Playing FPS Games with Deep Reinforcement Learning

Guillaume Lample, Devendra Singh Chaplot
Presented by Mark Iwanchyshyn

Introduction

Doom, the video game

Make an agent that can play deathmatch games in Doom

The input is the 60x108 colour screen

The agents actions are: turn {left, right}, walk forward, shoot, etc, (a subset of what the game provides)

Doom Details

The game is early 3D and automatically compensates for aiming differences in elevation. So only left and right are necessary.

In the ‘deathmatch’ game each agent tries to maximise their number of kills vs their number of deaths.

The agent can pick up health or ammunition throughout the level.

Proposed Agent (Simplified)

A deep neural network that is a Long Short Term Memory cell on top of a Convolutional Neural Net.

The intuition is that the CNN can process the raw image data and produce some higher level information that the LSTM can do something with.

The Proposed Solution

Deep Recurrent Q Networks (DRQN)

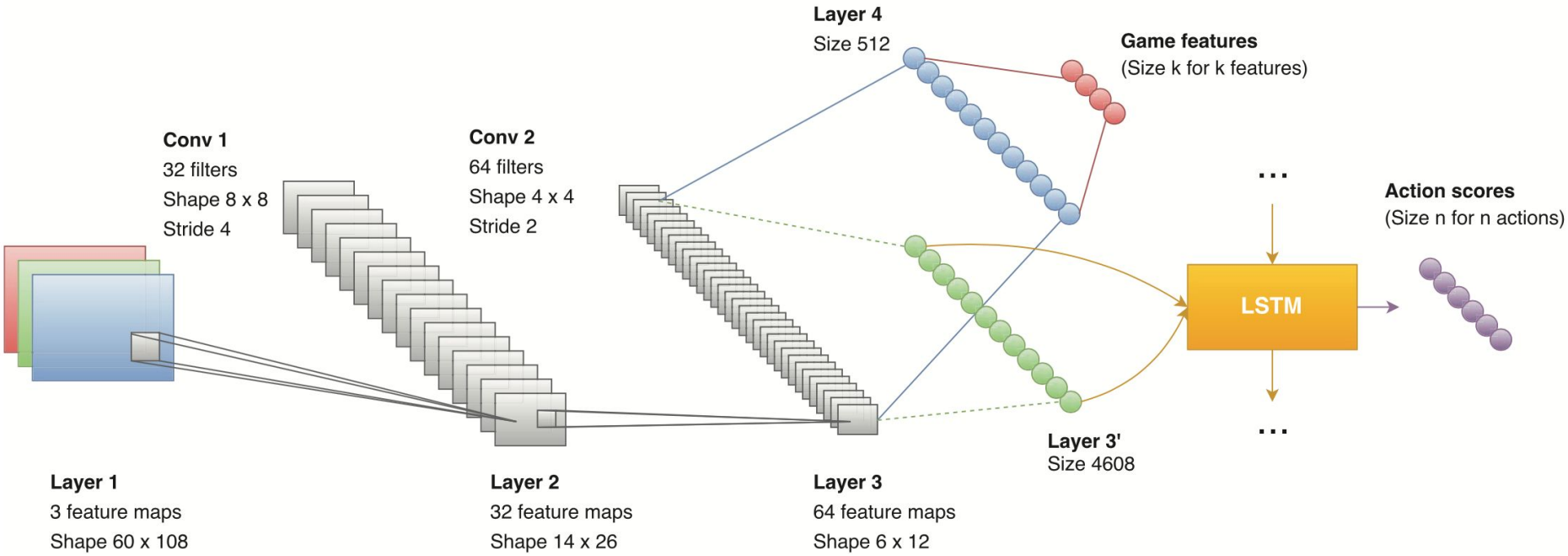
Instead of estimating $Q(o_t, a_t)$, we want $Q(o_t, h_{t-1}, a_t)$. Where h_{t-1} is some other output of our function at the previous timestep.

This is implemented as:

$h_t = LSTM(h_{t-1}, o_t)$

We estimate our Q as $Q(h_t, a_t)$

Network Structure



Notes on Network Structure

Layer 3’ is layer 3 flattened

Each convolution has a third input dimension that is the number of feature maps in the previous layer

The size of the LSTM hidden state is never specified

The entire structure seems to be strongly based on their citation of Hausknecht and Stone (2015): https://arxiv.org/abs/1507.06527

This source also talks about screen flicker in games which was covered in this course.

Game feature augmentation

To improve training the network is not only trained reinforcement-wise using the reward function.

During training the network is also trained to extract features about the world that their game engine provides: is there an enemy on the screen? Am I out of ammunition?

These are the size-k game features in the network.

This way the CNN is jointly trained, and the authors theorise this helps it extract information about the current frame.

Navigation Network

Two separately trained networks were used for the agent. Identical structure, but the navigation network could only move.

Swapping between the Navigation network and Action network was determined by the presence of enemies on the screen, an output that was trained from a game feature.

This network was easier to train and encouraged searching for health and ammo instead of ‘camping’.

Training

Reward shaping: Positive for picking up items, negative for losing health, negative for shooting, positive for distance traveled since last step (prevents turning in circles)

The navigation network was at times trained on a map without enemies just so it would learn to efficiently pick up items.

Frame skip: only each k^{th} frame is considered and the action decided is repeated (equivalent to key held down) for the next k frames. In the paper they decide on considering every 5^{th} frame.

Training Details

Used RMSProp algorithm

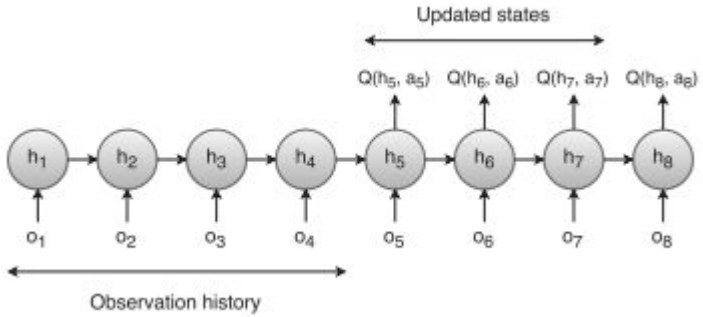
Replay memory of 1 million most recent frames

Minibatch size of 32

Epsilon greedy starting at 1 going to 0.1 over the first million frames

Discount factor of 0.99

Only experiences with enough history are backpropagated



Evaluation

Scenarios

Limited deathmatch on a known map

Only weapon is rocket launcher that all agents start with

Single known map

Full deathmatch on unknown maps

All agents start with pistol and must pick up other weapons

10 maps for training, 3 maps for testing

	Limited Deathmatch		Full Deathmatch			
	Known Map		Train maps		Test maps	
Evaluation Metric	Without navigation	With navigation	Without navigation	With navigation	Without navigation	With navigation
Number of objects	14	46	52.9	92.2	62.3	94.7
Number of kills	167	138	43.0	66.8	32.0	43.0
Number of deaths	36	25	15.2	14.6	10.0	6.0
Number of suicides	15	10	1.7	3.1	0.3	1.3
Kill to Death Ratio	4.64	5.52	2.83	4.58	3.12	6.94

Opponents

The opponents used in this paper were mostly the built-in doom ‘bots’

20 human players were also used to evaluate the agent. As best I can figure out these were university volunteers, definitely not professionals.

	Single Player		Multiplayer	
Evaluation Metric	Human	Agent	Human	Agent
Number of objects	5.2	9.2	6.1	10.5
Number of kills	12.6	27.6	5.5	8.0
Number of deaths	8.3	5.0	11.2	6.0
Number of suicides	3.6	2.0	3.2	0.5
K/D Ratio	1.52	5.12	0.49	1.33

Single player scenario is both humans and the agent playing against bots in separate games.

Multiplayer scenario is agent and human playing against each other in the same game.

Conclusions

Contributions

Another game humans are worse at!

Demonstrating the usefulness of truths (game features) in training rather than pure experience. And on a related note, the effectiveness of jointly training one network on multiple objectives.

Future Work

This paper expands a 2D game playing LSTM model to 3D.

This can be further extended to other 3D games or 3D environments.

My opinions

The use of separate Navigation and Action networks controlled by some pre-set (non-learned) criteria seems to indicate that the model used isn't expressive enough. It can also be cheated if the players are aware of this weakness, for example the agent can't fire a rocket if it expects an opponent to come around a corner before it has seen them.

Knowing how much hidden state the LSTM has is necessary to replicate the work.

A paper demonstrating exactly what we learned in class, seriously go look at the slides for 12: Deep recurrent Q-networks. Hausknecht and Stone (2016) cited in the notes are the same authors as Hausknecht and Stone (2015) cited by this paper.

Questions