Deep Q-Learning with Recurrent Neural Networks

Clare Chen
cchen9@stanford.edu

Vincent Ying
vincenthying@stanford.edu

Dillon Laird
dalaird@cs.stanford.edu

Abstract

Deep reinforcement learning models have proven to be successful at learning control policies image inputs. They have, however, struggled with learning policies that require longer term information. Recurrent neural network architectures have be used in tasks dealing with longer term dependencies between data points. We investigate these architectures to overcome the difficulties arising from learning policies with long term dependencies.

1 Introduction

Recent advances in reinforcement Learning have led to human-level or greater performance on a wide variety of games (e.g. Atari 2600 Games). However, training these networks can take a long time, and the techniques presented in the state of the art [0] perform poorly on several games that require long term planning. Deep Q-networks learn to estimate the Q-values (long term discounted returns) of selecting each possible action from the current game state.

Deep Q-networks are limited in that they learn a mapping from a limited number of past states, or game screens. In practice, DQN is trained using an input consisting of the last four game screens. Thus, DQN performs poorly at games that require the agent to remember information more than four screens ago. In other words, the game could no longer be modeled as a true Markov decision process; all of the information needed to make an optimal action would no longer be contained in a single state. This is evident from types of games DQN performs poorly at, near or below human-level Figure 1. We explore the concept of a deep recurrent Q-network, a combination of a long short term memory (LSTM) [6] and a deep recurrent Q-network (DRQN) similar to [5]. We wish to demonstrate with the introduction of recurrent network architecture into the deep Q-network, the network can retain information from previous frames of the game and achieve good performance on games that require long term planning.

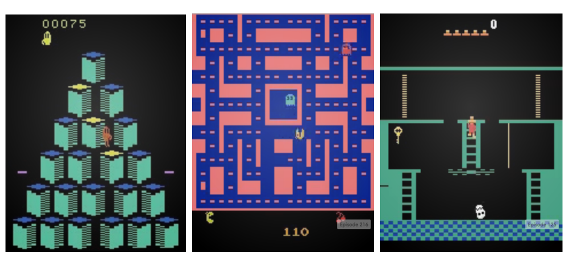

Figure 1: Q*bert, Ms. Pac-Man and Montezuma's Revenge

In addition, recent achievements of attention recurrent neural networks in long term sequence tasks [2, 3] have introduced the possibility of incorporating them into the structure of the DRQN architecture. The advantage of using attention is that it enables DRQN to focus on particular previous states it deems important for predicting the action in the current state. We investigate augmenting

DRQN with attention and evaluate its usefulness.

2 Deep Recurrent Q-Learning

We propose two architectures for the Deep Recurrent Q-Network (DRQN). The first is a very basic extension of DQN. Depicted in Figure 1, the architecture of DRQN augments DQN's fully connected layer with a LSTM. We accomplish this by looking at the last L states, $\{s_{t-(L-1)}, \ldots, s_t\}$ and feed these into a convolutional neural network (CNN) to get intermediate outputs $\text{CNN}(s_{t-i}) = x_{t-i}$. These are then fed into a RNN (we use an LSTM for this but it can be any RNN), $\text{RNN}(x_{t-i}) = h_{t-i}$, and the final output h_t is used to predict the Q value which is now a function of $Q(\{s_{t-(L-1)}, \ldots, s_t\}, a_t)$.

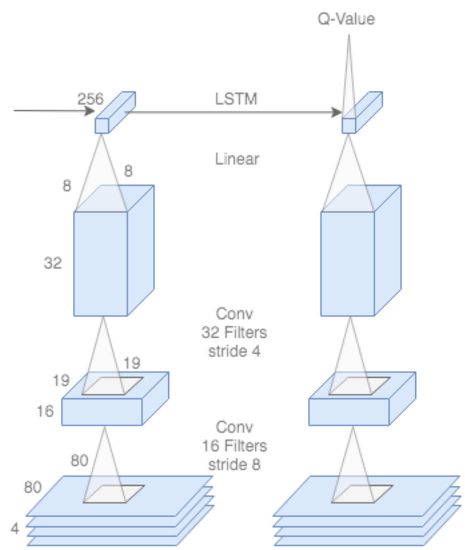

Figure 2: Architecture of the DRQN

Another architecture we used was a version of an attention RNN. For the attention RNN we take the L hidden states outputted by the RNN, $\{h_{t-(L-1)}, \ldots, h_t\}$ and calculate an inner product with v_a, $\{v_a^T h_{t-(L-1)}, \ldots, v_a^T h_t\}$. This allows us to focus more on nearby hidden states or further away states depending on what values of v_a are learned. We then take a softmax over these values, $a_{t-i} = \text{softmax}(v_a^T h_{t-i})$. We use this softmax to take a weighted sum over the hidden states to get a context vector, $c_t = \sum_{i=0}^{L-1} a_{t-i} h_{t-i}$. This context vector is then used to predict the Q value.

Learning sequences of observations creates some difficulty when sampling experiences and using the following loss function [0]:

$$L_i(\theta_i) = \mathbb{E}_{e_t \sim U(D)}\left[\left(r_t + \gamma \max_{a'_t} Q(s_{t+1}, a'_t; \theta_i^-) - Q(s_t, a_t; \theta_i)\right)^2\right]$$

where $e_t = (s_t, a_t, r_t, s_{t+1})$, $D_t = \{e_1, \ldots, e_t\}$, γ is the discount factor and $e_t \sim U(D)$ are drawn uniformly at random from the pool of stored experiences. We propose a simple solution where we sample $e_t \sim U(D)$ and then take the previous L states, $\{s_{t-(L+1)}, \ldots, s_t\}$ and zero out states from previous games. For example if s_{t-i} was the end of the previous game then we would have states $\{\mathbf{0}, \ldots, \mathbf{0}, s_{t-(i+1)}, \ldots, s_t\}$ and similarly for the next state $\{\mathbf{0}, \ldots, \mathbf{0}, s_{t-(i+1)}, \ldots, s_{t+1}\}$.

3 Experiments

For our experiments we chose to play the game Q*bert. We chose Q*bert because it was a challenging game for DQN which achieved scores only slightly above human-level [0] but it wasn't so challenging that DQN could not make any progress, such as Montezuma's Revenge [0].

For input the DQRN takes a specified number of screens per state. The screen images are grayscaled and resized to 80×80. The first hidden layer convolves 16, 19×19 filters with stride 8 across the input image and applies a rectified linear unit. The second hidden layer convolves 32, 8×8 filters with stride 4, again followed by a rectified linear unit. Convolutional outputs are fed to the fully connected LSTM layer. Finally, a fully connected linear layer outputs a Q-value for each possible action.

References

[0] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level Control through Deep Reinforcement Learning. *Nature*, 518(7540):529-522, 2015.

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] Dzmitry Bahdanau, Kyunghyun Cho and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Minh-Thang Luong, Hieu Pham and Chisopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint arXiv:1508.04025*, 2015.

[4] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov and Anastasiia Ignateva. Deep Attention Recurrent Q-Network. *arXiv preprint arXiv:1512.01693*, 2015.

[5] Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. *arXiv preprint arXiv:1507.06527*, 2015.

[6] Sepp Hochreiter and Jurgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735-1780, 1997.