# Deep Q-learning from Demonstrations

Todd Hester(google deepmind) et al
AAAI Conference on Artificial Intelligence(2018)

2023 ORLAB seminar

발표자 : 김변민

# Introduction

- **What is imitation learning ?**

  Imitation learning is a technique that involves an agent learns to perform a task by observing and Mimicking the actions of an expert

- **What is learning from demonstration ?**

  Learning from demonstrations is a subset of imitation learning, where an agent enhances its exploration efficiency by utilizing expert’s demonstrations

# Necessary of imitation learning in RL

- **Reduction in Training time**

  By mimicking expert actions, Learning from Demonstration achieves target performance faster with reduced training time compared to general reinforcement learning

- **Overcoming performance degradation due to inaccurate simulator**

  Agents must learn in the real domain with real consequences for its actions to perform well, but it is hard with with inaccurate simulators

Using Learning from Demonstration enables initial learning that reflects real-world

# Related works

- **J. Chemali et al, “Direct policy iteration with demonstrations,” in Proc. Int. Joint Conf on Artificial Intelligence, 2015**

  1. Obtain expert sample data $D_E$ and agent sample data $D_\pi$

  2. $D_E$ and $D_\pi$ are each used with a certain probability for updating the Q value (updated via the Monte Carlo method)

- **Ho et al, “Generative adversarial imitation learning" Advances in neural information processing systems, 2016.**

  1. The discriminator receives sample data from both the expert and the agent as inputs.

  2. The discriminator distinguishes whether the sample data is from the expert or the agent.

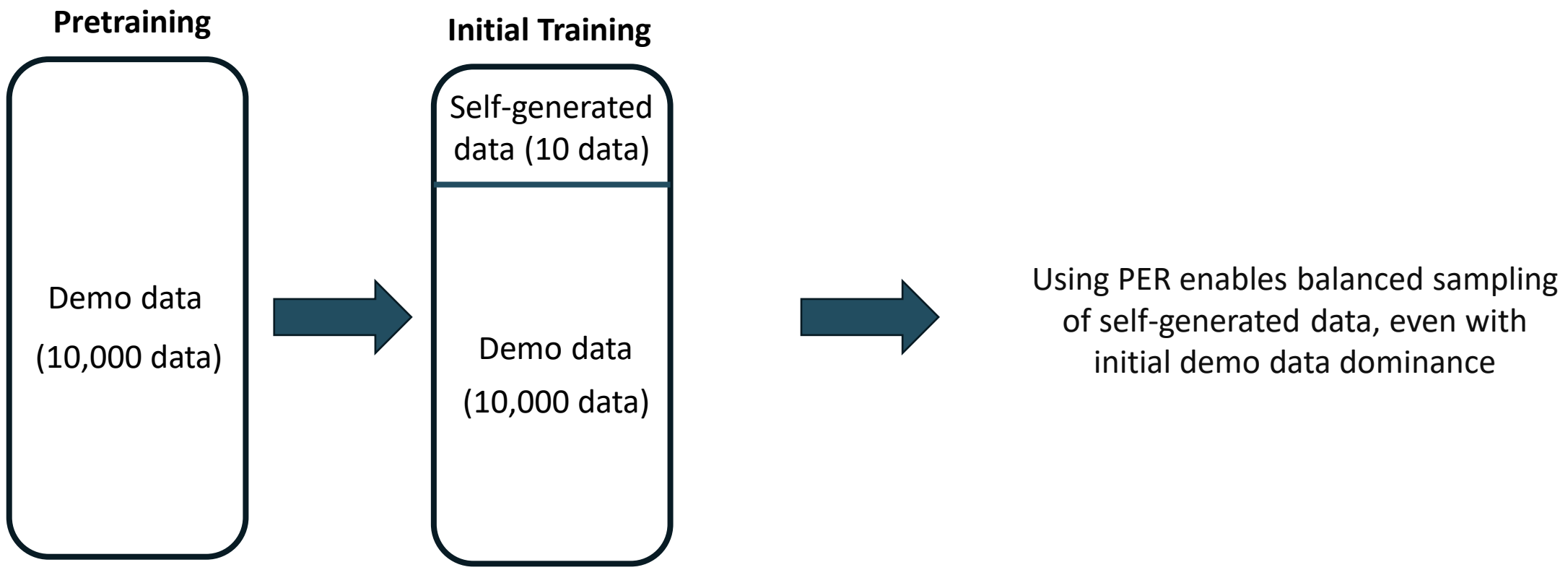
  3. If the discriminator can differentiate, a penalty is applied to the reward if not, an additional reward is given.

**Aim to develop a method for using demonstration data effectively within the DQN structure**

# Key Idea1

- **Utilizing PER(Prioritized Experience Replay) for balanced sampling of demo and self-generated data.**

Choosing the ratio between demonstration and self-generated data while learning is critical to improve the performance of the algorithm



# Key Idea2

- **Enables efficient utilization of demo data by separating loss into four types**

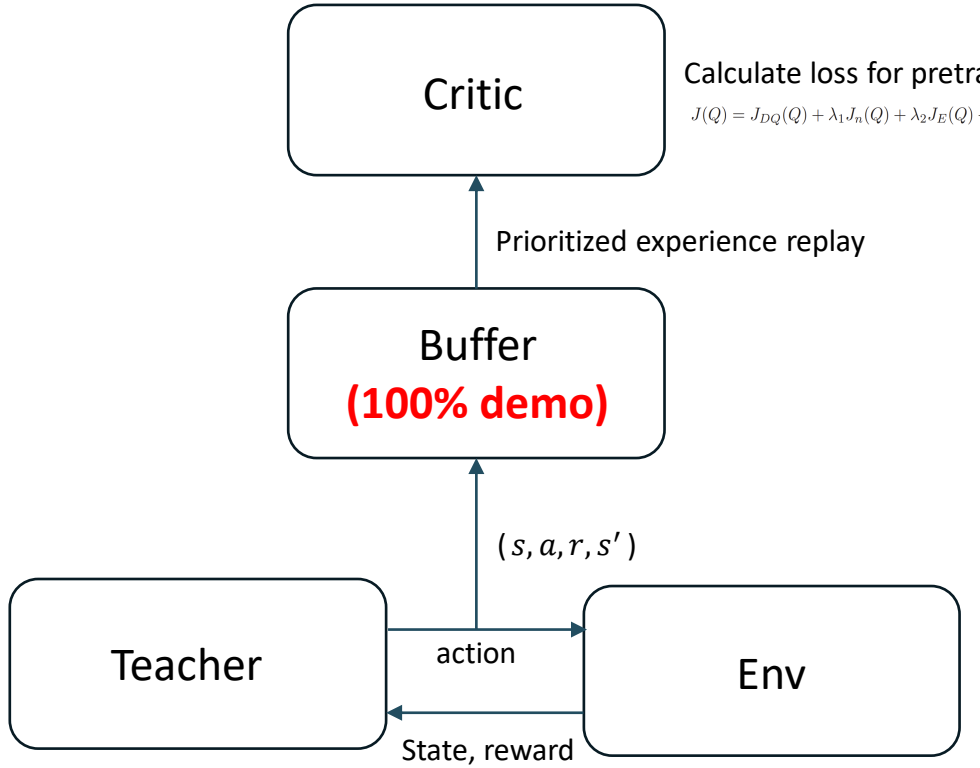
$$J(Q) = \underbrace{J_{DQ}(Q)}_{\text{TD-error}} + \underbrace{\lambda_1 J_n(Q)}_{\text{N-step TD error}} + \underbrace{\lambda_2 J_E(Q)}_{\text{Classification loss}} + \underbrace{\lambda_3 J_{L2}(Q)}_{\text{L2 regularization}}.$$

| Loss | Characteristics |
|---|---|
| N-step TD error | Propagates expert trajectory values to earlier states, enhancing pre-training |
| Classification loss | Pushes the value of the demonstrator's actions above the other actions value |
| L2 regularization | Prevent overfitting on the demonstration data |

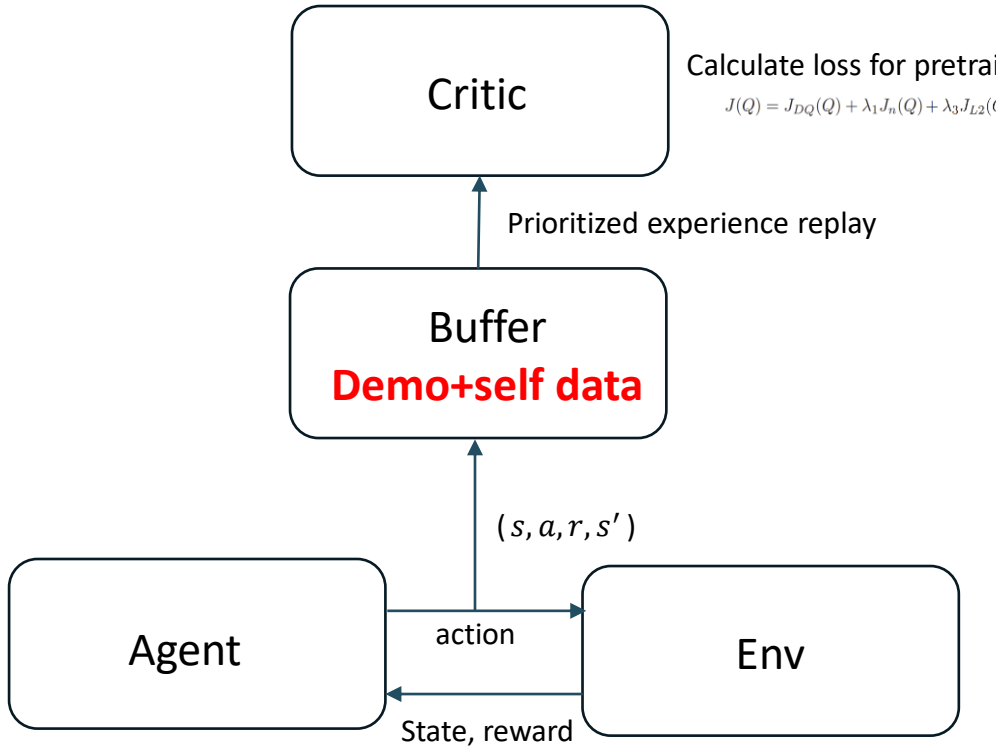
# Overall process

Utilizing demonstration data for pretraining [1] → Once a significant amount of data accumulates, begin gathering self-generated data. [2] → When the buffer is full, maintain the demo data and overwrite only the self-generated data with new data [2]

## Pretraining [1]




## Training [2]



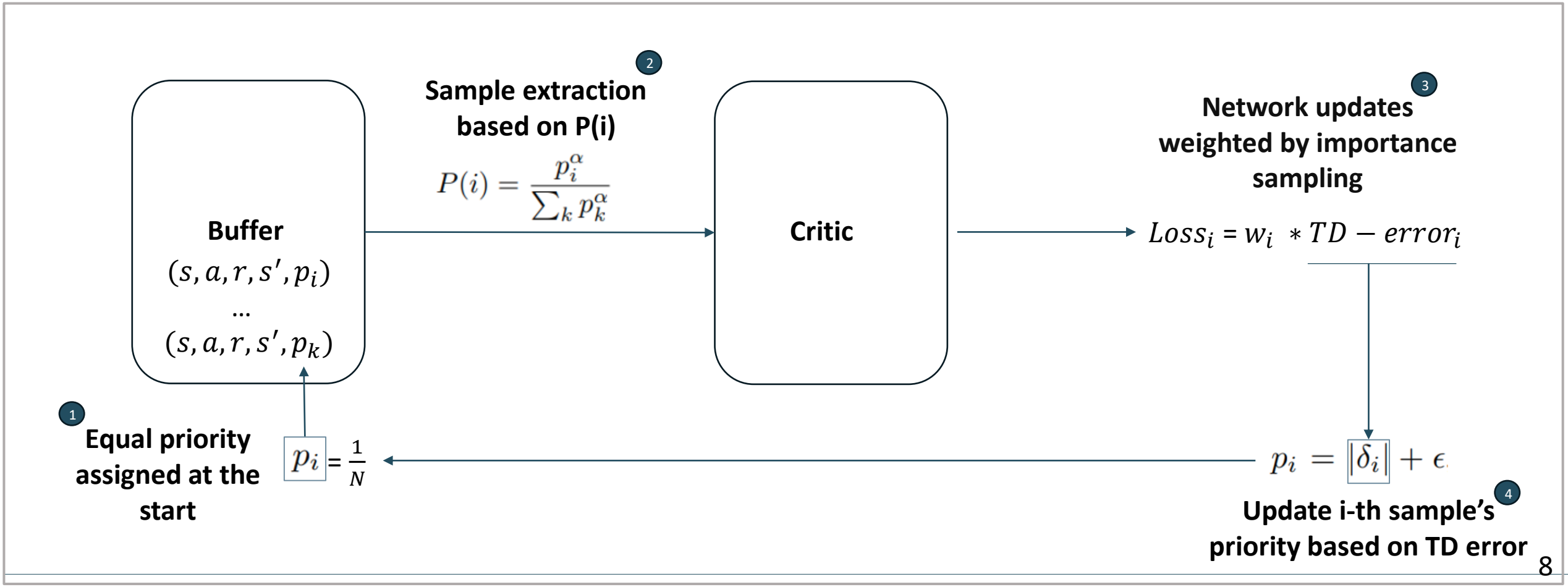
# PER (Prioritized Experience Replay)

$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^{\beta}$ = Sample i's importance sampling weight

i = sample index

N = current buffer length

- Prioritized sampling of samples with higher TD-error
- Utilize importance weights to ensure DQN's random sampling method



# Loss function

- Enables efficient utilization of demo data by separating loss into four types
- In pretraining phase, it use all loss but when self generated data selected, $\lambda_2$ becomes 0

$$J(Q) = \underbrace{J_{DQ}(Q)}_{\text{TD-error}} + \underbrace{\lambda_1 J_n(Q)}_{\text{N-step TD error}} + \underbrace{\lambda_2 J_E(Q)}_{\text{Classification loss}} + \underbrace{\lambda_3 J_{L2}(Q)}_{\text{L2 regularization}}.$$

| Loss | Characteristics |
|---|---|
| N-step TD error | Propagates expert trajectory values to earlier states, enhancing pre-training |
| Classification loss | Pushes the value of the demonstrator's actions above the other actions value |
| L2 regularization | Prevent overfitting on the demonstration data |

# Classification Loss

- Classification loss forces the values of the other actions to be at least a margin lower than the value of the demonstrator's action

$$J_E(Q) = \max_{a \in A} [\underbrace{Q(s,a)}_{\text{Q value}} + \underbrace{l(a_E, a)}_{\text{Margin function}}] - \underbrace{Q(s, a_E)}_{\text{Demo action's Q value}}$$

0 if Max(Q value) = demo action's Q value

Positive if Max(Q value) ≠demo action's Q value

# N-step TD Loss

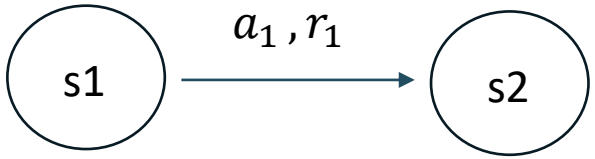
- Adding n-step returns helps propagate the values of the expert's trajectory to all the earlier states, leading to better pre-training

**N-step Loss**

$$J_n(Q) = \underbrace{r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + max_a \gamma^n Q(s_{t+n}, a)}_{\text{N-step return}} - \underbrace{Q(s, a_E)}_{\text{Demo action's Q value}}$$

**Example(1-step vs n-step)**

1-step TD




When updating Q($s_1, a_1$), only one expert action is used

n-step TD


When updating Q($s_1, a_1$), N expert actions are actually used

# L2 regularization

- Adding L2 regularization loss to help prevent it from over-fitting on the relatively small demonstration dataset
- Adding a weight term to the loss function prevents the weights from growing excessively

$$Loss = (TD - error)^2 + \underbrace{\frac{\lambda_3}{2}\sum_w w^2}_{\text{L2 regularization term}}$$

W = Model weights
$\lambda$ = Regularization strength

# ALE (Arcade Learning Environment)

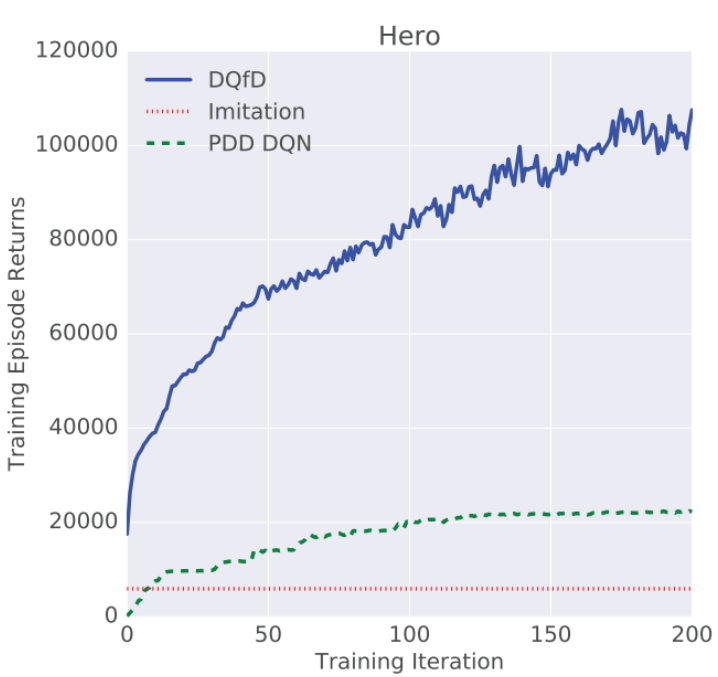
- ALE is a set of Atari games that are a standard benchmark for DQN
- Human player play each game between three and twelve time
- Human demonstrations range from 5,574 to 75,472 transitions per game

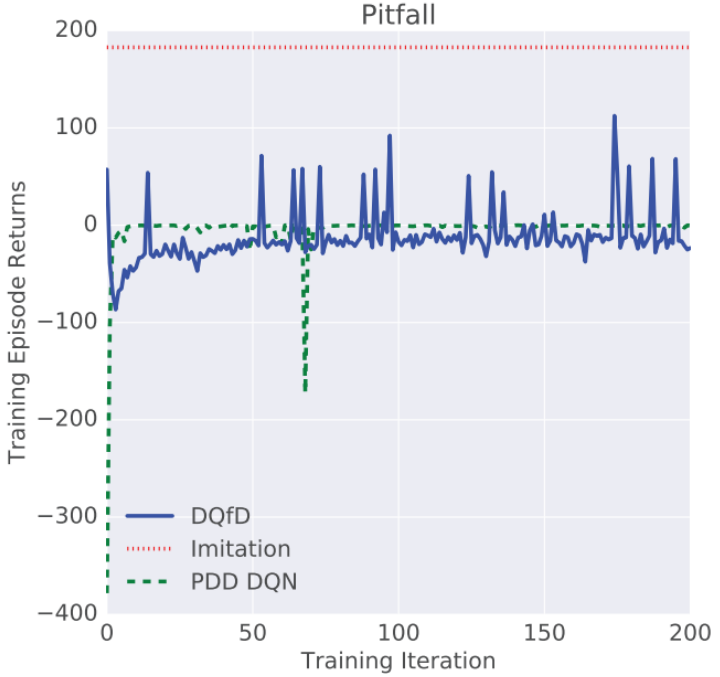
**Games in ALE**

# Results (Compare with PDD DQN and Imitation)

DQFD = proposed algorithm
Imitation = Only learns from pre-training
PDD DQN = PER+Dueling+Double DQN

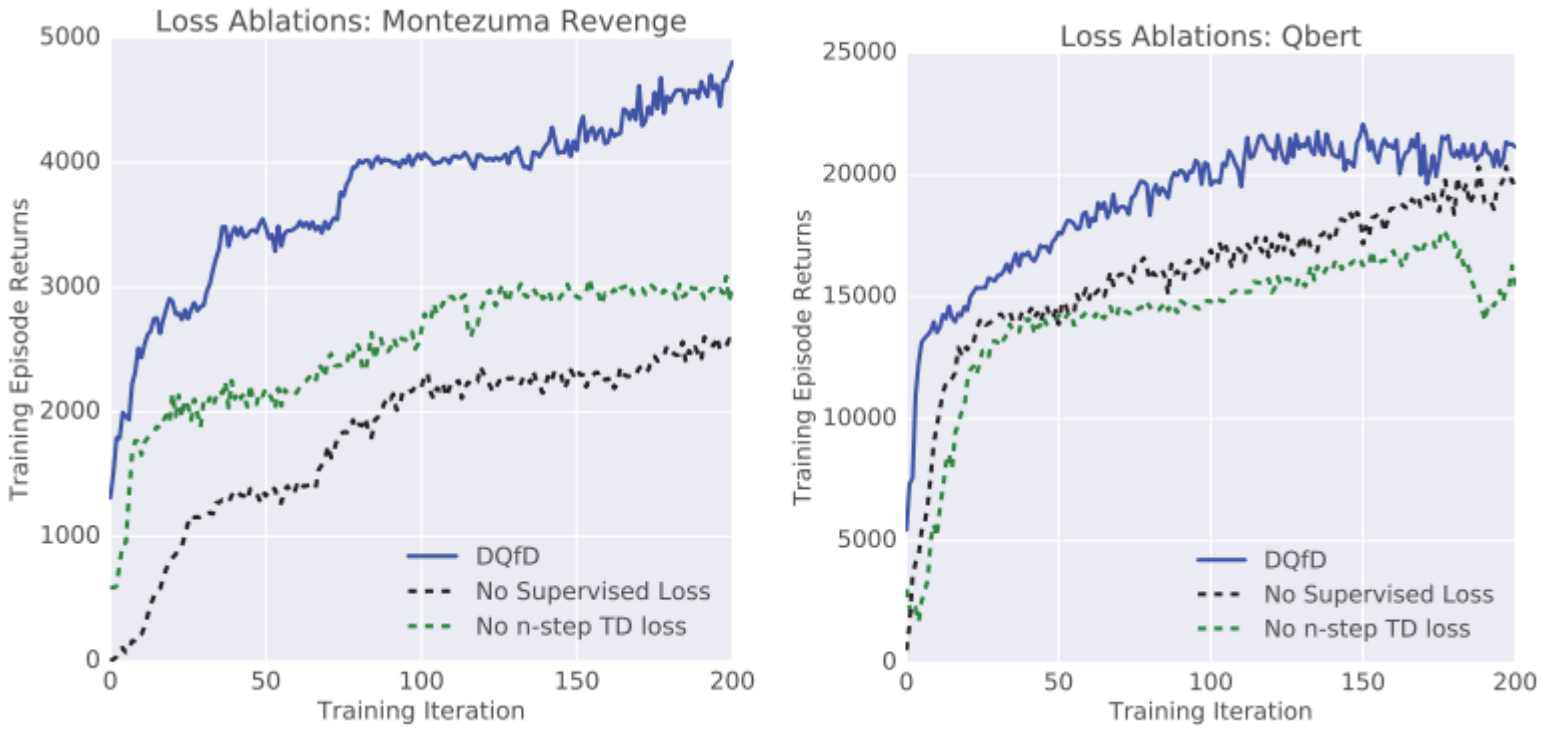
- In Hero, the performance of DQFD is significantly better compared to others
- In Pitfall, sparse positive and dense negative rewards result in the best outcomes from imitation




# Results (Loss Ablation)

- Without supervised loss in pre-training, the agent starts at a lower performance level and improves more slowly
- Removing n-step TD loss significantly affects initial performance as it aids learning from limited demonstrations



# Results (Compare with previous best)

- In 11 out of 42 games, DQFD outperforms previous best results
- Unlike the original method that evaluates 100 episodes at peak performance, DQFD assesses performance over 508 episodes

| Game | DQfD | Prev. Best | Algorithm |
|---|---|---|---|
| Alien | **4745.9** | 4461.4 | Dueling DQN (Wang et al. 2016) |
| Asteroids | **3796.4** | 2869.3 | PopArt (van Hasselt et al. 2016) |
| Atlantis | **920213.9** | 395762.0 | Prior. Dueling DQN (Wang et al. 2016) |
| Battle Zone | **41971.7** | 37150.0 | Dueling DQN (Wang et al. 2016) |
| Gravitar | **1693.2** | 859.1 | DQN+PixelCNN (Ostrovski et al. 2017) |
| Hero | **105929.4** | 23037.7 | Prioritized DQN (Schaul et al. 2016) |
| Montezuma Revenge | **4739.6** | 3705.5 | DQN+CTS (Ostrovski et al. 2017) |
| Pitfall | **50.8** | 0.0 | Prior. Dueling DQN (Wang et al. 2016) |
| Private Eye | **40908.2** | 15806.5 | DQN+PixelCNN (Ostrovski et al. 2017) |
| Q-Bert | **21792.7** | 19220.3 | Dueling DQN (Wang et al. 2016) |
| Up N Down | **82555.0** | 44939.6 | Dueling DQN (Wang et al. 2016) |

# Conclusions

- **It first pre-trains solely on demonstration data, using a combination of 1-step TD, nstep TD, supervised, and regularization loss**
- **Effects of additional losses were verified through experimentation**
- **DQfD is able to leverage the human demonstrations to achieve state-of the-art results on 11 Atari games**

# Future works

- **Plans are to conduct experiments on Dual Stokcer with the addition of PER and n-step TD loss**

# Q & A