Deep Q-Network for Angry Birds

Ekaterina Nikonova[1], **Jakub Gemrot**[1]
[1] Faculty of Mathematics and Physics, Charles University, Prague
nikonovae95@gmail.com, gemrot@gamedev.cuni.cz

Abstract

Angry Birds is a popular video game in which the player is provided with a sequence of birds to shoot from a slingshot. The task of the game is to kill all green pigs with maximum possible score. Angry Birds appears to be a difficult task to solve for artificially intelligent agents due to the sequential decision-making, nondeterministic game environment, enormous state and action spaces and requirement to differentiate between multiple birds, their abilities and optimum tapping times. We describe the application of Deep Reinforcement learning by implementing Double Dueling Deep Q-network to play Angry Birds game. One of our main goals was to build an agent that is able to compete with previous participants and humans on the first 21 levels. In order to do so, we have collected a dataset of game frames that we used to train our agent. We present different approaches and settings for DQN agent. We evaluate our agent using results of the previous participants of AIBirds competition and results of volunteer human players.

1 Introduction

Angry Birds has been one of the most popular video games for a period of several years. The main goal of the game is to kill all green pigs on the level together with applying as much damage as possible to the surrounding structures. The player is provided with a sequence of (sometimes) different birds to shoot from a slingshot. Usually, pigs are hidden inside or outside of the complex structures. The game requires a player to find and destroy some weak points of the structure such as its supports or hit a dynamite hidden inside them. Angry Birds was written using Box2D open source physics library and all objects in its game environment are following the laws of Newton Physics (in 2D). As the game is played by many, it seems that Angry Birds can be easily understood by almost everyone: from kids to adults. However, the task of playing the game is a challenge for the artificially intelligent agents, conversely to human players, for whom reasoning and planning in complex physical 2D world appears to be easy. This is due to a number of facts. Firstly, this game has a large number of possibilities of actions and nearly infinite amount of possible levels, which makes it difficult to use simple state space search algorithms for this task. Secondly, the game requires a planning of sequence of actions, which are related to each other or finding single precise shot. For example, a poorly chosen first action can make a level unsolvable by blocking a pig with a pile of objects. Therefore, to successfully solve the task, a game agent should be able to predict or simulate the outcome of it is own actions a few steps ahead.

2 Related work

Back in 2012, the first Angry Birds AI competition (called AIBirds) was held [The Australian National University, 2012]. Many agents have been created since then. In this section we describe two agents which implementation details were publicly disclosed, and which performed well in the competition. One of the best agent that has been created during AIBirds competition was Datalab Birds' 2014 made by a team from Czech Technical University in Prague. This agent still holds 3rd place in overall benchmark of all participating agents so far. As they describe in their paper [DataLab Team, 2014], their main idea was to build a planning agent that decides which strategy to play for the current move based on the environment, possible trajectories and birds available. The agent always plans only one move ahead. The next move is chosen by taking a strategy with maximum estimated utility.

The second participant that we have looked at is Eagle's Wings artificial player made by University of Waterloo and Zazzle Inc. for 2017 Angry Birds AI Competition. Their agent currently holds 16th place in overall benchmark. According to the description of their agent, it uses a simple multi-strategy affordance based structural analysis with a manually tuned utility to decide between the strategies [Eagle's Wing Team, 2017]. This form of analysis gives multiple decisions and a manually tuned utility to decide between them. They were using the machine learning method called xgboost to learn the utility function of each decision.

In this paper, we are adapting Deep Q-networks as used to solve Atari games [Mnih *et al.*, 2013]. We hoped our agent to find weak structural points while maximizing outcome of each bird shot in Angry Birds game using reinforcement learning.

3 Background

In order to solve a sequential decision problem such as the Angry Birds game, we consider the game environment ϵ over discrete time steps. At each time step t agent receives an observation s_t and chooses an action from a set of available actions $a_t \in A = \{a_1 ... a_n\}$ for which it then receives reward r_{t+1} from the environment. The agent goal then is to maximize the expected discounted reward following the policy π

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + ... | s, a]$$

where s is current state, a is a chosen action, r_{t+1} is the reward and $\gamma \in [0,1]$ is a discount, which specifies the importance of the rewards further into future. We now define our optimal Q value to be

$$Q^{*}(s,a) = max_{\pi} Q^{\pi}(s,a)$$

Where our optimal policy can be retrieved from it by choosing action of the highest value in each state.

4 Deep Q-Networks - Theory

To approximate the optimal action-value function, we use deep neural networks [Mnih *et al.*, 2015] as the nonlinear function approximator. For this, we define an approximate value function

$$Q(s,a,\theta_i)$$

where θ_i are the weights of the Q-network at i-th iteration.

As stated in the original paper, reinforcement learning using nonlinear function approximator such as neural networks can become unstable or even diverge due to the following problems: a) the correlation in the sequence of observations, b) from correlations between Q values and target values $r_t + \gamma max_a Q(s_t, a)$ and c) by having a policy that is extremely sensitive to changes of Q value.

Deep Q-learning is addressing the first problem by technique called experience replay. It is implemented by storing and later sampling the observations, which agent has experienced, randomly. This technique removes the correlation between the sequences of the observations by randomizing over the collected data. We define the experience to be

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1})$$

, where s_t is a state at time t, a_t is an action taken at time t, r_{t+1} is a reward received at time $t+1$ and s_{t+1} is a state an agent ended up after taking an action a_t in. We store the experiences in the experience set

$$M = \{e_1, \ldots, e_t\}$$

. We later sample the experience set M for a minibatches to update Q values inside a Q-network with.

In order to address the second problem, following loss function is used:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(M)}[(r + \gamma max_{a'} Q(s^{'}, a^{'}, \theta_i^{-}) - \\ - Q(s,a,\theta_i))^2]$$

where i is the iteration, γ is a discount factor, θ_i are weights of so called online Q-network and θ_i^{-} are weights of target network or so called offline Q-network. The target network is called offline since its weights are only updated every C steps with online network weights, while online network is updating every iteration i.

We then define the target that is used by DQN to be:

$$Y_t = r_{t+1} + \gamma max_a Q(S_{t+1}, a, \theta_t^{-})$$

where θ_t^{-} are weights of target Q-network and γ is a discount factor.

4.1 Double Deep Q-Networks

The original DQN uses the max operator for both selection and evaluation of the action. According to the [van Hasselt *et al.*, 2015] this makes it more likely to select overestimated values, which results in overoptimistic value estimates. For this reason the Double Q-learning algorithm was invented. The general idea behind it is that instead of using the parameters of only one network we will use the first network to select the action and the second network to evaluate that decision. In a more formal way, we will learn two value functions such that for each update one set of weights is used to determine the greedy policy and another one to determine the value of that policy. Thus we can rewrite the original Q-learning target to be:

$$Y_t = r_{t+1} + Q(S_{t+1}, argmax_a Q(S_{t+1}, a, \theta_t), \theta_t^{'})$$

where θ_t are online weights and $\theta_t^{'}$ are the second set of weights. Note that the we are using the online weights to select the action and second set of weights to evaluate it. Both of the weights can be updated by swapping their roles at random [Hasselt, 2010]. Fortunately for us, the same idea of reducing the overestimations can be applied to Deep Q-learning as we can replace the update function with the following:

$$Y_t = r_{t+1} + \gamma Q(S_{t+1}, argmax_a Q(S_{t+1}, a, \theta_t), \theta_t^{-})$$

where θ_t are weights of the online Deep Q-network and θ_t^{-} are weights of the target (offline) Deep Q-network. In comparison to previous Double Q-learning formula second set of weights $\theta_t^{'}$ is now replaced with weights of offline network θ_t^{-}. As in the original Double Q-learning we are using the same idea of using online set of weights to determine the greedy policy and offline weights to evaluate current greedy policy. The update function itself stays the same as we have defined before, we are replacing offline weights with online weights each τ steps, where τ is a hyperparameter to be chosen.

4.2 Dueling Deep Q-Network Architecture

In some situations, the values of the different actions are very similar, and it is unnecessary to estimate the value of each action. In case of Angry Birds, sometimes player would end up with an unsolvable level because pig was blocked with a pile of objects. It does not matter which action to take in this state, since the all of them will have a similar value and we only care about the value of the state itself. As an improvement for these types of situations, the idea of Dueling architecture was introduced [Wang *et al.*, 2015]. To implement the Dueling Q-learning architecture, we will have to use two streams

of fully connected layers. Both of the layers should be constructed in a way that they have the ability to provide separate estimates of the advantage and value functions. After we have separated the output of convolutional layer to two streams we have to combine them back in order to obtain our Q function back. As for the beginning, we first define the value function to be:

$$V^{\pi}(s) = \mathbb{E}_{a\ \pi(s)}[Q^{\pi}(s,a)]$$

And our advantage function to be:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

In other words, the value function tells us how good the particular state is, while advantage function tells us how important each action is by subtracting the value of the state form the value of choosing a particular action a in state s. From these two functions we can now define our new dueling module for Deep Q-network. We are going to make our first fully connected layer to have a single value output $V(s,\theta,\beta)$ and our second fully connected layer to have an output $A(s,a,\theta,\alpha)$ of dimension $|A|$. Here α and β are weights of the advantage and the value fully connected layers and θ is weights of convolutional layers. In order to combine those two values together and to obtain Q values back, we define our last module of the network to be the following:

$$Q(s,a,\theta,\alpha,\beta) = V(s,\theta,\beta) + (A(s,a,\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A(s,a',\theta,\alpha))$$

Thus, we get two streams of estimates of value and advantage functions which we will be using with our previously defined Deep Q-network and Double Q-learning for our playing agent in order to beat Angry Birds game in the following section.

5 Deep Q-Networks for AIBirds

In order to apply Deep Q-network to Angry Birds we first need to define a) the state, b) actions, c) a Q-network architecture and d) a reward function. The provided software by AIBirds tournament software provides us with the ability to take screenshots of the game of size 840x480 pixels. On the Figure 1 we present the step by step process of preprocessing the image. We first crop those frames by 770x310 pixels area that roughly captures the game play area and crops out the UI interface elements such as menu and restart level buttons. After the cropping we are resizing the cropped image to 84x84 pixels with RGB channels. At the end, we normalize the resulted image to have zero mean and unit norm and pass it as input to our Deep Q network.

We define the action to be $a \in \{0,1,2,3,\ldots,90\}$ where each discrete number represents the angle of the shot. We then find the final release point for a given angle using the provided software. To calculate the release point, trajectory module first finds the reference point for a sling and then calculates the release point using it. We decided to choose such a big action space since there are times where even 1 degree can make a huge difference. During our experiments the typical situation we would be observing is that agent shots at 49 degrees, receives some small amount of points for that, the next time it shots at 50 degrees and finishes the level with a new high score.

Our Deep Q-network architecture was based on the Google DeepMind Deep Q-network [Mnih *et al.*, 2013] that was used to play multiple Atari games. On the Figure 2 we present the architecture of our Deep Q-network. The model was consisted of 4 convolutional layers of kernel sizes 8x8, 4x4, 3x3, 7x7 and strides 4x4, 2x2, 2x2, 1x1. The last convolutional layer was followed by two flatten streams of value and advantages values that are later combined into final Q values as described in 4.2.

As for the reward function, we have experimented with two different versions. At first, our reward function was defined as follows:

$$reward = \begin{cases} 1, & \text{if } score > 3000 \\ -1, & \text{otherwise} \end{cases}$$

This reward function was using technique that is usually called reward clipping [van Hasselt *et al.*, 2016]. it squeezes the potentially unbounded score to two extremes of reward 1 for a "good" action and -1 for a "bad" action. With this reward function agent was able to perform quite well achieving on average the score of 850,000 (for the first 21 levels of "Poached Eggs" episode). However, with this reward function agent had no way to learn the important fluctuations between actions. For example, the action that obtained the score of 70000 was not any more useful than action that obtained score of 3001. One of the main goals in Angry Birds besides winning the level, is to obtain a maximum possible score. Because of that, we had to change our reward function to somehow encapsulate the importance of the magnitude of the score without simply assigning the obtained score as our reward. Thus, we have redefined our reward function to be:

$$reward = \frac{score}{maximum\_score\_of\_current\_level}.$$

where score is obtained score for some action a at some state s, and maximum score of current level is a maximum score in this level ever obtained either by one of our volunteers or by an agent itself. With this definition we were expecting agent to learn the importance of the score and eventually set new maximum scores for the levels by learning the most rewarding actions.

5.1 Training set

Our training set of levels consists of 21 levels of "Poached Eggs" episode of Angry Birds Classic game. For the training process we have collected over 45 thousands screenshots while agent was playing those 21 levels using greedy epsilon policy and approximately 20 thousands using completely random policy. Thus, by mixing both data sets together, we had over 65 thousands of the images that we used to train our network. The agent was waiting for 5 seconds before taking a screenshot of the game playing area. This small adjustment was needed in order to let physics to settle after the shockwave produce by the bird shot.

5.2 Validation set

Our validation set of levels consists of 10 levels of "Poached Eggs" episode of Angry Birds Classic game. Since our agent

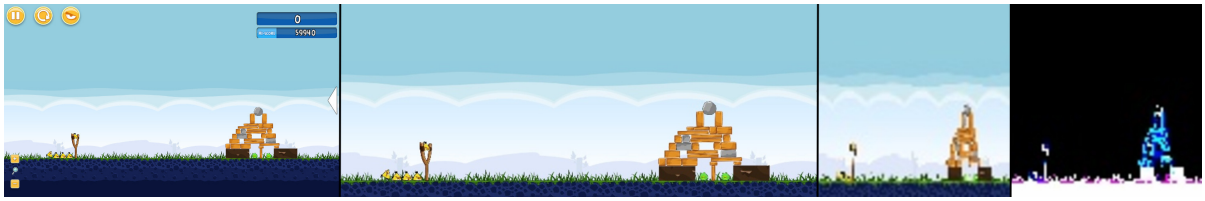

Figure 1: Game frame at each state of preprocessing. From left to right: original captured frame, cropped, resized and normalized.

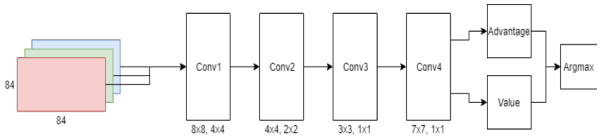

Figure 2: Double Dueling Deep Q-Network architecture

was trained on levels with only red, blue, yellow birds we had to select the validation levels with no new birds. Our validation levels included levels 1,2,3,4,10 and 12 from the second page of the episode and levels 1,3,4 and 5 from third page of the episode. We have selected this set of levels since they were previously unseen by our agent and they did not contain any new birds. Levels from second and third page of the "Poached Eggs" episode are considered to be of a higher difficulty than those from the first page on which our agent was trained. Third page episodes are considered to be hard to complete even for humans. These levels usually require player to find some non-obvious weak point of the structure and to carefully plan actions ahead of a time.

6 Results

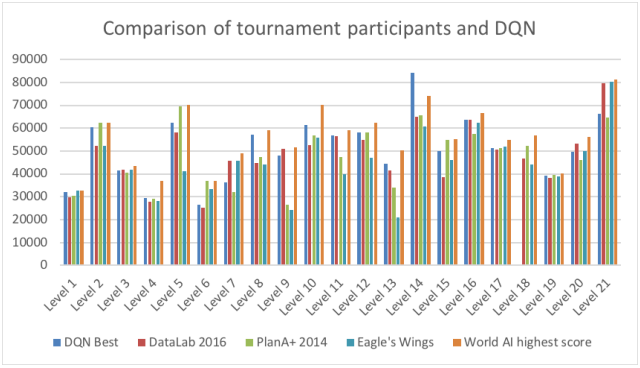

Figure 3: Comparison of best DQN agent to previous participants of AIBirds

On the Figure 3, we compare results of several participants of the tournament on 21 levels. Our agent had only 1 try to beat each level; in case of a failure agent received 0 score for that level. For the comparison, we have selected results of Datalab 2016 agent which holds the 1st place in the tournament benchmark overall years and PlanA+ 2014 agent which holds the 2nd place. We have selected these two agents since they showed the highest scores on first 21 levels. For the better comparison, we also present overall highest achieved score for each level by artificial agents. Our agent was able to compete with some of the best agents ever presented on the AIBirds competition. Surprisingly enough, the agent was able to beat the high score for level 14. At first, we thought it was a mistake or a lucky outcome, however multiple reevaluation of the agent showed the same results as presented here.

Comparing the results of different agents can help to understand which AI technique for solving Angry Birds game works better. However, what still reminds unclear after such a comparison is how well the agent compares to the human. For this paper, we have also conducted the Artificial Intelligence versus Human competition. The goal of the experiment was to achieve the maximum possible score for each level. Our participants were given an unlimited amount of time and tries. They were constantly replaying the levels, slowly achieving what they thought was their personal highest score. Once the participant decided that he or she achieved that maximum the experiment was stopped for him or her. We have also selected participants with different level of experiences. Thus, Human 1 and Human 3 claimed to have at least 4 years of experience in playing Angry Birds game. Human 3 has claimed to play Angry Birds almost every day at the time of the experiment. Human 2 claimed to played Angry Birds a couple times a long time ago and thus did not have much experience. Human 4 claimed that she has played Angry Birds a lot in the past, but did not played much recently, thus we considered her to be close to the average player.

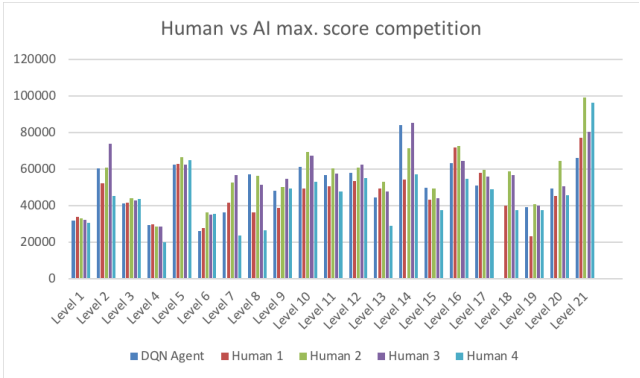

Figure 4: Comparison of best DQN agent to human players

As we can see on the Figure 4, our agent was able to beat one human but lost to others in a total sum of obtained scores for 21 levels. One of the main reasons agent has failed to obtain higher total score was due to its inability to beat level 18 with this particular hyper parameters setting. On the rest

of the levels, agent was able to outperform some of the human players in some of the levels. Overall, our DQN agent was able to achieve 1007189 points on first 21 levels of the Poached Eggs episode. Comparing it to total scores of humans: 1019190, 983840, 1191660 and 1152830 - it lost to humanity by losing in three out of fourth comparisons. Even in the level 14 where agent was able to beat the highest score ever achieved by artificially intelligent agents it lost to the third human player. Thus we can conclude that there are still place for the improvements for our agent.

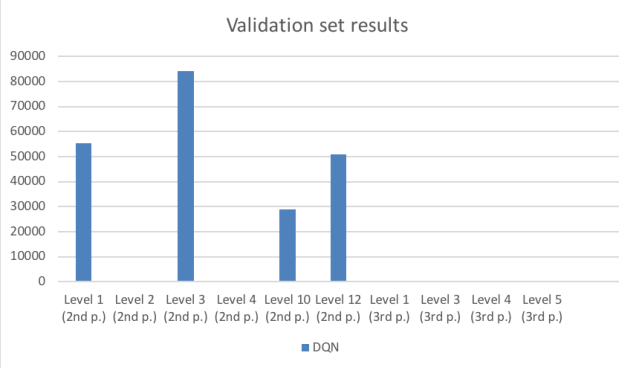

Figure 5: DQN agent results on validation set. Here 2nd p. and 3rd p. means second and third pages of the "Poached Eggs" episode respectively.

On the Figure 5 we present the results of the agents on validation data set. As we were expecting, the agent did not pass most of the previously unseen levels. It was still surprising however, to see it completing some of the levels. From these results, we can conclude that our training data set did not include the whole variety of possible levels in the game, which lead to poor generalization. As a possible improvement, we could extend our training set to more difficult levels.

7 Discussion

Angry Birds game has a continuous shooting angle space which is problematic for a neural network approximation using DQN method. This is due to the fact that in our approach we had a single output for each valid action, thus, for a continuous space we would had to define an arbitrary large output layer. To overcome this obstacle, we have defined the action space to be discrete. We defined the action to be $a \in \{0, 1, 2, 3, \ldots, 90\}$ with each discrete number representing the angle of the shot. By limiting the shooting angle in such a way, we were trying to reduce the chance of missing the optimal shot while keeping the size of output layer as low as possible. As a part of the experiment, we have also tried to reduce the action space to every 2 degrees, but it has dramatically reduced the overall agent's performance. One of the limits introduced with this action space, was agent inability to shot at points located lower than 0 degrees relatively to the center of the slingshot. However, this limitation will not significantly affect agent's performance since all of the first 21 levels do not require such shots.

On regards of hyper parameters of our network we have achieved the best results with the following setting: batch size = 32, learning rate = 0.00001, update rate = 4 and normalized reward function as was described in previous section.

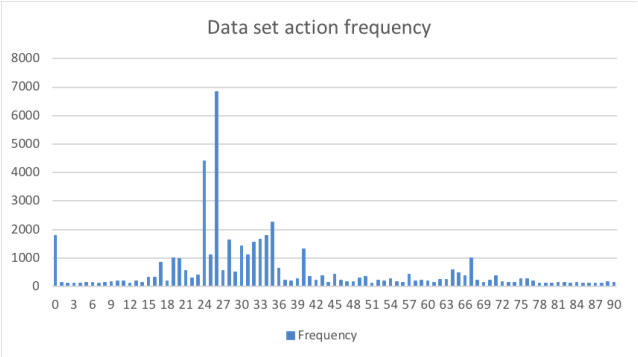

Figure 6: Data distribution

On the Figure 6, one can see the distribution of actions overall collected data. Here we note that most of the actions in our data were distributed between 17 and 36 degrees, with strong outliers of 0, 40 and 67 degrees. Based on this data we were expecting the agent to learn the actions that closely resemble the graph above. For example, we were expecting that it would mostly shoot at the range of 24 to 40 degrees, as well as have some odd shots of lower and higher degrees in some levels. In fact, this was the case, as we can see in the action distribution over time on Figure 7.

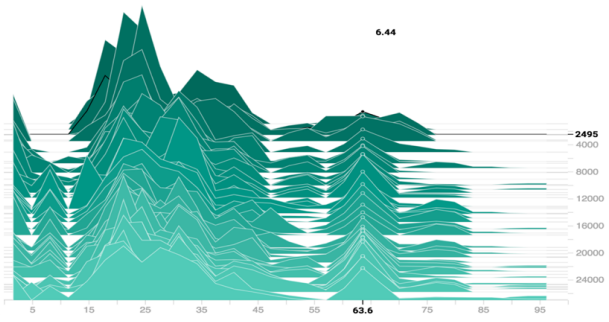

Figure 7: Distribution of predicted actions over time. Vertical axis shows us the time, while horizontal axis shows the action (degree).

8 Conclusion

The Angry Birds game still remains a difficult task for artificially intelligent agents. We have presented a playing agent that was using Double Dueling Deep Q-Network in order to solve sequential decision problem in Angry Birds game. In order to train our agent, we have collected over 65,000 of game frames using greedy epsilon and partially random policy. One of the goals that we did not quite achieve in this work, is to outperform humans in Angry Birds game. Despite the fact we have come close to a human-level performance on selected 21 levels, we still lost to 3 out of 4 humans in obtaining a maximum possible total score. On a good side, our

agent was able to learn to complete the levels using only one try. Another interesting point, is that most of the times it was using only one precise shot to some weak point which lead to the completion of the level. While our agent had outperformed some of the previous participants of AIBirds agents and set new high-score for one of levels, there are still a lot of room for improvements. As an example, we could have experiment more with hyper parameters and definition of reward function. As a more radical improvement, we could have used any of the nowadays published enhancements for Deep Reinforcement Learning [Hessel *et al.*, 2017]. As our next step, we would like to train the agent on the next 21 levels of "Poached Eggs" episode as well.

Acknowledgements

This Research was funded by the Czech Science Foundation (project no. 17-17125Y).

References

[DataLab Team, 2014] DataLab Team. Angry Birds AI. `https://datalab.fit.cvut.cz/research/projects/45-angry-birds-ai`, 2014. [Online; accessed 19-May-2018].

[Eagle's Wing Team, 2017] Eagle's Wing Team. AI Angry Birds Eagle Wing Developer Guide. `https://github.com/heartyguy/AI-AngryBird-Eagle-Wing/blob/master/docs/developerguide.md`, 2017. [Online; accessed 21-May-2018].

[Ge *et al.*, 2014] XiaoYu. Ge, S. Gould J., Renz S. Abeyasinghe, and P. Zhang J. Keysand A. Wang. Angry birds basic game playing software, version 1.32. 2014.

[Hasselt, 2010] Hado V. Hasselt. Double q-learning. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2613–2621. Curran Associates, Inc., 2010.

[Hessel *et al.*, 2017] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *ArXiv e-prints*, October 2017.

[Lample and Singh Chaplot, 2016] G. Lample and D. Singh Chaplot. Playing FPS Games with Deep Reinforcement Learning. *ArXiv e-prints*, September 2016.

[Mnih *et al.*, 2013] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with Deep Reinforcement Learning. *ArXiv e-prints*, December 2013.

[Mnih *et al.*, 2015] 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*, 518(7540):529–533, 2015.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* The MIT press, Cambridge MA, 1998.

[The Australian National University, 2012] The Australian National University. AI 2012 Angry Birds: AI Challenge. `http://ai2012.web.cse.unsw.edu.au/abc.html`, 2012. [Online; accessed 11-June-2018].

[van Hasselt *et al.*, 2015] H. van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-learning. *ArXiv e-prints*, September 2015.

[van Hasselt *et al.*, 2016] H. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver. Learning values across many orders of magnitude. *ArXiv e-prints*, February 2016.

[Wang *et al.*, 2015] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *ArXiv e-prints*, November 2015.