به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

درس شبکه‌های عصبی و یادگیری عمیق

تمرین اضافه

نام و نام خانوادگی	محسن فیاض
شماره دانشجویی	810100524
تاریخ ارسال گزارش	12 خرداد 1400

فهرست گزارش سوالات

سوال 1 – Lunar Lander **1**

الف) محیط Lunar Lander را مطالعه کرده و به صورت خالصه ویژگی های آن را شرح دهید. ویژگی های مد نظر عبارتند از مشخصات فضای حالت/ مشخصات فضای عمل/ سیستم پاداش 1

ب) عملکرد عامل را با رسم پاداش تجمعی در هر episode برای size batch های 32 ,64 و 128 بررسی کنید. تنها برای بهترین حالت به ازای episode های 50 ,100 ,150 ,200 و 250 فیلمی از عملکرد عامل تهیه کنید.

نکته: در صورتی که عملکرد عامل به ازای هر سه مقدار size batch مشابه یکدیگر شد، یکی از آن ها را به دلخواه به عنوان بهترین حالت انتخاب کنید. در رابطه به انتخاب بهترین حالت علاوه بر معیار سرعت همگرایی به پاداش بهینه معیار regret را نیز به صورت شهودی بررسی کنید. 2

ج) عملکرد مدل DQN و DDQN را با رسم پاداش تجمعی در هر episode و به ازای size batch برابر مقایسه کنید. برای هر دو مدل به ازای episode های 100 و 250 فیلمی از عملکرد مدل تهیه کنید.

نکته: هر بار آموزش عامل با استفاده از gpu های رایگان colab google بین 10-15 دقیقه زمان لازم خواهد داشت.

نکته: برای تهیه خروجی نمی توانید از سرویس colab google استفاده نمایید و می بایست checkpoint های مدل را دانلود کرده و روی سیستم خود فیلم ها را تهیه کنید. 6

سوال 1 – Lunar Lander

الف) محیط Lunar Lander را مطالعه کرده و به صورت خالصه ویژگی های آن را شرح دهید. ویژگی های مد نظر عبارتند از مشخصات فضای حالت/ مشخصات فضای عمل/ سیستم پاداش

محیط Lunar Lander شامل یک فرودگر است که باید در نقطه مشخص شده با ظرافت بنشیند. فضای حالت state شامل 8 مورد است که در زیر آمده است.

1. Horizontal Position موقعیت افقی
2. Vertical Position موقعیت عمودی
3. Horizontal Velocity سرعت افقی
4. Vertical Velocity سرعت عمودی
5. Angle زاویه
6. Angular Velocity سرعت زاویه‌ای
7. Left Leg Contact تماس بازوی چپ
8. Right Leg Contact تماس بازوی راست

و کنترل فرودگر که به ما سپرده شده است یعنی همان فضای عمل یا action شامل 4 مورد زیر است:

1. Do Nothing کاری نکند
2. Fire Main Engine روشن کردن موتور اصلی
3. Fire Left Engine روشن کردن موتور چپ (حرکت به راست)
4. Fire Right Engine روشن کردن موتور راست (حرکت به چپ)

از نظر پاداش ابتدا باید گفت که هر اپیزود وقتی تمام می‌شود که فرودگر تصادف کند یا فرود آید. تصادف -100 امتیاز و فرود موفق +100 امتیاز دارد. هر کدام از پایه های فرودگر اگر به زمین برسند +10 امتیاز دارد. روشن کردن موتور اصلی -0.3 امتیاز به ازای هر فریم دارد و روشن کردن موتورهای کناری -0.03 به ازای هر فریم. و در نهایت اگر مسئله حل شود +200 امتیاز دارد.

فرودگر از بالای صفحه همیشه شروع می‌کند ولی سرعت اولیه رندومی دارد و محل فرود همیشه 0و0 است.

ب) عملکرد عامل را با رسم پاداش تجمعی در هر episode برای size batch های 64, 32 و 128 بررسی کنید. تنها برای بهترین حالت به ازای episode های 50, 100, 150, 200 و 250 فیلمی از عملکرد عامل تهیه کنید.

نکته: در صورتی که عملکرد عامل به ازای هر سه مقدار size batch مشابه یکدیگر شد، یکی از آن ها را به دلخواه به عنوان بهترین حالت انتخاب کنید. در رابطه به انتخاب بهترین حالت علاوه بر معیار سرعت همگرایی به پاداش بهینه معیار regret را نیز به صورت شهودی بررسی کنید.

نتیجه بچ 32 در زیر آمده است: 26 دقیقه

Episode 25 Average Reward: -75.33 Epsilon: 0.47

Episode 50 Average Reward: 9.78 Epsilon: 0.22

Episode 75 Average Reward: 148.94 Epsilon: 0.10

Episode 100 Average Reward: 231.13 Epsilon: 0.05

Episode 125 Average Reward: 199.42 Epsilon: 0.02

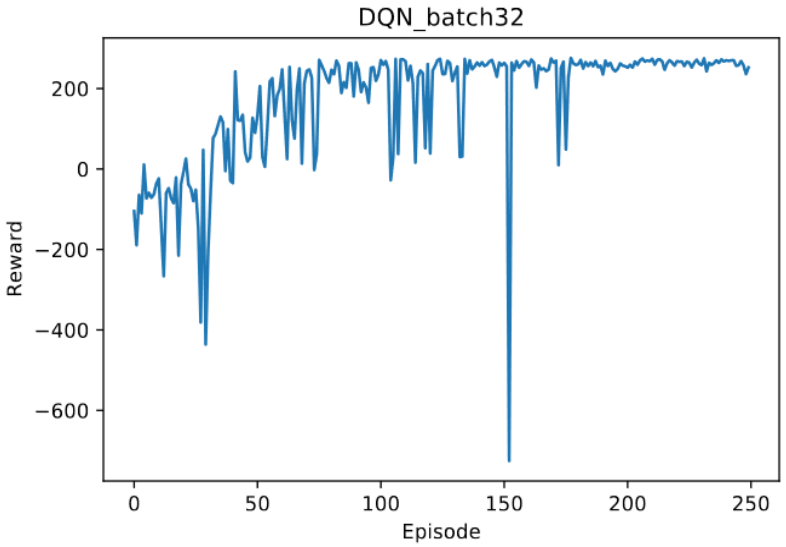
Episode 150 Average Reward: 236.35 Epsilon: 0.01

Episode 175 Average Reward: 207.80 Epsilon: 0.01

Episode 200 Average Reward: 248.65 Epsilon: 0.01

Episode 225 Average Reward: 264.41 Epsilon: 0.01

Episode 250 Average Reward: 262.38 Epsilon: 0.01



شکل 1 - پاداش تجمعی در طی اپیزودها با بچ سایز 32

نتیجه بچ 64 در زیر آمده است: 24 دقیقه

Episode 25 Average Reward: -80.90 Epsilon: 0.47

Episode 50 Average Reward: 67.15 Epsilon: 0.22

Episode 75 Average Reward: 206.00 Epsilon: 0.10

Episode 100 Average Reward: 215.65 Epsilon: 0.05

Episode 125 Average Reward: 262.25 Epsilon: 0.02

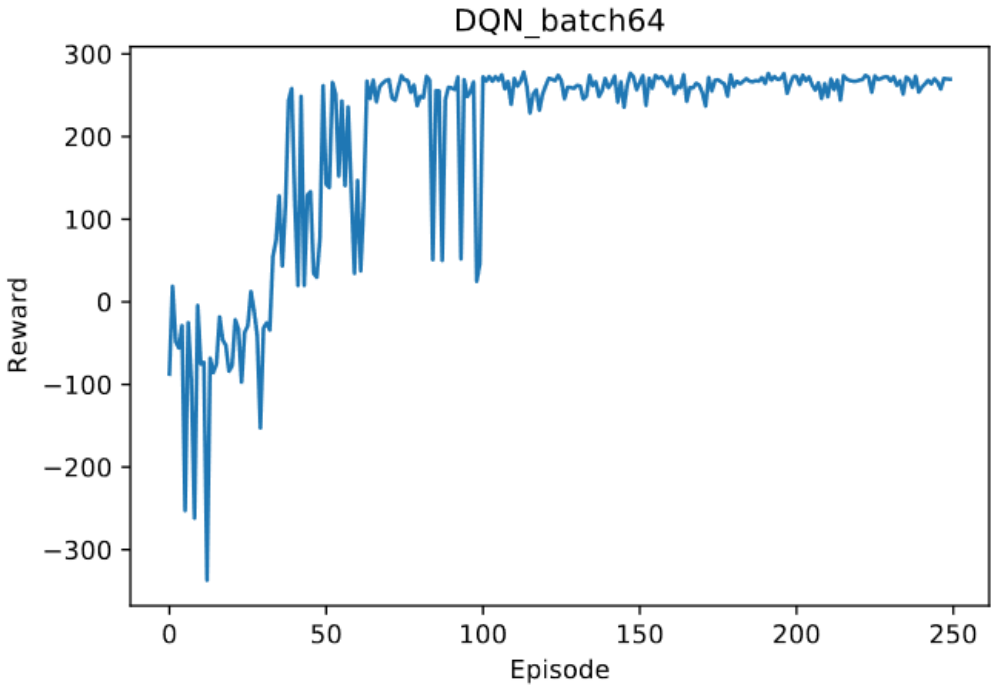
Episode 150 Average Reward: 259.72 Epsilon: 0.01

Episode 175 Average Reward: 261.68 Epsilon: 0.01

Episode 200 Average Reward: 267.01 Epsilon: 0.01

Episode 225 Average Reward: 264.49 Epsilon: 0.01

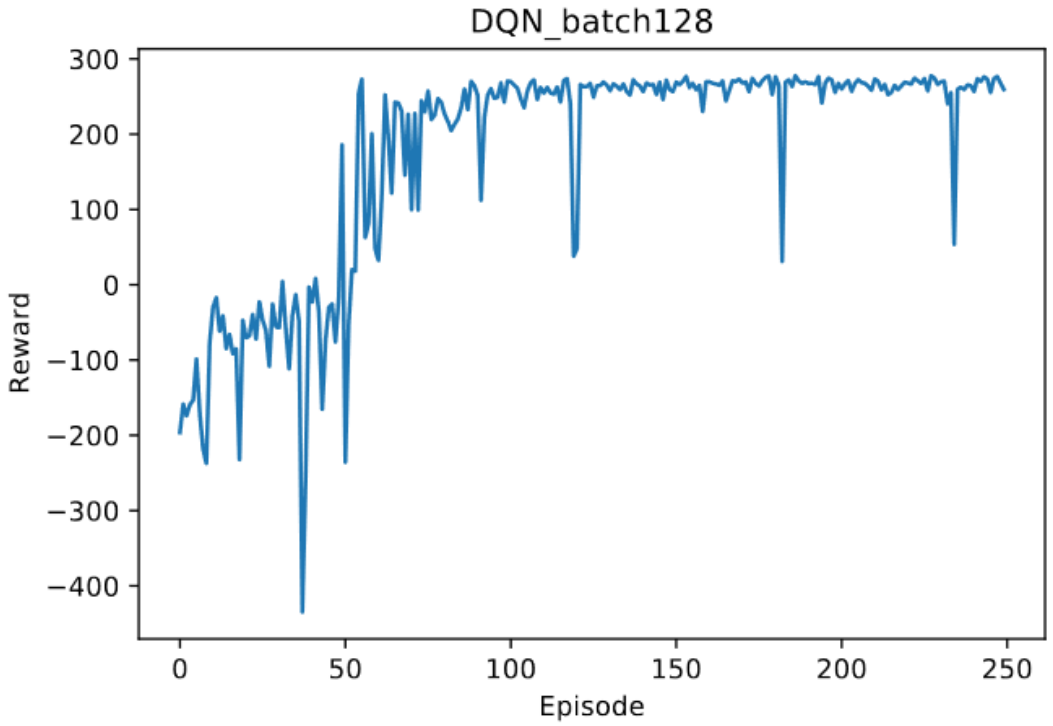
Episode 250 Average Reward: 266.51 Epsilon: 0.01



شکل 2 - پاداش تجمعی در طی اپیزودها با بچ سایز 64

نتیجه بچ 128 در زیر آمده است: 21 دقیقه

```
Episode 25   Average Reward: -106.90 Epsilon: 0.47
Episode 50   Average Reward: -62.81  Epsilon: 0.22
Episode 75   Average Reward: 135.10  Epsilon: 0.10
Episode 100  Average Reward: 236.46  Epsilon: 0.05
Episode 125  Average Reward: 241.89  Epsilon: 0.02
Episode 150  Average Reward: 262.01  Epsilon: 0.01
Episode 175  Average Reward: 265.04  Epsilon: 0.01
Episode 200  Average Reward: 258.39  Epsilon: 0.01
Episode 225  Average Reward: 265.40  Epsilon: 0.01
Episode 250  Average Reward: 257.35  Epsilon: 0.01
```

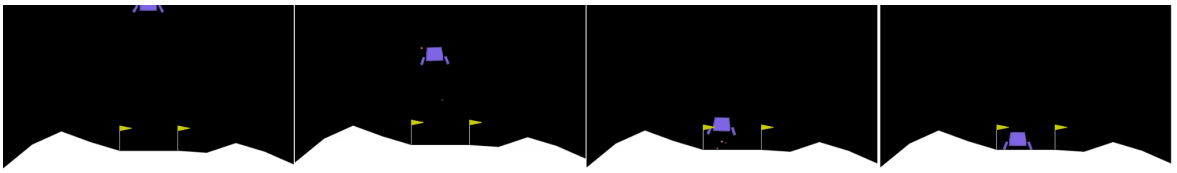


شکل 3 - پاداش تجمعی در طی اپیزودها با بچ سایز 128

برای مقایسه ابتدا به سرعت همگرایی می‌پردازیم، مشخص است که طبق نمودارها و همچنین میانگینی که هر 50 اپیاک چاپ می‌شد مشخصا بچ سایز 64 بهتر از بقیه عمل کرده است و سریعتر همگرا شده است. معیار regret یا پشیمانی به عنوان تفاوت بین پاداش یا reward یک

عمل احتمالی و پاداش عملی که واقعاً انجام شده است بیان می شود. یعنی می‌خواهیم هزینه فرصت باقی عمل ها را کمینه کنیم. البته در بازی‌های دو طرفه بهتر است همیشه بهترین را هم انتخاب نکنیم و به صورت احتمالی عمل کنیم تا policy برای رقیب مشخص نشود. این معیار را می‌توان اختلاف optimal reward و actual reward هم دانست که در اینجا باز هم بچ 64 توانسته بهترین پاداش ها را دریافت کند و طبق نمودار و اعداد، پشیمانی کمتری نسبت به دو حالت دیگر داشته است.

بنابراین 64 مناسب ترین است. البته فیلم تمام حالتها در کنار این فایل ارسال شده است، و 64 هم در اپیزودهای خواسته شده موجود است. به عنوان نمونه عکسهای زیر از فیلم 250 اپیزود برداشته شده است.



شکل 4 - فیلم فرود

ج) عملکرد مدل DQN و DDQN را با رسم پاداش تجمعی در هر episode و به ازای size batch برابر مقایسه کنید. برای هر دو مدل به ازای episode های 100 و 250 فیلمی از عملکرد مدل تهیه کنید.

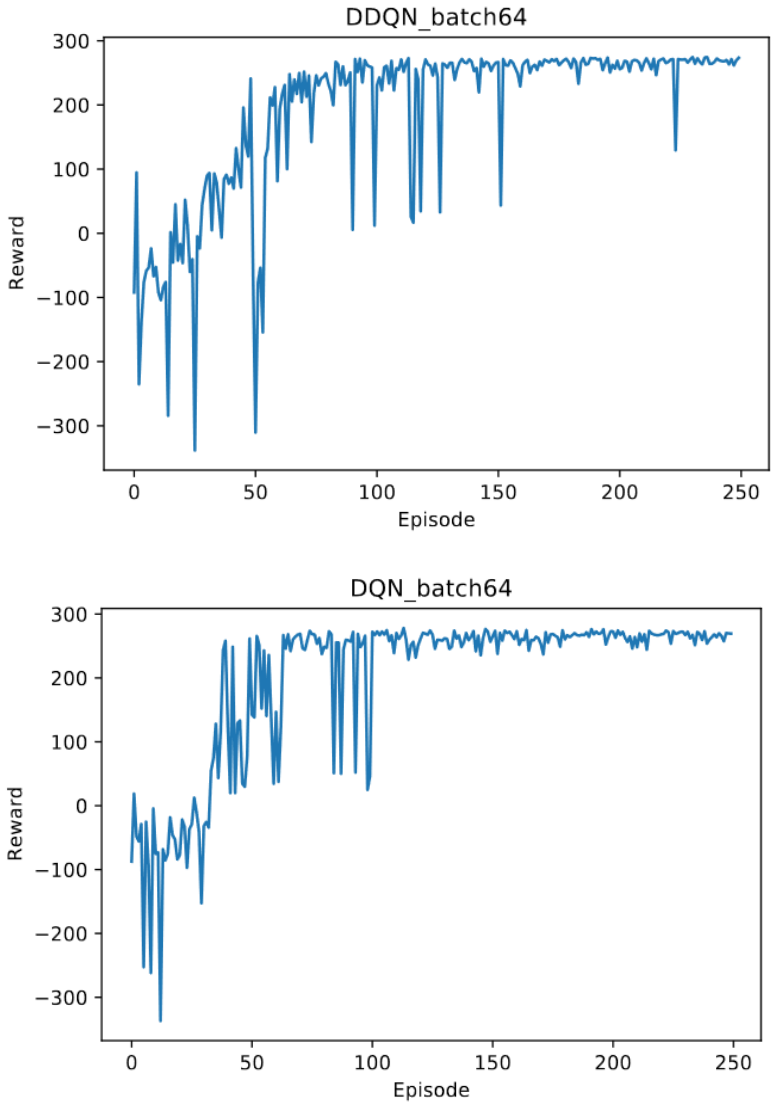
نکته: هر بار آموزش عامل با استفاده از gpu های رایگان colab google بین 10-15 دقیقه زمان لازم خواهد داشت.

نکته: برای تهیه خروجی نمی توانید از سرویس colab google استفاده نمایید و می بایست checkpoint های مدل را دانلود کرده و روی سیستم خود فیلم ها را تهیه کنید.

نتیجه مدل DDQN با بچ سایز 64 در زیر آمده است.

```
Episode 25   Average Reward: -59.39  Epsilon: 0.47
Episode 50   Average Reward: 59.12   Epsilon: 0.22
Episode 75   Average Reward: 142.09  Epsilon: 0.10
Episode 100  Average Reward: 228.12  Epsilon: 0.05
Episode 125  Average Reward: 225.84  Epsilon: 0.02
Episode 150  Average Reward: 250.73  Epsilon: 0.01
Episode 175  Average Reward: 253.84  Epsilon: 0.01
Episode 200  Average Reward: 264.92  Epsilon: 0.01
Episode 225  Average Reward: 259.71  Epsilon: 0.01
Episode 250  Average Reward: 268.91  Epsilon: 0.01
```

نمودار DDQN و DQN برای مقایسه در زیر آمده است.



شکل 5 - پاداش تجمعی در طی اپیزودها با بچ سایز 64 برای DDQN و DQN

DDQN یا Dueling Deep Q Networks یک الگوریتم یادگیری تقویتی است که سعی می کند یک مقدار Q را از طریق دو بخش که در این مسئله دو شبکه عصبی است ایجاد کند. یکی که تابع مزیت را تخمین می زند و دیگری که تابع ارزش را تخمین می زند. یکی از مشکلاتی که این کار قرار بود حل کند، وجود max بوده که باعث overestimation می‌شود. هیچ مدرک کاملی نظری یا تجربی وجود ندارد که نشان دهد Double DQN بهتر از DQN عادی است. کارهای مختلف زیادی وجود دارد، مقالات و آزمایشات بسیاری انجام شده است اما در انتها چیزی که می‌توان از آن برداشت کرد این است که در برخی از تسک‌ها DDQN بهتر است. (ضمنا باید اشاره کرد که در اینجا طبق اطلاعیه ای که زده شد، هر 5 حرکت به صورت سخت آپدیت انجام می‌دهیم که خودش اگر به صورت نرم انجام شود یا روی تعداد حرکت بهینه

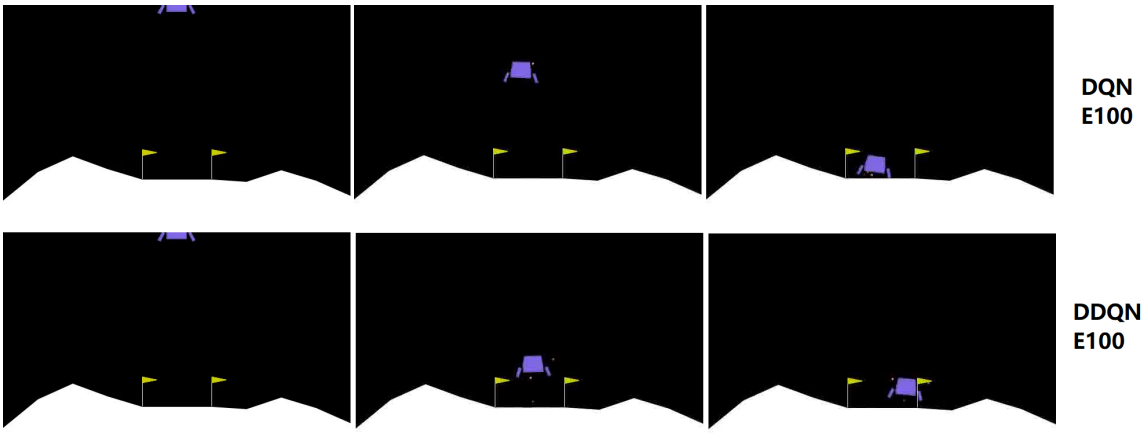
جستجو شود شاید می‌توانست نتایج بهتری دهد.)

در اینجا از نظر همگرایی هنوز DQN عادی بهتر عمل کرده است. اما از نظر پاداش نهایی، هر دو در یک محدوده هستند و طبق اعداد، DDQN توانسته کمی بهتر عمل کند.

از نظر فیلم، هر 50 اپیزود فیلم هر دو حالت در کنار فایلها قرار داده شده است.

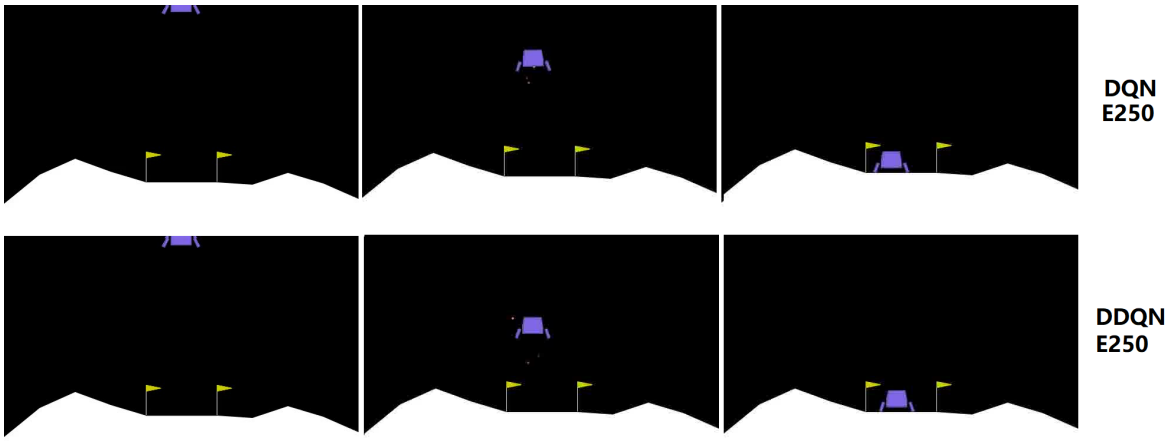
به طور خاص تصویر دو حالت 100 و 250 هم در اینجا می‌آوریم.

اپیزود 100:



شکل 6 - فیلم اپیزود 100

اپیزود 250:



شکل 6 - فیلم اپیزود 250

در مقایسه فیلمها اولا دیده می‌شود که اپیزود 250 فرود خیلی نرم تر و دقیق تر و سریعتری نسبت به 100 دارد و به نظر کمتر هم از موتورها استفاده می‌شود که طبیعی است. اما بین دو روش در 250 اپیزود تفاوت چشمی خاصی نیست و هر دو به خوبی یاد گرفته‌اند و فرود مناسبی را هر دو دارند.

در آخر باید اشاره کنیم که همانطور که در اطلاعیه ها آمد از 4 لایه پنهان و سایز 512 نورون استفاده کردیم و ضمنا برای جلوگیری از overfitting احتمالی از لایه dropout که درصدی از نورونها را به صورت رندوم خاموش می‌کند استفاده کردیم و همچنین از لایه LayerNorm که فضای فیچر بین لایه ها را نرمال می‌کند بهره بردیم که در مقاله‌اش تاثیرات مثبتش ذکر شده است. بخشهای مورد نیاز کد نیز با توجه به DQN و DDQN نوشته شده است.

Basic Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

Double Q-Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(R_{t+1} + \gamma \boxed{Q'(s_{t+1}, \boxed{a})} - Q(s_t, a_t))$$

estimated/expected Q-value

$$\boxed{a} = \max_a Q(s_{t+1}, a)$$

$$q_{estimated} = \boxed{Q'(s_{t+1}, \boxed{a})}$$

شکل 7 - DQN و DDQN

بخشها پر شده کد در فایلی که در کنار این گزارش پیوست شده ارسال شده است.