# ADL Hw3 Report
# Deep Reinforcement Learning

B06902104 吳由由

## 1 Models

- Policy Gradient Model
  - Architecture

$$action\_prob = softmax(L(Relu(Linear(observation\_state, 64)), action\_space))$$

    * $L$ is Linear layer
  - Make action
    When select action, using the state of game as input of the model and output the probability for each action. Select best action corresponds to the given state.
  - Update model parameter
    1. Calculate discounted reward
$$R_i = r_i + Gamma * R_{i+1}$$
    2. Normalize R
$$R_i = \frac{R_i - R_{mean}}{R_{std}}$$
    3. Calculate PG loss
$$loss = \sum_{i=1}^{n} -R_i * log(action\_prob)$$
- DQN Model
  - Architecture (online network and target network are in same architecture)

$$action\_prob = L(LeakyRelu(L(Relu(Conv(Relu(Conv(Relu(Conv(4, 32)), 64)), 64)), 512)), num\_action)$$

    * $L$ is Linear layer
  - Make action
    When select actions, use the frame of game as input state and pass the frame to model. The model then output the probability for each action. Select best action corresponds to the given frame state.
    * Calculate threshold for epsilon-greedy to decide whether to selcet a random action or not.

$$threshold = epsilon_{end} + (epsilon_{start} - epsilon_{end}) * exp(\frac{-1 * steps}{epsilon_{decay}})$$

      · $epsilon_{start} = 0.9$
      · $epsilon_{end} = 0.05$
      · $epsilon_{decay} = 100000$
    * In training mode, if I get random number less than threshold, I explore undone actions by random choose. Otherwise, I choose action with highest probability predicted by model.
  - Update model parameter
    1. Sample a batch of experience from training experience pool. Experience includes
       (a) State
       (b) Action
       (c) Reward
       (d) Next state

(e) If the state is the terminate state or not

2. Compute $Q(s_t, a)$ with online model. Use online model to get $Q(s_t, a)$ for each action and select the action $a$ token in game.
3. Compute $Q(s_{t+1}, a)$ with target net and select the action that can get max value.
4. Compute expected Q value for now.

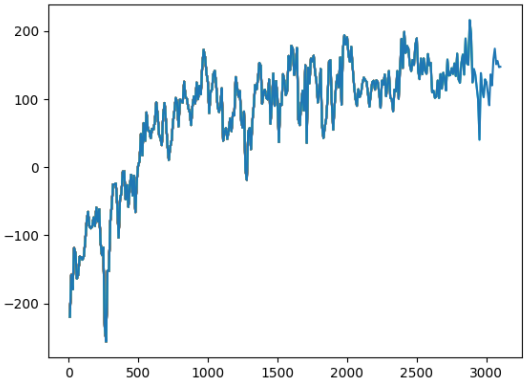
$$ExpectedQ = r_i + Gamma * \max_{a \in A} Q(s_{t+1}, a)$$

5. Compute temporal difference loss between expected Q value and real Q value

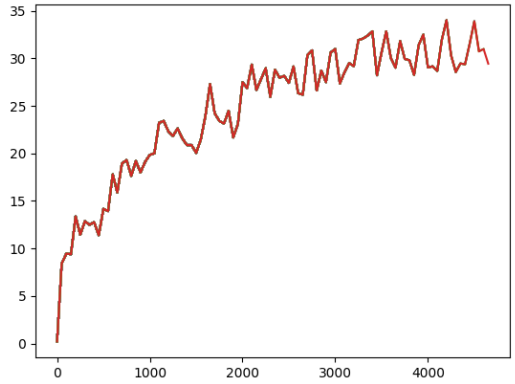
$$loss = (ExpectedQ - Q(s_t, a))^2$$

– Training process

1. Run 10000 steps before start training our network, use those steps to fill event buffer first.
2. When playing game, make action and store experience (state, action, reward, next state) in replay buffer.
3. update online network every 4 steps
4. update target network every 100 steps by copying online network parameter.




(a) policy gradient on Lunar Lander game




(b) DQN on Mspacman game

Figure 1: Learning Curves of Reward
Moving window size = 50 episode

## 2 Hyperparameters of DQN

- Hyperparameter choosed: Target network update frequency
- Update Strategy
  1. Update target network every step, blue line
  2. Update target network every 1000 step, orange line
  3. Update target network every 10000 step, green line
  4. Update target network every 100000 step, red line
- Target network are used to improve the stability of DQN training process. It fix weight of target network to freeze Q-value targets temporarily so we don't have a moving target to chase. This experiment aims to discuss reward got under different target network update frequency.
- Result
  1. When we update target network every step of training, our target Q-value is oscillating which is not stable. It's performance is not good even trained after thousands of episode.
  2. When we update target network every 1000 steps, it reaches highest reward among all other setting.
  3. When updating target network every 10000 or 100000 steps, the target network fixed period is too long so we do not get optimal Q-value when training. The performaces are relative worse as update period become longer.

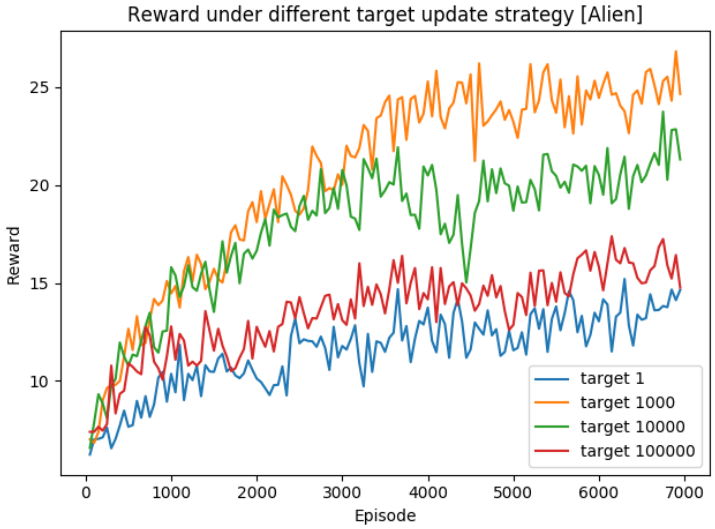



Figure 2: Reward under different target network update strategy on Atari Alien Game

# 3 Improvement of DQN

- DQN

$$\mathrm{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}[(r + \gamma \max_{a'} \hat{Q}(s', a', w^-) - Q(s, a, w))^2]$$

- Double DQN

$$\mathrm{L}(w) = \mathbb{E}_{s,a,r,s' \sim D}[(r + \gamma \hat{Q}(s', arg\max_{a'} Q(s', a', w), w^-) - Q(s, a, w))^2]$$

  The origin DQN uses the same values both to select and to evaluate an action, which makes it more likely to select overestimated values. It will result in overoptimistic values to be estimated. Double DQN can reduce this overestimations by decomposing the max operation in the target into action selection and action evaluation. Double DQN select best action for next state base on online net by greedily choose the action which gives max Q value. Then evaluate the Q value by target network. Reducing overestimation can significantly benefit the stability of learning.

- Duel DQN

$$Q(s, a) = V(s) + A(s, a)$$

  It seperate Q-network into two channels

  1. Action-independent value function $V(s)$ which estimate the value of ths state
  2. Action-dependent advantage function $A(s, a)$ which estimates the additional benefit

  This architecture can learn if the state is valuable or not without having to learn the effect of each action for each state. It is useful when actions do not affect environment.

- Improvement
  The experiment is done on Atari Alien Game, 4000 episode of games in tatal, with moving average window size 50.

  1. The green line is origin DQN, which is the baseline of this experiment.
  2. The blue line is Double DQN method. It can be observed that this method gives higher average reward than orgin DQN method after 1000 episodes.
  3. The orange line is duel DQN method. This architecture reaches siginificantly higher reward than the two other method after 500 episodes and converge faster than the other two method.

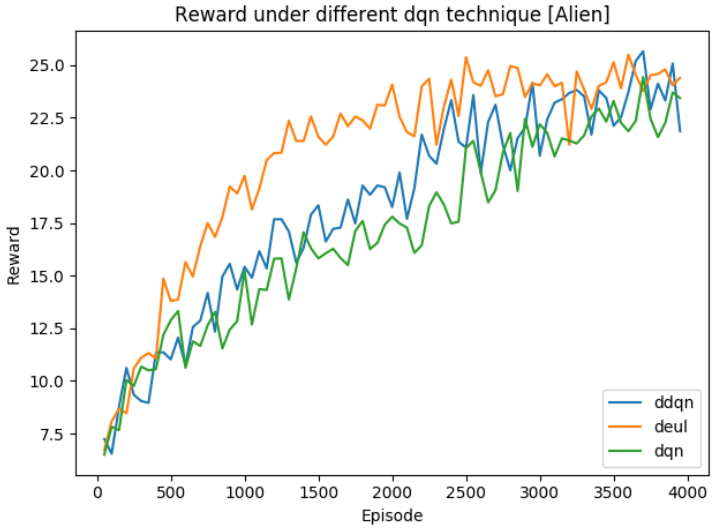



Figure 3: Learning curves on atari Alien game

# 4 Bonus: Fine-tuning Summarization

- RL algorithm
  - State: Decoder state $s_t$ with context vector $c_{t-1}$
  - Action: sample出一個word
  - 將Hw1的Attention Decoder 改成接受兩種deocde方式
    1. Greedy method: 找output vocabulary中機率最大的字
    2. Multinomial sampling: 使用torch.distributions.Categorical 從vocabulary 中sample出要decode的字, 並得到選擇此字的log probability
  - Training Process
    1. Objective, try to minimize

$$Ł_{RL} = -\mathbb{E}[rougeL(y_1, y_2, \ldots, y_T)]$$

    $y_1, y_2, \ldots, y_T$代表model生成的summary
    2. 將batch送入Attention model 中以greedy method decode, 得到的預測結果作為baseline, $baseline = rougeL(\hat{y_1}, \ldots, \hat{y_T})$
    3. 將同筆batch 送入Attention model中以multinomial sampling的方式得到結果, $sample = rougeL(y'_1, \ldots, y'_T)$
    4. Define reward, 使用rouge-l作為training reward

$$Reward = sample - baseline$$
$$Reward = rougeL(y'_1, \ldots, y'_T) - rougeL(\hat{y_1}, \ldots, \hat{y_T})$$

    This is self-critic policy gradient approch, model 使用greedy output作為baseline. Model會focus在比greedy selection更好的sample($Reward > 0$)，penalize比greedy還差的sample($Reward < 0$).
    5. 將$baseline$和ground truth算出cross entropy loss
    6. Cross Entropy Loss function

$$L_{crossEntropy} = -\sum_{t=1}^{T} logp^*_\theta(y_t | e(y_{t-1}), s_t, c_{t-1})$$

    where $e(.)$ is word embedding of token, $\theta$ is model parameters
    7. RL Loss function

$$Ł_{RL} = \sum_t logp^*_\theta(y_t | y'_{t-1}, s_t, c_{t-1}) \times Reward$$

    , which is equal to

$$Ł_{RL} = \sum_t logp(y_t | y'_{t-1}, s_t, c_{t-1}) \times (rougeL(y'_1, \ldots, y'_T) - rougeL(\hat{y_1}, \ldots, \hat{y_T}))$$

8. 實際上RL的loss function $L_{RL}$ 只有在model已經能表現的不錯時，才有機會sample到比baseline更好的結果，因此實際上訓練時的loss混合了以上兩種方法，以$\eta$ 調控混合兩種loss的比例, 使用cross entropy loss將model訓練到一個程度後再用RL loss fine tune

$$Loss = (1 - \eta)L_{crossEntropy} + \eta L_{RL}$$

- Result

| | RL ratio | rouge-1 | rouge-2 | rouge-l |
|---|---|---|---|---|
| Attention | 0 | 0.2555 | 0.0708 | 0.2093 |
| Attention with RL | 0.3 | 0.2544 | 0.0706 | 0.2106 |
| Attention with RL | 0.5 | 0.2548 | 0.0708 | 0.2104 |
| Attention with RL | 0.7 | **0.2583** | **0.0734** | **0.2134** |
| Attention with RL | 0.9 | 0.2498 | 0.066 | 0.206 |

I get best performance when RL ratio is set to 0.7, 表示使用rouge-L分數作為reward還是能slightly improve rouge performace在rouge-1 0.003%, rouge-2 0.003% rouge-l 0.005%. 此外在RL ratio設為0.3, 0.5時，雖然在rouge-1, rouge-2上的改善不明顯，但在rouge-l上還是有些微improve, 因為使用rouge-L作為reward的關係。當RL ratio設為0.9時可能因為reinforce learning loss的比例太大，而此model需要在已經train到一個程度後才比較有機會sample出比baseline好的結果，得到¿0的reward, 所以只用reinforce learning訓練時達到的結果沒有太好。

- Sample some example from both model
  - Attention model預測出的句子比較完整，readability 比RL model高
    - RL: a man has died in a suspected murder investigation after the death of a man in his flat
    - Attention: a man has been arrested on suspicion of murder after a man was found dead at a house in southampton.
  - RL model 預測出了和答案相同較長的句子很多遍，且出現比attention model更多在answer裡的字，但sentence quality和readability差
    - RL: the world anti doping agency wada is to be banned from the world anti doping laboratory and the world anti doping agency wada is to be banned from the world anti doping agency wada
    - Attention: russia has banned the country anti doping agency wada for failing to throw doping bans
    - answer: A drug testing laboratory in Qatar has been banned from carrying out its work for four months by the World Anti-Doping Agency (Wada).
- 可以發現由RL model predict出來的summary中平均預測summary的長度較attention model的長，可能是因為我選擇rouge-L作為reward，所以model預測出長一點的句子更有可能得到較高reward.