Many real-life applications of reinforcement learning have delayed rewards, e.g., a self-driving car should only be rewarded if it safely reaches its final destination. Delayed-reward settings are particularly challenging, as they require a significant amount of exploration to even begin getting a reward signal. 

**Goal:** Analyze performance of modern RL algorithms on a set of simulated robot navigation delayed-reward tasks. The learning algorithms are Deep Q-Networks (DQN) and Policy Gradients (PG).

---

### Task Definition

**Goal:**
The agent must learn to navigate from its starting position to an unspecified goal location.

**Environment:**
The simulated 2D environments contains the agent, walls, and a goal position.

**Agent:**
The agent can take 5 actions, move in any of the four cardinal directions, or use a sensor to learn more about its immediate surroundings.

**States:**
The state representation is a greyscale image of the agent's current beliefs about its environment.

---

### Conclusions and Future Work

- **Conclusion:** The vanilla PG method can suffer from lack of exploration, as it is an on-policy algorithm, while DQN is generally able to efficiently estimate action-values for similar tasks. Both methods have trouble handling large state spaces.

- **Future work:** There are many additional ways to apply policy gradients in the literature that I would like to try using, such as TRPO[3], PPO[4], and DDPG[5].

---

### References


