Human Level Control Through Deep Reinforcement Learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver

Presented By
Kaivalya Anand Pandey
Novosibirsk State University

March 23, 2020

Introduction

- The theory of reinforcement learning provides a normative account, deeply rooted in psychological and neuroscientific perspectives on animal behaviour, of how agents may optimize their control of an environment.
- To use reinforcement learning successfully in situations approaching real-world complexity, however, agents are confronted with a difficult task: they must derive efficient representations of the environment from high-dimensional sensory inputs, and use these to generalize past experience to new situations.

- Remarkably, humans and other animals seem to solve this problem through a harmonious combination of reinforcement learning and hierarchical sensory processing systems.
- While reinforcement learning agents have achieved some successes in a variety of domains, their applicability has previously been limited to domains in which useful features can be handcrafted, or to domains with fully observed, low-dimensional state spaces.
- Recent advances in training deep neural networks9–11 to develop a novel artificial agent, termed a deep Q-network, that can learn successful policies directly from high-dimensional sensory inputs using end-to-end reinforcement learning.
- Testing Platform - Atari 2600 Games.

Deep Q Network Agent

- Receiving only pixels and the game score as inputs, was bale to surpass the performance of all previous algorithms and achieve a level as compared to that of professional human game tester across a set of 49 games.
- This work bridges the divide between high-dimensional sensory inputs and actions, resulting in the first artificial agent that is capable of learning to excel at a diverse array of challenging tasks.
- Goal is to create a single algorithm that would be able to develop a wide range of competencies on a varied range of challenging tasks,a central goal of general artificial intelligence.

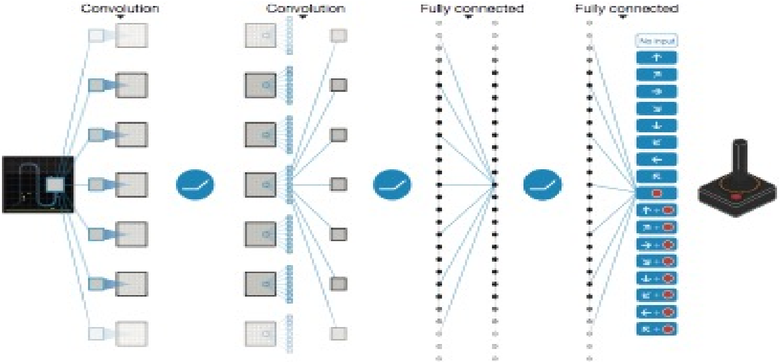
Deep Q Network Agent

- To achieve this, we developed a novel agent, a deep Q-network (DQN), which is able to combine reinforcement learning with a class of artificial neural network known as deep neural networks.
- Notably, recent advances in deep neural networks, in which several layers of nodes are used to build up progressively more abstract representations of the data, have made it possible for artificial neural networks to learn concepts such as object categories directly from raw sensory data.
- We use one particularly successful architecture, the deep convolutional network,which uses hierarchical layers of tiled convolutional filters to mimic the effects of receptive fields, thereby exploiting the local spatial correlations present in images, and building in robustness to natural transformations such as changes of viewpoint or scale.

- We use deep convolutional neural network to approximate the optimal action value-function.

$$Q*(s,a) = maxE[r_t + \gamma * r_t + 1 + \gamma * \gamma * r_t + 2 + ...|s_t = s, a_t = a, \pi] \quad (1)$$

which is maximum sum of rewards $r_t discountedby \gamma$ at each time step t, achievable by a behaviour policy $\pi = P(a|s)$, after making an observation (s) and actions (a).



- We address these instabilities with a novel variant of Q-learning which uses two key ideas. Firstly, we used a biologically inspired mechanism termed as experience replay that randomises the data meaning removing correlations in the observation sequence. Secondly, we used an iterative update that adjusts the action values(Q) towards target values that are periodically updated, thereby reducing correlations.
- The Q-learning update at iteration i uses the following loss function :

$$L_i(\Theta_i) = \mathbb{E}_(s, a, r, s') \ U(D)[(r + \gamma maxQ(s', a', \Theta_i^{`}) - Q(s, a, \Theta_i))^2] \tag{2}$$

- To evaluate our DQN agent, we use Atari 2600 games platform whcih offers a diverse array of tasks (n=49).

Methods

- Preprocessing - Working directly with raw Atari 2600 frames which are 210 X 160 pixel images with 128 color palette, can be demanding in terms of memory and computation. We apply a basic preprocessing step aimed at reducing the input dimensionality and dealing with some artefacts of Atari 2600 emulator.
- First to encode a single frame we take maximum value for each for each pixel color value over the being frame encoded and the previous frame. This was necessary to remove flickering in games as some objects only appear in even frames while some in only odd frames.
- Second then we extract the Y-channel, also known as luminance, from the RGB frame and rescale it to 84 X 84.

Methods

- Code Availability - The source code can be accessed at https://sites.google.com/a/deepmind.com/dqn.
- Model Architecture - The input to neural networks consists of 84X84 X4 image produced by preprocessing. The first hidden layer convolves 32 filters of 8X8 with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 64 filters of 3X3 with stride 1 follows by a rectifer. The final hidden layer is fully connected and consists of 512 rectifier units. The output layer is fully connected linear layer with a single output for each valid action. The number of valid actions varied between 4 to 18 on the games we considered.
- Training Details - RMS Prop algorith with mini batch size of 32. The behaviour policy was from 1.0 to 0.1 over the first million frames and was fixed at 0.1 after. We trained over a 50 million frames and used a replay memory of over 1 million most recent frames.

Evaluation Procedure

- The trained agents were evaluated by playing each game 30 times for up to 5 minutes each time with different initial random conditions.
- This procedure is adopted to minimise the possibility of overfitting during evaluation. The random agent served as a baseline comparison and chose a random action at 10Hz which is every sixth frame, repeating its last action on intervening frames.
- 10 Hz is the fastest a human can select the 'fire' button, and setting the random agent to this frequency avoids spurious baseline of scores in a handful of the games.

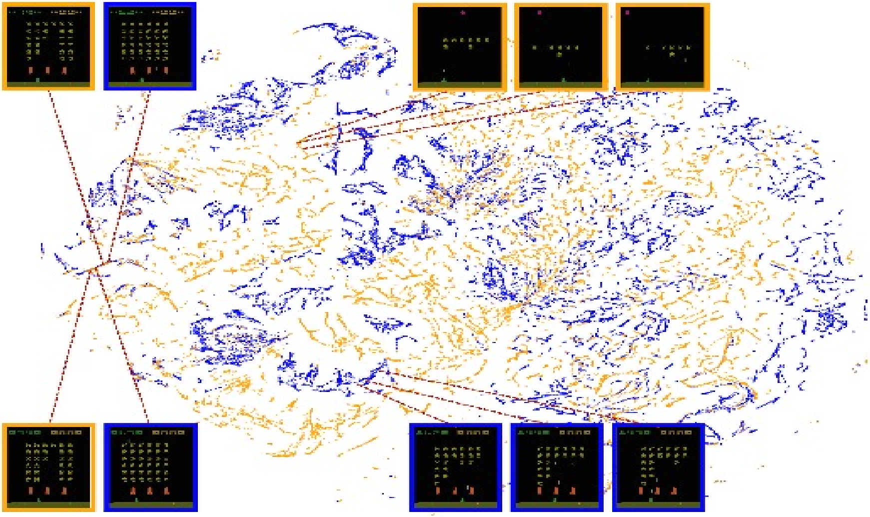
Algorithm

- At each time-step the agent selects an action at from the set of legal game actions, A= 1, . . . ,K . The action is passed to the emulator and modifies its internal state and the game score. In general the environment may be stochastic.
- The emulator's internal state is not observed by agent, instead agent observe an image$x_t \epsilon R^d$ which is a vector pixels representing current screen.
- In addition it receives a reward rt representing the change in game score. Note that in general the game score may depend on the whole previous sequence of actions and observations; feedback about an action may only be received after many thousands of time-steps have elapsed.

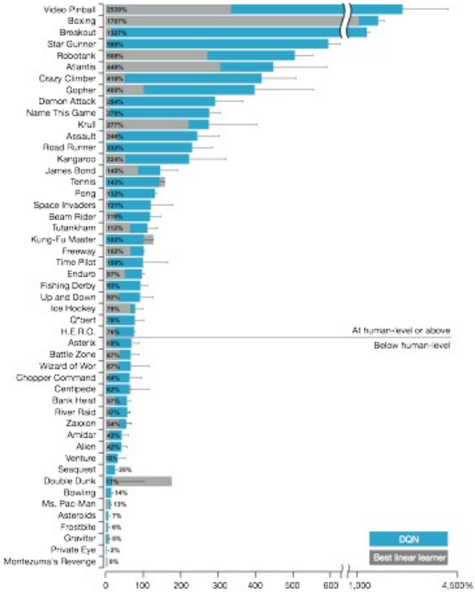
Algorithm

- Because the agent observes the current screen, the task is partially observed and many emulators state are perceptually aliased.
- Therefore the sequences of actions and observations, $s_t = x_1, a_1, x_2, ..., a_t - 1, x_t$ are input to algorithm, which then learns game strategies, depending upon the sequences.
- All sequences in the emulator are assumed to terminate in a finite number of time- steps. This formalism gives rise to a large but finite Markov decision process (MDP) in which each sequence is a distinct state. As a result, we can apply standard rein- forcement learning methods for MDPs, simply by using the complete sequence.

2-D t-SNE embedding of last layer assign by DQN



Results


Video Pinball
Boxing
Breakout
Star Gunner
Robotank
Atlantis
Crazy Climber
Gopher
Demon Attack
Name This Game
Krull
Assault
Road Runner
Kangaroo
James Bond
Tennis
Pong
Space Invaders
Beam Rider
Tutankham
Kung-Fu Master
Freeway
Time Pilot
Enduro
Fishing Derby
Up and Down
Ice Hockey
Q*bert
H.E.R.O.
Asterix
Battle Zone
Wizard of Wor
Chopper Command
Centipede
Bank Heist
River Raid
Zaxxon
Amidar
Alien
Venture
Seaquest
Double Dunk
Bowling
Ms. Pac-Man
Asteroids
Frostbite
Gravitar
Private Eye
Montezuma's Revenge
At human-level or above
Below human-level
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%

Extended Data Table 2 | Comparison of games scores obtained by DQN agents with methods from the literature[12,15] and a pro human games tester

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%
Boxing	0.1	44	9.8	4.3	71.8 (±8.4)	1707.9%
Breakout	1.7	5.2	6.1	31.8	401.2 (±26.9)	1327.2%
Centipede	2091	8803	4647	11963	8309(±5237)	63.0%
Chopper Command	811	1582	16.9	9882	6687 (±2916)	64.8%
Crazy Climber	10781	23411	149.8	35411	114103 (±22797)	419.5%
Demon Attack	152.1	520.5	0	3401	9711 (±2406)	294.2%
Double Dunk	-18.6	-13.1	-16	-15.5	-18.1 (±2.6)	17.1%
Enduro	0	129.1	159.4	309.6	301.8 (±24.6)	97.5%
Fishing Derby	-91.7	-89.5	-85.1	5.5	-0.8 (±19.0)	93.5%
Freeway	0	19.1	19.7	29.6	30.3 (±0.7)	102.4%
Frostbite	65.2	216.9	180.9	4335	328.3 (±250.5)	6.2%
Gopher	257.6	1288	2368	2321	8520 (±3279)	400.4%
Gravitar	173	387.7	429	2672	306.7 (±223.9)	5.3%
H.E.R.O.	1027	6459	7295	25763	19950 (±158)	76.5%
Ice Hockey	-11.2	-9.5	-3.2	0.9	-1.6 (±2.5)	79.3%
James Bond	29	202.8	354.1	406.7	576.7 (±175.5)	145.0%
Kangaroo	52	1622	8.8	3035	6740 (±2959)	224.2%
Krull	1598	3372	3341	2395	3805 (±1033)	277.0%
Kung-Fu Master	258.5	19544	29151	22736	23270 (±5955)	102.4%
Montezuma's Revenge	0	10.7	259	4367	0 (±0)	0.0%
Ms. Pacman	307.3	1692	1227	15693	2311(±525)	13.0%
Name This Game	2292	2500	2247	4076	7257 (±547)	278.3%
Pong	-20.7	-19	-17.4	9.3	18.9 (±1.3)	132.0%
Private Eye	24.9	684.3	86	69571	1788 (±5473)	2.5%
Q*Bert	163.9	613.5	960.3	13455	10596 (±3294)	78.5%
River Raid	1339	1904	2650	13513	8316 (±1049)	57.3%
Road Runner	11.5	67.7	89.1	7845	18257 (±4268)	232.9%
Robotank	2.2	28.7	12.4	11.9	51.6 (±4.7)	509.0%
Seaquest	68.4	664.8	675.5	20182	5286(±1310)	25.9%
Space Invaders	148	250.1	267.9	1652	1976 (±893)	121.5%
Star Gunner	664	1070	9.4	10250	57997 (±3152)	598.1%
Tennis	-23.8	-0.1	0	-8.9	-2.5 (±1.9)	143.2%
Time Pilot	3568	3741	24.9	5925	5947 (±1600)	100.9%
Tutankham	11.4	114.3	98.2	167.6	186.7 (±41.9)	112.2%
Up and Down	533.4	3533	2449	9082	8456 (±3162)	92.7%
Venture	0	66	0.6	1188	380.0 (±238.6)	32.0%
Video Pinball	16257	16871	19761	17298	42684 (±16287)	2539.4%
Wizard of Wor	563.5	1981	36.9	4757	3393 (±2019)	67.5%
Zaxxon	32.5	3365	21.4	9173	4977 (±1235)	54.1%

Results

Extended Data Table 3 | The effects of replay and separating the target Q-network

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0