S2}, P_{S3}, P_{S4}] = [0,\ 0,\ 1,\ 0,\ 1]$ where each element represents the probability that the agent will take the action a_1, in the other word, one minus that probability denotes the probability that agent will take action a_2 (8%)

a_1

state	left	right
S0	0.2	0.8
S1	0.7	0.3
S2	0.3	0.7
S3	0	1
S4	0.2	0.8

a_2

state	left	right
S0	0.9	0.1
S1	0.2	0.8
S2	0.7	0.3
S3	0	1
S4	0.8	0.2

Note: For each policy, you need to calculate a 8×3 array to describe the value and Q value table. Then save the array into .npy file. For example, for the policy in (a) you need to name the file as value_a.npy

2. In this part, you need to implement the Deep Q Network algorithm to estimate the Q value from the previous finite MDP by using the DQN agent to interact with the environment.

   Note: You don't need to implement the environment by yourself. The environment file finite_MDP_env.py is provided by TA. In this environment, the state is defined as a one-hot vector. And the index mapping of each state is [S0, S1, S2, T1, S3, S4, T2, T3] = [0, 1, 2, 3, 4, 5, 6, 7].

   (a) Please follow the algorithm shown below to implement the DQN (10%)

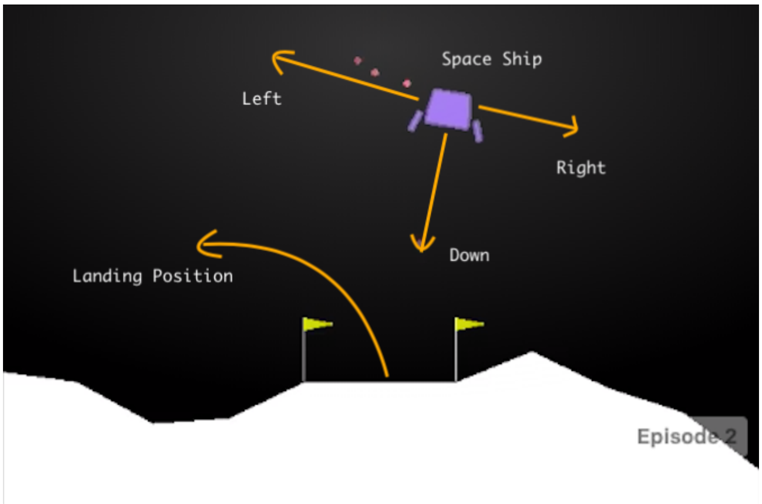
   (b) Please analyze the difference between your answers by the calculation and the output of the DQN (8%)

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

Note: In the implementation, you can check the provided format of DQN here to write your own program.

3. After the practice of DQN, now we can modify the input and the output sizes and the other parameters of the DQN to train on the openAI gym environment LunarLander-v2



(a) Please compare the difference between the total return curves of the previous finite MDP and the LunarLander-v2 environment. (8%)

(b) Please choose some hyper-parameters and analyze how these hyper-parameters affect the training result. (8%)

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.