Problem Formulation and Critical Analysis

In this question I have analysed "Playing Atari with Deep Reinforcement Learning" paper published by David Silver and his colleagues in Deepmind in 2013.

Framing the question

Finding a phenomenon or a question to ask about

Deep learning made many breakthroughs recently. One of the reasons for this is its ability to approximate functions.

One of the biggest challenges in Reinforcement Learning (RL) was to control agents by taking high dimension sensory inputs. Advancement in Deep Learning made lead to many breakthroughs in the field of computer vision and speech recognition the main question is whether it would be beneficial to use neural networks architectures like CNN in RL with sensory data.

The goal is to create a single neural network agent which uses neural network for feature extraction and RL for learning a particular task and to check the performance of the algorithm on seven different Atari 2600 games. Atari 2600 was selected because these games consist of various set of task and also the high dimensional visual input.

Understanding the state of the art

The best-known success full paper of RL was TD-*gammon* a backgammon playing RL program which learnt the game entirely through RL. It used model free algorithm similar to Q-learning. When it was tested on other games like Chess or Go, it didn't perform well. It used a multi-layered perceptron with one hidden layer to extract features.

A work similar to combination of neural network and Q-learning was neural fitted Q-learning (NFQ). It uses RPROP as its optimizer and also batch update which made its computation cost per iteration proportional to size of the dataset in this paper, we have used stochastic gradient updates.

Previously work has also been done on ATARI 2600 games. Q-learning combined with linear function approximation has been used. The results also go improved by using a large set of input features.

Determining the basic ingredients

For the Deep Q-Network, the state of the game is fed as an input to the network in the form of pixels. For this task we can ignore the colour of the screen, so we converted RGB to grey-scale and also cropped it down to 84X84 pixels. The input is the stack of four consecutive states so that our CNN could extract velocity from it. We also need memory for storing each transition so that it could train itself by randomly sampling from the set of transitions. The output will consist of Q-value estimates which will be needed to for computing the loss function.

Formulating mathematical hypothesis

The deep neural networks will be giving estimates of Q-values for a particular state-action pair denoted by $Q(s,a,\theta_i)$ where θ_i represents weights of neural network in iteration i. For obtaining our target label we will use Bellman Equation on our previous Q-value estimate $y_i = E[\, r + \Upsilon \max_{a'} Q(s',a',\theta_{i-1}) | s,a]$.

This network uses MSE Loss, $L_i(\theta_i) = E_{s,a\sim\rho(.)}[(y_i - Q(s,a,\theta_i))^2]$ and derivative of loss function with respect to weights gives:

$\nabla_{\theta i} L_i(\theta_i) = E_{s,a\sim\rho(.)}[\, r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\,] \nabla_{\theta i} Q(s, a; \theta_i)$

Here ρ represents probability distribution of state and action pair with respect to behavioural distribution. For Exploration we used ϵ-greedy method. Where we choose action having maximum Q-value estimate with probability 1- ϵ and random action from action space with probability $\epsilon/|A(s)|$ where A(s) is action space for state s.

Implementing the Model

Selecting Toolkit

For this task I would use pytorch framework as it has some basic libraries already built in it which would help in implementing CNN as well as DQN with better ease.

For getting Atari 2600 environments I would use OpenAI Gym as it has different environments already implemented and it is also easy to get rewards, transition probability and next state.

Planning and implementing the model

Atari frames are 210X160 pixel images with RGB representations. Basic pre-processing techniques can be applied like converting it to grey-scale and then cropping the final image to 84X84 pixels inputs.

Input of the neural network consist of 84X84X4 dimensions and first hidden layer convolves 16 8X8 filters with strides of 4 over input images with rectifier non-linearity. The second hidden layer consist of 32 4X4 filters with strides of 2 followed rectified non-linearity. The final hidden layer consists of 256 rectified non-linearity and the output layer consist of units equal to the number of possible actions in each game. All the positive rewards were set to +1 and negative rewards were set to -1.

For learning weights, we used RMSProp algorithm with batch size of 32. We used ϵ-greedy policy for exploration where ϵ varied linearly from 1 to 0.1 over one million frames and was set to 0.1 thereafter. We trained for ten million frames and used replay memory of one million most recent frames.

In these Atari games a simple frame skipping technique was also used i.e. agent select the action on every k^{th} frame and the last action is copies in the previous frames. In all games k=4 except Space Invaders where k=3 to make laser visible.

Model Testing

Completing the model

The aim of the paper was to use a single neural network and learn different Atari 2600 games successfully which was achieved in the paper. It was showed that high dimension sensory inputs can be used to extract features and RL algorithms can be used to take various decisions not only for a particular task but for different set of tasks (games) without incorporating game-specific information.

Testing and evaluating the model

The model was tested on seven Atari games like Breakout, Space-Invader, Q*Bert, Beam Rider, Endure, Pong and Seaquest. The network architecture, hyperparameters were same indicating the network is robust. One of the main evaluation criteria was to maximise the cumulative reward during each episode or game averaged over number of games we periodically compute it during training. This was achieved by minimising the Mean Squared Error.

In the end the results were also compared with other pre-existing algorithms, results are shown below: -

	B. Rider	Breakout	Enduro	Pong	Q*Bert	Seaquest	S.Invaders
Random	354	1.2	0	-20.4	157	110	179
Sarsa	996	5.2	129	-19	614	665	271
Contingency	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	3616	52	106	19	1800	920	1720
HNeat Pixel	1332	4	91	-16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	**1075**

The upper table compares average total reward for various learning methods by running an ϵ-greedy policy with ϵ= 0.05 for a fixed number of steps. The lower table reports results of the single best performing episode for HNeat and DQN. HNeat produces deterministic policies that always get the same score while DQN used an ϵ-greedy policy with ϵ = 0.05.

New questions/hypothesis

It can be observed that combining neural networks with RL has given better results in many cases in which consist of difficult control policies. In this task for sampling, experienced replay was used, instead using prioritized replay with DQN could give better results.

Implementing other variants like Double DQN and Dueling DQN could also beat the state-of-the art performance achieved by DQN.

Links-

1. Prioritized Experience Replay with DQN can be found here https://arxiv.org/pdf/1511.05952.pdf
2. Deep Reinforcement Learning with Double Q-learning paper can be found here https://arxiv.org/pdf/1509.06461.pdf
3. Dueling DQN on Atari Games can be found here https://arxiv.org/pdf/1511.06581.pdf