در ابتدا، ماژول‌های مورد نیاز برای اجرای کد و تعریف پارامترهای مهم مانند تعداد قسمت‌ها (episodes)و تعداد مراحل در هر قسمت (iterations) و اندازه دسته‌های آموزش (batch size) تعریف شده‌اند. سپس، محیط Gym بازی CartPole-v1 ایجاد می‌شود.

سپس، یک کلاس به نام DQN تعریف شده است. این کلاس شامل توابع و متغیرهای مربوط به شبکه عصبی و الگوریتم DQN است. در تابع سازنده، هیپرپارامترها و متغیرهای مورد نیاز برای الگوریتم تعریف شده‌اند و شبکه عصبی ساخته شده است. همچنین توابع دیگری مانند تابع epsilon_greedy برای انتخاب عمل با استفاده از استراتژی epsilon-greedy و توابع store و train_net برای ذخیره و آموزش تجربه‌ها در حافظه بازی (replay memory) تعریف شده‌اند.

در بخش اصلی برنامه، یک شی از کلاس DQN ایجاد شده و حلقه اصلی آموزش اجرا می‌شود. در هر قسمت، محیط تنظیم می‌شود و حلقه زمان در هر قسمت اجرا می‌شود. در هر گام زمان، عمل با استفاده از استراتژی epsilon-greedy انتخاب می‌شود و بازده، وضعیت بعدی و اطلاعات دیگر محاسبه می‌شوند. سپس تجربه برای استفاده در آموزش ذخیره می‌شود و در صورتی که تعداد تجربه‌ها بیشتر از اندازه دسته آموزش باشد، شبکه عصبی آموزش داده می‌شود. در نهایت، نتایج آموزش و مدل نهایی در فایل‌های جداگانه ذخیره می‌شوند.

در ابتدا، ماژول‌ها و کتابخانه‌های مورد نیاز برای اجرای کد و تعریف پارامترهای مهم مانند تعداد قسمت‌ها (episodes) و تعداد مراحل در هر قسمت (iterations) و اندازه دسته‌های آموزش (batch size)استفاده شده‌اند. همچنین محیط Gym بازی CartPole-v1 ایجاد شده است.

سپس، یک کلاس به نام DQN تعریف شده است که شامل توابع و متغیرهای مربوط به شبکه عصبی و الگوریتم DQN است. در تابع سازنده، هیپرپارامترها و متغیرهای مورد نیاز برای الگوریتم تعریف شده‌اند و شبکه عصبی ساخته شده است. همچنین توابع دیگری مانند تابع epsilon_greedy برای انتخاب عمل با استفاده از استراتژی epsilon-greedy و توابع store و train_net برای ذخیره و آموزش تجربه‌ها در حافظه بازی (replay memory) تعریف شده‌اند.

در بخش اصلی برنامه، یک شی از کلاس DQN ایجاد شده و حلقه اصلی آموزش اجرا می‌شود. در هر قسمت(episode) ، محیط تنظیم می‌شود و حلقه زمان در هر قسمت اجرا می‌شود. در هر گام زمان، عمل با استفاده از استراتژی epsilon-greedy انتخاب می‌شود و بازده، وضعیت بعدی و اطلاعات دیگر محاسبه می‌شوند. سپس تجربه برای استفاده در آموزش ذخیره می‌شود و در صورتی که تعداد تجربه‌ها بیشتر از اندازه دسته آموزش باشد، شبکه عصبی آموزش داده می‌شود. در نهایت، نتایج آموزش و مدل نهایی در فایل‌های جداگانه ذخیره می‌شوند.

نتایج آموزش در یک فایل CSV با نام "dqn_training_results.csv" ذخیره می‌شود و مدل نهایی در فایل "dqn_model.h5" ذخیره می‌شود.

test_trained_modle.py

در این قسمت ابتدا ماژول‌ها و کتابخانه‌های مورد نیاز برای اجرای کد تعریف شده‌اند. سپس مدل آموزش دیده شده با استفاده از تابع `load_model` از کتابخانه Keras بارگذاری می‌شود. توجه داشته باشید که تابع `mean_squared_error` به عنوان تابع سفارشی برای محاسبه خطا در آموزش مدل استفاده شده است.

سپس، محیط Gym بازی CartPole با حالت رندرینگ ایجاد می‌شود. تابع `epsilon_greedy` تعریف شده است که با دریافت وضعیت فعلی و مدل، عمل را با استفاده از استراتژی epsilon-greedy انتخاب می‌کند.

تابع `test_model` تعریف شده است که با دریافت مدل و تعداد قسمت‌های مورد نیاز، مدل را بر روی محیط تست می‌کند. در هر قسمت، محیط تنظیم می‌شود و حلقه زمان در هر قسمت اجرا می‌شود. در هر گام زمان، محیط نمایش داده می‌شود و با استفاده از تابع `epsilon_greedy` عمل بعدی انتخاب می‌شود. سپس وضعیت بعدی، پاداش و وضعیت پایانی محاسبه می‌شوند و نمره کلی قسمت به روزرسانی می‌شود. در صورت پایان قسمت، نتیجه قسمت چاپ می‌شود.

در آخرین بخش، تابع `test_model` بر روی مدل آموزش دیده شده فراخوانی می‌شود و محیط رندرینگ بسته می‌شود.