Final Project Update

Interactive Robot Learning

Team: Pramodith
Tanmay

Outline

- Recap
- 2D Block World
- Deep Q Learning
  - Architecture
  - Replay Experience
  - Training
- Reinforcement Learning
  - State Representations
  - Reward Function
- Learning from Demonstrations
  - Oracle
  - Human Demonstrations
- Experiments and Analysis

Recap

Task: Given a configuration of cubes on a plane, teach a robot to assemble them into a goal configuration(pile(s) of cubes) by providing exactly 1 demonstration of the task

Input: 1 Demonstration of the task

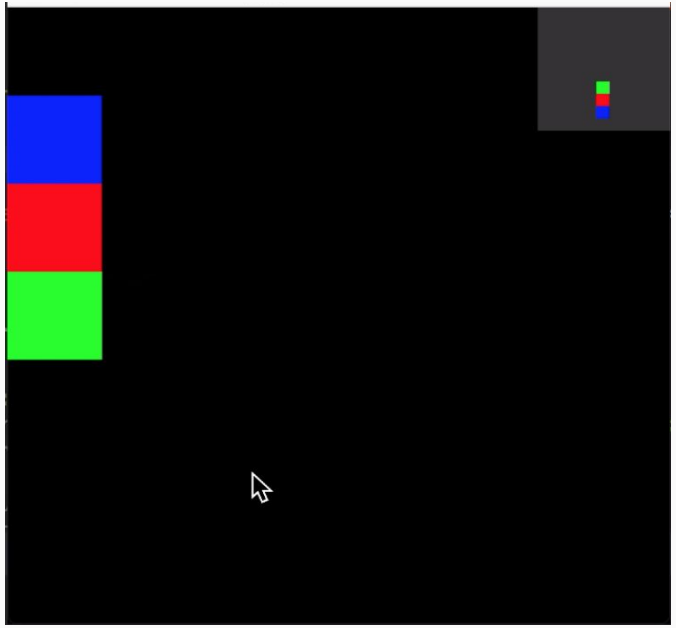
Output: Action to perform given another start(or intermediate) configuration

Importance: Extensible complexity and generalization of the problem; One shot learning

Update

- ~~One Shot~~ Imitation Learning
- 2D ~~3D~~ Block World
- Single stack
- 2 or 3 blocks in our environment

2D Block World



2D World

- 7 x 7 Grid
- Actions: PICK, DROP, MOVE UP, MOVE DOWN, MOVE LEFT, MOVE RIGHT
- Environment defines constraints on Actions
- Goal Location invariance
- In a 3 block world, state space: ${}^{7x7}C_3$ x 6 = 3.3M states

Deep Q Learning

- Only require images and corresponding reward to the state as input.
- Record demonstrations, two methods of training.
- Reward Structure
  - Sparse Reward
  - Dense Reward: Inverse Manhattan Distance scaled to a maximum of 1

DQN: Network Architecture

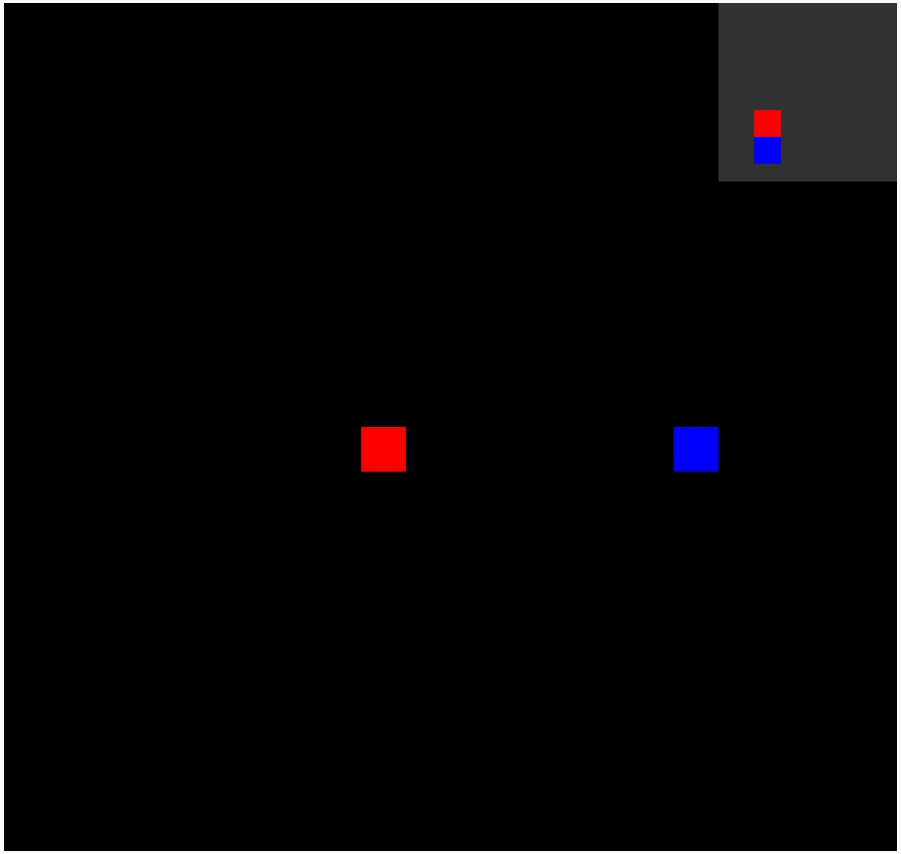
- 2 Conv layers followed by 2 Linear Layers
- Input images are resized to 100 x 100
- Goal is part of the state
- Replay Experience
- Policy and Target Network

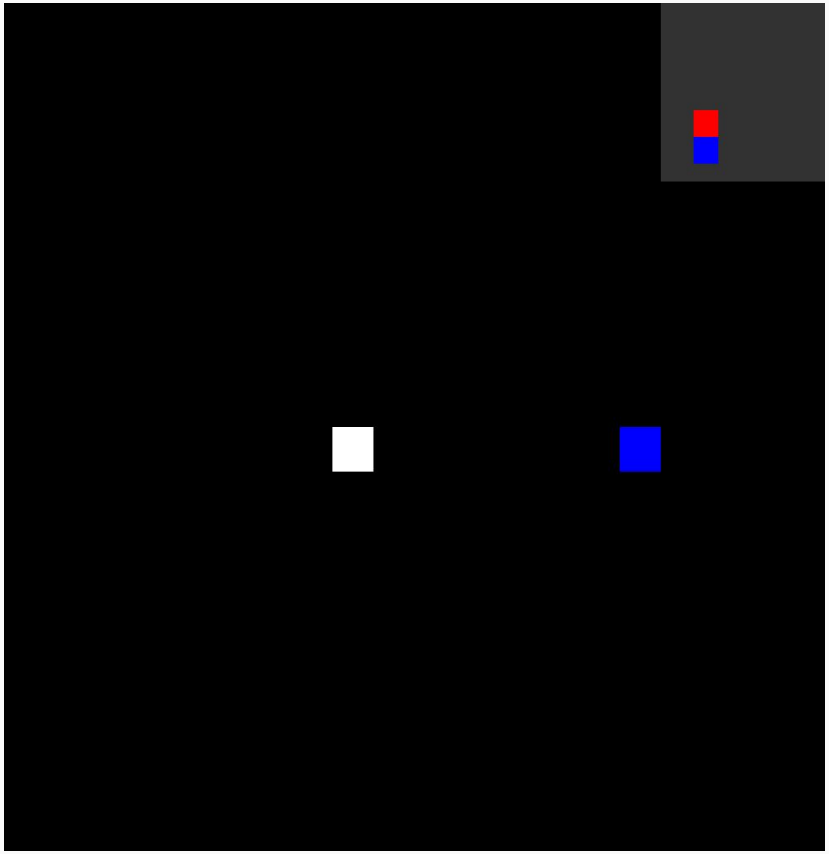
DQN: Training

- Had to teach the agent to pick and drop blocks
- Loss and Update

$$J_E(Q) = \max_{a \in A}[Q(s,a) + l(a_E, a)] - Q[s, a_E]$$

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot),s'\sim\epsilon}[(r+\gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)))\nabla_{\theta_i} Q(s, a; \theta_i)]$$





DQN: Results

- Poor results despite a lot of hyper parameter tuning
- Agent just learnt to Pick a block - all actions after that weren’t meaningful

Q Learning: State Representation

- Basic State Representation
  $((x_1,y_1), (x_2,y_2), (x_3,y_3)$, Selected Block Index, Goal Configuration)
- Problems
  - Can't handle different goals
  - Depends on the location of blocks
  - Depends on the location of the goal
- State representation should be location invariant
- State representation shouldn't be specific to a goal configuration to generalize for all goal combinations

Q Learning: State Representation

- Notion of chasing blocks
- Teaching the agent the means to achieve the goal
- (Distance to target block, Total distance between all blocks, (Direction of chased block, Goal blocked), Picked block, (Source , Target))

  Example: ((5625, 46250, (r, d, -), 1, (1, 0))
- Advantages
  - Partially extensible to new goals,
  - Independent of location

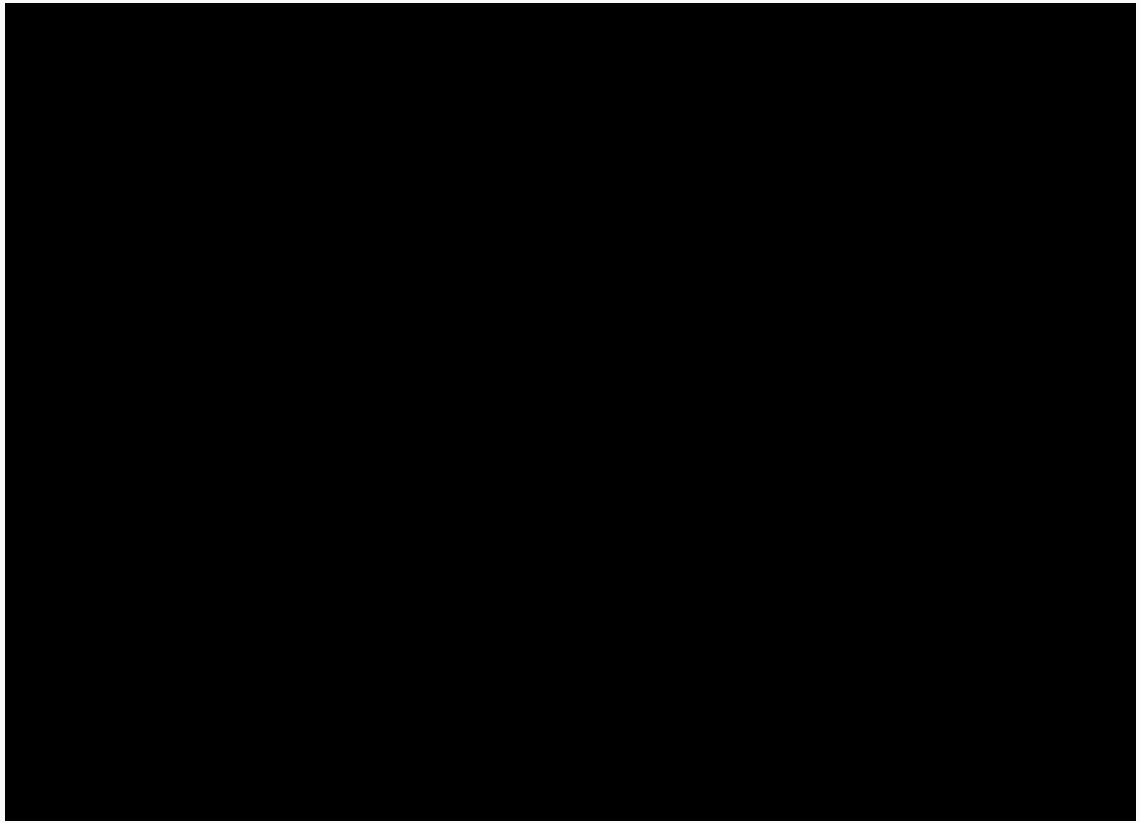
Reward Structure

- A reward after every action
  - 1 if the distance between the target block and the picked block has reduced
  - -5 if the distance between the target block and the picked block has increased
  - 0 if no change in the distance
- A reward of 5 for every pair of blocks that are adjacent to each other
- A reward of 50 for every matching pair of block according to goal configuration
- A *penalty* of 0.1 for dropping a block if the distance between the block it's chasing is greater than 5000
- A *penalty* of 0.1 if the agent just picks and drops blocks

Oracle

- Analytical solution
- Computes Goal location at the centroid
- Loose optimality in number of actions - focus on completion

Demo of RL algorithm



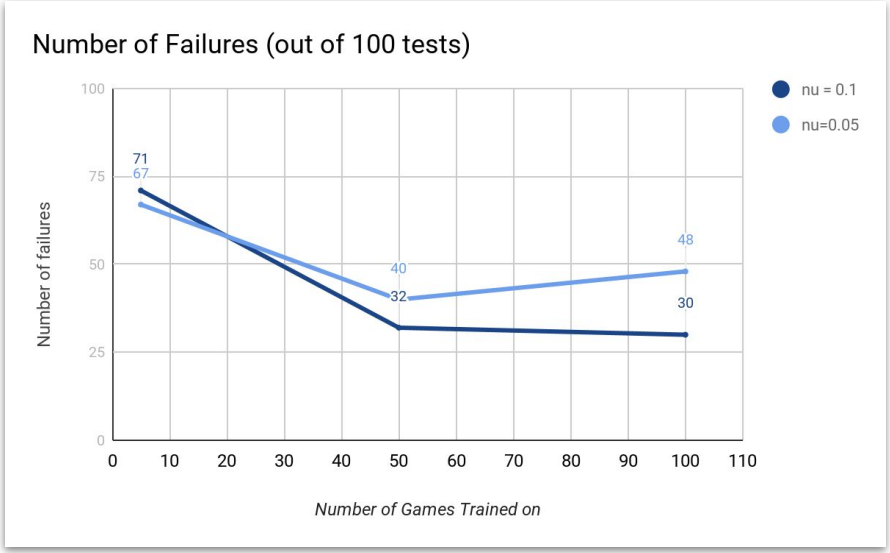
Experiments

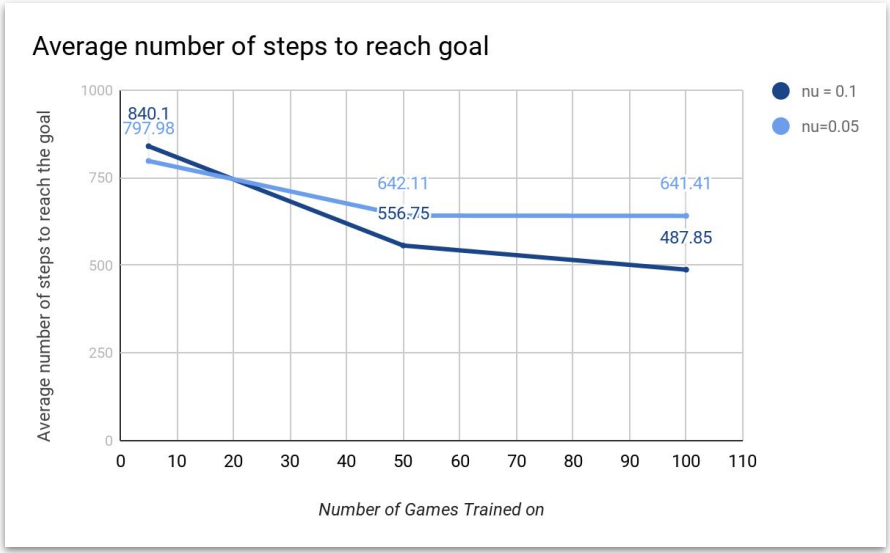
- Single stack 7x7 Grid
- Blocks start at random initial positions
- Performance metric
  - Number of successful games
  - Average number of iterations required to reach goal

Performance on 2 block world

- Baseline: Random Actions
  - 100 Games
  - Horizon length: 1000
  - 4 Failures with an average of 263.55 actions to reach the goal state
- Reinforcement Learning
  - nu=0.1: 75.25 Actions (average) to reach goal state
  - nu=0.05: 77.92 Actions (average) to reach goal state

Performance on 3 block world





Experiments for Learning from Demonstrations

- Experiments with training with the oracle
  - Querying probabilistically
  - Querying when the block oscillates between states
- Using Demonstrators to correct a pre trained version of the agent.
  - 10 demos per person
  - 3 participants
  - Trained only on 4 goals
  - User can pause the algorithm to make corrective actions

Effect of Oracle on performance

Experiment type	Number of failures out of 500	Average number of steps to reach goal	Wilcoxon p value against baseline
Baseline	186	578.626	-
Query the oracle during oscillation	157	554.322	3.395×10^{-1}
Query during oscillation with 0.1 probability of choosing action decided by oracle	125	518.668	1.60×10^{-2}

Effect of Demonstration on performance

	Failures (out of 500)	Average number of steps to reach goal	Wilcoxon P value against baseline
Baseline	159	527.632	
Demonstrator 1	97	399.14	1.169×10^{-5}
Demonstrator 2	100	483.979	3.23×10^{-8}
Demonstrator 3	137	511.83	1.3×10^{-3}

Goal Generalization: Unseen Goals

	Unseen Goal: [2, 0, 1]			Unseen Goal: [0, 2, 1]		
	Failures (out of 500)	Average number of steps to reach goal	Wilcoxon P value	Failures (out of 500)	Average number of steps to reach goal	Wilcoxon P value
Baseline	131	478.596		99	384.962	
Demonstrator 1	140	519.504	0.811	143	492.878	0.022(Reject)
Demonstrator 2	143	519.254	0.449	94	392.414	0.976
Demonstrator 3	109	446.834	0.223	92	404.016	0.957

Conclusion

- We created an agent that could make stacks of 2 and 3 blocks independent of the starting location
- The agent is partially extensible to new goals
- Corrective Human Demonstrations and an Oracle can be used to improve the agents performance.

Progress

1. Simulation environment/agent's world
    a. Open AI Gym
    b. PyDart
    c. Gazebo

2. Deciding the number of training and test demonstrations required

3. Design of train and test tasks

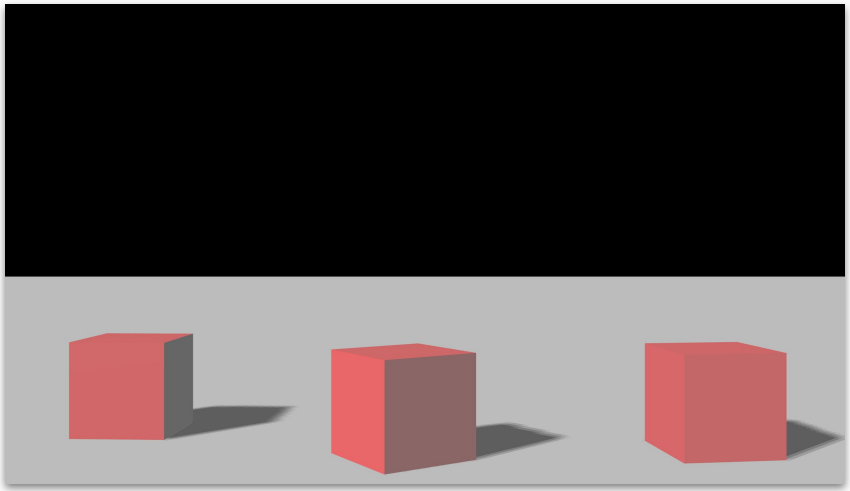
Explorations

- Experiment with a much simpler architecture.
- Try to build a system that is completely image based
  Input configuration, Demonstrations, Goal State are all provided as images

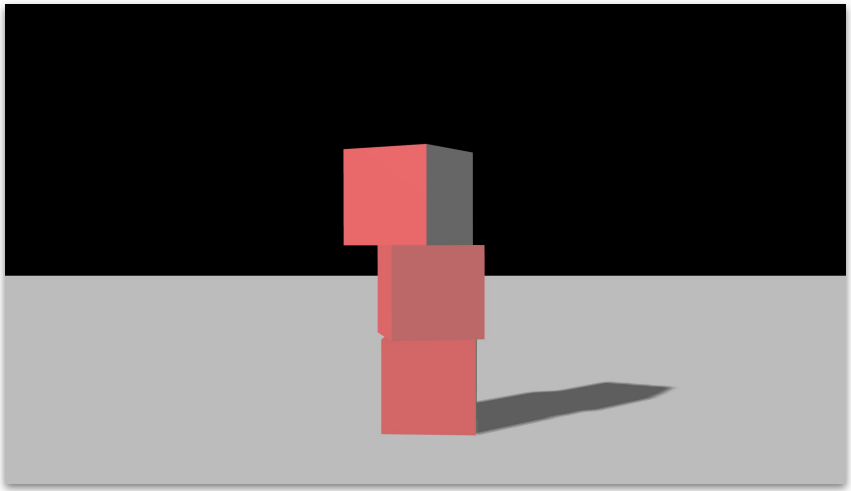
- Could use an intermediate block to learn the *pick location* of the gripper using images

- Modularizing Task difficulty
  - All blocks of the same color
  - Blocks of multiple colors
  - Blocks of multiple colors with unique ids(numbers) on them
  - Constrain Orientation

Custom Simulation Environment



Initial Configuration



Goal Configuration

Data point

Trajectories: As a sequence of actions(move x, move y, move z, pick, release)

Block information: The initial and final configuration of blocks as image and (x, y, z, color) coordinates of blocks

Learnings from Pilot Demonstration

- Interface Improvements
  - Demonstrator should be shown the goal alongside
- Physics Improvements
  - More realistic inertia for the blocks
- Add camera capabilities to the environment (to overcome occlusion related limitations for demonstrator)
  - This will require more thought as the input will change (image frames)
  - Another possibility is to provide fixed camera views and use that as an augmented input feature

Questions