CPSC 481 Project Three: Go bot engine

due Wednesday, 13 Nov at beginning of class

Your name TitanGO

Repository: https://github.com/bhdang95/TitanGO /

Finished	Not Finished	Verify each of the following items and place a checkmark in the correct column. Each item incorrectly marked will incur a 5% penalty on the assignment's grade.
☒	☐	Created an end-to-end application (DeepLearningAgent) to train/run a Go bot (ch 8.1)
☒	☐	Used the web interface and flask server to play your Go bot using an attractive UI (ch 8.2)
☒	☐	Have created an AWS account (one for yourself and one for your bot) to allow training of your bot and deploying it. (ch. 8.3 and Appendix C)
☒	☐	Have installed **gnugo** as a LOCAL GTP server using the Go Text Protocol. The server can be run using a user interface such as Sabaki, Lizzie, GoRilla, or q5Go (Appendix C).
☒	☐	Create a web application using a Flask server that allows you to play against your bot using a browser to: localhost:5000/static/play_random_99.html . The browser will show a traditional (graphic) view of a Go game, with black and white stones on a wooden board (ch 8) (pp. 229-30).
☒	☐	Created a GTP frontend for your bot (chs. 8.4 and 8.5)
☒	☐	Your bot can play against two other local Go bots (gnugo and pachi). Gnugo has strength 12 kyu. Pachi has strength 2d to 7d, depending on the strength of the computer running it.
☒	☐	Your bot has been deployed on the OGS (online Go Server) platform (Appendix E)
☒	☐	Make a self-improving Deep Learning agent using Reinforcement learning, collecting experience data by playing copies of itself. (ch. 9)
☒	☐	Made a self-improving Deep Learning agent that uses Keras to develop its policy gradient algorithm (ch. 10).
☒	☐	Made a self-improving Deep Learning agent with the Q-learning algorithm (ch. 11)
☒	☐	Made a self-improving Deep Learning agent with the actor-critic method (based on advantage: A = R – V(s), where R is an estimate of the action-value method Q(s, a). (ch. 12)
☒	☐	Create a 48 plane board encoder, to make your Go bot more powerful.
☒	☐	Create TWO deep CNN policy networks for move prediction – one for more accurate results, and the other for faster evaluation (ch. 13)
☒	☐	Use the strong self-play CNN policy network to build your self-play value network.
☒	☐	Use the fast self-play CNN policy network to guide your tree-search algorithm.
☒	☐	Train a value network using the AlphaGo board encoder, and by having the Go bot play itself
☒	☐	Improve your MCTS rollout policy to use your policy network to guide rollouts, instead of just making moves at random
☒	☐	Winning percentage against other Go bot engines gets you into the top six in the class.
☒	☐	Winning percentage against other Go bot engines gets your bot into the top three.

☒	❑	Train your Go bot using different hyper-parameters to get best performance.
☒	❑	Be written in Python. No issues are shown in PyCharm (all source code screens shown a green checkmark at the top right hand corner).
☒	❑	Project directory pushed to new GitHub repository listed above
Comments:		