Assignment 4 - Reinforcement Learning with TF-Agents

Machine Learning Engineering Practicum - Fall AY '23-'24

1 Your Challenge

Your mission is to train machine learning algorithms to play an Atari video game, other than Breakout, of your choosing using the tf_agents python package. I will rank your submission by combining several factors:

1. The performance your agent achieves in playing the games, as roughly determined by watching your network play.
2. The level of relative difficulty of the game you select as measured by its position in Figure 3 of `https://homl.info/dqn2`.
3. You will use the checkpointer and other code you develop to assess the progress of the models' training, by assessing their present performance at three different amounts of cumulative experience playing the game and training that you select. These three cumulative amounts of experience should be selected to demonstrate improvement during the training process - poor, intermediate, and best game play, respectively to show that your training parameters were well-selected.

Additionally, I will asses whether your submission is compliant with "what you must submit" below, whether your relevant code files described therein run without errors, and whether each of your videos seems long enough to assess the effectiveness of the agent's game play, and if your sequence of poor, intermediate and best game play demonstrate visual improvements in playing the came.

2 Training and Testing your Learning Agent

You will use a game simulator from the open AI gym to train and test your algorithm. The *Breakout* example from Chapter 18 of version 2 of the text and its notebook `https://github.com/ageron/handson-ml2/blob/master/18_reinforcement_learning.ipynb` may be helpful.

3 What you must submit

You will upload to bblearn a file assignment4.zip containing the following files

1. buildAndTrainAgent-GAME.py – Here, GAME is replaced by the game you selected, and there should be two of these files. This code will build and train your model for the associated game, periodically saving checkpoints and associated videos of your agent playing the game. Save 10 different times during the training process, and be sure to
   (a) save enough that the agent you learned can be deployed elsewhere, and
   (b) save the model and checkpoint information to enable resuming training later.
2. resumeTraining.py – Code which demonstrates loading your final saved checkpoint model and resuming training.
3. deployGamePlayer.py – Code which demonstrates loading your saved policy and playing the game using it, creating and saving a video while doing so. This should take an argument that specifies the path to the policy/model/checkpoint to use.
4. createTrainingCurvesAndVideos.py – Code which loads the 10 checkpoints associated with different culumative amounts of gameplay experience and generates videos of that agent playing the game.
5. Based on your own visual inspection, subselect three of the videos created in the previous item, that, in order of cumulative experience indicate poor, intermediate, and best game play quality. Save these as myAgentPlays-GAME-QUAL.gif or myAgentPlays-GAME-QUAL.mp4 – where GAME is replaced by the name of the game selected and QUAL is taken from (poor,intermediate,best). Take care that sufficient time is recorded to visually assess game play quality.
6. savedAgentInfo.zip or savedAgentInfo.tgz – Archive containing the necessary files and directories for the final checkpoint, which, when decompressed enables resumeTraining.py and deployGamePlayer.py to work properly.