Human-level control through deep reinforcement learning

Thierry Deruyttere & Armin Halilovic

Overview

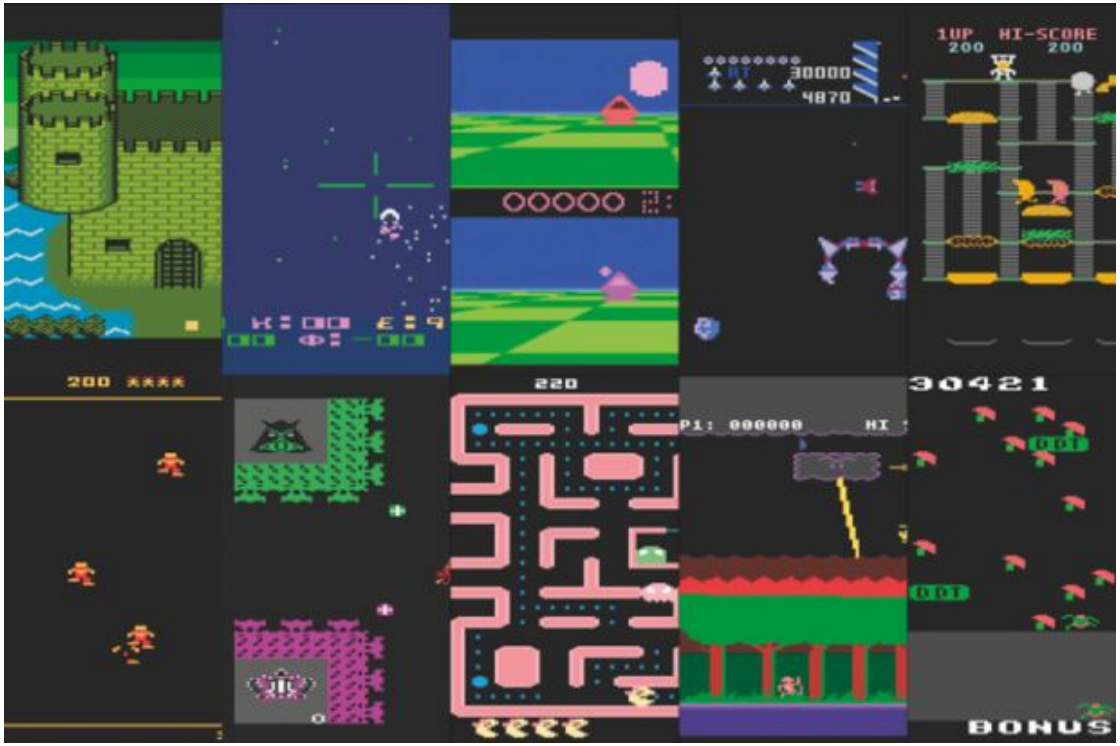
1. Introduction
2. Reinforcement learning
3. Deep learning
4. Deep reinforcement learning
5. Demo!
6. What may be next

Introduction - Problem statement

We want to create an agent that can:

- learn to solve problems
- with human-level performance
- without any human interference

Introduction - Problem statement



retrogamer.net/top_10/top-ten-atari-2600-games

Introduction - An approach

A combination of:

- Reinforcement learning
- Deep learning

Deep reinforcement learning!

Reinforcement learning

Reinforcement learning

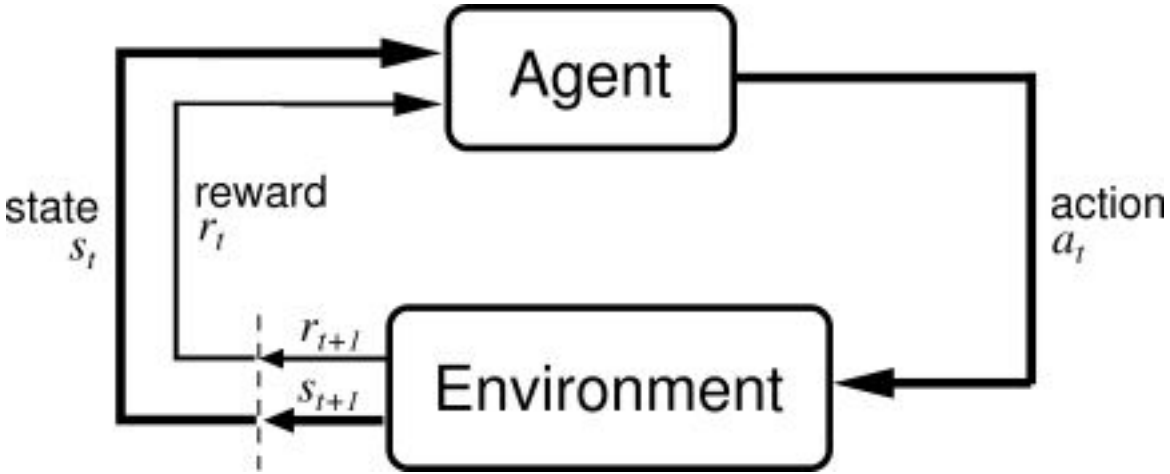
We want to create an agent that:

- given a state **s** (e.g: lives left, current pos, pos of enemies, …)
- returns an action **a** (e.g: move left, move right, fire, …)
- gets a reward **r** for that action (from environment)

Agent's goal:

- learn a **policy π** that will maximise the sum of rewards
- **π** = function that says which action **a** to take in state **s**

Reinforcement learning (visual)



Richard Sutton and Andrew Barto, Reinforcement Learning: An Introduction (1st Edition, 1998)

Reinforcement learning - Q-learning

Computes the optimal **policy π**

Iterative improvement

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}^{\text{learned value}}$$

Implemented using tables or neural networks (function approximation)

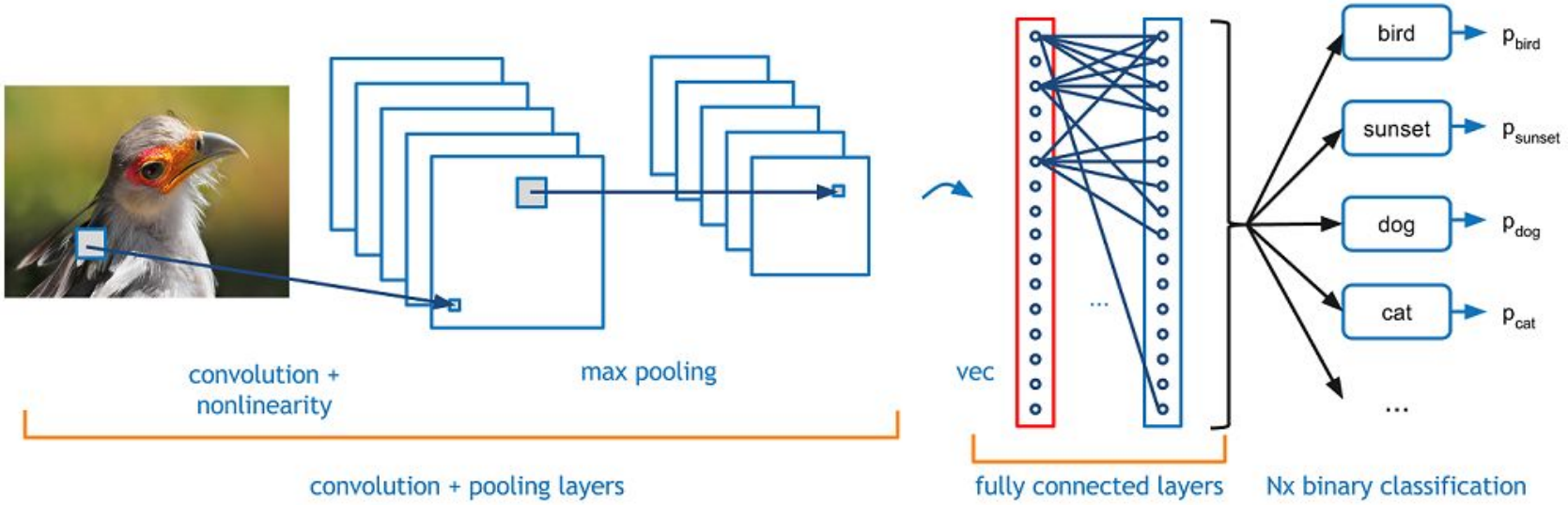
wikipedia.org/wiki/Q-learning

Deep learning

Deep learning - CNN

Maps image to class probabilities

Classification example: return the likeliest category for this picture



adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

CNN - How it’s done

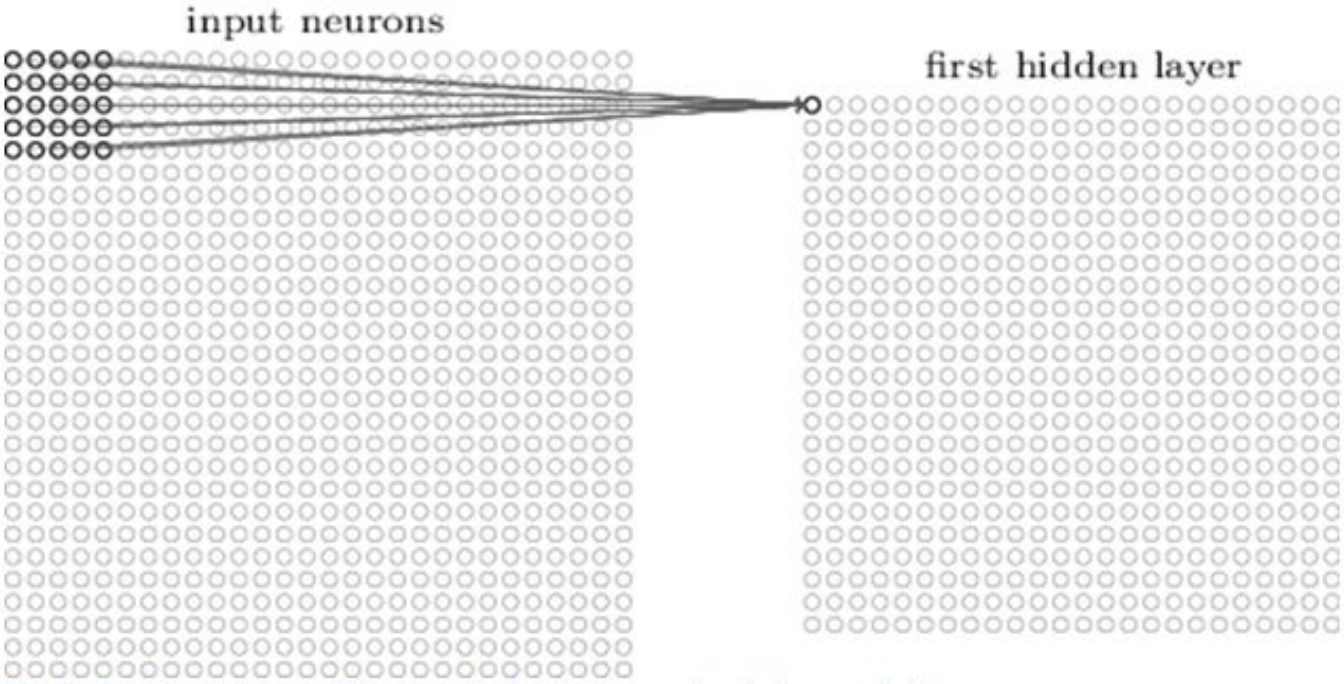
What We See

```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What Computers See

adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

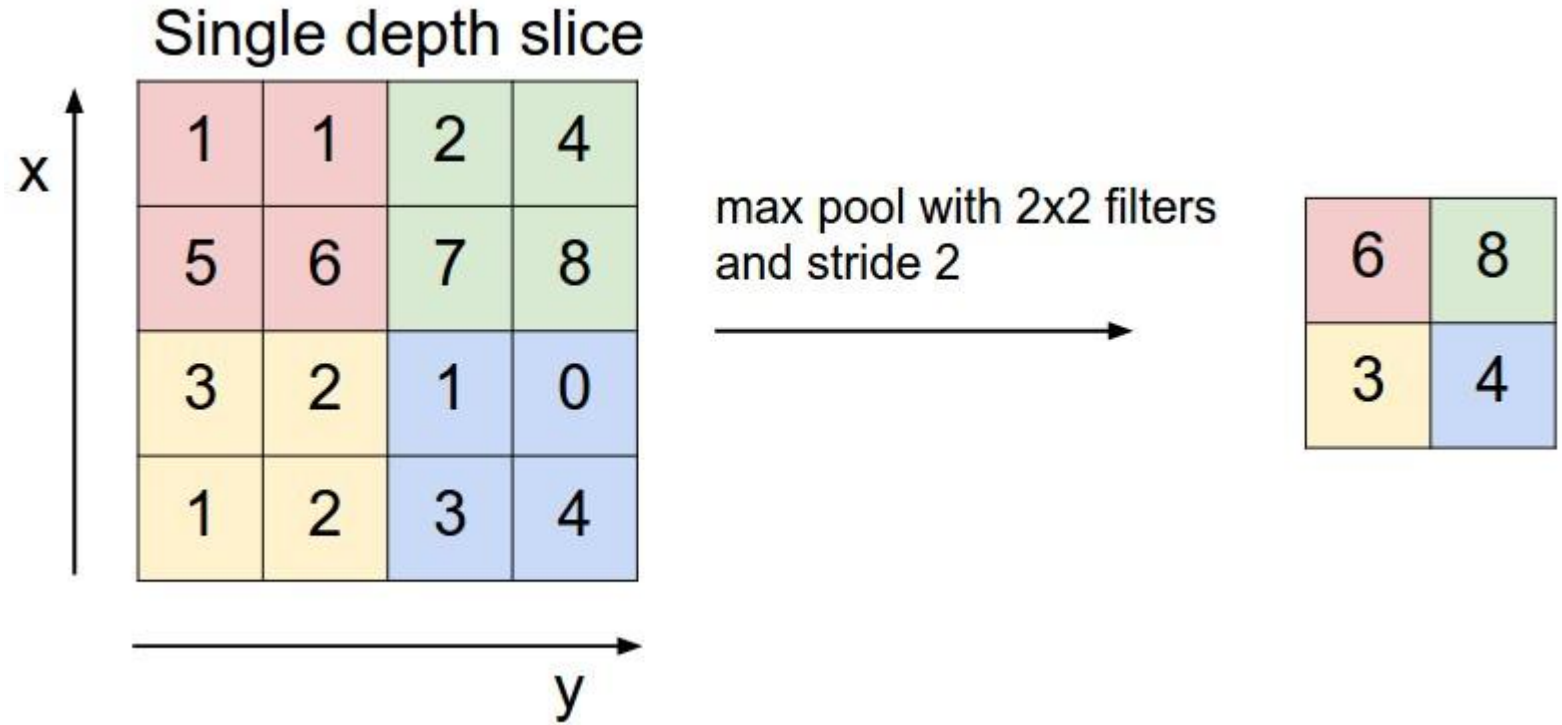
CNN - Convolution



Visualization of 5 x 5 filter convolving around an input volume and producing an activation map

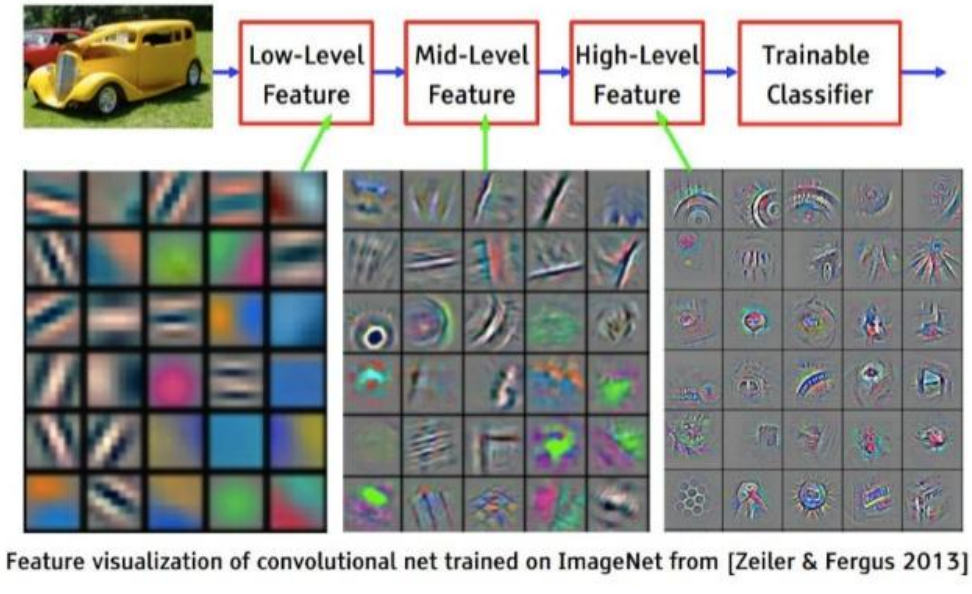
http://neuralnetworksanddeeplearning.com/images/tikz44.png

CNN - Max pooling



http://cs231n.github.io/convolutional-networks/

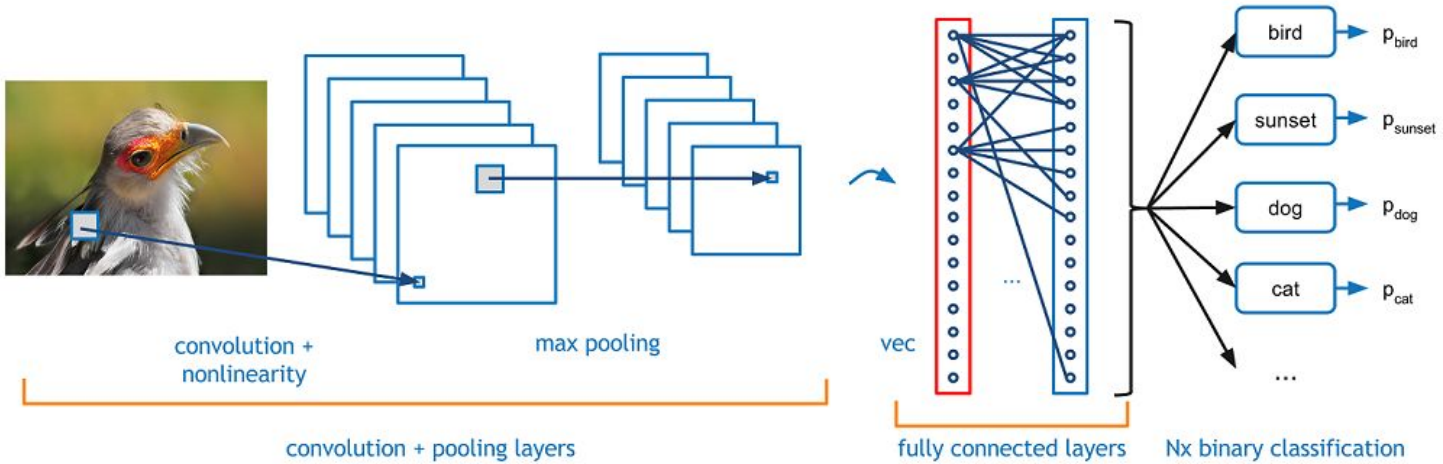
CNN - What it sees



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Stack convolutional layers and then stack a classifier on top to predict the class

CNN - How to train it (loss function)



Input: image **x_i** with known class **y_i**
Output: prediction **h(x_i)**
Loss: distance metric between real class **y_i** and output **h(x_i)**

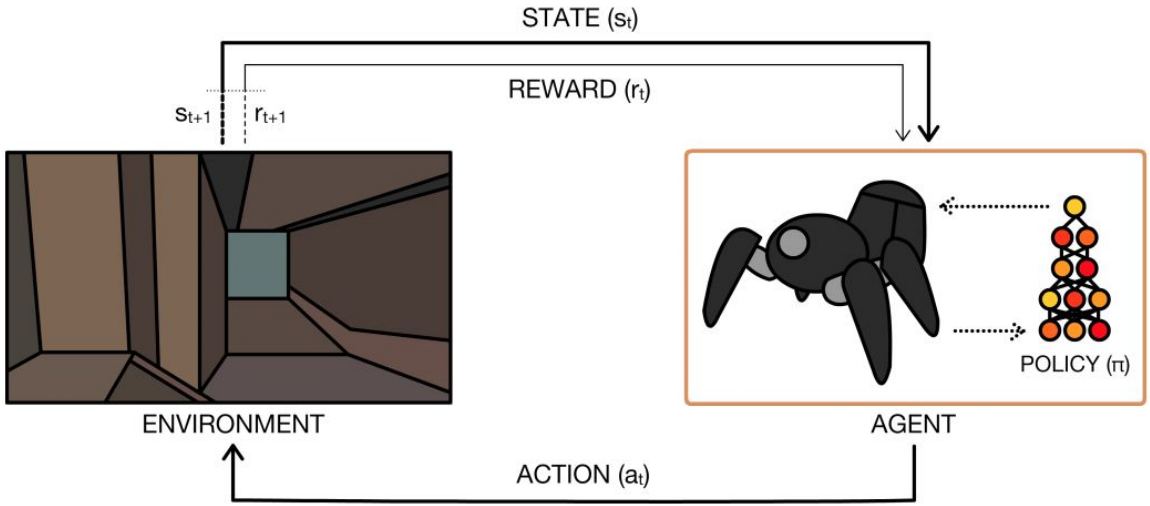
Example loss function: $$S = \sum_{i=0}^{n} (y_i - h(x_i))^2$$

adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

Deep reinforcement learning

Deep reinforcement learning (DRL)

Combination of reinforcement learning and deep learning!
We introduce the concept using **Deepmind's Deep Q-Network** (DQN).



A Brief Survey of Deep Reinforcement Learning, [arXiv:1708.05866]

Why combine RL with DL

Traditional RL agents perform well in a variety of specific domains (robot soccer, games, …) but it has **drawbacks**:

- Feature extraction (by hand)
- Not scalable (curse of dimensionality)

Cannot apply the same RL model to a variety of video games

Solution:
- Neural network for feature extraction
- + Neural network for policy function **π**!

RL vs DRL - Differences

Differences between standard reinforcement learning and DRL:

- Representation of states
- Policy function π
- Stored data

RL vs DRL - Differences

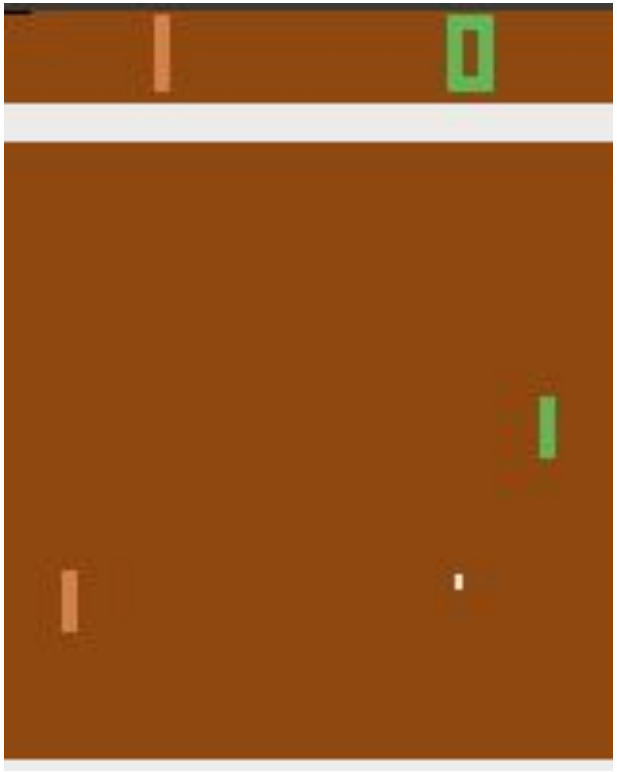
Differences between standard reinforcement learning and DRL:

- **Representation of states**
- Policy function **π**
- Stored data

RL vs DRL - Representation of states

RL:
Handcrafted features:

Which features would you use for this?



RL vs DRL - Representation of states

RL:
Handcrafted features:

Which features would you use for this?

https://github.com/aleju/mario-ai

RL vs DRL - Representation of states

RL:
Many handcrafted features, e.g.:

- Pong: location of player and ball
- Mario: location of Mario, map? enemies? power up locations? secret tubes?

Tons of time for even simple games.

DRL:
Want something more universal, something that every game uses!

Pixels!

DRL - Representation of state **s**

Not just raw pixels as input

Preprocessing:

1. Frames are represented as RGB arrays
   Luminance Y from RGB: $$Y = 0.2126R + 0.7152G + 0.0722B$$
2. Resize Y to an 84 x 84 matrix

State **s** is the result of these transformations!

RL vs DRL - Differences

Differences between standard reinforcement learning and DRL:

- Representation of states
- **Policy function π**
- Stored data

RL vs DRL - Policy function π

RL:

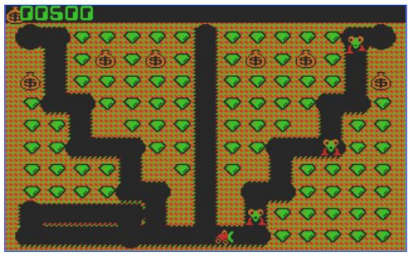
if **π** = determined by a table of (state, action) pairs

=> not scalable (memory & sample complexity)

RL vs DRL - Policy function π

Number of State-Action Pairs

- Own position: 150 squares
- At most 5 Monsters (?) : 150^5
- Emeralds & Bags
  of Gold : 3^{150}
- Tunnel structure: 2^{2*150}
- Fireball, bonus, ...



Multiply and take one for each of 8 actions

KATHOLIEKE UNIVERSITEIT LEUVEN

43

DTAI

If using images for state:

Image = 210x160 8-bit RGB

$|S|$
$= 256^{210*160*3}$
$\approx 4 * 10^{242750}$

RL vs DRL - Policy function **π**

RL:

if **π** = determined by a table of (state, action) pairs

=> not scalable (memory & sample complexity)

if **π** = neural network

=> also not scalable for our goal (feature extraction for each game)

DRL:

π = deep neural network for **feature extraction and policy function**

=> can automatically find compact low-dimensional representation

RL vs DRL - Differences

Differences between standard reinforcement learning and DRL:

- Representation of states
- Policy function **π**
- **Stored data**

RL vs DRL - Stored data

RL:
A table entry for every possible (state, action) pair or the weights of a NN

DRL:
Store 4-tuples in a list:

- Current state **s**
- Action **a** taken in **s**
- Reward **r** for taking action **a**
- Next state **s'**

Such a 4-tuple is called a **transition**.
Stored inside a **replay memory** that can hold N transitions.

DRL - Deep Q-Network

Plug in Neural network and problem solved?

NO!

Reinforcement learning **caveats when using neural networks**:

- unstable
- can diverge when a nonlinear function approximator such as a neural network is used to represent the policy **π**

DRL - Instability

Causes of instability:

- **Correlations** between sequences of frames
  If the player moves in one direction on a certain frame, it will often make the same move in the next frame

- **Feedback loop** during training
  Every step of training, the network's output shifts
  If using a constantly shifting set of output to adjust network weights, then the output can spiral out of control

DRL - Reducing instability

Solutions for instability:

- **Correlations** between sequences of frames
  - sample transitions uniformly from **replay memory**

- **Feedback loop** during training
  - **target network** = copy of main network
  - use target network in loss function
  - allow network updates to accumulate
  - synchronize target network periodically

DRL - Preprocessing step

Some more preprocessing before passing to the network:

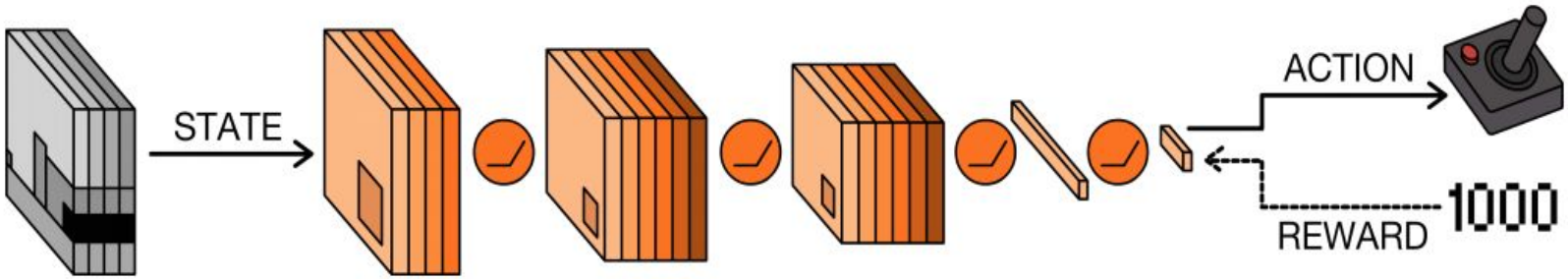
- Stack **4 consecutive states** into an 84 x 84 x 4 array
  This is **1 sample**

So… What does the neural network look like?

DRL - Deep Q-Network Architecture

Actually the architecture doesn’t matter that much here.

This approach works for small and big networks!



A Brief Survey of Deep Reinforcement Learning, [arXiv:1708.05866]

DRL - DQN Code (Summary)

Algorithm 1: deep Q-learning with experience replay.

replay memory → Initialize replay memory D to capacity N

policy → Initialize action-value function Q with random weights θ

target network → Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

For episode $= 1, M$ **do**

Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$

For $t = 1$,T **do**

With probability ε select a random action a_t

otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$

Execute action a_t in emulator and observe reward r_t and image x_{t+1}

Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$ ← preprocess image

store in replay memory → Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

fetch from replay memory → Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$

Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ

synchronise target network → Every C steps reset $\hat{Q} = Q$

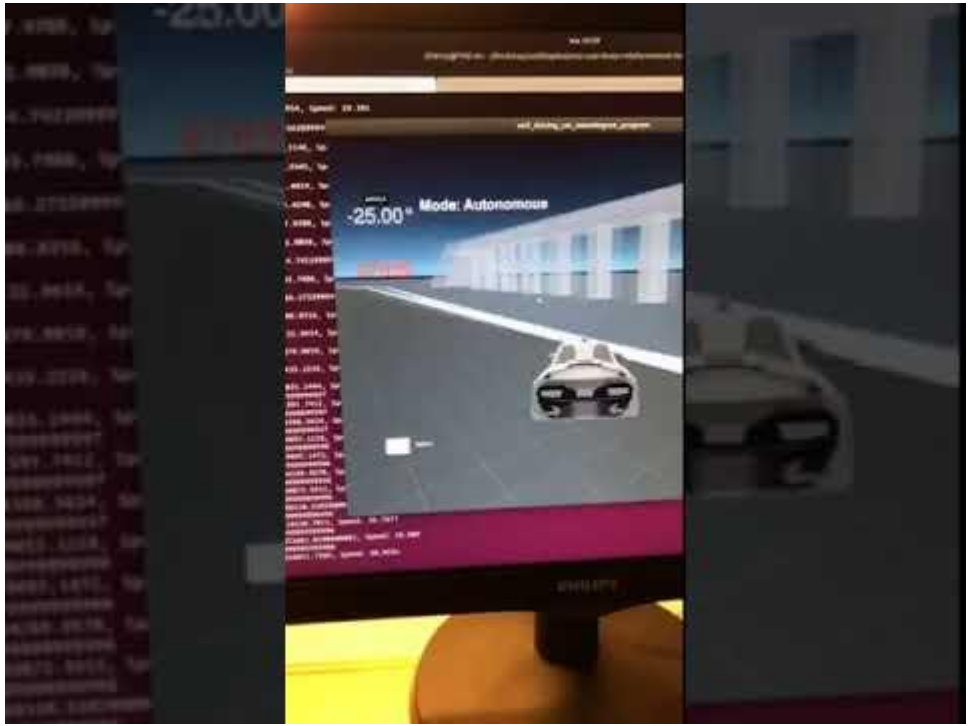
End For

End For

Human-level control through deep reinforcement learning, https://www.nature.com/articles/nature14236

Demo!

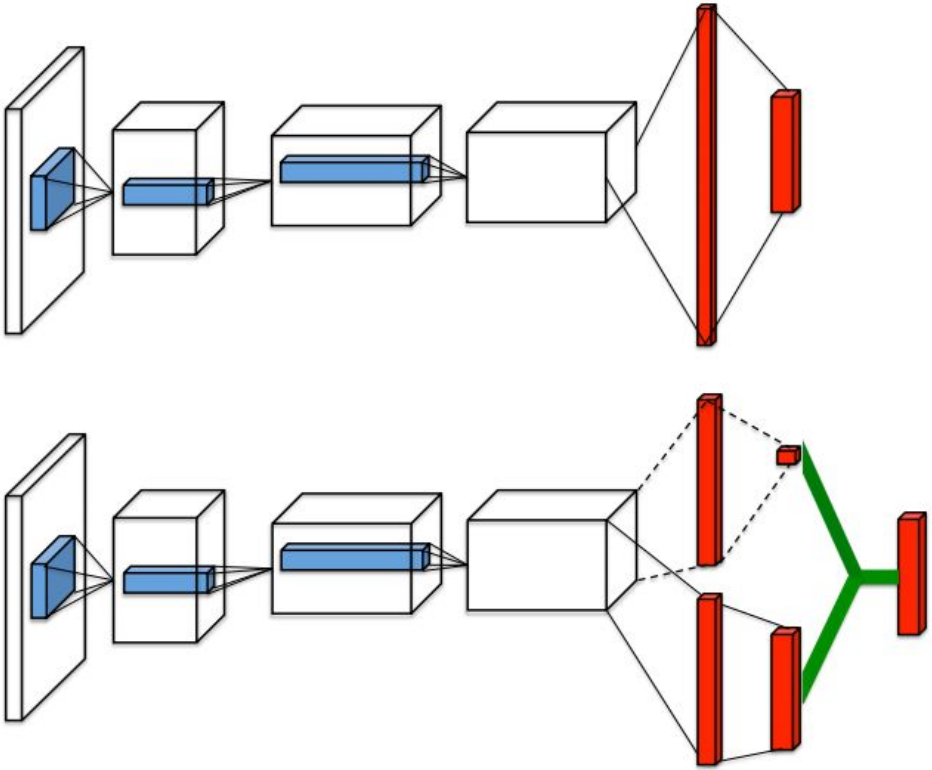
Reward function

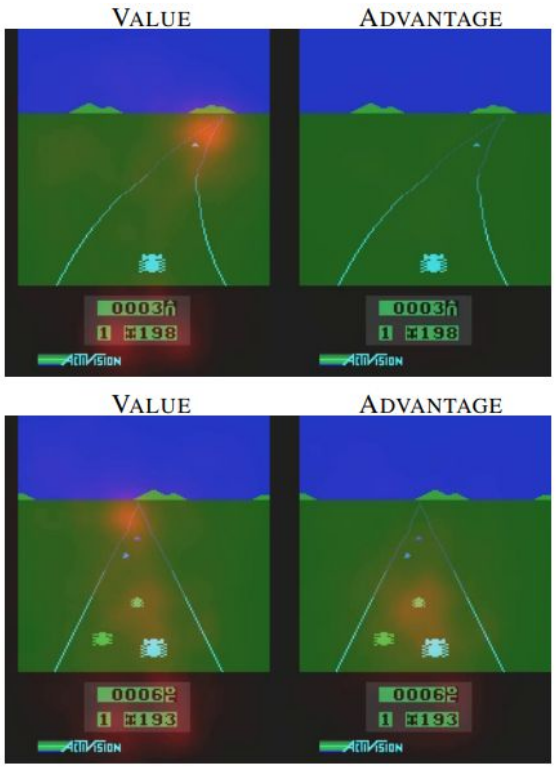


DRL - More methods

- Dueling Network Architectures
- Deep Recurrent Q-Learning (DRQN)

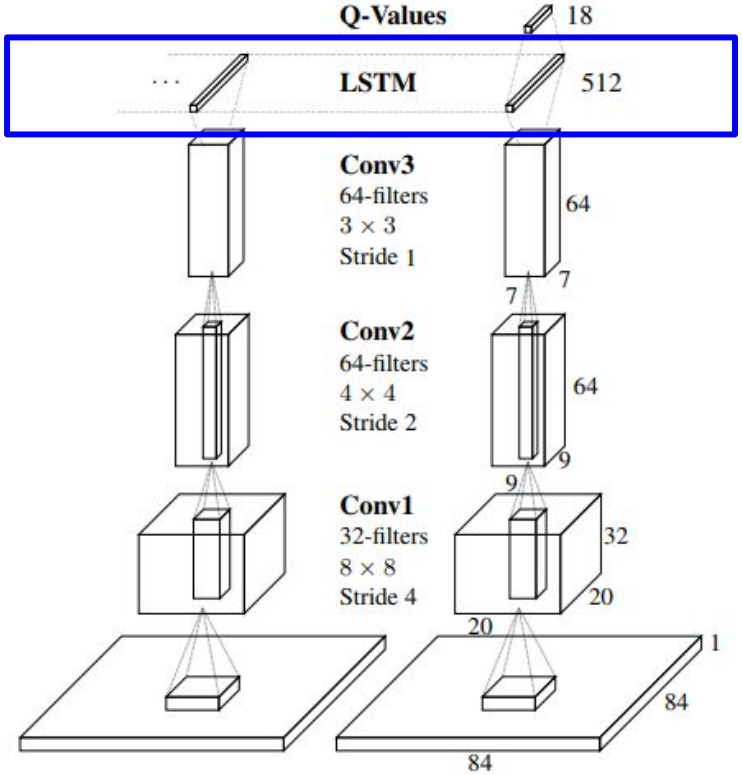
DRL - Dueling Network Architectures





Dueling Network Architectures for Deep Reinforcement Learning, [arXiv:1511.06581]

DRL - Deep Recurrent Q-Learning (DRQN)



Deep Recurrent Q-Learning for Partially Observable MDPs, [arXiv:1507.06527]

DRL - More methods

Many more:

- Deep Attention Recurrent Q-Network (DARQN)
- Asynchronous advantage actor-critic (A3C)
- Generative Adversarial Imitation Learning (GAIL)
- Normalized Advantage Function (NAF)
- ...

Key differences:

- Model based <--> Model free
- Gradient based <--> Gradient free

What may come after DRL

What may come after DRL

How does a human learn? Do we have a reward function?

NO

We are curious and try to explore many different options.

Curiosity reinforcement learning

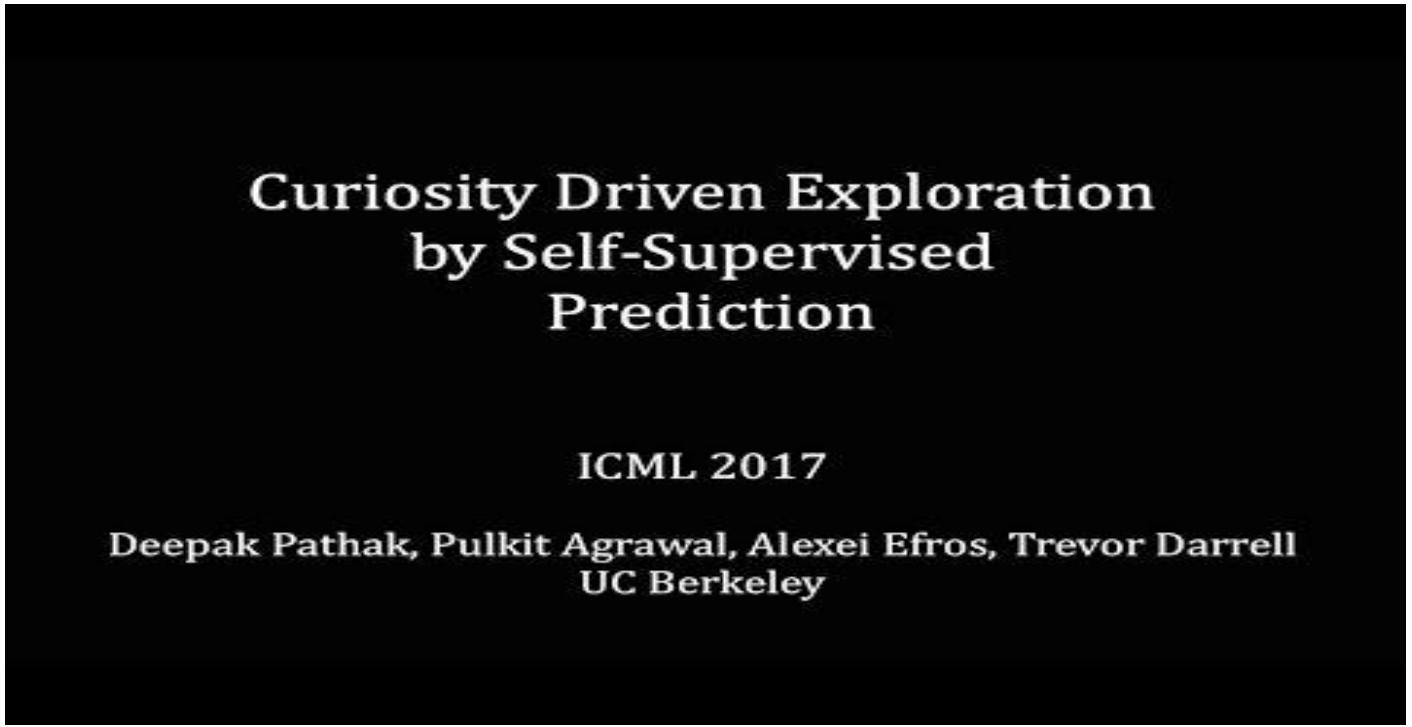
Curiosity reinforcement learning - How it’s done

Instead of us giving it a reward function, it creates its own reward function.

No need for rewards from the game anymore!

Curiosity-driven Exploration by Self-supervised Prediction, arXiv:1705.05363

Curiosity reinforcement learning - Example



References

- Richard Sutton and Andrew Barto, Reinforcement Learning: An Introduction (1st Edition, 1998)
- Human-level control through deep reinforcement learning, https://www.nature.com/articles/nature14236
- A Brief Survey of Deep Reinforcement Learning, arXiv:1708.05866
- Dueling Network Architectures for Deep Reinforcement Learning, arXiv:1511.06581
- Deep Recurrent Q-Learning for Partially Observable MDPs, **arXiv:1507.06527**
- Curiosity-driven Exploration by Self-supervised Prediction, arXiv:1705.05363
- Visualizing and Understanding Convolutional Networks, arXiv:1311.2901
- Convolutional Neural Networks (CNNs / ConvNets), cs231n.github.io/convolutional-networks/
- A Beginner's Guide To Understanding Convolutional Neural Networks, adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/
- Resource Management with Deep Reinforcement Learning, people.csail.mit.edu/hongzi/content/publications/DeepRM-HotNets16.pdf