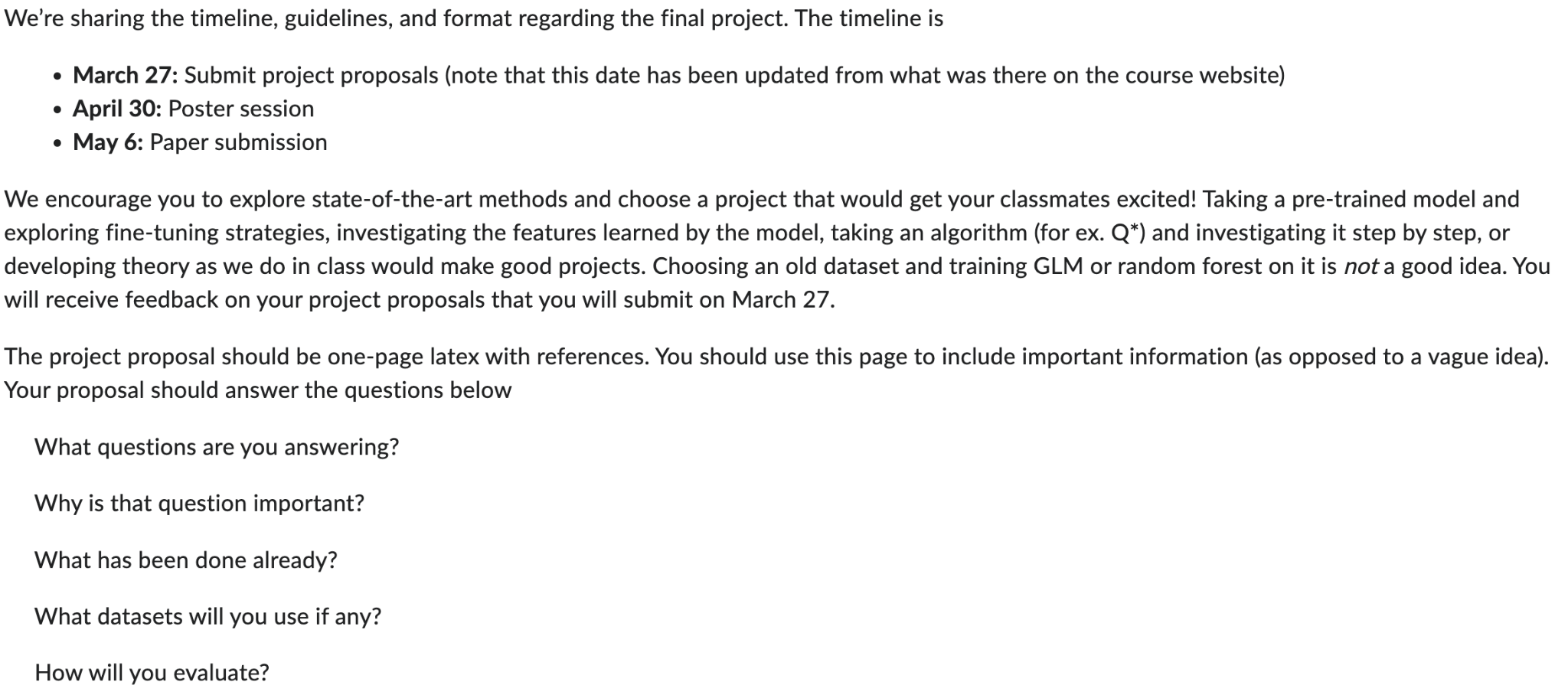
Investigate an Algorithm

[**https://www.overleaf.com/7128745935gnbbcfqywmxv#8a7190**](https://www.overleaf.com/7128745935gnbbcfqywmxv#8a7190)



* Reinforcement learning or other alg?
* Go through each step of algorithm and discuss its features and what it does to the data

ccm course website (contains slides on reinforcement learning and Q learning):

<https://brendenlake.github.io/CCM-site/>

3 types of reinforcement learning: <https://www.linkedin.com/pulse/types-machine-reinforcement-learning-madhavan-vivekanandan/>

Q-learning, DQN, and SARSA

Another article on reinforcement learning: <https://www.turing.com/kb/reinforcement-learning-algorithms-types-examples>

Sophie: Do we focus on one type of reinforcement learning? Or compare different types?

Sophie: Probably should avoid doing Q\* as they mentioned in the details in case a bunch of ppl decide to do it as well?

**Sophie’s Project Idea 1/3:** Compare 3 different types of reinforcement learning (value-based, policy-based, model-based)

1. Questions answering: What are the different types of reinforcement learning? How is each type constructed? What are the differences/similarities? What tasks are they optimized for? How do they compare on similar tasks?
2. Questions important bc: Want to know how to choose the best algorithm for the system’s goal. Want to know the tradeoffs between different algorithms so you’re able to identify and account for any potential cons to the one you select.
3. Done already: Provide specific algorithms of each type - what they are, what they do, what they’re designed for
4. Datasets: None, not building a model. But find example(s) of the different algorithms applied to a similar problem (automated navigation, translation, etc)
5. Evaluation: Compare algorithms’ performances across a similar task - compare scores and what each is optimized for.

Joy:

1. Can we find a problem where we can apply all three different types of reinforcement learning and compare?

**Sophie’s Project Idea 2/3:** Look at one state of the art application of reinforcement learning (Reinforcement learning in physical rehabilitation RoboREHAB - [paper](https://eds-p-ebscohost-com.proxy.library.nyu.edu/eds/detail/detail?vid=0&sid=c52d2b7a-e399-46c0-959b-0f374b7b6991%40redis&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=175058180&db=aci) )

1. Question answering: Where is reinforcement learning going - what is a state of the art application of reinforcement learning? What does this system look like/how does it work? Pros/cons? What’s next?
   1. Another question of interest: does it learn the specific gait of the person (if so how) or does it conform everyone’s gait to be the same?
2. Questions important because: This kind of tech has direct impact on human well-being. Has the potential to improve patient care, medical system, and access to health care. Important to identify flaws that could in the long run be potentially harmful since again, this has a direct impact on human life.
3. Done already: This paper. Previous work that they’re proposing an improvement on.
4. Datasets: Look at what data this paper uses. Not using to build program but should see how it is used by the system
5. Evaluation: Model performance compared to previous model. What has improved? What still needs work?

Joy:

1. discusses the implementation of RL in improving the functionality and effectiveness of a robotic device designed for physical rehabilitation, specifically focusing on gait training.
2. doesn't specify a particular reinforcement learning algorithm

**Sophie’s Project Idea 3/3:** Look at one state of the art application of reinforcement learning (Generative data augmentation to boost reinforcement learning - [paper](https://eds-p-ebscohost-com.proxy.library.nyu.edu/eds/detail/detail?vid=0&sid=54b7d8f3-7c75-4d87-a508-72fea539588c%40redis&bdata=JnNpdGU9ZWRzLWxpdmU%3d#AN=174715463&db=aci) )

1. Question answering: Where is reinforcement learning going - what is a state of the art application of reinforcement learning? What does this system look like/how does it work? Pros/cons? How does it improve upon past work? Why should this be the standard?
2. Questions important because: Want to know if this new method is better than previous methods. What are the tradeoffs? Want to know if this is something that should become standard practice or why it shouldn’t be.
3. Done already: This paper. Related work? Previous similar applications?
4. Dataset: Look at what data this paper uses. Not building program but should see how model compares with and without generated data, how data is generated, how data compares before and after.
5. Evaluation: Model performance compared to previous models. Is there something better?

**Joy’s Project Idea 1/1:**

<https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf>

Paper "Playing Atari with Deep Reinforcement Learning":

Background:

In traditional Q learning, we have an agent that learns to make decisions in an environment. The goal is to learn a policy to maximize cumulative reward.

What makes Q learning not applicable to Atari 2600 games?

In Atari 2600 games, imagine if we try to take a snapshot of every single possible screen we could see while playing a game, the number of possible screens is huge. Traditional Q-learning uses a Q–table to store the value of every state-action pair, it tries to remember what to do in every possible screen it may encounter, which is unmanageable.

Deep Q-Network, a variant of Q-learning (a value-based reinforcement learning algorithm), can successfully learn control policies directly from raw pixels to play Atari games competently.

What questions are you answering?

* How does the DQN algorithm learn to play Atari 2600 games from raw pixel inputs, and what features does it prioritize during learning?

Why is that question important?

* Learn DQN in detail, learn how deep reinforcement learning models interact with complex environments

What has been done already?

* In this paper, DQN is able to learn control policies for 7 games directly from screen pixels, and outperforming previous methods (human players)

What datasets will you use if any?

<https://github.com/Farama-Foundation/Arcade-Learning-Environment?tab=readme-ov-file>

* The Arcade Learning Environment (ALE) is a simple framework that allows researchers and hobbyists to develop AI agents for Atari 2600 games.

How will you evaluate?

* Since we are taking an algorithm and investigating it step by step, we look at how the algorithm is evaluated in the paper.

1. The DQN model’s ability to play multiple games with the same architecture (without game-specific tuning)
2. Average reward
3. Comparison to baselines (random action selection)

References:

* Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. DeepMind Technologies.

Joz:

Some potential methods for Investigating RL Algorithms Step by Step:

1. Exploration vs. Exploitation: Analyzing how the algorithm balances exploration (trying new actions to discover more about the environment) versus exploitation (leveraging known information to maximize rewards). This balance can be crucial for the learning efficiency of the agent and we can take a look at how it changes.

2. Policy Updates: Examining how the agent's policy (a mapping from states to actions) evolves over time. This could involve looking at the changes in action probabilities in policy gradient methods or the updates in Q-values for Q-learning algorithms after each episode or even each step.

3. Reward Signal: Understanding the reward signal and how immediate rewards lead to long-term gains. This involves analyzing the reward structure and how it influences the agent's learning process and decision-making.

4. Learning Rate and Discount Factor: Investigating the impact of hyperparameters such as the learning rate and the discount factor (γ) on the learning process. These parameters significantly affect how quickly the agent learns and how much importance it gives to future rewards.

5. Temporal Difference Learning: For algorithms that use TD learning (like Q-learning and SARSA), examining the TD error and how it guides the update process can be insightful. The TD error reflects the difference between the predicted rewards and the actual rewards received, driving the model's updates.

Joy - Final Project idea:

The idea behind reinforcement learning is that we have an environment and an agent that takes actions and receives feedback from the environment. The feedback includes the reward of his past action and information of his new state.

**Model-free reinforcement learning**, the agent doesn’t need to understand or predict the environments’ responses to its actions. Instead, it learns policy and value function based directly on the observed rewards and penalties it receives from its interactions with the environment. This contrasts with model-based RL, where the agent builds a model of how its actions affect the environment and uses this model to plan its actions.

Model-free reinforcement learning methods don't require any prior knowledge or model of the environments, so generally it’s more applicable across a wide range of environments, especially those where the environments are complex or unknown.

**Two popular examples of model-free RL are policy gradient and Q-learning**. They both operate directly using the information they gather from interacting with the environment and use these data to optimize their decision-making policy. In this paper I want to focus on 2 algorithms: PPO and DQN and apply those to 2 atari 2600 games and see how their performace compare to each other.

**PPO (proximal policy optimization):**

The example we see in class is a standard policy gradient method, which involves updating policies in a direction that maximizes expected return by using the gradients of the expected return with respect to policy parameters. This can lead to high variance in updates.

PPO modifies the standard policy gradient approach by using a clipping mechanism, which helps in controlling the size of the policy update. This clipping mechanism is used to prevent large destabilizing updates, which can be a significant problem in standard policy gradient methods.

**Frozen lake (reward range 0-1):**

**Game introduction:**

The "Frozen Lake" environment from OpenAI Gym presents a grid world where an agent must navigate from a starting point to a goal. The environment is composed of four types of tiles:

* Start (S): Where the agent begins.
* Frozen (F): Tiles the agent can safely walk on.
* Hole (H): Tiles that will cause the agent to "fall" and end the episode.
* Goal (G): The destination tile where the agent aims to reach.

The agent can move in four directions: left, down, right, and up, there's a probability that the agent will not move in the intended direction but rather slip to a side tile. This introduces uncertainty and complexity in navigation.The episode ends when the agent falls into a hole or reaches the goal.

**PPO:**

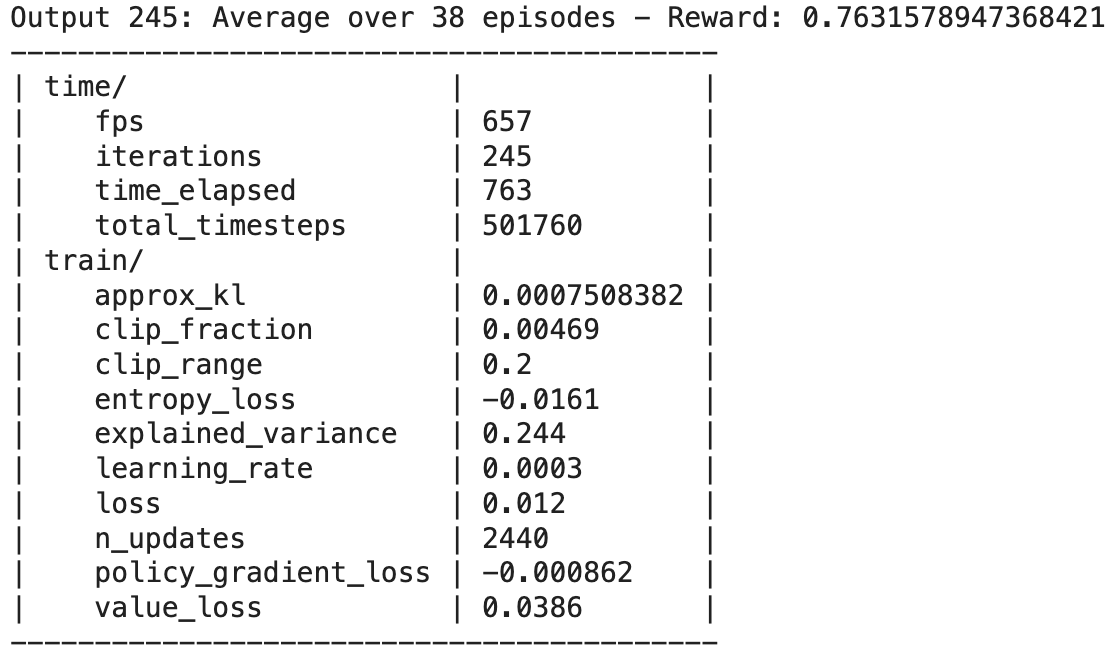
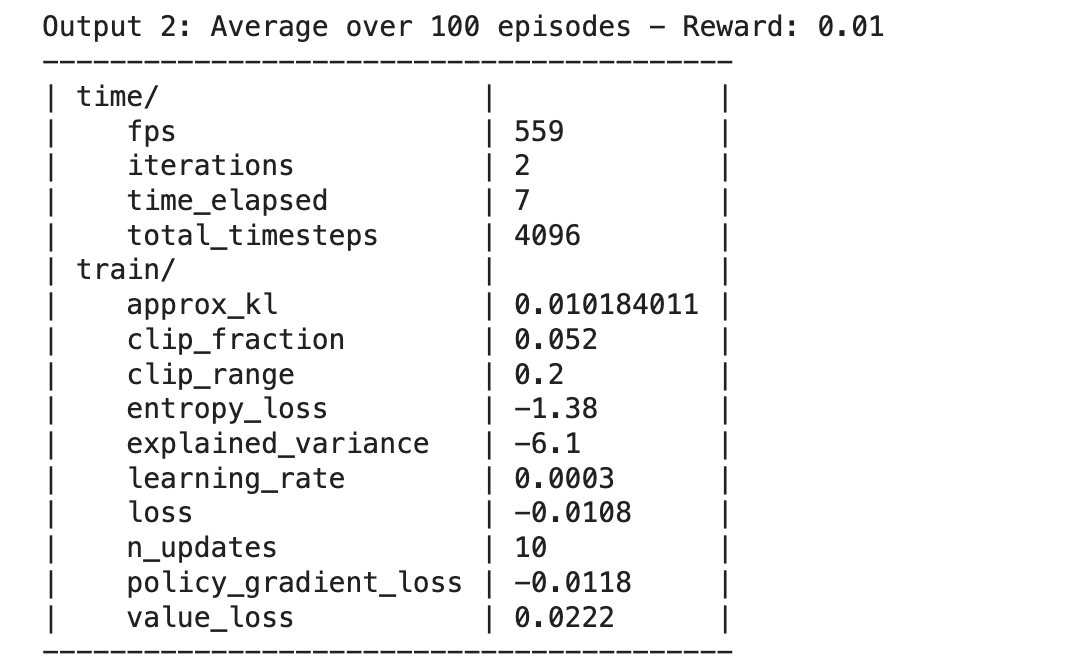
**ep\_rew\_mean:** The average reward per episode that the agent received

**approx\_kl:** The approximate Kullback-Leibler divergence between the new policy and the old policy. A measure of how much the policy changes after an update. Low values mean less change.

**clip\_fraction:** The fraction of the time the clipping is activated in the PPO loss function. It indicates how often the policy's updates are being clipped.

**explained\_variance:** How much variance in rewards is explained by the value function.

Episode 1: Episode 245:





**Observation from PPO:**

* Early training phase:

Initially, the rewards are very low (0.01, 0.0, 0.02...), which is normal as the model starts with a randomly initialized policy and must learn to navigate the environment.

KL divergence values are relatively low but show some variation, indicating that the policy is undergoing adjustments, but these changes are not drastic due to the clipping mechanism

* Late training phase:

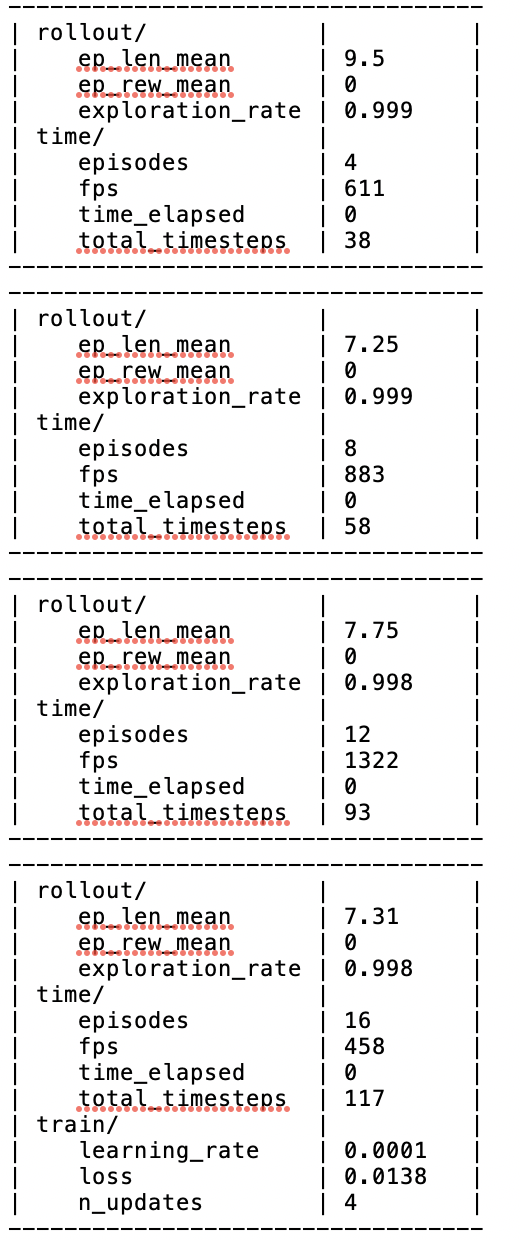
Towards the end of the training, the average rewards have increased significantly, reaching values like 0.8077, 0.5909, and 0.7636. This improvement suggests that the agent has learned effective strategies for reaching the goal more frequently.

The KL divergence is very low, suggesting that the policy updates are now resulting in only minor modifications to the policy. This is a sign of convergence.

Overall, the training appears successful, with the agent learning to significantly improve its performance in the "Frozen Lake" environment.

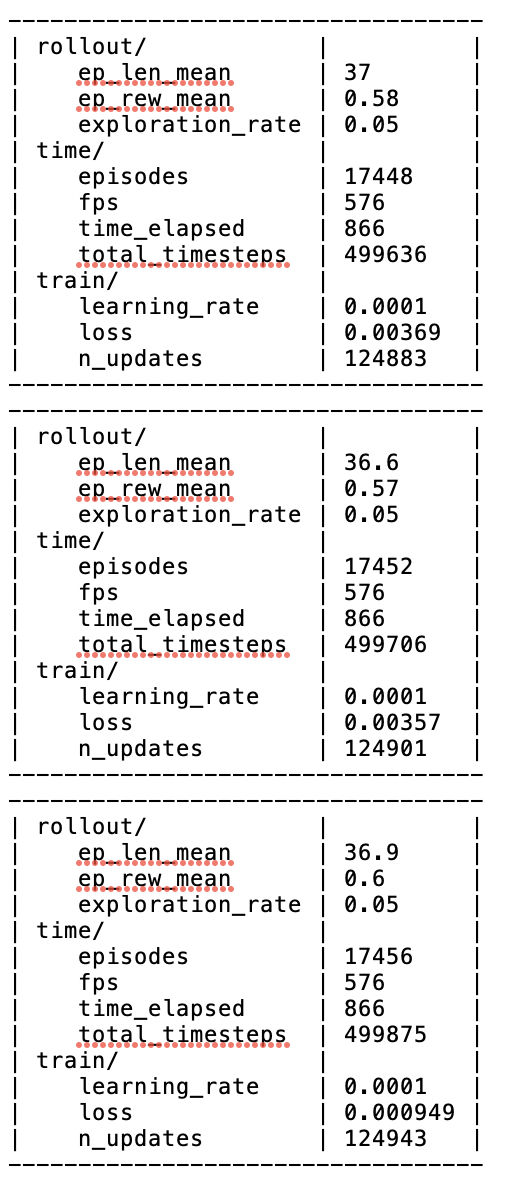
**DQN:**

Early training phase:



* The reward mean is consistently 0, suggesting that the agent never reaches the goal in the early stages of training.
* Exploration rate starting close to 1, and slightly decreasing, this high exploration rate is typical of early training where the model priortizes discovering new strategies over exploiting known paths.
* The number of updates to the model's parameters is relatively low, this is consistent with the early phase where less data has been collected

Late training phase:

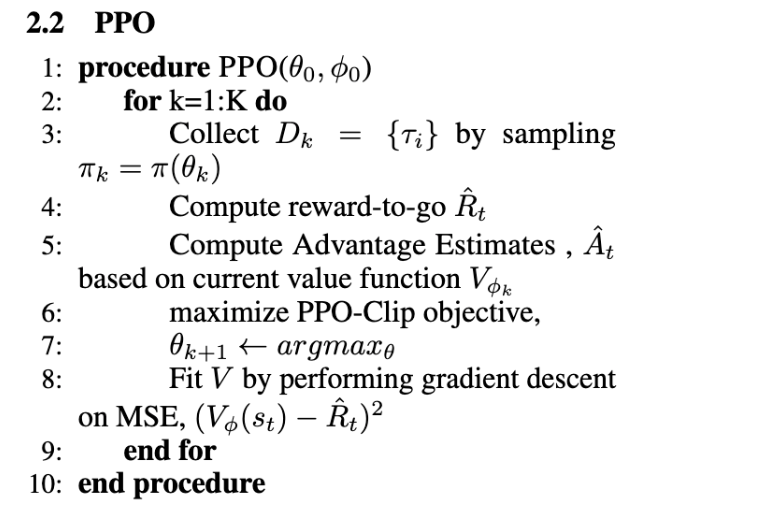
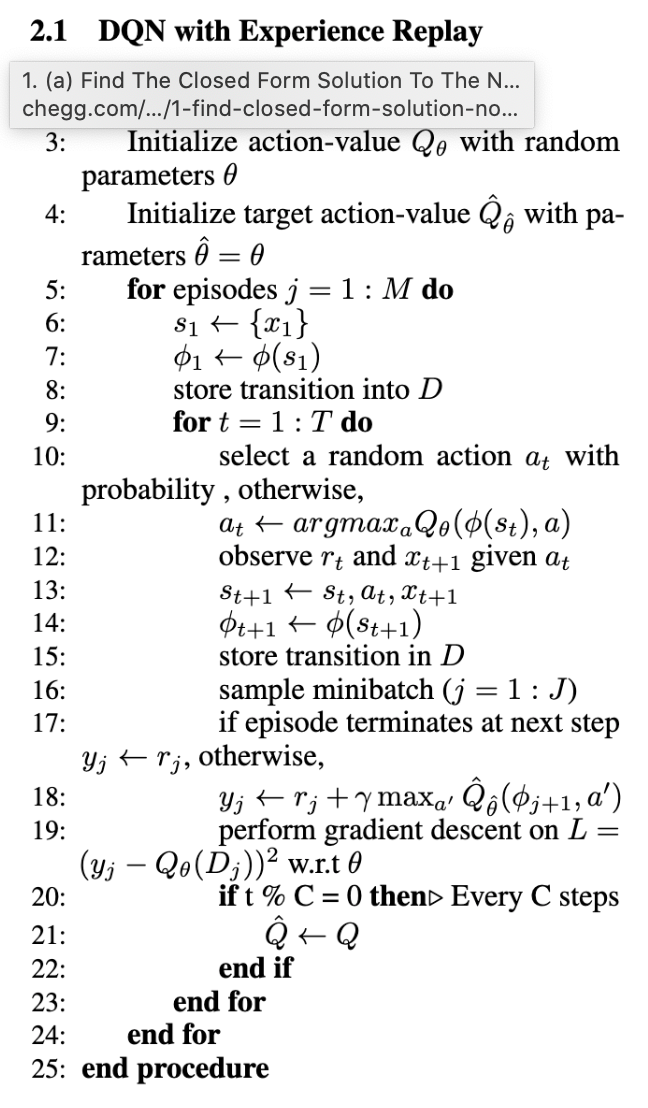


* The reward improves significantly. An increase to approximately 0.56 to 0.6 shows the agent regularly reaches the goal, showing successful learning.
* Exploration rate reduced to 0.05, this shows a shift from exploration to exploitation. The agent now leveraged learned strategies to maximize rewards.

Overall, transitioning from exploration to exploitation is evident.

The average reward 0.69 is higher than the average rewards towards the end of the episodes (ranging from 0.56 to 0.6), this seems counterintuitive at first, but could be caused by:

1. In environments like frozen lake, stochastic elements (like slipping on ice) can significantly affect outcomes. Actions do not always lead to the expected results because of the slipping mechanism. Therefore, a well trained agent might experience episodes with poorer outcomes simply due to the randomness of the environment.
2. Even in the later stages of training, the DQN agent is still engaging in some exploration (exploration rate at 0.05). This means it occasionally still tries less optimal actions to explore less frequent states, which can lead to lower rewards on some episodes.
3. It’s possible towards the end of training, the model begins to overfit.

.

04/11 notes:

* Test on interactive game
  + Suggestions: breakout or frozen lake
  + Breakout - Sophie & Joz
  + Frozen lake - Joy & Nicolas
* Look at 2 algs
  + DQN
    - <https://github.com/KJ-Waller/DQN-PyTorch-Breakout> ?
    - <https://github.com/He-Ze/DQN-breakout> ?
* For next monday - try to get game running & training

04/15 notes

* For this thursday:
  + For breakout:
    - #1 connect RL with our gameplay (see Joz & Joy’s code)
    - #2 train and see results
      * Feed state & reward to model (outputs of gymnasium vector)
    - #3 optimize and see step-by-step changes
  + For frozen lake:
    - #1 turn off slippery (concern abt model trying to learn randomness of that action
    - #2 run on both 4x4 and 8x8 matrices
    - #3 check how many episodes are in each batch (100 in first and 38 in last - guessing that all batches are 100 until last batch which just doesn’t contain enough)
  + resources/suggestions
    - Refer to gymnasium vector documentation ([LINK](https://gymnasium.farama.org/api/vector/))
    - Use GPT to debug
    - Link from Joy for breakout agent training ([LINK](https://github.com/MelvinMo/Agent-Training-for-Breakout-Game/blob/main/final%20code.ipynb))