Internship Log

Weekly Meetings – 22 hours

- February 8th
- February 15th
- February 29th
- March 7th
- March 14th
- March 21st
- March 28th
- April 4th
- April 11th
- April 18th
- April 25th

- General Notes I found useful:
  - Paper name: V-IRL: Grounding Virtual Intelligence in Real Life
    - LLM uses knowledge about agents using bibliographies and the environment using APIs to develop AI agents that can sense, think, and act as humans in the real world
  - Paper name: Private Fine-tuning of Large Language Models with Zeroth-order Optimization
    Paper name: LMO-DP: Accurately Fine-Tuning Language Models with Stronger Differential Privacy
    - Introduces a better alternative to DP-SGD for differential privacy in LLMs without major memory and performance loss

Project Meetings – 5 hours

- February 14th
  - Went over the basis paper and reviewed the test environments
  - Discussed why the paper was rejected
  - Task: Look into benchmarking tools that utilize PyTorch instead of Tensorflow
- February 28th
  - Quickly went over a list of possible reinforcement learning environments
  - Brainstormed ideas for tasks and environments
  - Received access to a previously built environment but with no RL model
  - Task: Work on a simple DQN to train a policy that can use the CliffWalk environment to navigate to the goal
- March 6th
  - Went over code for DQN and explained the functionality
  - Task: Finish the DQN, save the trained model in a .pth file, and implement it back into the CliffWalk environment to ensure the optimal path is being taken
- March 13th
  - Go over the literature review of the RL constraints field and understand how other environments use constraints
  - Task: Start implementing the second environment for testing
- March 20th
- March 27th
- April 3rd
- April 10th
- April 17th
- April 24th

Personal Work

- Reviewed paper – 14 hours
  - Paper name: Safe Reinforcement Learning with Natural Language Constraints
  - Went through original paper working to solve the issue of adding natural language constraints to RL models
  - Notes:
    - Analogy of constraints: A cleaning robot must be careful to not knock the television over, even if the television lies on the optimal path to cleaning the house
    - Two key limitations of Safe RL algorithms:
      - Provide constraints in mathematical or logical forms, which calls for specific domain expertise
      - Policies trained with a specific set of constraints cannot be transferred easily to learn new tasks with the same set of constraints
    - Figure 1
      - Safety training – agent learns to interpret textual constraints while learning the task
      - Safety evaluation – agent learns to interpret textual constraints while learning the task
    - Types of constraints:
      - Budgetary constraints – limit the frequency of being in unsafe states
      - Relational constraints – specify unsafe states in relation to surrounding entities
      - Sequential constraints – activate certain states to be unsafe based on past events
    - Why not instructions?
      - Instead of instructions, which specify what to do, textual constraints only inform the agents on what *not to do*, independent of maximizing rewards
        - To obtain rewards, the agent has to explore and figure out optimal policies on its own
      - Constraints are decoupled from rewards and policies, so agents trained to understand certain constraints can transfer their understanding to respect these constraints in new tasks, even when the new optimal policy is drastically different
    - Methodology – POLCO (Policy Optimization with Language Constraints)
      - Constraint interpreter to encode language constraints into the representation of forbidden states
      - Policy network operates on these representations and state observations to produce actions
      - Factorizing the model in this manner allows the agent to retain its constraint comprehension capabilities while modifying its policy network to learn new tasks
- Researched RL benchmarking tools – ___ hours
  - OpenAI Gym with PyTorch
    - Library: PyTorch
    - Applications: General reinforcement learning research and experimentation
  - Stable Baselines with PyBullet
    - Library: PyBullet
    - Applications: Training reinforcement learning agents in simulated environments with physics simulations
  - PySC2
    - Library: PySC2 (StarCraft II library)
    - Applications: Reinforcement learning research and experimentation in StarCraft II environments
  - Habitat
    - Library: PyTorch

  - Applications: Embodied AI research, exploration, and navigation tasks
- AI Gym
  - Library: PyTorch
  - Applications: Reinforcement learning research and education, customized environments
- AllenAct
  - Library: PyTorch
  - Applications: Research in embodied AI, supports various tasks and environments
- Meta-World
  - Library: PyTorch
  - Applications: Benchmarking meta-reinforcement learning and multi-task learning algorithms
- DeepMind Control Suite
  - Library: PyTorch
  - Applications: Continuous control tasks for deep reinforcement learning research
- PyBullet
  - Library: PyTorch
  - Applications: Physics simulation for robotic tasks and reinforcement learning experiments
- OpenAI Baselines
  - Library: PyTorch
  - Applications: Collection of implementations of reinforcement learning algorithms
- R2D2 (Recurrent Experience Replay in Distributed Reinforcement Learning)
  - Library: PyTorch
  - Applications: Deep reinforcement learning with recurrent neural networks, distributed training

- Learning Reinforcement Learning – 19 hours
  - Youtube Playlist: Stanford CS234: Reinforcement Learning | Winter 2019
    Accompanying Information: Slides & Lecture Notes
    - Lecture Notes:
      - Lecture 1 – Introduction to Reinforcement Learning
      - Lecture 2 – Making Good Decisions Given a Model of the World
      - Lecture 3 – Model Free Policy Evaluation: Policy Evaluation Without Knowing How the World Works
      - Lecture 4 – Model Free Control
      - Lecture 5 – Value Function Approximation
      - Lecture 6 – CNNs and Deep Q Learning
      - Lecture 7 – Imitation Learning
      - Lectures 8 & 9 – Policy Gradient
      - Lecture 9 – Advanced Policy Gradient
      - Lecture 10 – Policy Gradient III
      - Lectures 11 & 12 – Exploration and Exploitation
      - Lecture 13 – Fast Reinforcement Learning
      - Lecture 14 – Model-Based RL, Monte-Carlo Tree Search
      - Lecture 15 – Batch RL
      - Lecture 16 – Monte Carlo Tree Search
- DQN for CliffWalk environment – ___ hours
  - Covered example of DQN in PyTorch as an example before implementation in the CliffWalk environment
    - Resource: Reinforcement Learning (DQN) Tutorial
  - Worked on implementing DQN in the CliffWalk environment

- Learned how to use TensorBoard to graph the reward of the agent from every round to ensure the reward is improving
  - Resource: How to use TensorBoard with PyTorch
- After creating the DQN, the model was saved as learned_10000episode.pth
- Created a script to load the saved model and implement it in the CliffWalk environment to ensure the optimal path is being taken by the RL agent