Playing Atari with Deep Reinforcement Learning

CS519 Home Work #5

Matthew Frantz

June 7, 2019

1 Non-technical background

The paper that I chose was Playing Atari with Deep Reinforcement Learning from DeepMind Technologies. Authors for this paper are: Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Daan Wierstra, Alex Graves, Ioannis Antonoglou, and Martin Riedmiller. Each of the authors is from DeepMind Technologies which is a British artificial intelligence (AI) company founded in 2010 and was purchased by Alphabet (Google's parent company) in 2014. Several important AI algorithms have come out of DeepMind over the past several years, most famous is the AlphaGo algorithm that bested humans in the game of Go. The conference paper appears to have Martin Riedmiller as the PI for this paper, but the similar journal has Demis Hassabis who is the founder of DeepMind Technologies. Since I will be focusing on the conference paper I will assume Martin Riedmiller as the PI. All of the Co-Authors were employees of DeepMind at the time of this publishing.

This paper has two separate versions; a conference and journal version. The conference version was originally published in Conference on Neural Information Processing Systems (NIPS) 2013 and is available via arXiv [1]. NIPS is is a machine learning and computational neuroscience conference held annually. The journal version of the paper was published in Nature [2]. Both versions of the paper have fairly large impact factors with 2339 citations for the conference version and 5618 citations according to Google Scholars at the time of this writing. The trend for both articles being cited is similar with the number of citations roughly doubling each year since the papers were first published in 2014/2015. This tells me that the work has been gaining a lot of attention in the field. There were several articles over the years discussing this approach. In 2015 when the paper was published The New Yorker magazine published an article about the spectacular demonstration of the algorithm during NIPS [3] where it learned to play the game breakout allowing people see it learn in real time. The article mentioned that it went from totally inept, to world caliber in roughly 3 hours of training. In 2018 Venture Beat (a website which reports on technology) posted an article discussing how algorithms like Deep Q-Learning continue to surpass humans in video games [4].

Although there is not a dataset available for this experiment, they did make the code available[1] for use. They did not include the emulator they use but that can be downloaded separately. Since the author has been highly sited I imagine the code is also influential. There are several videos available demonstrating this algorithm but there is one that demonstrates the Breakout game performance described above that is particularly good[2]. Based on the journal article it appears that this code is very robust. They tried it on over a dozens different Atari games, with very little changing between games, and achieved excellent results on several of them by outperforming the human expert. This type of reproducibility most likely contributed to its large number of citations and spurred an entire area of research into Deep Q-learning.

2 Core

The authors were looking to tackle the challenge of learning control policies directly from high-dimensional inputs, images from a video game in this instance. According to the authors this had been, "one of the long standing challenges of reinforcement learning (RL)[2]". Similar approaches had been attempted previously, most notably was the TD-gammon which used temporal difference learning, a form of reinforcement learning, combined with a neural network to teach itself how to play backgammon from scratch [5]. This approach however had less success with games like chess and Go with the authors speculating that since backgammon requires dice rolls it can better explore the state space due to its more stochastic behavior. Another approach was neural fitted Q-learning (NFQ) [6] which required batch updating parameters and used auto-encoders to reduce the dimensionality of the feature space. Batch updating proved to scale with the size of the feature space and proved very costly and auto-encoders removes you from analyzing the raw data. The author manages to overcome both of these problems and will be discussed below.

The reason why this problem is so hard is because deep learning and reinforcement learning work on different paradigms. Deep learning had traditionally relied on large amounts of hand labeled training data, where as reinforcement learning requires a scalar reward signals from sequences of examples. Since the reward signals in video games can often be delayed by many time steps it can be difficult for deep learning to learn such a sparse reward signal. Also they mention that deep learning assumes independence between samples, where as reinforcement learning assumes a dependence over a sequence of examples. Finding a way for these two different communities of AI to come together to tackle more difficult problems is definitely an important moment.

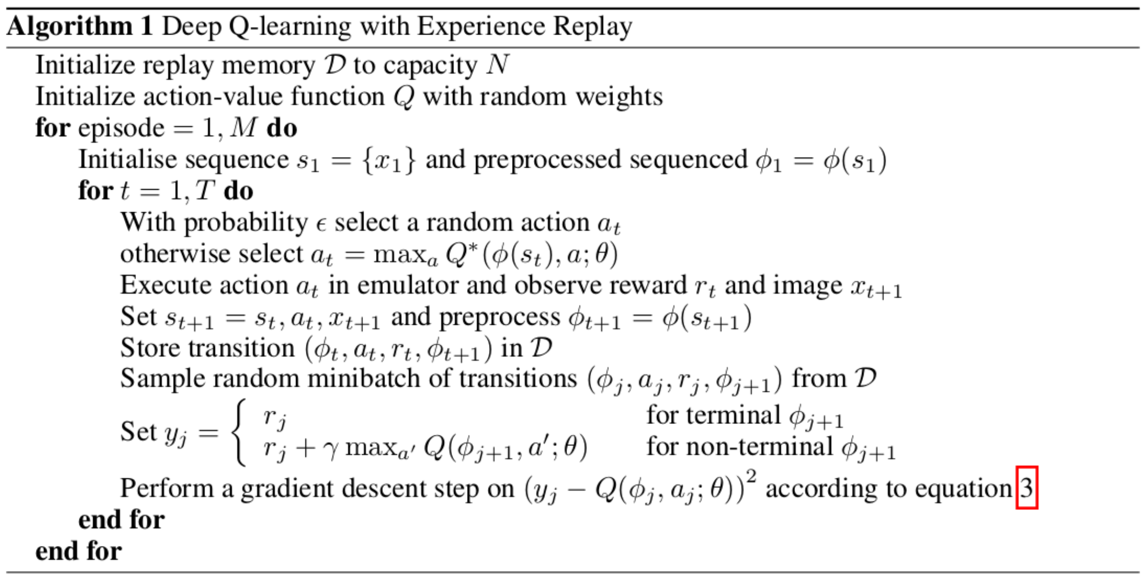
Before the authors could apply any learning techniques they first had to pre-process the images as follows:

[1]https://sites.google.com/a/deepmind.com/dqn/

[2]https://www.youtube.com/watch?v=V1eYniJ0Rnk

1. First the images are preprocessed
    (a) Raw images are 210x160 pixels with 128 color palette.
    (b) Images are down-sampled to 110x84 pixes and colors turned to gray scale.
    (c) Then those images get cropped to 84x84 pixels to allow for square convultions.
2. The last 4 frames of a history is taken and stacked for use.

They then use this sequence of 4 images as the input to the Deep Q-Network (DQN). These preprocessed sequences were then stored for use, this is called experience replay and was key to this algorithm. This was chosen because it allows the network to do only one forward pass per sequence to obtain an action to perform rather than have to learn different Q-functions for different actions which would require multiple forward passes. Also it is difficult to build a network that allows for arbitrary sizes of input vector so they needed to fix the value. Inside of their network the input images went through 2 convolutions and rectifier stages followed by a hidden layer with 256 fully connected rectifier units, followed by the final fully connected layer which has the available actions for the specific game.



```
Algorithm 1 Deep Q-learning with Experience Replay
  Initialize replay memory $\mathcal{D}$ to capacity $N$
  Initialize action-value function $Q$ with random weights
  for episode $= 1, M$ do
      Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
      for $t = 1, T$ do
          With probability $\epsilon$ select a random action $a_t$
          otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
          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 $\mathcal{D}$
          Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
          Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
          Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
      end for
  end for
```

Figure 1: DQN algorithm.[1]

The entire DQN algorithm is show in Figure 1 and I will explain it here. In order to learn the weighting parameters in the DQN the system is first initialized with a state and a preprocessed sequence. Then for some number of time steps it selects an action randomly at probably ϵ or $a_t = max_a Q^*(\phi(s_t), a; \theta)$ which means it takes the best action given its known state and its current parameters. This is known as the ϵ-greedy approach where an action is either selected randomly, or greedily to be the best known action the agent can take at that time. This approach helps avoid local minimum that can result from taking only the max Q-value action. From there it executes the action in the Atari emulator, receives a reward, a new image of the game world, and stores that as the next state and saves it into memory replay. From here it draws random samples of transitions from the replay memory and updates the output value y_i by using the immediate reward and the discounted Q value for the best action sequence in the replay memory samples. It does this for every sample drawn to get try and learn what the best action sequence might be. This gives us a good idea of what the actual value of the Q values are by providing the best discounted future actions values. From here they performed stochastic gradient descent to update the weights from the samples drawn. This repeats for a predefined number of episodes.

An example of the DQN algorithm in practice is shown in Figure 2 for the game Seaquest. What this demonstrates is that for sequences in frames 1 and 2 the immediate reward is 0 as indicated by the score at the top of the frame. However, the sequences of rewards and Q values increase once the algorithm sees an opportunity to attack a target thus it has learned that attaching a target will have higher rewards in the future as indicated by the 60 points in the third frame. It knows this because its Q value spikes at the first and second frame which lets the agent know it has high future rewards for the state action pair.

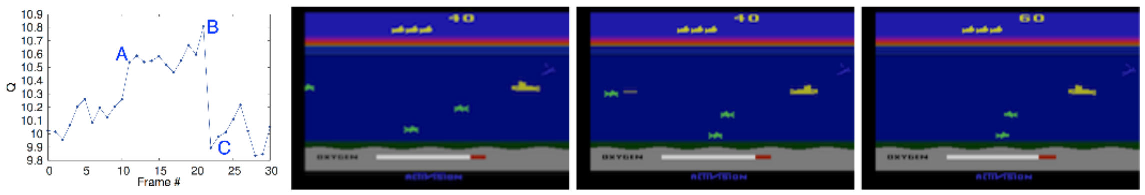
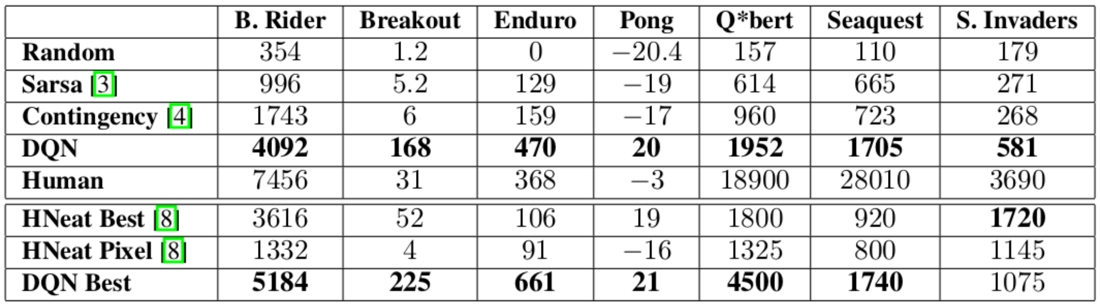


Figure 2: "The leftmost plot shows the predicted value function for a 30 frame segment of the game Seaquest. The three screenshots correspond to the frames labeled by A, B, and C respectively."[1]

Results for their approach are shown in Figure 3. Their approach vastly outmatched all other previous algorithms tested. This includes two other reinforcement learning algorithms; Sarsa and Contingency, which both required hand labeling of the features in order to be able to work, where as their approach had to learn the features as well as the reward sequences. Random just uniformly selects actions at random. They were also able to best human experts at 3 out of the 8 games which is very impressive. I think one of the most impressive results of this paper is the generality of their approach. Each of the games in Figure 3 has different rules and reward structures, yet they were able to train their DQN to learn superior policies to previous approaches. This to me shows that this approach is not a one-hit-wonder such as TD-gammon.

3 Further

The obvious flaw that comes to mind is the one will all deep learning models (and machine learning in general) and that is you need to have sufficient data to learn the features needed to



	B. Rider	**Breakout**	**Enduro**	**Pong**	**Q*bert**	**Seaquest**	**S. Invaders**
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Figure 3: Results between a random approach, SARSA, Contingency, DQN (theirs), and a human expert. DQN performed better that all othe algorithms and better than the human on 3 of 8 games.

adequately perform the task. Even though many advances have been made in deep learning over the past decade, deep learning still suffers from the problem of needing large amounts of data and training in order to function reasonably well. Also, for each game you have start from scratch in this approach even if the game is similar since there is no ability to transfer what was learned previously. It also uses no knowledge of the game and could be improved using heuristics specific to the game or be fed data from a human example to help speed things up.

I would like to look into combining other techniques from machine learning and artificial intelligence to speed up the learning rate by using heuristics. Also, I would like to see how the training would be effected if using human examples combined with the reinforcement learning would help it or actually just end up stunting it.

I found this paper to be very inspiring and fascinating. Since this work came out there have been many advances in this field including the success of OpenAi winning in Dota [7], and the very recent victory in capture the flag mode of Quake III [8]. Each of these more recent examples use techniques that were described in this paper and have continued to show success. Even though the above examples are related to video games, there is no reason that this type of work could not be extended to other task such. This could include learning better multi-robot team task such search and rescue through the use of simulators.

4 Relevance

For the most part this paper was outside the scope of this course. Fortunately I have taken several classes on robotic learning so I am familiar with reinforcement learning and have learned a little about deep learning as well. What this course helped with was understanding the importance of feature mapping to ensuring your algorithms can easily and correctly process data in a desired manner. In this paper they discussed transforming the frames of the game into specific image sizes (84x84x3 pixels) per frame prior to going through convolutions. Furthermore, it required roughly a million sequences saved to be able to learn from which is much larger than we were used to dealing with meaning they would need a better structure for data management.

I would say this paper is relevant to the field but the subject matter is not covered by this course so it may be difficult for someone to understand without further background. I think if someone had no prior knowledge of reinforcement learning or deep learning they would at least understand the idea of learning from data to maximize (or minimize) an objective function.

For this paper I reviewed some old class notes on reinforcement/Q-learning as well doing some refresher studying on convulsional neural networks [3]. This seemed to be sufficient to be able to understand this paper.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[2] 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, pp. 529 EP –, Feb 2015.

[3] N. Twilley, "Artificial intelligence goes to the arcade," 2015.

[4] K. Wiggers, "Openai and deepmind ai system achieves superhuman performance in pong and enduro," 2018.

[5] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, pp. 58–68, Mar. 1995.

[6] M. Riedmiller, "Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method," in *Machine Learning: ECML 2005* (J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, eds.), (Berlin, Heidelberg), pp. 317–328, Springer Berlin Heidelberg, 2005.

[7] N. Statt, "Openais dota 2 ai steamrolls world champion e-sports team with back-to-back victories," 2019.

[3]http://ufldl.stanford.edu/tutorial/

[8] E. Gach, “Google’s deepmind ai takes down human players in quake iii’s capture the flag mode,” 2019.