Deep Reinforcement Learning for Flappy Bird

Kevin Chen
Stanford University

Abstract

Reinforcement learning is essential for training an agent to make smart decisions under uncertainty and to take small actions in order to achieve a higher overarching goal. **In this project, we combined reinforcement learning and deep learning techniques to train an agent to play the game, Flappy Bird.** The challenge is that the agent only sees the pixels and the rewards, similar to a human player. Using just this information, it is able to successfully play the game at a human or sometimes super-human level.

Related Work

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattle, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, *Nature* **518**, 529-533 (2015).

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv: 1312.5602, 2013.*

Reinforcement Learning

State: Sequence of frames and actions

$$s_t = x_1, a_1, x_2, a_2, \ldots x_{t-1}, a_{t-1}, x_t$$

Action: Flap (a = 1) or Do nothing (a = 0)

Rewards:

rewardAlive	rewardPipe	rewardDead
+0.1	+1.0	-1.0

Q-learning:

$$Q^*(s, a) = \mathbf{E}_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q^*(s', a') \mid s, a]$$

$$Q_{i+1}(s, a) \leftarrow \mathbf{E}_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q_i(s', a') \mid s, a]$$

$$\text{Loss } L_i(\theta_i) = \mathbf{E}_{s, a \sim p(\cdot)}[(y_i - Q(s, a; \theta_i))^2]$$

$$y_i = \mathbf{E}_{s' \sim \varepsilon}[r + \gamma \max_{a'} Q_{target}(s', a'; \theta_{target}) \mid s, a]$$

Pipeline

initialize replay memory and DQN → new episode → next frame → update state and replay memory → choose action based on ε-greedy policy → sample minibatch from replay memory and update DQN

if crash (→ new episode)

if not crash (→ next frame)

Feature extractor

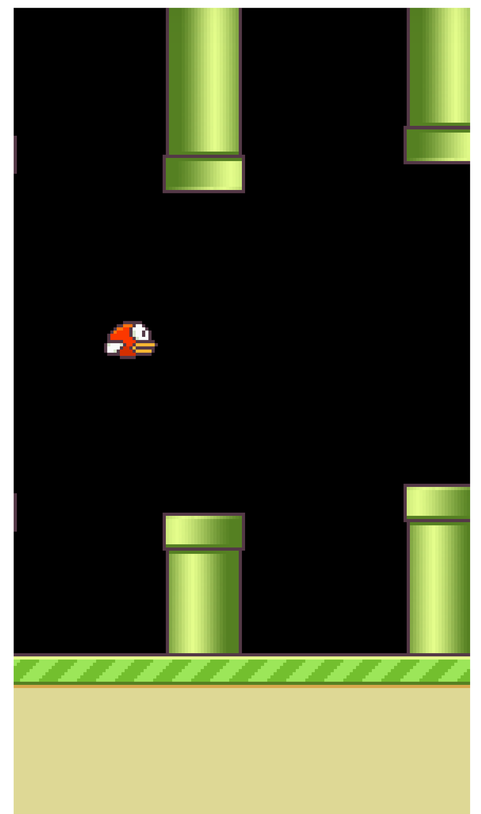
extract images from state → convert to grayscale → downsample to 84x84

Deep Q-Network (DQN)

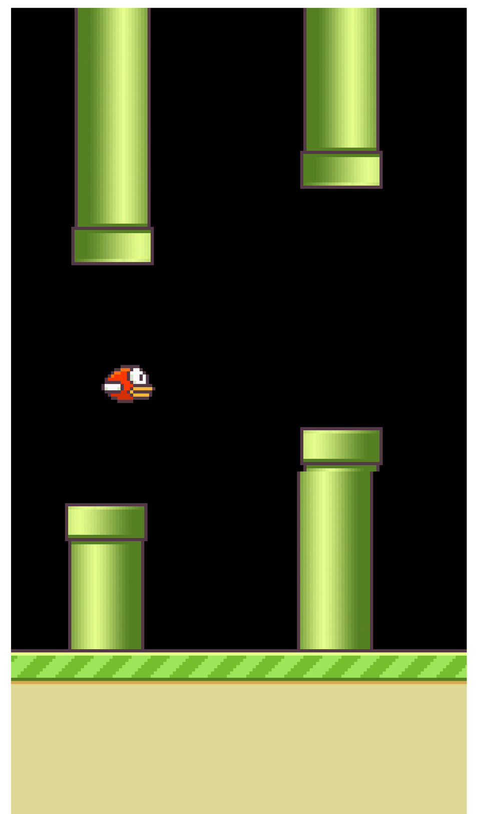
n-channel image (**n** most recent frames) → Deep Q-Network (DQN) → Q-value for each action

Experimental Results

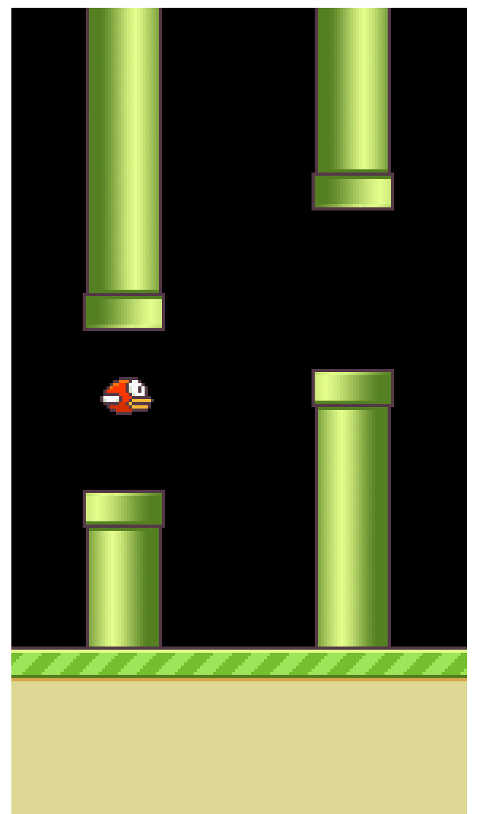
Easy



Medium



Hard



Average Score

Game difficulty	Human	Baseline (flap every n)	DQN (easy)	DQN (medium)	DQN (hard)
Easy	Inf	Inf	**Inf**	Inf	Inf
Medium	Inf	Inf	0.7	**Inf**	Inf
Hard	21	0.5	0.1	0.6	**82.2**

Highest Score Achieved

Game difficulty	Human	Baseline (flap every n)	DQN (easy)	DQN (medium)	DQN (hard)
Easy	Inf	Inf	**Inf**	Inf	Inf
Medium	Inf	11	2	**Inf**	Inf
Hard	65	1	1	1	**215**