<u>Human-level control through Deep Reinforcement learning</u> (Aravind S - EE14B013 - RL project RA 5)

Humans learn from continuous interaction from environment and the way we learn things is still a question to the entire scientific community. Due to success of many RL algorithms, people believed RL will lead to general AI. But if we observe how humans learn - we just use very basic sensory inputs and raw feedback from the environment. Somehow we are able to develop an understanding about what our environment is, what actions to take and we then infer from that. Understanding complex representations posed a problem to researchers till the advent of deep learning. David Silver believes that a combination of Deep learning and reinforcement learning algorithms leads to General Artificial Intelligence. It is still a debated topic, but it makes sense. Deep learning helps in understanding complex representations and Reinforcement Learning helps in achieving our end goal - taking proper actions at the right time. This paper is about **“Deep Q networks”** and it talks about this idea, how DL and RL can be used to understand different environments and draws parallels to we humans learn. For experiments, they have used **“Atari games”** and compared the agent’s performance with other humans. As the model should be as close to human as possible, the inputs should be as raw as possible. So raw pixels with minimal preprocessing should be fed as input to the network. Since we can’t interpret much from a single frame a sequence of consecutive frames seems to be a valid alternative. Outputs from the network is a probability distribution over all actions which can be taken by the agent from the current state. In other words, the deep net models Q(s, a). Network should be deeper to learn complex features whereas optimisation involves a pinch of RL, Q-learning. Training data is a sequence of frames along with the reward obtained. Here, they have also used “number of lives left” as we want the agent to understand that it can die as long as it has at least one more life left.

Researchers have worked on similar problems before but they couldn’t succeed due to a variety of reasons. One problem with this approach is that training samples are correlated i.e there won’t be much of a difference between successive training samples. Another problem is due to the way in which the samples are generated. Suppose we are generating samples by interacting with a simulator following some behaviour policy. We want to learn the optimal policy somehow. If we use an algorithm like Q-learning with GPI, the training samples obtained later will be due to a different policy i.e the training data comes from a non-stationary distribution. Neural networks learning from correlated instances with non-stationarity tend to fail as they may fall into some unwanted feedback loop or diverge. These 2 problems are addressed by the paper and there are some modifications made to the original Q-learning algorithm. As a result, the model learnt to play 49 Atari games to near/better than human performance with the same network architecture and hyperparameter setup - thus taking a step towards **“General AI”**.

To avoid correlated inputs, they used **Experience Replay**. Experience replay is a biologically inspired technique where we learn things by thinking about past experiences. The recent ‘N’ training samples are stored in a buffer and a randomly sampled batch is taken and fed to the network. Weight updation is done using RMSProp algorithm. This reduces correlation between the inputs fed to the network. Instead of a randomly sampled batch, a more intelligent sampling can be done which emphasizes on samples which aren’t learnt properly - a possible extension to the current approach.

The current parameters of the network do not control the {input, output} of the network. So it is necessary to go for Off-policy methods where we use a sufficiently explorative behaviour policy and the network learns the optimal target policy: the choice of Q-learning is justified. One more thing to note here that, the target values for outputs depend on Q(s, a) itself. It is as if the network learns from its own understanding! As awkward as it sounds, practically this leads to instability. To overcome this issue, targets are computed using an **earlier version** of Q(s, a). Targets derived from older Q-values reduce oscillations and hence reduce the chances of divergence. We haven’t completely solved the problem, but deep nets trained using this idea learn better in spite of being unstable. **Reward clipping** is used in this approach. In all these experiments, reward was the change in score of the player and it was clipped between -1 and 1. Rectifier units in the network have gradients -1 or 1. So, while backpropagating weights if the rewards are limited to [-1, 1], it enhances stability while training. But the model tends to lose information about the relative magnitude of rewards - one obvious drawback in this method.

Interesting things have been found after analysing what representations the deep net has learnt. As expected, representations of perceptually similar states are nearby (DL ensures this). Representations of perceptually dissimilar states which have similar expected reward are also nearby (RL algorithms!). t-SNE was used to analyse the “closeness” between the different states. Another interesting thing to note is that the same network (same architecture, hyperparameters) generalised well and learnt to play many Atari games to levels comparable to human.

There are some Atari games where the logic is complicated and even humans find it tough to figure out ways to obtain higher rewards. Learning such complicated methods is still a problem for Deep Q networks. Many improvements have been suggested to this approach which try to fix one issue or the other. **Double DQN** is a framework derived from DQN using the idea of Double learning and it reduces the maximisation bias due to Q-learning updates. Another method is the usage of **Prioritized replay** where experiences are weighed according to DQN error. This makes sense, because higher the error the newer it is to the network (a surprise to the network) and hence it should be given priority over other “familiar” experiences. **Duelling network** tries a different approach by decomposing Q into a an action-independent value function and an action-dependent advantage function.