Comparison of Reinforcement Learning Methods Using Deep Q-Learning

Trevor Rizzo
Robotics Engineering, Worcester Polytechnic Institute, MA, USA
Email: tarizzo@wpi.edu

***Abstract*— This paper will demonstrate and compare the performance of three methods of training a deep reinforcement learning model; traditional reinforcement learning, transfer learning, and imitation learning. These methods will be used to train a Deep Q-learning model to act as an agent in the Super Mario World video game. The implemented models will be evaluated based on their final model performance, initial performance, and time to converge.**

I. Introduction

This paper uses Deep Q-learning models to compare the characteristics of multiple reinforcement learning training techniques. This section will provide a brief introduction into the fundamental concepts of Deep Q-learning, and it's broader category of reinforcement learning methods, Temporal Difference (TD) learning.

Temporal difference learning is a category of model-free reinforcement learning methods that combines concepts from both Monte Carlo methods and Dynamic Programming methods. Like Monte Carlo learning, Temporal Difference learning learns through raw experiences and does not require a predefined model of the environment. Both Monte Carlo and Temporal Difference learning use sampling, meaning the value function is only updated based on the raw experiences of the agent and not the expected value of state and/or action. Temporal Difference learning methods share the concept of bootstrapping, a concept introduced in the Bellman equations, with Dynamic Programming methods. With bootstrapping, the value estimate of a state/action is updated based on later learned estimates in the episode. This method allows Temporal Difference learning methods to update value estimates before the terminal state is reached, unlike Monte Carlo methods. Equations 1 and 2 show the value function evaluation functions of Monte Carlo learning and TD(0), the most simple Temporal Difference learning implementation.

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \tag{1}$$

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2}$$

These equations show that while Monte Carlo learning evaluates the value function based on the actual value returned at time t (Gt), while Temporal Difference learning uses an estimate of the the return at time t+1 to update the estimate the value at time t. Unlike Monte Carlo methods, Temporal Difference learning methods can learning without knowing the ultimate outcome of an action/epsiode.

Q-learning is an off-policy Temporal Difference Learning method that works by learning the value-action function (Q) for an action (a) and a particular state (s). The off-policy nature of Q-learning allows for an estimate of the optimal policy to be learned independent of the agent's actions during the learning process. In other words, the policy being followed during the process is not the same policy that is being learned. As in other Temporal Difference learning methods, Q-learning uses the Bellman equations to bootstrap the value estimation process. Q-learning uses an epsilon-greedy strategy to balance the exploration, exploitation process. At the start of the learning process the agent has it's highest epsilon values, enabling the agent to explore the environment thoroughly. As the agent progresses through the learning process, the epsilon value decreases, which makes the agent more likely to exploit the environment. Just as in other Temporal Difference learning implementations, Q-learning will converge to a final policy [1], [2].

Deep Q-learning follows the same fundamental concepts as Q-learning, but takes advantage of DNNs to expedite the learning process. Deep Q-learning uses DNNs to represent the agent's value function, policy and the model.

II. Related Work

The development of Artificial Intelligence (AI) techniques for video games is an expansive research field that encompasses both academia and industry. Very commonly deeep learning methods are used to develop these video game playing AIs. Deep reinforcement models have been developed and trained to perform in various categories of video games including 2D arcade games, racing games, strategy games, text adventure games and first-person shooter games. These categories and the games within them encompass many different degrees of inputs and outputs and environments that exist in 2D and 3D [3].

The video game environment used for this paper, Super Mario World, fits into the 2D arcade category of video games. Deep Q-learning was the first reinforcement learning method that was able to reach expert-level human abilities in a 2D arcade game. [3], [4] This performance was achieved by reducing the computations required to train a Deep Q-Network (DQN) by parameterizing an approximate value function which had not achieved previously [4].

Mnih et al. have developed a generalized DQN architecture for learning 2D video games for the Atari 2600. This network is trained using only the raw pixel information as input; 210 x 160 pixel images with 128 possible colors. The input is down sampled from RGB color space to a grayscale image with dimensions of 110 x 84 pixels, then cropped to

84 x 84 pixels. Unlike previous DQN implementations for video games. This implementation does not use the history or action as inputs to the neural network, only the pre-processed pixel data. This allows for all of the Q-values for all possible actions to be update with a single forward pass. The remaining sections of this DQN include 3 hidden layers and a full connected output layer with an output for each possible action for the chosen game. Additionally, the DQN proposed by Mnih et al. is trained using small stochastic batch sizes and experience replay memory. The experiments performed with this network used the same architecture and hyperparameters for each game that the DQN was trained on. The resulting DQNs reached state-of-the-art level of performance for 6 out of the 7 games tested [5].

The General Video Game AI (GVGAI) competition is an AI development competition that provides a method for benchmarking AIs against various genres of video games. This tools enables AI models to be evaluated in a method that shows the models' level of generalization. Torrado et al. connected this benchmark to OpenAI gym and used this new tool to evaluate the performance of three common deep reinforcement models; 1) DQNs, 2) Dueling DQNs, and 3) Advantage Actor-Critic (A2C). The GVGAI-OpenAI benchmark showed the for most of the games, the chosen reinforcement learning models were able to converge. The performance across the various games was extremely variable and showed that the reinforcement learning algorithms in their standard form do not have the necessary attributes to generalize across a wide variety of video games [6].

Reinforcement learning methods including DQN and transfer learning have many applications outside of video games. Sharma et al. demonstrate how DQNs and Q-matrix transfer learning can be used to learn fire evacuation routes. In addition to implementing these models, Sharma er al. also design an OpenAI gym environment to model dynamic fire evacuation environments. The novelty in this paper is how the transfer learning as applied to the DQN. First tabular Q-learning is used to learn the q-values for the shortest exit path in a none-emergency environment. These q-values are transfer to a DQN agent which learns in a fire evacuation environment. The results show that this transfer learning method decrease the time to converge of the DQN and was able to outperform all the other Deep Learning methods analyzed including SARSA, A2C, and Dueling-DQN [7]. This study shows how using a simple model to initialize the weights of a more complex model through transfer learning can reduce convergence time and increase overall performance.

III. Problem Statement

The goal of this paper is to implement and evaluate a Deep Q-learning model training using three popular methods of training reinforcement learning models. The models will be training to play the video game Super Mario World using the OpenAI Gym Retro reinforcement learning environment. First a Deep Q-learning model will be training using traditional deep reinforcement learning techniques, this will be the baseline for performance and time to converge. Then two additional models will be training one using transfer learning and another using imitation learning. The transfer learning method will explore pre-training a model on a video game that is similar to Super Mario World and and pre-training a different DQN on an environment that is unlike Super Mario World. The imitation learning implementation will be trained from the actions of a human agent in the Super Mario World environment.

IV. Solution

This section will detail the methodologies used to develop a solution for the problem statement

A. Deep Q Network Design

The first step in evaluating methods for training Deep Q Networks was to design a DQN implementation that was modular and could be easily adapted to the different training techniques. The class at the center of this modular DQN implementation is the "Agent" class. The agent class contains all the necessary functions and components to train a DQN for any environment. The ultimate goal of the agent is to populate it's replay memory with experiences from the environment and use those experiences to train a target network to accurately predict the Q-values for the environment. The implemented uses the same neural network architecture for the policy DQN and the target DQN. The DQNs take a batch of stacked game frames (batch of 3D tensors) as inputs and output a single-dimension tensor with a length equal to the total number of possible button-press combinations for the environment. The stack frames used as the input are the last n frames output by the environment and are passed to the DQN to allow for the perception of motion. The architecture of the DQN consists of 3 2D conventional layers with RELU activation and 4 fully connected layers with RELU activation. The fully connected layers also have a 80% dropout rate to contribute to the generalization of the trained model. While training, the policy net is used to choose an action based on the current environment state. After each action is performed in the environment, the replay memory is update and the policy DQN and target DQN are used to approximate the loss of the policy DQN and its weights using backpropagation. During the training process both Huber loss and Mean Squared Error (MSE) were tested. Huber loss showed immediatly better results over MSE and was used for the remainder of the training sessions.

B. Deep Q Network Training

After the basic architecture of the agent and the Deep Q Networks was completed, it took several iterations of training technique adjustments and hyperparameter tuning to achieve a trained network with satisfactory results. In the first attempts at training the network, the only preprocessing performed on each frame was converting the RGB color-space frame into a grayscale frame. After many attempts at training it became apparent that this was not a sufficient level

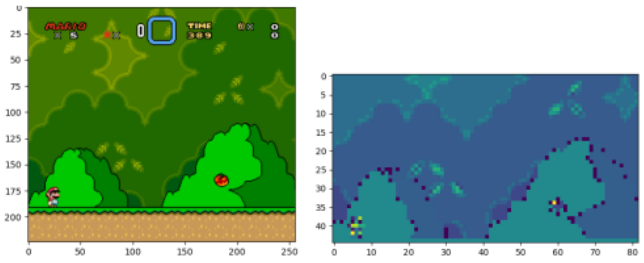

Fig. 1. Raw frame (left) and processed frame (right).

of prepossessing. The large size of the frame limited the maximum size of the replay memory due to RAM limitations. Additionally, the larger frame size also caused the training process to be extremely slow, even when training on a GPU. Only certain areas of each frame contain information that is relevant to the agent's task of progressing through the level. The final iteration of the frame prepossessing crops out all the necessary areas of the frame and then down samples the remaining pixels to 1/3 of their original resolution. Improving the frame prepossessing improved the training speed and the results of the DQN. An example of a raw and processed frame can be seen in Fig. 1

In addition to prepossessing the frames from the environment, frame skipping was also implemented to improve the performance of the DQN. The frame skipping was implemented by applying each chosen input to the environment for 5 frames before calculating the reward and updating the model's weights. Applying the input over 5 frames exaggerates the impact of the action on the environment and allows the DQN to associate actions and rewards more easily.

The agent's possible outputs were also limited to reduce the number of episodes required to train an agent. The objective of Super Mario World is to move right while jumping on and/or avoiding enemies. To limit the output space of the agent's DQNs, the possible outputs were reduced to only the buttons that allow the agent to move right, run and jump. Additionally, the agents ability to not move was taken away to avoid situations where a local minima may cause the agent to not move in certain states.

The game score was used for the reward when training the agent's DQNs. In Super Mario World, the score is increased when an enemy is jumped on, when a coin is collected, and when the end of the level is reached. Furthermore, the score increase caused by reaching the end of the level changes inversely with the amount of time it took to reach the end. As a result, using the game score as the reward incentives completing levels as quickly as possible. The training set for Super Mario World consists of 4 levels, each representing a subset of the most basic levels in Super Mario World. The 4 level in the training set are "YoshiIsland2", "YoshiIsland1", "DonutPlains1", and "Forest1". During the training process, agents were trained on each level individually and one agent was trained using all 4 training levels. Each level was trained on for 150 episodes.

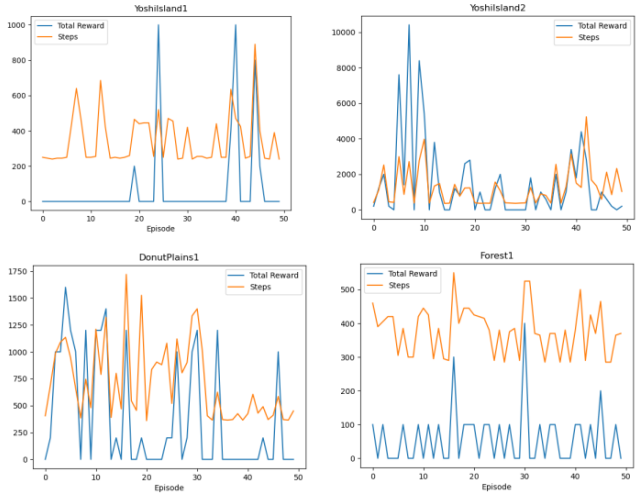

Fig. 2. Results of training on training levels individually.

C. Imitation Learning

D. Transfer Learning

The transfer learning experiment took place after the initial training and tuning of the Super Mario World DQN agent. Using the same agent parameters and DQN architecture as the Super Mario World DQN agent, a new agent was training using the game Super Mario Bros. Super Mario Bros. Was chosen for the transfer learning experiment because it has a very similar environment to Super Mario World, but it is a simpler game that has the potential to be easier to train on. Once the agent was trained on Super Mario Bros., it's policy DQN's weights were saved and used as the starting point to train a new Super Mario World agent. The hyperparmeters used when training the Super Mario Bros. agent and the new Super Mario World agent were identical to the original Super Mario World Agent.

V. Results

The DQN developed to play Super Mario World was able to successfully complete the levels that it was trained on. The results of training on each level individually can be seen in Fig. 2, the results of training the DQN on all the levels together can be seen in Fig. 3. Both of these results show that the developed DQN is able to achieve human-like results. Their are significant variations in the rewards achieved at each episode in every level. This inconsistency could mean that more episodes are need to further train the model or that the hyperpararmeters, such as learning rate and batch size, need to be adjusted further. Overall the DQN that was trained using all of the training levels outperformed the individually trained DQNs. The agent was able to complete all of the training levels except for "Forest1". This can be attributed to "Forest1" being both the most difficult level in the training set and "Forest1" having the most unique features when compared to the rest of the training set.

Training the agent using all the training levels was as performed after loading the weights from the Super Mario

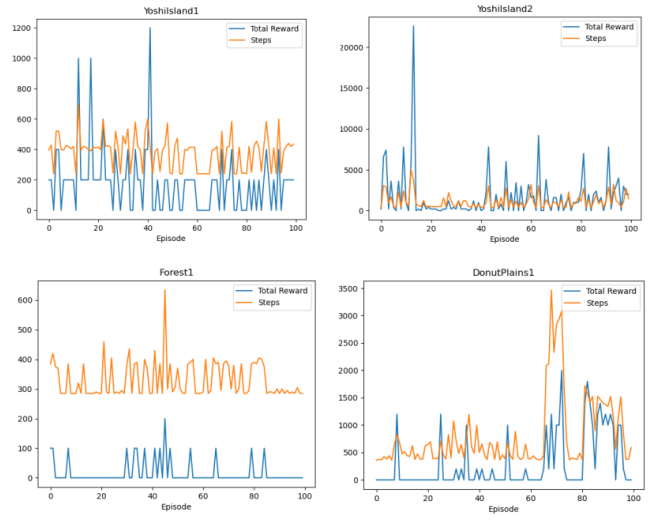

Fig. 3. Results of training on training levels as one training set.

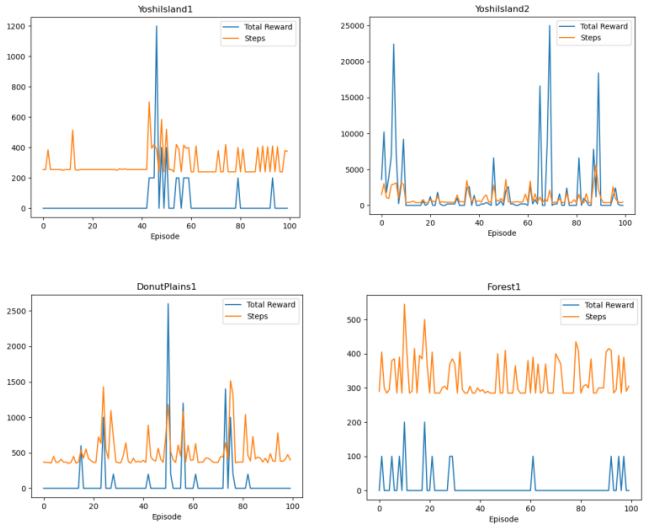

Fig. 4. Results of training on training levels as one training set after initializing weights from the Super Mario Bros. model.

Bros. agent. The results of this training can be seen in Fig. 4. Applying transfer learning to the training process resulted in no significant difference in the agent's performance. This result aligns with the results from the normally-trained agent, showing that the agent's parameters must be further developed to achieve a more consistent result.

VI. Conclusion

The results of this research show that while Deep Q Learning is a powerful algorithm that can provide human-like performance in many types of environments, it is also a very complex algorithm that even with a significant amount of time and research, may not yield desired results. For future work I would like to expand on this research and determine what steps need to be taken to make the resulting agent more consistent.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[2] C. Morato, *Class Lecture, Topic: Deep Reinforcement Learning*. Worcester Polytechnic Institute, Massachusetts: RBE595, Robotics Engineering, April., 11, 2021.

[3] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *IEEE Transactions on Games*, vol. 12, no. 1, pp. 1–20, Mar. 2020. [Online]. Available: https://doi.org/10.1109/tg.2019.2896986

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[5] V. Mnih, "Playing atari with deep reinforcement learning."

[6] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game AI," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Aug. 2018. [Online]. Available: https://doi.org/10.1109/cig.2018.8490422

[7] J. Sharma, P.-A. Andersen, O.-C. Granmo, and M. Goodwin, "Deep q-learning with q-matrix transfer learning for novel fire evacuation environment," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–19, 2020. [Online]. Available: https://doi.org/10.1109/tsmc.2020.2967936