Deep Learning (Homework 3)

Due date : 2020/6/19

1 Generative adversarial network (GAN)

In this exercise, you will implement a Deep Convolutional Generative Network (DCGAN) to synthesis images by using the provided celebrities' face dataset. You can download dataset imgalignceleba.zip which is collected from the origin website CelebFaces[1].

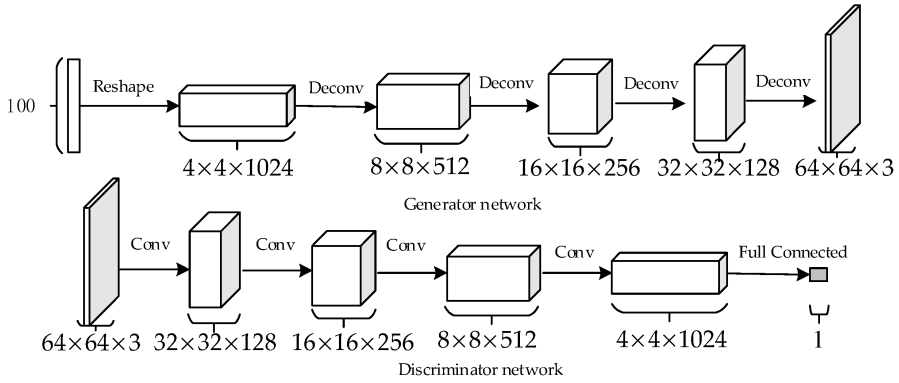


Figure 1: Structure of DCGAN

1. Data augmentation can be used to enhance GAN training. Describe how you preprocess the dataset (such as resize, crop, rotate and flip) and explain why.
2. Construct a DCGAN with vanilla GAN objective, plot the learning curves for both generator and discriminator, and draw some samples generated from your model.

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

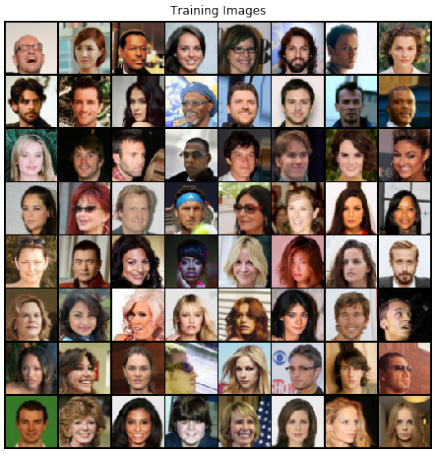


Figure 2: Samples drawn by using DCGAN

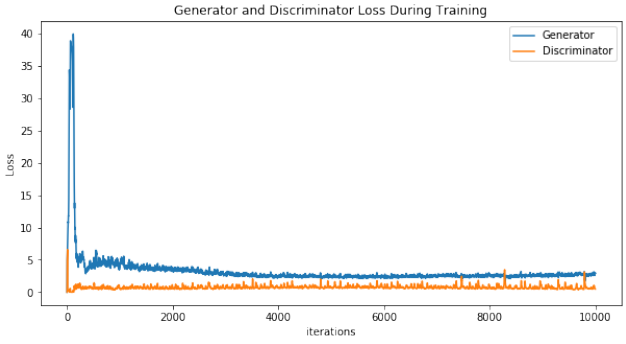


Figure 3: Learning curves of generator and discriminator in DCGAN

3. Implementation details are addressed as follows

   (a) Models has already been designed in Model.py. Feel free to modify the generator and discriminator, and you can write down how you design your model and why. (bonus 5 points)

   (b) In data preprocessing, the ImageFolder and Dataloader provided by pytorch are recommended. The customized dataset (without using ImageFolder) can be implemented for extra points. (bonus 5 points)

   (c) In main.py, you have to complete three functions. main(), train(), and visualizaion.

   - main() : you have to set up dataset (dataloader), models, optimizers, and the criterion. After preparation, the train function is called to start the training procedure.
   - train() : In every iteration, you have to perform the following jobs
     - send true data into the discriminator, and update the discriminator.
     - use generator to create fake data, and send them to the discriminator. Calculate loss for both models and update them.
     - record the loss in every iteration and draw some samples by the current generator after the fixed number of iterations.
     - after finishing all epochs, you have to save the files of your models, losses, and samples. (sampling should not be more than 20 times)

   (d) Visualization
   (If you don't use jupyter notebook, attach your visualization code in the folder and chart in report.)

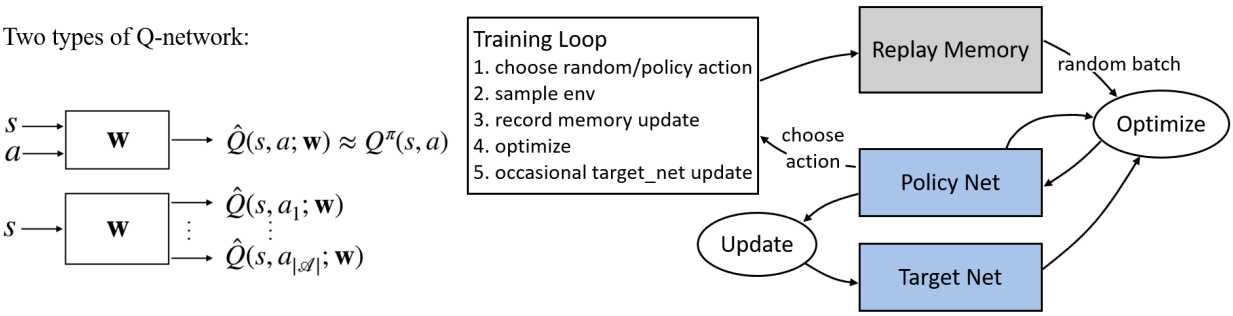
   - write down the visualization code in Visualization.ipynb, you can open it by using jupyter notebook.
   - plot the following two charts and show them in the report
     i. original samples and generated samples
     ii. learning curves for generator and discriminator
   - training procedure of GANs is unstable, when visualizing the loss curve you can do moving average every N steps (smooth the curve) to observe the trend easily.
   - we provided an extra visualizaion code in Visualization.ipynb, you can use it to see what does the generator generates through the training procedure.

   (e) Please do some discussion about your implementation. You can write down the difficulties you face in this homework, e.g. hyperparameter settings, analysis of the generated images, or anything you want to address.

2 Deep Q Network (DQN)

Deep neural networks are widely used to solve reinforcement learning problems. In this exercise, you will implement the Deep Q-Learning algorithm. Three components should be constructed.

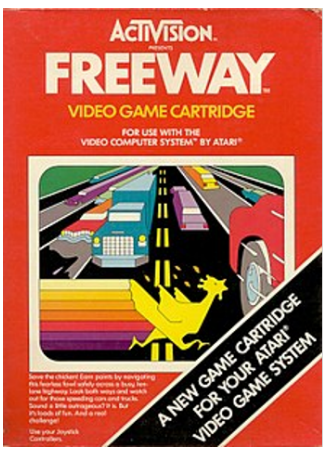
- Main component of this algorithm is the deep Q-networks. You can use them to approximate the state-action value.
- Experience replay buffer keeps the state transition information and removes the correlation in observation sequence.
- During the training phase, there is another target Q-network adopted to increase the training stability.



Environment:

It takes a lot of efforts to apply reinforcement learning in real world. Here, we use the classic Atari 2600 games as the simplified environment and try to maximize the score in the video game. We use the "FreewayDeterministic-v4" environment built in the OpenAI gym toolkit [2].

- Player could control the character across a ten lane highway filled with traffic and get to the other side [5].
- There are 2048 steps in one game episode in Freeway.
- The observation is an RGB image of the screen, which is an array of shape (210, 160, 3).
- There are 3 choices in the action space, which are NOOP (a0), UP (a1) and DOWN (a2). At each step, we should take one action in the environment.
- A reward 1 point will be awarded for reaching the other side.



Hint:

- Part of the source code is provided for your DQN implementation. The method similar to the DQN paper published in Nature [3].

Algorithm 1 Deep Q-learning with experience replay

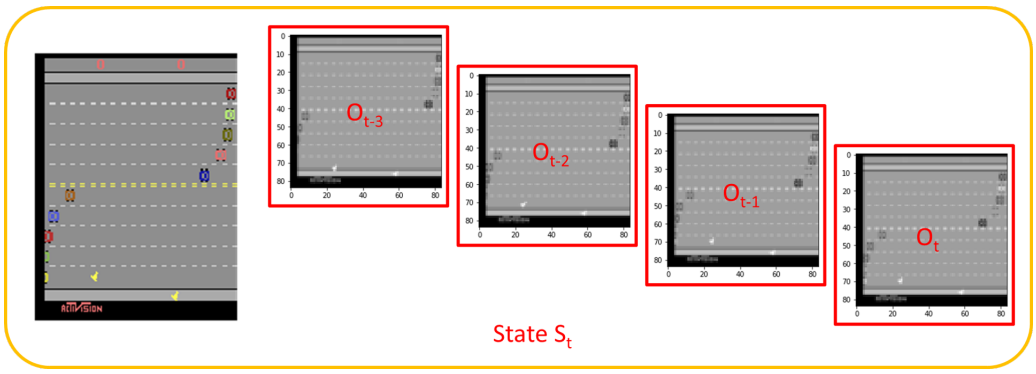
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^- = \theta$
for episode $= 1, \ldots, M$ **do**
  initialize sequence $S_1 = \{O_1\}$ and preprocessed sequence $\phi_1 = \phi(S_1)$
  for $t = 1, \ldots, T$ **do**
    with probability ϵ select a random action a_t
    otherwise select $A_t = \arg\max_a Q(\phi(S_t, a; \theta))$
    execute action A_t in emulator and observe reward R_t and new data O_{t+1}
    set $S_{t+1} = S_t, A_t, O_{t+1}$ and preprocess $\phi_{t+1} = \phi(S_{t+1})$
    store transition $(\phi_t, A_t, R_t, \phi_{t+1})$ in D
    sample random minibatch of transitions $(\phi_j, A_j, R_j, \phi_{j+1})$ from D

$$\text{set } y_j = \begin{cases} R_j, & \text{if episode terminates at step } j+1 \\ R_j + \gamma \max_{a'}(\hat{Q}(\phi_{j+1}, a'; \theta^-)), & \text{otherwise} \end{cases}$$

    perform a gradient descent step on $(y_j - Q(\phi_j, A_j; \theta))^2$ with respect to the network parameters θ
    every C steps reset $\hat{Q} = Q$
  end for
end for

You can use this code or implement DQN by yourself. The hyperparameter settings can be referred from the original paper.

- In order to obtain the time change information without violating the Markov assumption, we stack four luminance images to be the Markov state as shown below



- You can compare the reward with the score in the paper to judge the goodness of your result.

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Freeway	0	19.1	19.7	29.6	30.3 (±0.7)	102.4%

- You can use the computing resources given from NCTU or Google Colab [4] to help you finish the exercises.

Exercise:

1. Please indicate the code paragraph about the updating based on the temporal difference learning in your implementation or from the given source code

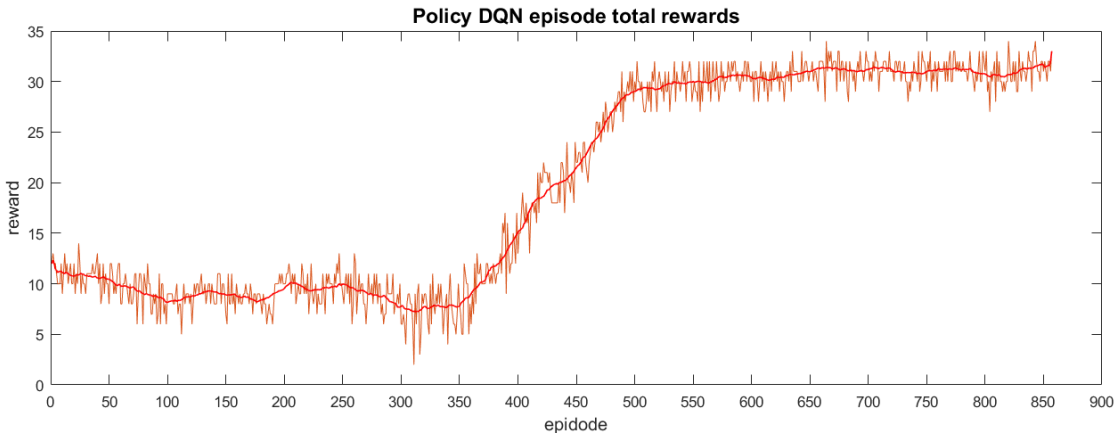
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \cdot \max_a Q(s_{t+1}, a_t) - Q(s_t, a_t)).$$

   Explain the purpose of the following hyperparameters: updating step α, discount factor γ, target network update period τ, and ϵ for ϵ-greedy policy.

2. For deep Q-Learning, exploring the environment is an important procedure. Because the reward signals in the given environment is very sparse, it takes a lot of computation time to explore the state space and perform the updating. To speed up the training process, you can simply change the probability of random agent: [NOOP (0.3), UP (0.6), DOWN (0.1)]. Please show the total reward of sample episodes for this configuration.

```
Episode:      0, interaction_steps:   2048, reward: 10, epislon: 1.000000
Episode:      1, interaction_steps:   4096, reward:  9, epislon: 1.000000
Episode:      2, interaction_steps:   6144, reward: 11, epislon: 1.000000
Episode:      3, interaction_steps:   8192, reward: 10, epislon: 1.000000
Episode:      4, interaction_steps:  10240, reward: 12, epislon: 1.000000
Episode:      5, interaction_steps:  12288, reward: 10, epislon: 1.000000
Episode:      6, interaction_steps:  14336, reward: 10, epislon: 1.000000
Episode:      7, interaction_steps:  16384, reward: 14, epislon: 1.000000
Episode:      8, interaction_steps:  18432, reward: 10, epislon: 1.000000
Episode:      9, interaction_steps:  20480, reward: 12, epislon: 1.000000
```

3. Use the modified random agent in the ϵ-greedy and keep training. Here, you should tune hyperparameters to let model converge. Plot the episode reward in learning time and evaluation time (ϵ = final epsilon) (2 charts). Show your configuration and discuss what you find in training phase.



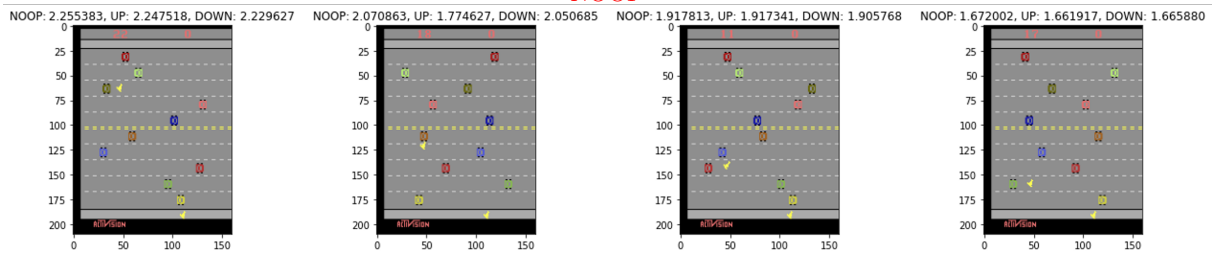
4. After training, you will obtain the model parameters for the agent. Show total reward in some episodes for deep Q-network agent.

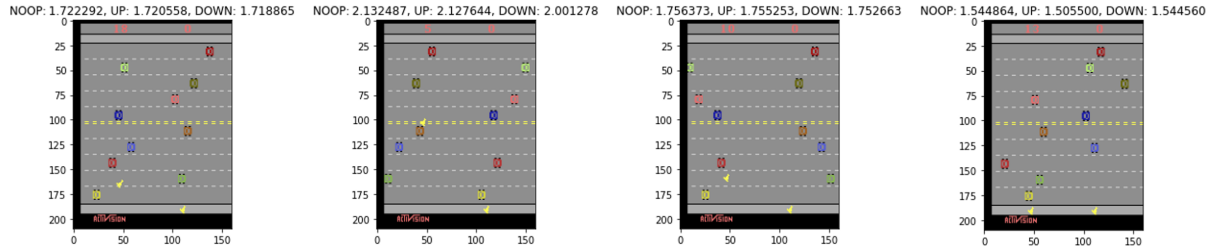
```
Use device: cuda
[Info] Restore model from './source_model/q_target_checkpoint_1538048.pth' !
Episode:      0, interaction_steps:      0, reward: 32, epislon: 0.100000
Episode:      1, interaction_steps:      0, reward: 30, epislon: 0.100000
Episode:      2, interaction_steps:      0, reward: 29, epislon: 0.100000
Episode:      3, interaction_steps:      0, reward: 31, epislon: 0.100000
Episode:      4, interaction_steps:      0, reward: 31, epislon: 0.100000
Episode:      5, interaction_steps:      0, reward: 32, epislon: 0.100000
Episode:      6, interaction_steps:      0, reward: 32, epislon: 0.100000
Episode:      7, interaction_steps:      0, reward: 32, epislon: 0.100000
Episode:      8, interaction_steps:      0, reward: 30, epislon: 0.100000
Episode:      9, interaction_steps:      0, reward: 31, epislon: 0.100000
```

5. Sample some states, show the Q values for each action, analyze the results, and answer

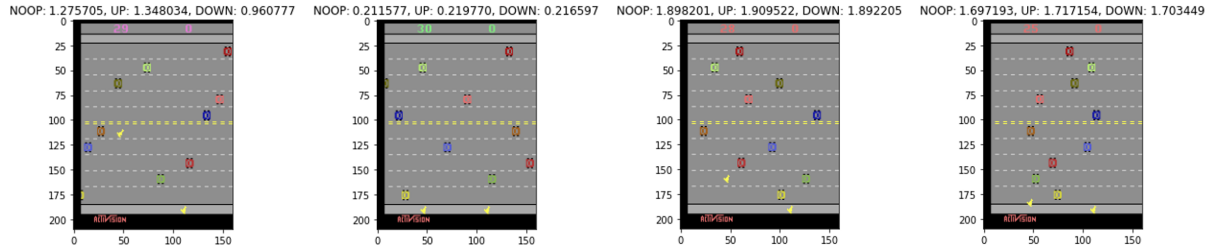
   - Is DQN decision in the game the same as yours? Any good or bad move?
   - Why the averaged Q-value of three actions in some state is larger or less than those of the other states?

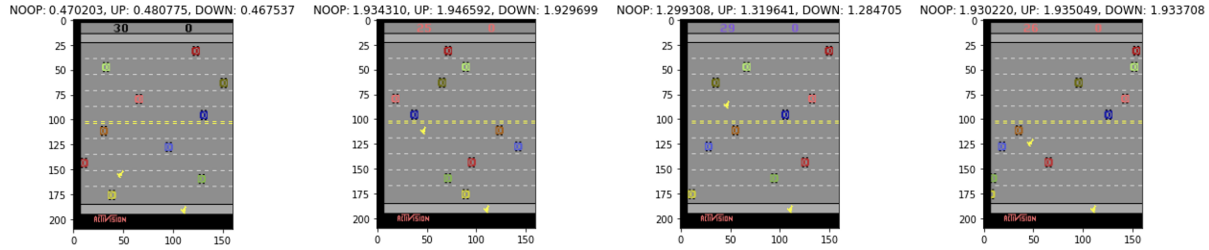
NOOP



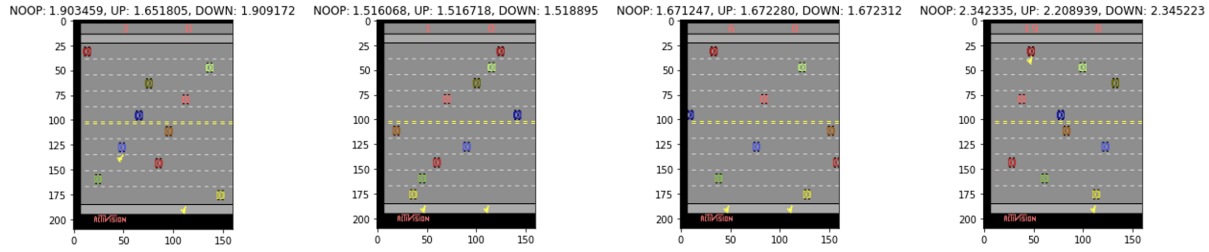


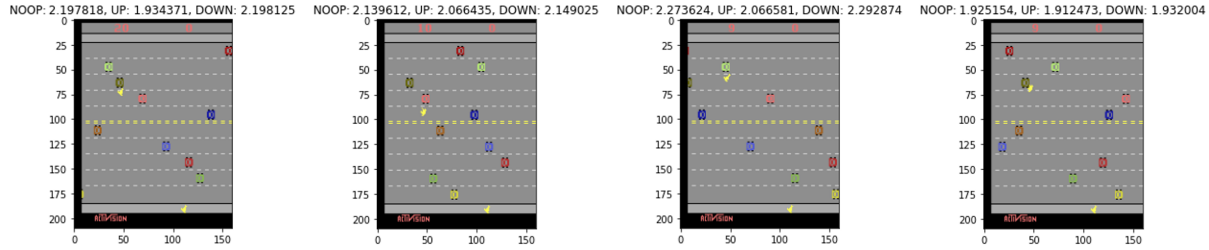
UP





DOWN





3 References

– [1] CelebA Dataset in the first problem is collected from http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

– [2] OpenAI Gym is a toolkit for reinforcement learning. You can refer to the installation guide at: https://gym.openai.com/docs/

– [3] DQN algorithm is cited from Human-level control through deep reinforcement learning (V Mnih, 2015).

– [4] Google Colab can provide some computational resources at https://colab.research.google.com/

– [5] Atari 2600 Freeway description is at https://en.wikipedia.org/wiki/Freeway_(video_game)