Deep Reinforcement Learning: DQN, Double DQN, Dueling DQN, & Prioritized Experience Replay

James Chapman
CIS 730 Artificial Intelligence– Term Project
Kansas State University
jachapman@ksu.edu

I. Introduction

Artificial intelligence (AI) has proven to be a revolutionary frontier in society, evolving into a multibillion-dollar industry with far-reaching implications [1]. The demand for AI-related job skills is increasing across every industrial sector in the USA [2]. AI will continue to evolve rapidly, with new approaches being developed for increasingly complex tasks. Machine learning, a subset of AI, has been widely implemented and has improved decision-making in many applications. Perhaps the most impactful side of AI is deep learning and the various architectures of neural networks. The ability of deep learning models to perform pattern recognition and function approximation has proven to be invaluable to society, and deep learning will continue to drive the success of AI.

Reinforcement learning (RL) is a paradigm of machine learning that intersects control theory and involves decision-making within dynamic environments. The structure of reinforcement learning problems is characterized by an agent that interacts with the environment. The agent perceives the environment with observations (called states, s) and the agent takes actions, a, that subsequently affect the environment. These actions are guided by some policy, π, making decisions that are motivated toward achieving some goal, or reward, R. The state, action, reward relationship of agents in environments is foundational to reinforcement learning [3].

The Markov decision process is a mathematical framework that allows this agent-environment interaction to be formalized. A defining characteristic of a Markov decision process is that when an agent takes an action, in a given state, the outcome is nondeterministic, shown by the state-transition probability equation (1). The effects of actions have probabilistic results, making decision-making difficult. Furthermore, these actions affect not only the immediate rewards, but also future rewards. When faced with this uncertainty, reinforcement learning has a variety of methods and algorithms for optimizing the agent's decision-making. Generally, the goal of reinforcement learning is to maximize the cumulative reward, R. The state-value function, V, is a prediction of the cumulative reward, given policy π, (2). Traditional reinforcement learning involves dynamic programming and iterative methods that seek to optimize either the state-value function or the policy.

$$P(s'|s,a) = \Pr(s_t = s'|s_{t-1} = s, a_{t-1} = a) \tag{1}$$

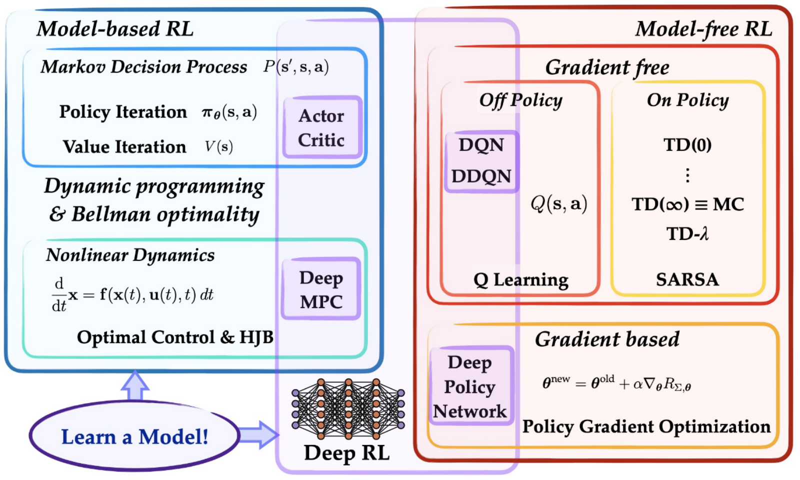
$$V^{\pi}(s) = \mathbb{E}[\, R_t \mid S_t = s\,] \tag{2}$$

$$V^{\pi}(s) = \textstyle\sum_{s'} P(s'|s,a)[R + \gamma V^{\pi}(s')] \tag{3}$$

The Bellman equation is an essential element in understanding the dynamics of reinforcement learning. The Bellman equation provides a recursive definition of the value function, V, under policy π (3). Gamma γ is a discount factor and an important hyperparameter that balances immediate and future rewards. This equation is important when rewards are sparse, a major problem in the reinforcement learning paradigm.

Reinforcement learning is a diverse field with a variety of algorithms to optimize action-choice. Figure 1 provides an overview of the numerous RL algorithms. A large division in the field is the use of model, or model-free, design. Modeling the environment with state-transition probabilities (1) and value function (2) defines model-based RL. The alternative, model-free design, generally optimizes either the policy directly (on-policy methods) or the Q-value (off-policy methods). Q-learning operates by learning an action-state function that gives the expected rewards of taking an action in a given state (Q-value). Q-learning is beneficial in complex environments and, since it does not learn the policy directly, Q-learning can learn from exploratory actions or even from simply observing.

Q-learning is a form of temporal difference learning, and the Q-value is based not only on the current reward but also on future potential rewards. Therefore, the Q-function can be represented as a Bellman equation (4). This recursively defines the current state-action pair, based on future Q-values (discounted by gamma γ) of actions following policy π. The goal of the algorithm is to learn an optimal policy of taking actions with the



Model-based RL
Markov Decision Process $P(s', s, a)$
Policy Iteration $\pi_\theta(s, a)$
Value Iteration $V(s)$
Actor Critic
Dynamic programming & Bellman optimality
Nonlinear Dynamics
$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t)\, dt$
Deep MPC
Optimal Control & HJB
Learn a Model!
Deep RL
Model-free RL
Gradient free
Off Policy
DQN
DDQN
$Q(s, a)$
Q Learning
On Policy
TD(0)
⋮
TD(∞) ≡ MC
TD-λ
SARSA
Gradient based
Deep Policy Network
$\theta^{new} = \theta^{old} + \alpha \nabla_\theta R_{\Sigma,\theta}$
Policy Gradient Optimization

Fig.1 An overview of algorithms in the field of reinforcement learning [4]

maximum Q-value, which is the Bellman optimality equation of Q-values (5).

$$Q^{\pi}(s,a) = \sum_{s',r} P(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s') Q^{\pi}(s',a')\right] \quad (4)$$

$$Q^{*}(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^{*}(s',a') \mid s,a\right] \quad (5)$$

Q-learning can be a powerful technique when a model of the environment is not fully known in advance. However, for many years reinforcement learning was held back and lacked practical application. In part, reinforcement learning was unable to handle large state-space problems and required too much hand-crafted tuning.

II. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines the principles of classical reinforcement learning with deep learning and the use of neural networks. This fusion has led to significant breakthroughs, enabling agents to operate in environments with high-dimensional state/feature spaces.

A. Deep Q-Network (DQN)

In 2013, Google's DeepMind research laboratory introduced Deep Q-Networks (DQN) [5,6]. DQNs utilize deep neural networks to approximate the Q-function. Neural networks proved to be robust enough to learn the state-action pair relationship. DQN's were a catalyst in DRL, launching Q-learning into a wide variety of applications. Key to this design is the loss function presented in the publication. Equation 6 defines Q-learning in terms of neural network weights, θ and θ-, where θ- denotes the parameters of a target network, a novel concept introduce. The target network is a copy of the main neural network that is updated at a frequency (treated as a hyperparameter). This is a form of temporal difference learning, where the Q-value is discounted by gamma γ.

$$L(\theta) = E\left[\left(r + \gamma \max_{a'} Q(s',a';\theta^{-}) - Q(s,a;\theta)\right)^{2}\right] \quad (6)$$

The training process of a DQN involves adjusting the weights of the neural network to minimize the difference between the predicted Q-values and the target Q-values, which are estimated using the Bellman equation. Algorithm 1 presents the steps of training. Neural network weight updates occur in batches of (randomly) sampled experiences from a buffer of experiences. Each experience is a tuple containing data on each state, action, reward, and next state (s, a, r, s'). Algorithm 1 displays this as (ϕ_t, a, r, ϕ_{t+1}). Experience replay was necessary for the networks to cope with the sequential nature of reinforcement learning. Neural network training suffers when data is correlated, and the Q-function needs to make decisions for all states.

Algorithm 1: deep Q-learning with experience replay.

```
Initialize replay memory $D$ to capacity $N$
Initialize action-value function $Q$ with random weights $\theta$
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ do
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T do
    With probability $\varepsilon$ select a random action $a_t$
    otherwise select $a_t = \mathrm{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the
    network parameters $\theta$
    Every $C$ steps reset $\hat{Q} = Q$
  End For
End For
```

B. Double Deep Q-Network

Since DQN was introduced, DRL research has exploded alongside the AI industry. It is important to study some of the advancements of DQN, and how they improve learning capabilities. In 2015, double Q-learning implemented in DRL by Hasselt, et al [7]. Double DQN is a subtle adjustment to the algorithm where the action selection comes from the primary network, but evaluated by the target network. Equation 7 shows the loss function of double DQN, notice the argmax function representing the primary network selecting the action. The motivation for this change is if the network chooses the action, it will also value that action higher. This creates an unstable measure of error, and the loss function is said to overestimate. When the primary network selects the action, the loss function becomes more representative of the primary network's need for update/backpropagation.

$$L(\theta) = E\left[\left(r + \gamma\, Q\left(s', \underset{a'}{\operatorname{argmax}}\, Q\,(s', a'; \theta); \theta^-\right) - Q(s, a; \theta)\right)^2\right] \quad (7)$$

C. Prioritized Experience Replay (PER)

In 2016, Schaul, et al. (Google DeepMind) published a new state-of-the-art in Atari DRL with prioritized experience replay [8]. Ordinary experience replay samples randomly from a buffer of experiences. This randomness breaks the correlation of sequential updates, as the environment proceeds. However, intuitively, certain experiences have higher impact on learning, such as key points in a game, and the model would learn more from those experiences. PER prioritizes experiences by their temporal difference error, which describes the surprise that the experience witnessed in the state transition.

PER does not greedily select the experiences with the highest priority for each batch update. Instead, experiences are sampled according to equation 8, where each experience, i , has priority, p_i , and probability, $P(i)$, of being sampled. This is governed by hyperparameter α (α = 0 is uniform random sampling). The loss function is modified, shown in equation 10, which includes weights, w_i . Weights are calculated at each update by equation 9. The use of weights in the loss function is governed by the hyperparameter, β, which increases to 1 during the training process.

$$P(i) = \frac{{p_i}^\alpha}{\sum_k {p_k}^\alpha} \quad (8)$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)}\right)^\beta \quad (9)$$

$$L(\theta) = \sum_i w_i \cdot \left(y_i - Q(s_i, a_i; \theta)\right)^2 \quad (10)$$

D. Dueling Deep Q-Network

Dueling DQN introduces another change to the standard DQN outline [9]. The Dueling DQN design changes the architecture of the neural network, which approximates Q-values. This new architecture splits the fully connected layers of the Q-network into 2 pathways (Fig. 2). Both pathways have interpretable definitions. Conceptually, the value stream, V(s), estimates the value of the state, independent of the action (see equation 10). The advantage stream, A(s,a), computes the advantage of certain actions over others. This has shown to have significant benefits in certain learning environments and more efficient learning because the network can learn which states are (or are not) valuable, independent of the action taken.

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a)\right) \quad (10)$$

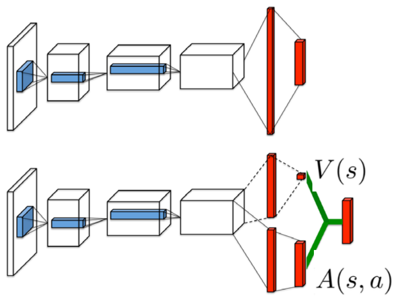


Fig. 2 An outline of DQN neural network design (above) compared to a Dueling DQN design (below) [9]

III. Experiment Design

The Arcade Learning Environment (ALE) was originally published and released in 2012 [10]. ALE is an evaluation platform for reinforcement learning research and other AI technologies. It provides an interface to the Atari 2600 games which is great for research projects because of its variety in complexity and ability to benchmark and compare algorithms. With ALE, researchers can study the interpretability of machine learning methods in a reproducible and controlled environment. The complete control of environment allows researchers to focus on foundations of reinforcement learning such as the trade-off between exploration and exploitation. OpenAI's Gym provides additional support and is widely popular in the research community [11]. In 2022, the Farama Foundation took over Gym (re-naming it to Gymnasium) [12].

This project uses Gymnasium as an interface for 2 of the Atari 2600 games. The Gymnasium environment "PongNoFrameskip-v4" was used to play the game of Pong and "BreakoutNoFrameskip-v4" was used to pre-the game of Breakout. Much of the original design from DeepMind's DQN paper was used. Figure 3 outlines the general schematic of DQN in Gymnasium [6]. Atari 2600 games supply screen images of 210 x 160 pixel RGB images at a frequency of 60 Hz.

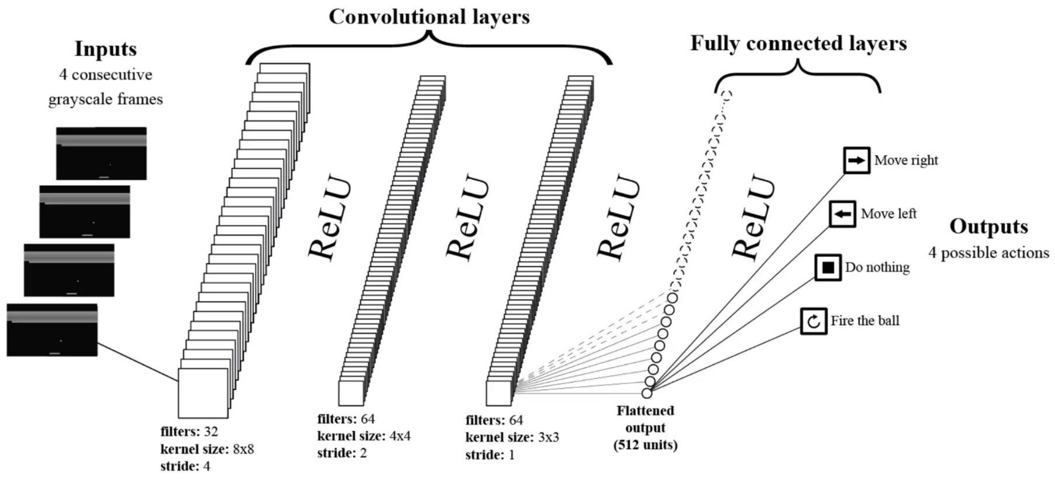


Fig. 3 Schematic of DQN processing in Gymnasium

Essentially, the only output from our agent design is the choice of action at each frame.

A Jupyter notebook was created for training the agent & network for this project (see section V. Data & Code). The original design of the training notebook extends from an online tutorial [14], which implements DQN for Pong in OpenAI's Gym. However, this proved problematic when transferring to Gymnasium, especially when implementing Breakout. Gymnasium uses an object orientated module of Wrappers classes that distills the images from the high dimensional feature space into features suitable for the convolutional neural network (see figure 3). Wrappers not only process pixels (by interpolating pixel values and converting to grayscale), but also concatenates consecutive frames (usually 4) so to capture the environment dynamics of the current state, s. The wrappers were chosen based on other projects found online. This project seeks to examine the connective tissue of deep reinforcement learning so the implementation details and hyperparameters were not extensively explored.

To evaluate models, the neural network parameters are saved every 250,000 frames of training. Another Jupyter notebook was created that used these saved networks in a game environment with slight stochasticity. The networks play 150,000 frames of the game and the mean and standard deviation of the final game scores are recorded. The interval with the highest mean are reported for each model.

IV. Results & Discussion

The 4 DQN variants were implemented in both environments (Breakout & Pong). Table 1 lists the highest mean value of test intervals for each of the 8 models. Appendix A provides the training curves for each model. Notice Pong is a much simpler game and training is very flexible. The Breakout environment has greater complexity and makes the training very sensitive. This is been investigated since the first publication DQN. There is a major theme amongst reinforcement learning algorithms that contrasts exploration and exploitation. Notice the large standard deviation as the Breakout agents grow in mean rewards. This may be evidence of a lack of generalization capabilities of the network. A more robust agent should not witness this deviation.

There is a wide variety of implementation details and large combinatorial hyperparameter space. Deep reinforcement learning has continued to progress. Table 1 includes the improvements as these variants were introduced. Furthermore, the current state of art as reported by Papers with Code [13] shows the potential for neural networks to capture complex behavior in high dimensional feature spaces. Longer training and increased computing power (GPU, RAM, etc.) will also drive the success of DRL.

Table 1: Results of Deep Reinforcement Learning in Gymnasium

Model	Breakout	Pong
DQN	112.4 +/- 57.6	20.8 +/- 0.5
Double DQN (DDQN)	96.5 +/- 43.5	20.8 +/- 0.5
DDQN w/ PER	43.5 +/- 13.4	20.8 +/- 0.5
Dueling DDQN w/ PER	**207.3 +/- 124.0**	**20.9 +/- 0.4**
Random [5]	1.2	- 20.4
Human [5]	31	- 3
DQN [5]	168	20
DDQN [7]	**375.0**	**21.0**
DDQN w/ PER [8]	371.6	18.9
Dueling DDQN w/ PER [9]	366.0	20.9
State-of-the-art [13]	864	21

V. Conclusion & Future Work

Researchers have continued to improve DQN methods. Notably, the Rainbow algorithm [15] uses the 4 models of this project, along with 3 other variants (A3C, Distributional DQN, and Noisy DQN It's adaptability across various domains makes it popular in areas such as robotics, finance/trading, and gameplay.

V. Data & Code

CODE

https://github.com/JamesChapmanNV/Deep_Reinforcement_Learning

References

[1] Maslej, N. et al. (2023). Artificial Intelligence Index Report 2023 Stanford University. doi.org/10.48550/arXiv.2310.03715

[2] Chui, M. (2017). Artificial intelligence the next digital frontier. McKinsey and Company Global Institute, 47(3.6). odbms.org/2017/08/artificial-intelligence-the-next-digital-frontier-mckinsey-global-institute-study/

[3] Sutton, R. S., & Barto, A. G. (2020). Reinforcement learning: An introduction. The MIT Press.

[4] Brunton, S. L., & J. Nathan Kutz. (2019). Data-Driven Science and Engineering. Cambridge University www.databookuw.com

[5] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. doi.org/10.48550/arXiv.1312.5602

[6] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015). doi.org/10.1038/nature14236

[7] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." Proceedings of the AAAI conference on artificial intelligence. Vol. 30. No. 1. 2016. doi.org/10.48550/arXiv.1509.06461

[8] Schaul, T., Quan, J., Antonoglou, I. & Silver, D. (2015). Prioritized Experience Replay Published at ICLR 2016 doi.org/10.48550/arXiv.1511.05952

[9] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2016. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48 (ICML'16). JMLR.org, 1995–2003. doi.org/10.48550/arXiv.1511.06581

[10] Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47, 253-279. https://doi.org/10.48550/arXiv.1207.4708

[11] A maintained fork of OpenAI's Gym library. Gym Documentation. (n.d.). https://www.gymlibrary.dev/

[12] Towers, M. et al. (2023, March 25). Gymnasium API. Zenodo. https://doi.org/10.5281/zenodo.8127026

[13] Papers with code - atari 2600 pong benchmark (Atari Games). The latest in Machine Learning. (n.d.). https://paperswithcode.com/sota/atari-games-on-atari-2600-pong

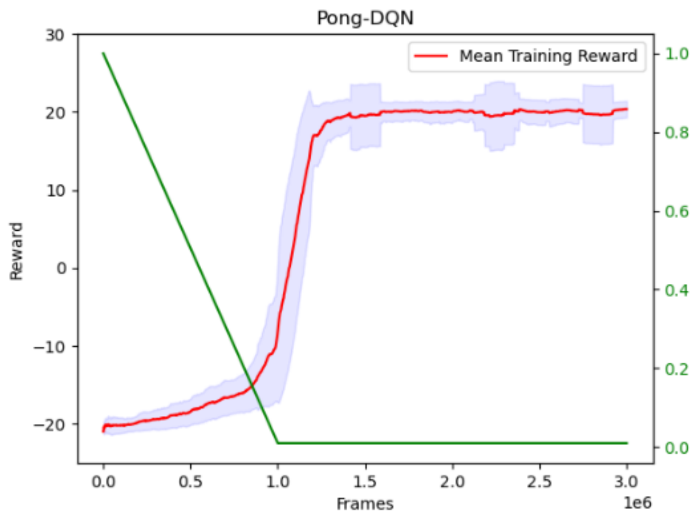
[14] Torres, Jordi. "Deep Reinforcement Learning Explained." *GitHub*, github.com/jorditorresBCN/Deep-Reinforcement-Learning-Explained/tree/master. Accessed 15 Dec. 2023.

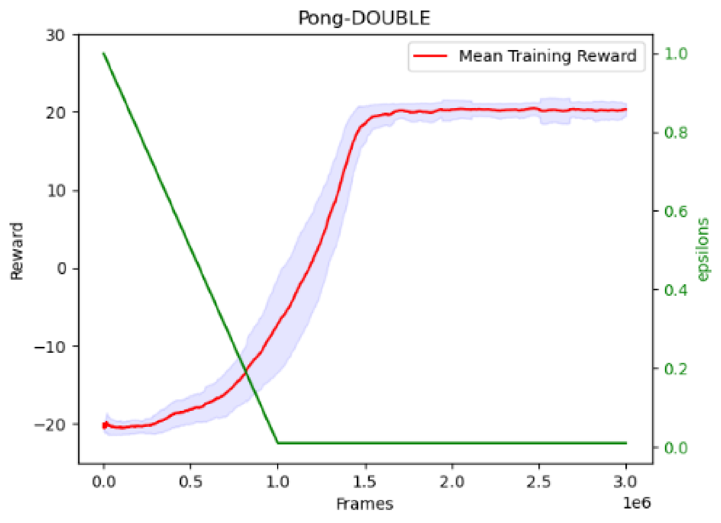
[15] Hessel, M., Modayil, J., Hasselt, H.V., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M.G., & Silver, D. (2017). Rainbow: Combining Improvements in Deep Reinforcement Learning. AAAI Conference on Artificial Intelligence. doi.org/10.48550/arXiv.1710.02298

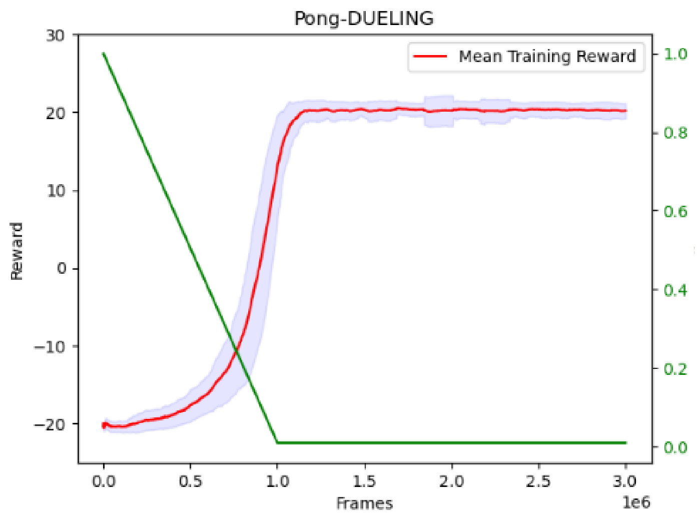
[16] Badia, Adrià & Piot, Bilal & Kapturowski, Steven & Sprechmann, Pablo & Vitvitskyi, Alex & Guo, Daniel & Blundell, Charles. (2020). Agent57: Outperforming the Atari Human Benchmark. doi.org/10.48550/arXiv.2003.13350

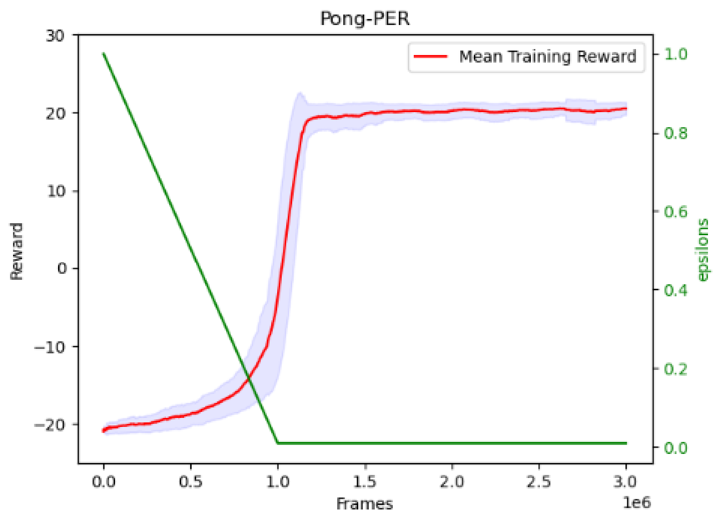
[17] Fan, J., & Xiao, C. (2022). Generalized Data Distribution Iteration. International Conference on Machine Learning. doi.org/10.48550/arXiv.2206.03192

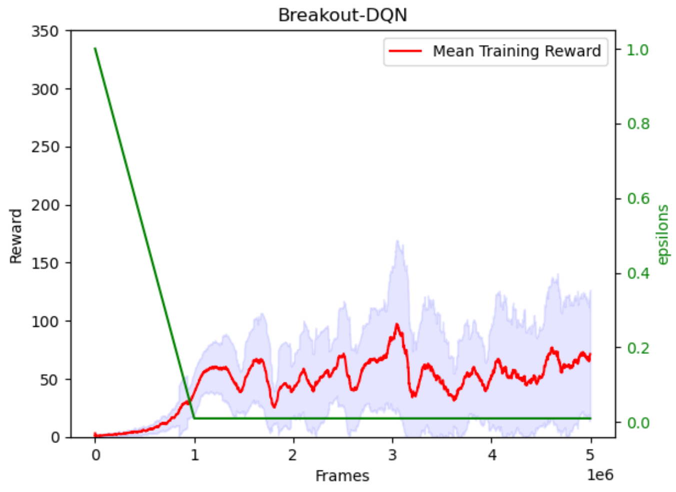
APPENDIX A:
Training Curves

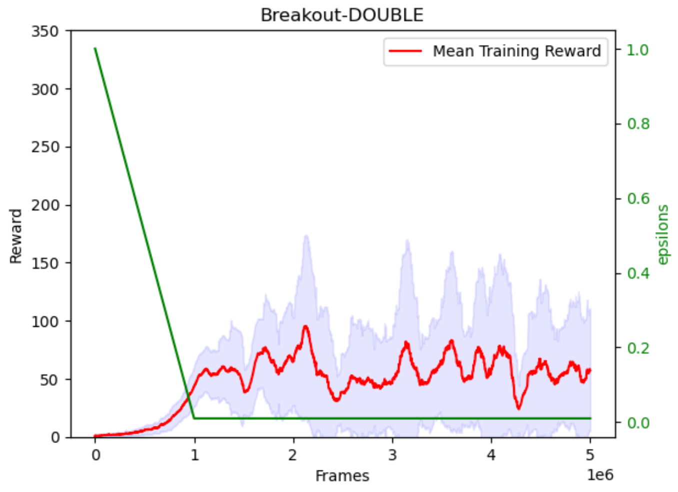


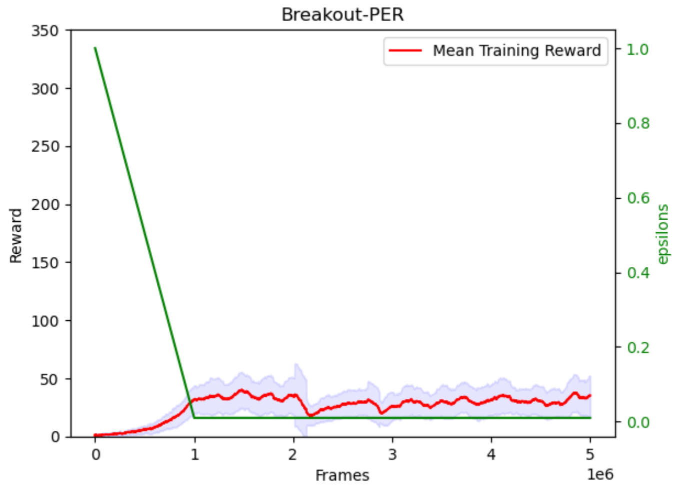


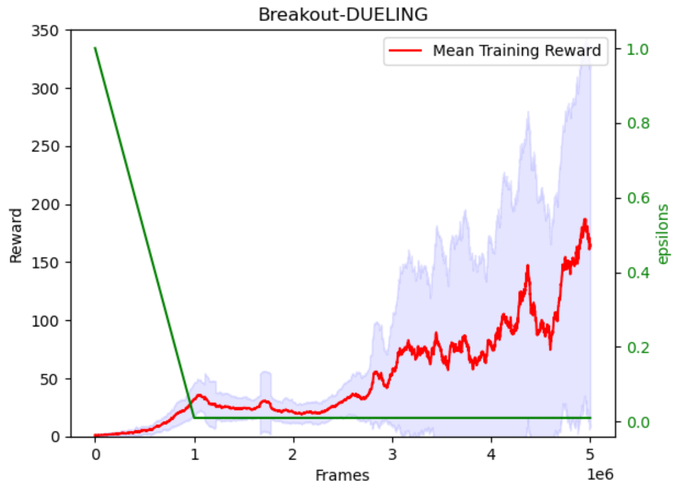





Breakout-DQN
Mean Training Reward
Reward
epsilons
Frames
1e6
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5


Breakout-DOUBLE
Mean Training Reward
Reward
epsilons
Frames
1e6
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5


Breakout-PER
Mean Training Reward
Reward
epsilons
Frames
1e6
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5


Breakout-DUELING
Mean Training Reward
Reward
epsilons
Frames
1e6
0
50
100
150
200
250
300
350
0.0
0.2
0.4
0.6
0.8
1.0
1
2
3
4
5