Requirement 5 Report - Reinforcement vs. Imitation Learning Research

What is the process of training an agent for each technique?

First, the process of Imitation Learning will be explored in-depth.

Imitation Learning is also known as Learning From Demonstration, which provides insight on the nature of the process. Data is provided to the model that allows the system to closely emulate the actions and processes involved in decision making. The key is for the agent to learn a control policy based on the policies inferred from observation. It essentially maps the set of actions performed to its own weights, which allows it to replicate the behaviors in future, similar applications. Imitation Learning uses a supervised policy network to achieve its functionality; this is the same technique used in classifier models, such as Convolutional Neural Networks in the Image Classification domain.

There are two primary types of Imitation Learning: Behavior Cloning and Off-Policy [1]. For Behavior Cloning, the primary goal is to minimize the difference between the expert and learned policy. This allows the model to exhibit a set of behaviors as similar as possible to the input provided. A potential problem with this type is covariate shift, when the environment requires actions that are not included in the training data, since the set of samples may not be evenly distributed across the state space. A solution to the data distribution problem is DAgger, Dataset Aggregation, where the future actions of the current policy are projected into several time steps in the future [3]. However, this may be computationally expensive and time consuming.
Off-Policy is the process of learning more general observed behaviors, which may be used in a wider variety of environments and situations. Varying degrees of Gaussian noise can be injected to induce slight variation of exhibited behaviors, allowing the agent to explore and experiment [1]. A goal in the field of Imitation Learning is achieving One-Shot learning, i.e. where the model only needs a single training input, such as a video, to learn a behavior proficiently [2]. This is a potential approach for us to consider in a few applications, though a more generalizable approach may be optimal for the majority of our tasks

The steps involved in Imitation Learning are Data collection, Preprocessing, Training, and Testing. We will be conducting Video PreTraining (VPT); our potential data sources are outlined in Report 2. The preprocessing steps may involve manually classifying segments into movement patterns or translating each movement into a keyboard control. The technique used in our application will depend on our final choice of dataset and if any additional information is provided.

Second, the process of Reinforcement Learning will be explored in-depth.

In Reinforcement Learning, agents learn a behavioral policy to maximize a reward, while minimizing a loss function. The overarching goal is to maximize the sum of rewards within an iteration of training. The key difference from Reinforcement is the incorporation of this explicit reward function, as opposed to the ingraining of specific behaviors [4].

One of the cornerstones of Reinforcement learning is Q-Learning. Q stands for quality, which represents the degree that an action contributes to gaining a future reward [4]. The Q-function represents the maximal rewards gained beginning at a particular point, and following the optimal policy [5]. Q-learning is an Off-Policy algorithm similar to those discussed in the Imitation Learning section, but with a greater degree of freedom in achieving a goal. Randomization of behavior is induced by entropy bonuses, which encourages exploration of the environment that varies among agents. Optimization of a Deep-Q network is achieved via stochastic gradient descent; this stochastic element further encourages exploration, with a diminishing degree as the model evolves [5].

There is minimal initial input and preprocessing involved in Reinforcement Learning; the agent learns its weights during interaction with the environment. The rewards are determined by the Machine Learning Engineer via hyperparameters. An example within our application is the goal of collecting stone. Points would be added to the agent's score when there is stone within its field of view, and the goal would be considered complete when it is in its inventory. There may be intermediate steps as well, such as the possession of a wooden pickaxe, which requires its own goals and processes. There may be a bonus for the shorter amount of time taken to achieve this; inversely, there may be a penalty if it takes too long. There may be other penalties as well, such as losing health points.

Overall, the process of Reinforcement Learning brings an agent with little to no knowledge of an environment to an experienced level without requiring the observation of expert behavior

What is preferred among ML engineers working in similar projects?

Recently, NVIDIA trained a model with 33 years of Minecraft gameplay footage: the project was named "MineDojo" [6]. It was trained on 730K YouTube videos, 2.2B words within transcripts, 7K Wiki pages (incorporating images, tables, and diagrams), 340K Reddit posts and their 6.6M comments in r/Minecraft subreddit [6]. The videos included narration, and this was incorporated into a Natural Language Processing (NLP) in the model, performed by MineCLIP. The final model additionally incorporated the NLP interface to direct the agent, involving abstract goals such as exploring. The agent is additionally able to multitask within the game to achieve several goals simultaneously [7].

More recently, OpenAI trained a Minecraft model that understands wider time horizons and can obtain diamonds, a notorious challenge in this domain [8][9][10]. It primarily learned from unlabeled videos, with a small amount of "contractor data." The contractor data denotes the information collected among a set of individuals involved in a case study OpenAI conducted, where keypresses were logged during the recorded gameplay. This is fed into an Inverse Dynamics Model, "which predicts the action being taken at each step in the video" [8]. It uses past and future information to produce its prediction. This is simpler and less data intensive than solely using previous information, because the end goal is more easily determined. The optimization technique used for the IDM is Behavioral Cloning. The goal of the OpenAI model

was to create a gameplay environment that is similar to that of a human player; this environment includes a 20Hz frame rate, along with simulated mouse and keyboard input [8]. There are some techniques in which it learned in a zero-shot manner: swimming, hunting animals, eating food, and pillar jumping. This is fascinating, since these behaviors were not directly trained for, but learned by context.

Which technique provides the most efficient training? Is one technique superior for the scope of this project, or should a combination be used?

Comparing the two: Imitation Learning is more efficient for simple, direct, and repeatable tasks; Reinforcement learning is more efficient for broad, abstract, and variable tasks. There is no superior technique in the context of playing Minecraft, since the game has self-determined goals within a sandbox environment; each technique has pros and cons for the applications in which they are used. It would be best to use a combination of both: Imitation Learning as pre-training, and Reinforcement Learning as formal training. A majority of the time should be spent on the Reinforcement Learning aspect, using Imitation Learning as an accelerator. This will leverage the temporal efficiency provided in learning foundational tasks in Imitation Learning, while utilizing the minimal data involved in Reinforcement Learning.

References

[1] D. Seita, “DART: Noise injection for robust imitation learning,” The Berkeley Artificial Intelligence Research Blog, https://bair.berkeley.edu/blog/2017/10/26/dart/ (accessed Sep. 24, 2023).

[2] D. Seita, “One-shot imitation from watching videos,” The Berkeley Artificial Intelligence Research Blog, https://bair.berkeley.edu/blog/2018/06/28/daml/ (accessed Sep. 24, 2023).

[3] Imitation learning - Stanford University, https://web.stanford.edu/class/cs237b/pdfs/lecture/lecture_10111213.pdf (accessed Sep. 25, 2023).

[4] A. Violante, “Simple reinforcement learning: Q-learning,” Medium, https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56#:~:text=What%27s%20%27Q%27%3F,in%20gaining%20some%20future%20reward (accessed Sep. 24, 2023).

[5] “Introduction to RL and Deep Q Networks ,” TensorFlow, https://www.tensorflow.org/agents/tutorials/0_intro_rl (accessed Sep. 24, 2023).

[6] “Building generally capable AI agents with Minedojo,” NVIDIA Technical Blog, https://developer.nvidia.com/blog/building-generally-capable-ai-agents-with-minedojo/ (accessed Sep. 24, 2023).

[7] “Nvidia’s AI plays Minecraft after 33 years of training! ,” YouTube, https://www.youtube.com/watch?v=5LL6z1Ganbw (accessed Sep. 24, 2023).

[8] “Learning to play Minecraft with video pretraining,” Learning to play Minecraft with Video PreTraining, https://openai.com/research/vpt (accessed Sep. 24, 2023).

[9] Openai, “Openai/video-pre-training: Video pretraining (VPT),” GitHub, https://github.com/openai/Video-Pre-Training (accessed Sep. 24, 2023).

[10] OpenAI’s new AI learned to play Minecraft!, https://www.youtube.com/watch?v=fJn9B64Znrk (accessed Sep. 25, 2023).