RL Lab 3

Summary of the DQN algorithm

<u>Training process outlined in figure 3:</u>

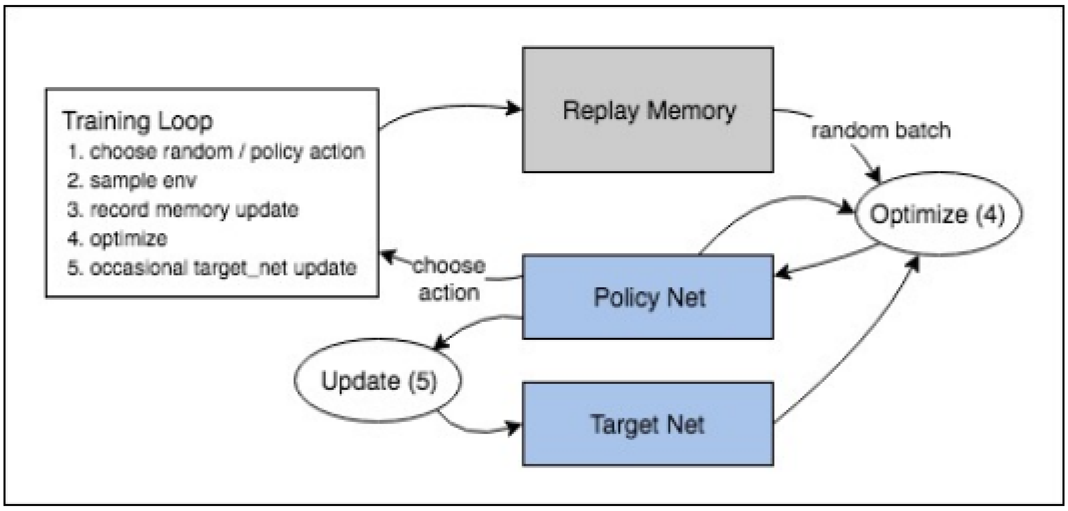

Figure 3: Training loop overview

This algorithm combines Q-learning and neural networks (in this case convolutional neural networks) to map states to their action value. The agent uses experience replay to basically learn from states it has already encountered. The algorithm plays randomly for the first 10000 steps in order to populate the replay buffer then starts learning after those 10000 steps.
The agent first chooses a random/policy action.

- A random action is chosen with probability epsilon and exploits with probability 1 - epsilon
- A random action is any action in the action space of the environment.
- A policy action is the action given by the policy network (given a state, it gives the action that has the highest q value).

Using this action, it takes a step in the environment.

- The environment returns the reward, next state and a indicator variable “done” for taking the action
    - “Done” indicates whether the state is terminal or not

It then records(saves) the state, action, reward, next state and “done” to the memory replay buffer.
The network learns using “experience replay” by sampling from the replay memory buffer.

- Learning means the policy network’s weights are optimised
- This updates the network weights by backpropagating using the sampled batch of data (the network trains using the batch from memory)
- The TD target is calculated using the target network and the bellman equation
- Using the TD target, the policy network weights are fitted to the target values

The target network is occasionally (e.g. every n steps) updated.

- The target network weights are set to be the policy network weights.

Q-Network architecture

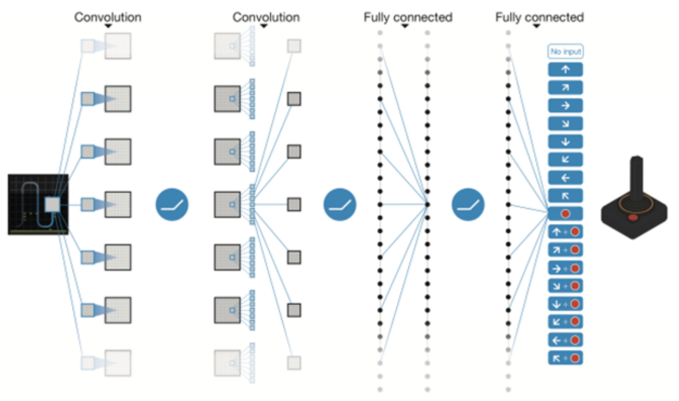

Figure 1: Deep Q Network (Source: Nature paper)

The policy and target networks take the form illustrated in the above figure. The network takes a state and passes it through a convolution neural network and then through a fully connected network. For each action, it produces a q-value given the state. The network basically maps input states with actions.The agent takes the action that has the highest q-value.
The first 2 hidden layers are convolutional layers and the last layers are fully connected layers. The hyperparameters of the CNN and fully connected layers will be subsequently discussed.

Hyper-parameters

Q-learning

Name	**Value**	**Role**
Learning rate	1e-4	Magnitude of the amount we update the weights of the network during backpropagation (specifically gradient descent).
Discount factor	0.99	How much the agent cares about future rewards. In this case the agent cares about future rewards.

Batch size	32	Number of transitions used when sampling from the replay buffer during optimization

Epsilon	Start = 1
	End = 0.01
	Fraction = 0.1
		We want to explore in the beginning but over time exploit thus we decrease the epsilon value by the fraction value from start to end.

Learning frequency	1 (only after 10000 steps)	Number of steps between every optimization of the policy network
Update frequency	1000 (steps)	Because episodes have different number of steps, it is more stable to copy the target weights to the policy weights after N steps instead of N episodes.

CNN

Name	**Value**	**Role**
Number of conv layers	3	Convolutions extract low-high level features from the images. Different-level characteristics are present in each conv layer. Lower layers tend to extract primitive features and higher layers extract more complex features.
Filters in a conv layer	Layer 1: 16 Layer 2: 32 Layer 3: 64	The conv layer has filters that scan through the state/image and extract certain features.
Kernel size	Layer 1: 8 x 8 Layer 2: 4 x 4 Layer 3: 3 x 3	This is the size of the filter. The size of the kernel affects the complexity of the features being extracted.
Stride size	Layer 1: 4 Layer 2: 2 Layer 3: 1	Filters shift through the image using the stride value. The stride value

		modulates the density of convolving. Larger strides lead to more loss of information but less correlation with the pixels.

Fully connected network

Name	**Value**	**Role**
Neurons in hidden layer	512	This acts like a multi-layer perceptron that takes the learnt features and classifies them.
Neurons in output layer	Number of actions	In the last layer the number of neurons amounts to the number of classes/actions so that the state can be classified.