# RL Project

**David S. Hippocampus**
Department of Computer Science
University of Bath
Bath, BA2 7AY
hippo@bath.ac.uk

## 1 Problem Definition

A clear, precise and concise description of your chosen problem, including the states, actions, transition dynamics, and the reward function. You will lose marks for an unclear, incorrect, or incomplete problem definition.

## 2 Background

A discussion of reinforcement learning methods that may be effective at solving your chosen problem, their strengths and weaknesses for your chosen problem, and any existing results in the scientific literature (or publicly available online) on your chosen problem or similar problems.

### 2.1 DQN (Mirco)

### 2.2 A2C (Chris)

- Intro to Policy Gradient Methods

  A set of alternative methods to value-based reinforcement methods that can be applied to this problem is policy gradient methods. Whilst value-based methods such as DQN learn values of actions and use these estimates to select actions, policy gradient methods do not need to consult a value function and, instead, learn a parameterised policy to select actions (Sutton and Barto, 2018).

  One commonly used policy parameterisation for discrete action spaces, *soft-max in action preferences*. In selecting actions using a parameterised policy, policy gradient methods use *action preferences* rather than action values. This distinction is crucial as it confers several benefits to policy gradient methods such as the ability to learn a stochastic optimal policy (Sutton and Barto, 2018). One benefit particularly relevant to Breakout is that the policy can approach a deterministic policy which would not be possible if action-values were used (Sutton and Barto, 2018). This is relevant to the problem of breakout as in some situations such as where the ball is close to the paddle, the desired behaviour is for the agent to deterministically move toward the ball to return it and have no chance of moving away from it.

  Policy Gradient techniques have seen success in high-profile reinforcement learning breakthroughs in game-playing. Most notably, AlphaGo (Silver et al. 2016) used the REINFORCE algorithm to train the policy network used in the agent.

- Intro to Actor Critic

  One Policy-Gradient method that has been shown to be effective on similar tasks to Breakout is the Actor Critic method.Actor Critic chosen as a promising method for this problem over other methods as it offers significant benefits over other policy-gradient methods such as REINFORCE. REINFORCE is a Monte Carlo algorithm and as such, suffer from problems such as slow learning and issues with online implementation. In contrast, Actor-Critic methods are analogous to TD methods and so avoid these issues (Sutton and Barto, 2018). Avoiding the issue of slow learning is particularly crucial for a complex environment such as Breakout. Training a DQN agent capable of performing above average performance on Atari games required a training time of 8 days on a GPU (Mnih et al. 2016). For a small team and the time window on this assignment, an agent that learned slower than this would not be feasible. Variants of Actor-Critic including Asynchronous Advantage Actor-Critic (A3C) and Advantage Actor-Critic (A2C) have shown strong performance on similar tasks to the problem considered in this paper such as in Minh et al. (2016).

### 2.3 Async Q-Learning (Peter)

## 3 Method

A description of the method(s) used to solve your chosen problem, an explanation of how these methods work (in your own words), and an explanation of why you chose these specific methods.

### 3.1 DQN (Mirco)

### 3.2 A2C (Chris)

- Basic Actor Critic explanation

  The second method chosen to solve our specific problem of Atari Breakout is Advantage Actor-Critic (A2C). As discussed in section 2, A2C is a variation of the Actor-Critic method which implements an advantage function. Whilst several variations of Actor-Critic methods are used in practice, all Actor-Critic methods are comprised of two key components at their core: an Actor and a Critic. The role of the actor is to learn a parameterised policy which is used to select the actions. The role of the critic is to provide a reinforcing signal to the actor based on actor performance which is used to update the parameters of the policy. This reinforcement signal is based upon the state-values learned by the critic.

- Implementation details for Actor Critic networks including architecture and pre-processing

  In this implementation, the actor policy is parameterised by a convolutional neural network (CNN) where the parameters are the weights of the neural network. The critic is also implemented as a convolutional neural network. The neural network architecture for both the actor and the critic is drawn from the architecture used in Silver et al. (2015) with some notable changes. For the Actor, the output layer uses a softmax activation function for the output layer with a single neuron for each possible action that the actor can take. This is to achieve the softmax in action preferences policy parameterisation described in section 2. This softmax output is sampled in order for the agent to take an action in a given state. The critic network uses the same architecture but uses a single output neuron with linear activation. This output value represents the action-value the critic prescribes to a given state. Both networks also use a RandomUniform initialisation to the output layers with a relatively small range between min and max values (0 - 0.02). This was included as initial runs of the algorithm showed high sensitivity to initial weights on the output layers, particularly on the actor.

  The input to both networks is a pre-processed 84x84x4 image that represents a state the Breakout environment. The preprocessing applied is the same as that applied in Silver et al. (2015), including greyscale, image cropping and frameskip. The 4 channels of the image are implemented using the StateHistory class. The instance of StateHistory passed to the agent as input includes the current frame and the 3 previous frames. This gives the agent context to the current frame such as the direction the ball is travelling in.

- Advantage-Actor-Critic explanation

  The issue with vanilla Actor-Critic as described above is that it is generally implemented as an on-policy learning algorithm and on-policy learning algorithms are unstable due to temporally correlated data across timesteps (Li, Bing and Yang, 2018). This was addressed by Minh et al. (2016) through the introduction of Asynchronous Advantage Actor-Critic (A3C). In A3C, the asynchronous component is that multiple agents are executed in parallel on multiple instances of an environment with the aim of removing the non-stationarity since agents will be experiencing a variety of different states at any one time step (Minh et al. 2016). The Advantage component of A3C refers to the use of an advantage function in the reinforcing signals sent to the actor. This advantage value quantifies how much better or worse an action is than the average available action by using the critic's valuation of states (Graesser and Loon, 2020). An algorithm that implements the advantage function but does not execute multiple agents in parallel is known as Advantage Actor Critic (A2C).

Results from OpenAI (2017) showed that A2C may be a more suitable candidate for the problem of Breakout than A3C. Indeed on a set of Atari games, including Breakout, their results showed that A2C outperformed A3C and the noise introduced by the asynchronous implementation failed to deliver any performance benefit (OpenAI, 2017). As such, A2C is chosen as the Actor-Critic implementation.

In A2C, the inclusion of the Advantage function is motivated in a similar way to inclusion of baselines in other policy gradient methods which is to decrease variance and thus improve the stability of training (Sutton and Barto, 2018). In the case of A2C, the state-value produced by the critic is used as the baseline. The update rule for A2C is given in Yoon (2019) and shown below:

$$AdvantageActorCritic$$

$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)$

Where:

A(s$_t, a_t) = Q_w(s_t, a_t) - V_v(s_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$ (1)

The critic uses a TD update to adjust it's estimate of the value of a state. The intuition behind which is as follows: Each step of the environment, the critic network calculates a value estimate for both the previous state and the next-state that our agent is in. In a similar fashion to TD(0), the critic uses the bootstrapped estimate of the next state, the reward at timestep t+1 and the discount factor to calculate a target value for the state. The mean squared difference between the critic's initial value of the state and the target value is used to update the critic and force it to adjust it's estimate of the state closer to the target value.

However, the target TD value is also used in the calculation of the advantage function and, hence, the actor loss. As seen in the above equation, The advantage function is calculated by subtracting the critic's valuation of the previous state from the target TD update. The intuition behind this is that the difference between these two values represents the value of taking the action that the actor took in that state against the average value of that state (The average value of that state) (Yoon, 2019). This is used in the actor update to provide a measure of how good that action was against the baseline for that state.

The update equation above is calculated for the actor. The first component $\nabla_\theta$ represents gradient (delta), i.e which action the agent took whether the agent went with or against the current policy which sets the direction of the update for each action. For example, if the agent is selecting from two actions and selects action one, delta will be positive for action one and the size of delta will be larger if the action went against the current policy. This value is multiplied by the log probabilities of the action taken in the given state. The negative log probabilities are used rather than the raw probabilities in order to provide numerical stability as multiplying together probabilities approaches 0 which can yield issues in computing these numbers quickly.

The advantage, which represents the reinforcement signal from the critic, tells the actor whether the action taken was better than the average reward for that state. Thus, combining this with the above variables, if the advantage is positive, the agent will be told to take more of the action it did take and less of the ones it did not take. Conversely, a negative advantage value tells the agent to take less of the action it did take and more of the actions it did not take.

A2C implementation including inclusion of entropy bias and details of custom loss function

In this implementation, the critic loss is easily implemented using mean squared loss of the TD update. However, the loss for the actor is calculated in a custom loss function called *actorcustomloss*. This function implements the update described above aswell as implementing techniques useful for ensuring stability in training such as clipping values. However, the loss function implemented here also includes an entropy regularisation term. This yields a new update equation for the actor

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T-1} -\nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t) + \beta H_t \tag{2}$$

The reason for incldluing this term was.... Entropy (reference) incentvises the actor to output a distribution closer to even. The weiht

### 3.3 Async Q-Learning (Peter)

## 4 Results

result comparison between the 3 approaches. how quickly each agent learns how well do they perform is absolute terms how do they compare with respect to a human player

### 4.1 DQN (Mirco)

### 4.2 A2C (Chris)

### 4.3 Async Q-Learning (Peter)

## 5 Discussion

An evaluation of how well you solved your chosen problem.

### 5.1 DQN (Mirco)

### 5.2 A2C (Chris)

### 5.3 Async Q-Learning (Peter)

## 6 Future Work

What other techniques can be used to improve further the performances of the 3 agents.

A2C - generalised advantage estimate as an improvement

## 7 Personal Experience

A discussion of your personal experience with the project, such as difficulties or pleasant surprises you encountered while completing it.

## References

file:///Users/mogul/Downloads/Mastering$_the_game_of_Go_with_deep_neural_networks.pdf - Silveretal.AlphaGoSuttonandBarto(2018)$

https://bath-ac-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=44BAT$_ALMA_DS5184633880002761context$
$Lvid = 44BAT_VU1lang = en_USsearch_scope = CSCOP_44BAT_DEEPadaptor = Local$

https://arxiv.org/pdf/1602.01783.pdf - Mnih et al.

https://arxiv.org/pdf/1806.06914.pdf - Li, Bing and Yang, 2018

https://openai.com/blog/baselines-acktr-a2c/ - A2C vs A3C

https://www.nature.com/articles/nature14236?wm=book$_wap_0005 - Silveretal2015, humanlevelcontrol$

https://towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f Yoon 2019

## Appendices

If you have additional content that you would like to include in the appendices, please do so here. There is no limit to the length of your appendices, but we are not obliged to read them in their entirety while marking. The main body of your report should contain all essential information, and content in the appendices should be clearly referenced where it’s needed elsewhere.

### Appendix A: Example Appendix 1

### Appendix B: Example Appendix 2