Test Plan and Results

Part I. Description of Overall Test Plan

There are three different types of tests that I run on my code. First, I test that individual components work as expected. For example, I might test that the image preprocessing works with particular images or that the DQN agent produces a valid action when given a valid state. Second, I test that the code gets the expected results on simpler environments. For example, I might test that my DQN implementation get the expected performance on the CartPole environment or that my CQL implementation matches the expected performance on the simple maze environments. Third, I test that my implementations get correct performance on the more difficult environments that I plan to test on. For example, I might check that offline DQN and CQL perform as expected on Atari Breakout.

Part II. Test Case Descriptions

IP1.1	**Image Preprocessing Test 1**
IP1.2	This will test that the image preprocessing for Atari works as expected.
IP1.3	The image preprocessor is given a collection of images to preprocess. We need to check that they are grey-scaled and down-sampled.
IP1.4	A collection of unprocessed Atari frames.
IP1.5	The processes Atari frames.
IP1.6	Normal
IP1.7	Whitebox
IP1.8	Functional
IP1.9	Unit

DQN1.1	**Deep Q-Learning Test 1**
DQN1.2	This will test that DQN produces a valid action when given a preprocessed frame.
DQN1.3	The DQN agent, represented as a neural network, is given a preprocessed frame, and is expected to produce a valid action.
DQN1.4	A collection of processed Atari frames.
DQN1.5	A collection of valid actions corresponding to the input frames.
DQN1.6	Normal
DQN1.7	Whitebox
DQN1.8	Functional
DQN1.9	Unit

DQN2.1	**Deep Q-Learning Test 2**
DQN2.2	Test that DQN performs as expected on CartPole.
DQN2.3	I will run my DQN implementation on CartPole and test that the output is similar to what is expected from the Mnih et al. implementation.
DQN2.4	The Gym CartPole environment
DQN2.5	A reward for each trajectory

DQN2.6	Normal
DQN2.7	Blackbox
DQN2.8	Performance
DQN2.9	Unit

DQN3.1	**Deep Q-Learning Test 3**
DQN3.2	This will test that DQN performs as expected on BreakOut.
DQN3.3	The DQN algorithms will be given the Gym BreakOut environment. Performance will be measure on this environment to ensure that it is in line with what is expected.
DQN3.4	The Gym BreakOut environment
DQN3.5	The reward for each trajectory
DQN3.6	Both
DQN3.7	Blackbox
DQN3.8	Performance
DQN3.9	Both

B1.1	**Buffer Test 1**
B1.2	This will test that the replay buffer is correctly saving the transitions from DQN.
B1.3	Given an agent and environment we will test that the transitions are being save correctly in the replay buffer.
B1.4	An agent and environment
B1.5	A replay buffer of transitions
B1.6	Normal
B1.7	Whitebox
B1.8	Functional
B1.9	Unit

B2.1	**Buffer Test 2**
B2.2	This will test that we can access transitions in the buffer.
B2.3	Given a buffer of transitions we test that we can sample from the buffer.
B1.4	A replay buffer
B1.5	Transitions sampled from the buffer.
B1.6	Normal
B1.7	Whitebox
B1.8	Functional
B1.9	Unit

O1.1	**Offline DQN Test 1**
O1.2	Test that offline DQN performs as expected.
O1.3	Given a buffer of transitions and an environment we test that the implementation of offline DQN matches the expected performance.
O1.4	A replay buffer and environment
O1.5	The rewards from online evaluation
O1.6	Normal

O1.7	Whitebox
O1.8	Functional
O1.9	Unit
CQL1.1	**Conservative Q-Learning Test 1**
CQL1.2	Test that CQL produces valid actions.
CQL1.3	Given a preprocessed frame check that the CQL agent takes that frame and produces a valid action corresponding to that frame.
CQL1.4	A preprocessed frame.
CQL1.5	A valid action corresponding to the given frame.
CQL1.6	Normal
CQL1.7	Whitebox
CQL1.8	Functional
CQL1.9	Unit
CQL2.1	**Conservative Q-Learning Test 2**
CQL2.2	Test that CQL produces expected performance on simpler environments.
CQL2.3	Given a replay buffer of transitions and the simple maze environments test that the CQL performance observed matches expectations.
CQL2.4	A replay buffer of transitions and the maze environments.
CQL2.5	The evaluated reward at each online evaluation.
CQL2.6	Both
CQL2.7	Blackbox
CQL2.8	Performance
CQL2.9	Both
A1.1	**ACORL Test 1**
A1.2	Test that ACORL outputs valid actions.
A1.3	Given a valid state we test that ACORL will output a valid action.
A1.4	A preprocessed state.
A1.5	A valid action.
A1.6	Normal
A1.7	Whitebox
A1.8	Functional
A1.9	Unit
A2.1	**ACORL Test 2**
A2.2	Test that ACORL learns as expected.
A2.3	Run ACORL on BreakOut and ensure that it get the expected performance.
A2.4	Replay buffer and BreakOut environment.
A2.5	Reward at each evaluation
A2.6	Both
A2.7	Blackbox
A2.8	Performance

A2.9	Both
E1.1	Evaluations Test 1
E1.2	Test that evaluations save correctly.
E1.3	Given and agent and environment test that the can save the evaluations as a numpy file.
E1.4	Agent and environment
E1.5	Dataset of evaluations
E1.6	Normal
E1.7	Whitebox
E1.8	Functional
E1.9	Unit
PLT1.1	**Plotting Test 1**
PLT1.2	Test that plotting works as expected.
PLT1.3	Given a dataset of evaluation returns check that the plots look correct.
PLT1.4	Dataset of evaluation returns
PLT1.5	Plots
PLT1.6	Normal
PLT1.7	Whitebox
PLT1.8	Functional
PLT1.9	Unit

Part III. Test Case Matrix

	Normal/ Abnormal	**Blackbox/ Whitebox**	**Functional/ Performance**	**Unit/ Integration**
IP1	Normal	Whitebox	Functional	Unit
DQN1	Normal	Whitebox	Functional	Unit
DQN2	Both	Blackbox	Performance	Both
DQN3	Both	Blackbox	Performance	Both
B1	Normal	Whitebox	Functional	Unit
B2	Normal	Whitebox	Functional	Unit
O1	Normal	Whitebox	Functional	Unit
CQL1	Normal	Whitebox	Functional	Unit
CQL2	Both	Blackbox	Performance	Both
A1	Normal	Whitebox	Functional	Unit
A2	Both	Blackbox	Performance	Both
E1	Normal	Whitebox	Functional	Unit
PLT1	Normal	Whitebox	Performance	Unit