DINK: Differently Initialized Q-Networks

Atiksh Bhardwaj
(ab2635)

Jonathan Chen
(jqc3)

GitHub Repository

1 Introduction

Q-Networks represent an off-policy reinforcement learning approach centered on estimating value functions from environments. As an off-policy method, it relies on an external dataset for learning and computing Q-values. This dependency on provided data can yield diverse outcomes, either enhancing or hindering the Q-Network's performance. To delve into this dynamic, we embarked on an investigation utilizing conventional imitation learning techniques like Behavior Cloning and DAgger to generate data for training Q-Networks. Our objective is to train and contrast these two approaches alongside expert data sourced from the Atari Grand Challenge (AGC), specifically in the OpenAI Space Invaders gym environment. This exploration aims to shed light on whether Deep Q-Networks possess the capability to learn effectively from various datasets, or if their performance is heavily contingent upon the quality of the initial dataset used for training.

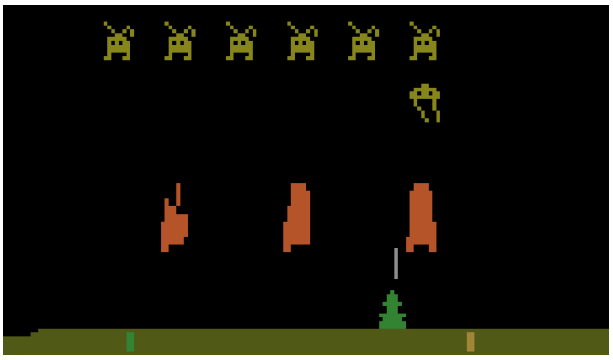


Figure 1: Atari Space Invaders

2 Problem

2.1 Motivation

We enjoy video games, and initially wanted to do a project on Space Invaders, and found several videos and projects online that did Space Invaders with Q-Networks. Thus we came up with an experiment to incorporating Q-Network as this is the first fitted model we learned and also one that is an off-policy algorithm, which allows our question to be more interesting as on-policy models have variation with the initial policies. However, Q-Networks, only rely on an initial dataset, which can come from any source including the untrained dataset. Thus, we wanted to see whether performance changes in Q-Networks based on this initialized dataset.

2.2 Problem Statement

We propose the following question: Does initialization by different datasets such as those created by expert humans, a BC policy, or a DAgger policy affect the performance of a Q-Network? This will help in determining what the ideal initialization for a Q-Network would be for ideal performance and learning.

2.3 Hypothesis

Based on the problem statement, define the function $J(\pi)$ as the performance of a policy:

$$J(Q_{\textbf{Human Expert Data}}) > J(Q_{\textbf{DAgger}}) > J(Q_{\textbf{BC}}) \tag{1}$$

where $J(Q_{\textbf{Human Expert Data}})$ is the Q-Network initialized with the human expert data, $J(Q_{\textbf{DAgger}})$ is initialized with the DAgger created data, and $J(Q_{\textbf{BC}})$ is initialized with the BC created data.

As we plan to initialize DAgger with BC:

$$J(\textbf{DAgger}) \geq J(\textbf{BC}) \tag{2}$$

and the human expert data will be strictly better than both policies as the human will have better knowledge of possible danger cases, whereas the above policies can only work in their data distribution.

3 Approach

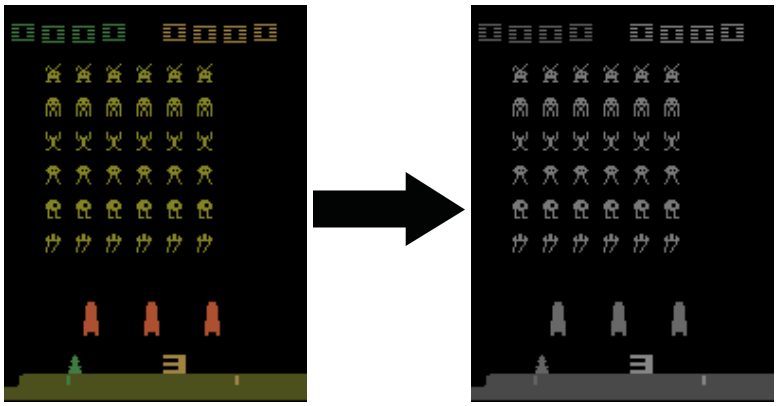


Figure 2: Gray scaling of input images

3.1 Atari Grand Challenge Data

In order to train an imitation learning agent, we need to collect expert trajectories for our models. As with most games, players can employ a variety of strategies to play Space Invaders. Thus it is difficult to classify any player or agent an "expert" with regards to expert policies.

Our solution was to use the open-source Atari Grand Challenge (AGC) [2] repository, which includes Space Invaders replays generated through a crowd-sourcing effort. This will serve as the basis for a separate, initial dataset of expert trajectories.

We first sample the top 15 trajectories from this repository. These top 15 trajectories all have a final score above 1400. Each trajectory encodes the game state and the action taken by the human player at a certain frame/timestep.

The raw data for game state are screenshots of the game taken at each frame; we read these images in and convert them to grayscale to decrease the observation space without any loss of information. We

then flatten this 2D array to prepare it for a linear neural network layer. The game state is combined with current score (cumulative reward) and current action to form one time step of the environment. Additionally, we map human misinputs, i.e. human players pressing buttons that have no effect on the game, to the NO-OP action (See Section 3.2).

3.2 Environment

We use the OpenAI gym environment to train our models. This environment has an observation space of 210*160*3, representing the height and width of the screen and 3 RGB channels. This environment has an action space of 6, representing the 6 valid controls for the game (NO-OP, FIRE, RIGHT, LEFT, RIGHTFIRE, LEFTFIRE).

We can further parameterize the base *SpaceInvaders-v0* environment with two options: sticky actions and frame skips. Sticky actions introduce stochasticity into the environment by repeating the same action from the last timestep instead of from the actual policy, with some probability β. Frame skips mean that the environment steps multiple frames at once for each action taken. These two parameters can affect the exploration vs. exploitation of learned policies. During training we set sticky actions $\beta = 0.15$, with no frame skips, and we maintained these hyperparameters in testing.

3.3 Models

3.3.1 Imitation Learning: BC and DAgger

We first implement two classic imitation learning models: Behavior Cloning and DAgger. The implementations of both were adapted from Assignment 2 [1]. Our BC model has a neural network architecture that takes in a 210*160 long vector from the images, then sends it through two hidden layers followed by ReLU activation functions. The output layer has 6 outputs, each representing the likelihood to take one of 6 possible actions. This architecture learns a policy from taking the maximum of these likelihoods as its action. The initial dataset are the top 15 trajectories from the Atari Grand Challenge dataset.

We then use the DAgger paradigm to improve the performance of our imitation learning agent. Starting with the same initial dataset and same architecture as the BC model, we query a pre-trained PPO agent [3] for expert actions. This pre-trained agent will allow us to query for high-quality expert actions on demand. DAgger is implemented with the following algorithm:

Algorithm 1 DAgger

procedure TRAIN($b, d, \pi^\star$) ▷ b is the BC learner policy
  Dataset $D \leftarrow d$
  Policy $\pi_0 \leftarrow b$
  Number of interactions $N \leftarrow 25$
  for $i \leftarrow 1$ to N **do**
    while not done **do**
      state, action $(s, a) \leftarrow$ environment stepped by π_i
      query expert π^* with s for optimal a^*
      trajectory $t \leftarrow t \cup (s, a^*)$
    end while
    $D \leftarrow D \cup t$
    Train π_{i+1} on updated D
  end for
  return best π_i
end procedure

3.3.2 Q-Network

Our Q-Network has a basic architecture which takes in a 210*160 long vector from the images, and then passed through 2 linear layers which each are followed by ReLU to generate Q-Values for training. The computed values are compared to target Q-Values that are computed based on the next

observation's maximum Q-values that are computed from the network, reward, and done flag for that specific state that is passed through as done in the equation below:

$$Q^{k+1} = \arg\min_{q \in \mathbb{Q}} \sum_{i=0}^{N-1} \left[q(s_i, a_i) - (r(s_i - a_i) + \gamma \cdot \max_a Q^k(s_{i+1}, a) \right] \tag{3}$$

Equation (3) represents the optimization problem for the Q-Network, where it finds the $q \in \mathcal{Q}$ that minimizes the above using the previous network's Q^k function.

During train time, we chose to take an online approach, where every four epochs we let the model run on the environment and collect data for it to learn. Additionally, the loss for the Q-Network takes the MSE between the computed Q-Values and the target Q-Values as seen in equation (3). We train the networks with a learning rate of $1 \cdot 10^{-6}$, and a discount factor of $\gamma = 0.99$. The discount factor had been chosen to be a large value has reward in the space invaders environment is sparse, so discounting would make it difficult to learn. Additionally, we have random exploration with $\epsilon = 0.01$ to allow the model explore and find better actions than what it generates, as Space Invaders is a fairly random game to the model.

Algorithm 2 Fitted Q-Value Iteration (online learning)

procedure TRAIN(τ) ▷ τ is the input dataset
    Initialize function $q \in \mathbb{Q}$
    Number of epochs $K \leftarrow 25$
    for $k \leftarrow 1$ to K **do**
        $Q^{k+1} \leftarrow \arg\min_{q \in \mathbb{Q}} \sum_{i=0}^{N-1} \left[q(s_i, a_i) - (r(s_i - a_i) + \gamma \cdot \max_a Q^k(s_{i+1}, a) \right]$
        Every 4 iterations, $\tau \leftarrow \tau \cup$ trajectories collected with $\pi_k = \max_a Q^k(s, a)$, which is ϵ greedy sampling uniformly if probability $\leq \epsilon$
    end for
    return best $\hat{\pi}(s) = \text{argmax}_{a \in \mathbb{A}} Q^K(s, a)\ \forall s$
end procedure

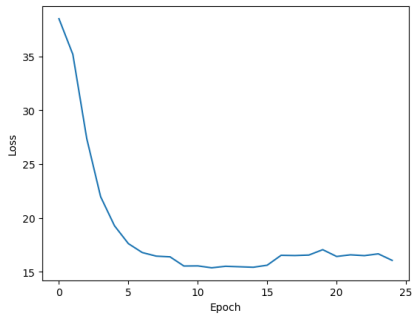
3.4 Methodology

The goal of our experiment is to determine how initializing Q-Networks might affect their performance. Our experiment setup is as follows:

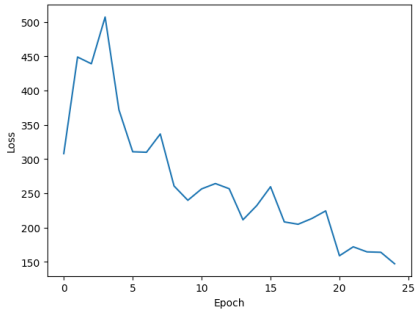
1. Preprocess and filter trajectories from the Atari Grand Challenge Dataset
2. Train BC and DAgger (with the help of expert PPO agent)
3. Collect datasets of trajectories using learned policies from BC and DAgger
4. Initialize three Q-Network models with the three datasets:
   original AGC trajectories, trajectories from BC policy, trajectories from DAgger policy
5. Compare average scores over 50 runs for BC, DAgger, AGC Q-Network, BC Q-Network, and DAgger Q-Network.

4 Results

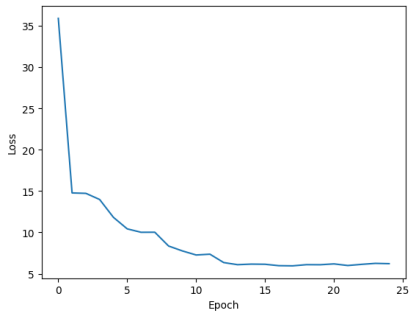
4.1 DQN Loss and Score Plots



Atari Expert QN

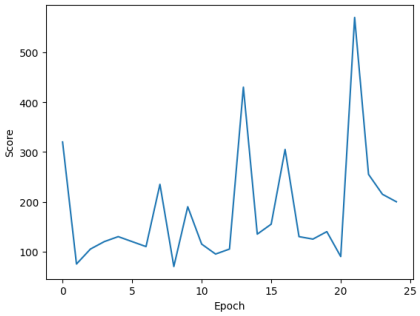


BC QN

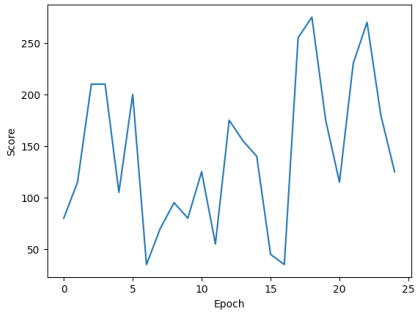


Dagger QN

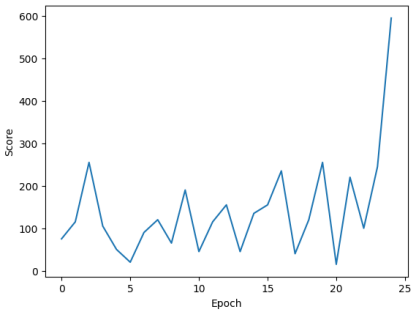
Figure 3: Comparison of Q-Network Loss



Atari Expert QN



BC QN



Dagger QN

Figure 4: Comparison of Q-Network Reward (Per Epoch)

4.2 Comparison of Models by Average Score

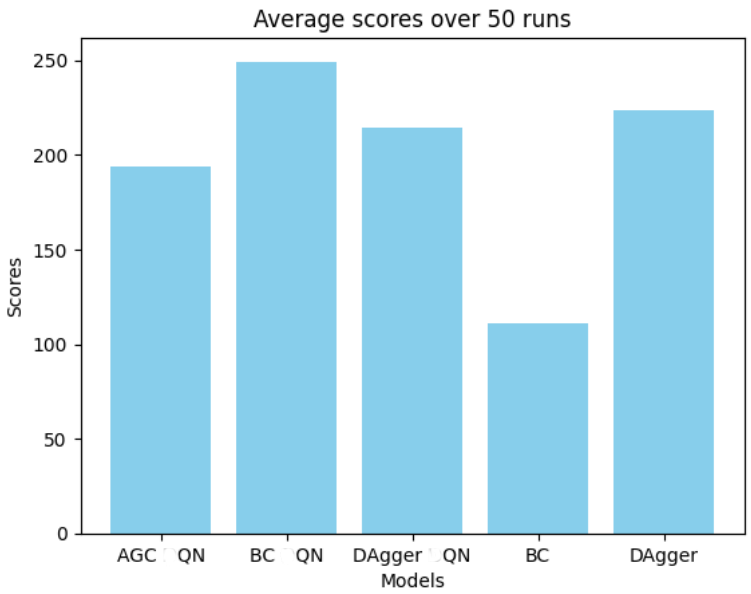


Figure 5: Average Score for each Model

4.3 Analysis

1. Looking at Figure 3, the graphs of the loss functions for the Atari Expert data Q-Network and the DAgger Q-Network are quite similar reaching fairly loss values and fitting well to their data. Training is stable and converges after around 20 epochs. In comparison, the BC Q-Network has a lot more randomness in the change in loss, but over time slowly drives down. With more epochs, the loss could be driven down further. Based on these graphs, we can interpret that the Atari and DAgger based Q-Networks have fitted well to the provided data and the online data it created, whereas with the BC Q-Network, the algorithm needs more training to fit as well as possible.

2. On Figure 4, all the models have high stochasticity during training partially due to the Space Invaders environment having built-in stochasity and partially from the stochastic exploration built into the Q-Network. However, we can observe that Atari and DAgger have fairly consistent rewards in comparison to BC which has high variation in rewards leading to incredible erratic results. The consistency for the Atari and DAgger QNs can be due to the data being provided has consistent reward in the environment, which can be verified in figure 5. BC, due to the small training data, had erratic decision making, but also had learnt more of the expert actions for the specific times where it ended in a similar situation as an expert leading to higher scores on the average.

3. For Figure 5, the results show that BC QN had the highest average score hovering around 240-250, with DAgger and DAgger QN behind at around 225 score. BC performed the worst due to most of the environments being out of distribution for the data that it had been trained on. BC QN ended with the highest score due the stochasticity described above allowing it learn Q-Values that would lead to better scores, thus leading to a higher average score in performance. This essentially allowed the model to explore more than the Atari QN and DAgger QN which were hampered by their datasets having more consistency in comparison to BC.

However, based on these graphs the three different initializations resulted in similar average scores varying from 190 to 250 score which is the difference between shooting two to three more aliens. Thus, the dataset initialization did not actively impact the Q-Network, and the performance could be more related to the relevant hyperparameters such as γ, ϵ, and the learning rate.

5 Conclusion

Our findings suggest that the initial dataset provided to a Q-Network model does not have a significant impact on its overall performance. This is despite the fact that the performance of the agents that produced those datasets vary drastically in performance. Our original hypothesis that the Q-Network initialized with expert human data would perform the best is not supported by our results. Rather, our findings support this conclusion:

$$J(Q_{\textbf{Human Expert Data}}) \approx J(Q_{\textbf{DAgger}}) \approx J(Q_{\textbf{BC}}) \tag{4}$$

These results are under the assumption that a small subset of expert trajectories and a low number of epochs are sufficient for training. These results may change with different hyperparameters, more epochs, and more training data.

6 Further Work and Implications

As mentioned, our Q-Network simply used linear layers to calculate the Q-Values, but the environment originally works with (210, 64, 3) shaped images. A followup would be to change our model to work with convolutional layers so that it can gain a better spatial understanding. Thus we can expand the model depth, so it can gain more understanding of the observations.

Additionally, for improved model performance, the Space Invader environment allows for frame skipping, allowing the model to skip images that do not have any relevant information such as frames between an Alien shooting or the Spaceship shooting. This would also allow for better performance gains and give a consistent model that can perform well.

7 Work Assignment

Atiksh:

- Q-Network Model
- BC Model
- DAgger Model
- Experiment setup

Jonathan:

- Dataloader
- Preprocessing
- Visualization
- Experiment setup

The writing of the report was equally split between Atiksh and Jonathan.

References

[1] S. Choudhury. Cs4756 robot learning spring 2024. URL `https://github.com/portal-cornell/cs4756-robot-learning-sp24`.

[2] V. Kurin, S. Nowozin, K. Hofmann, L. Beyer, and B. Leibe. The atari grand challenge dataset. *CoRR*, abs/1705.10998, 2017. URL `http://arxiv.org/abs/1705.10998`.

[3] D. Tyagi. Implementations of deep reinforcement learning algorithms and bench-marking with pytorch. URL `https://deepanshut041.github.io/Reinforcement-Learning/`.