Human-level control through deep reinforcement learning

Thierry Deruyttere & Armin Halilovic

Overview

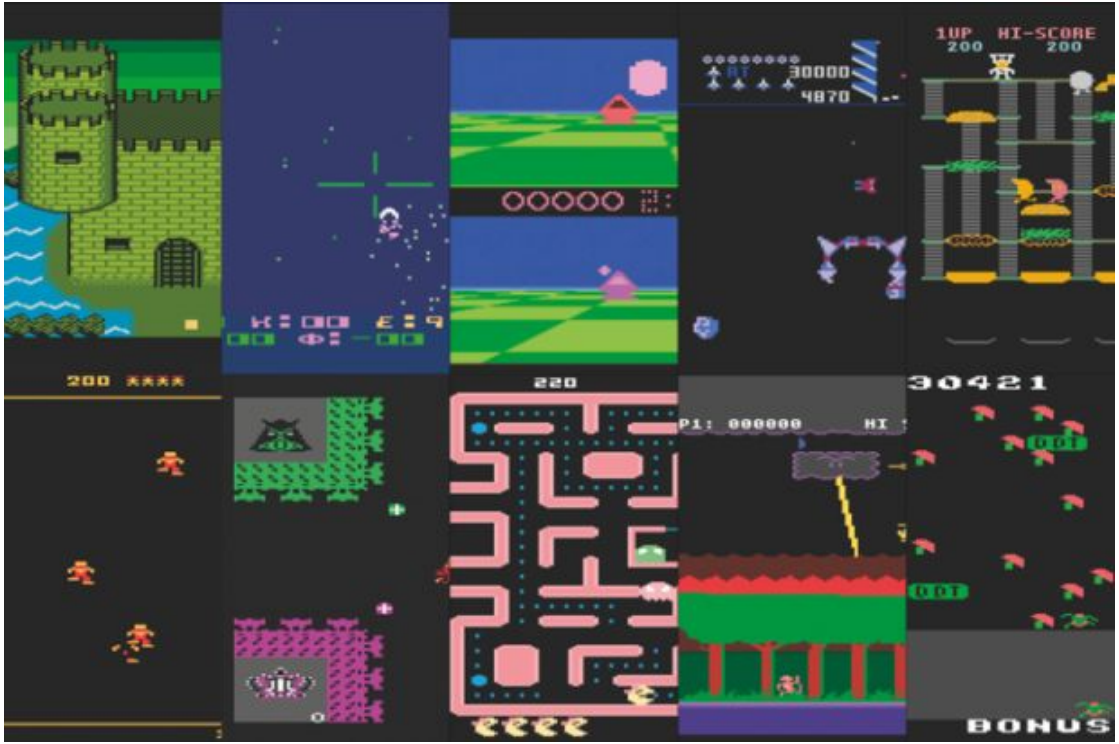
1. Introduction
2. Reinforcement learning
3. Deep learning
4. Deep reinforcement learning
5. Demo!
6. What may be next

Introduction - Problem statement

We want to create an agent that can:

- learn to solve a problem
- 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

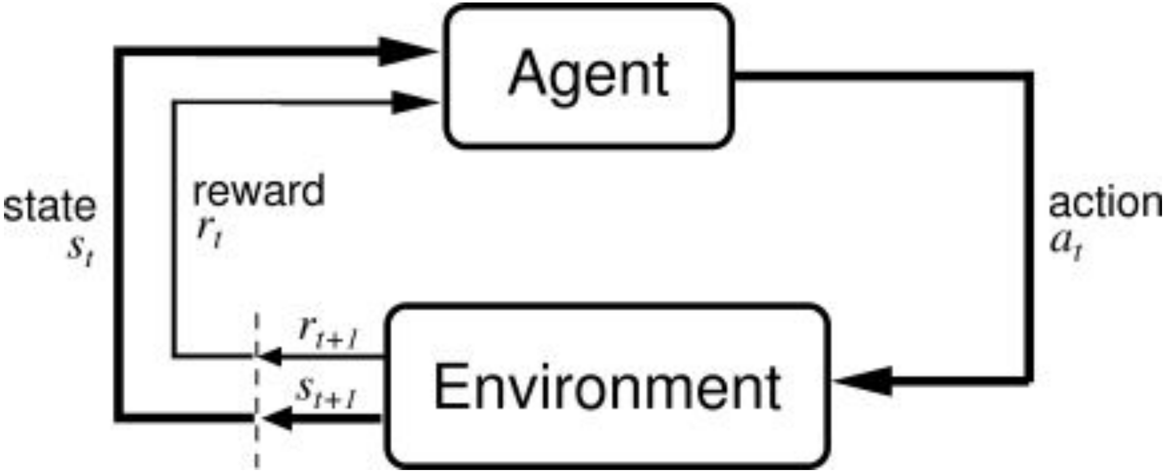
We want to create an agent that:

- 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

Model-free method to compute 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)

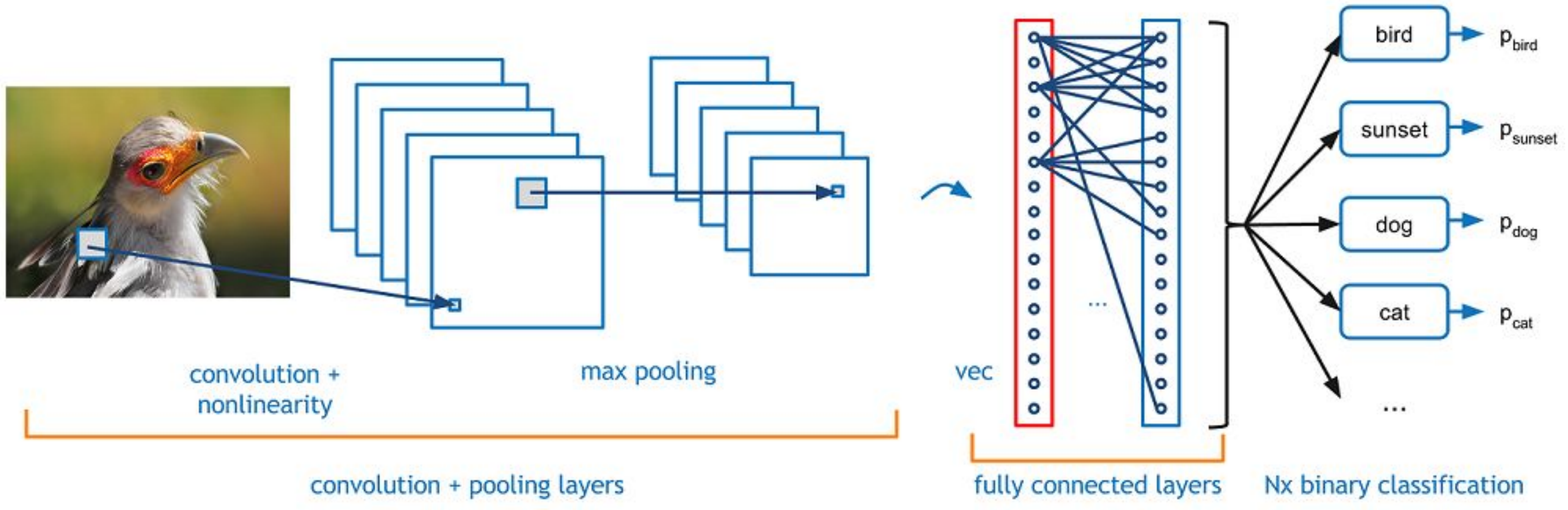
wikipedia.org/wiki/Q-learning

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

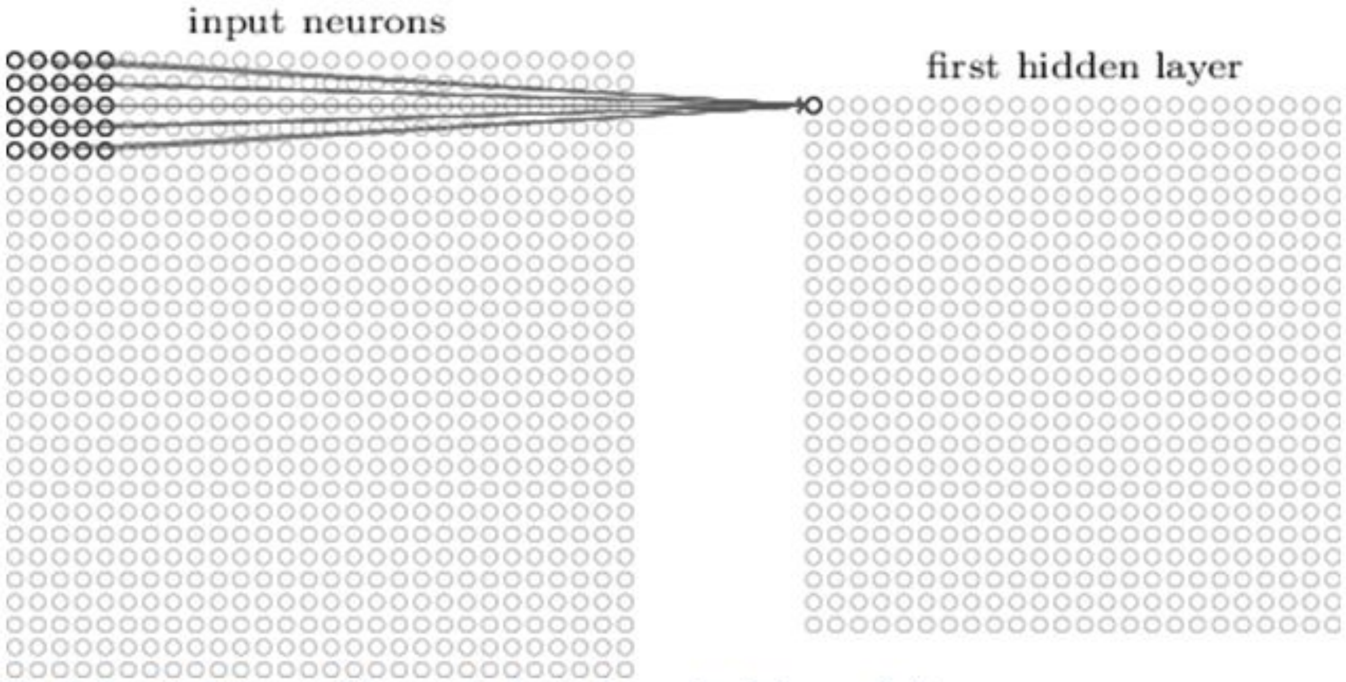
CNN - How it’s done

What We See

What Computers See

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

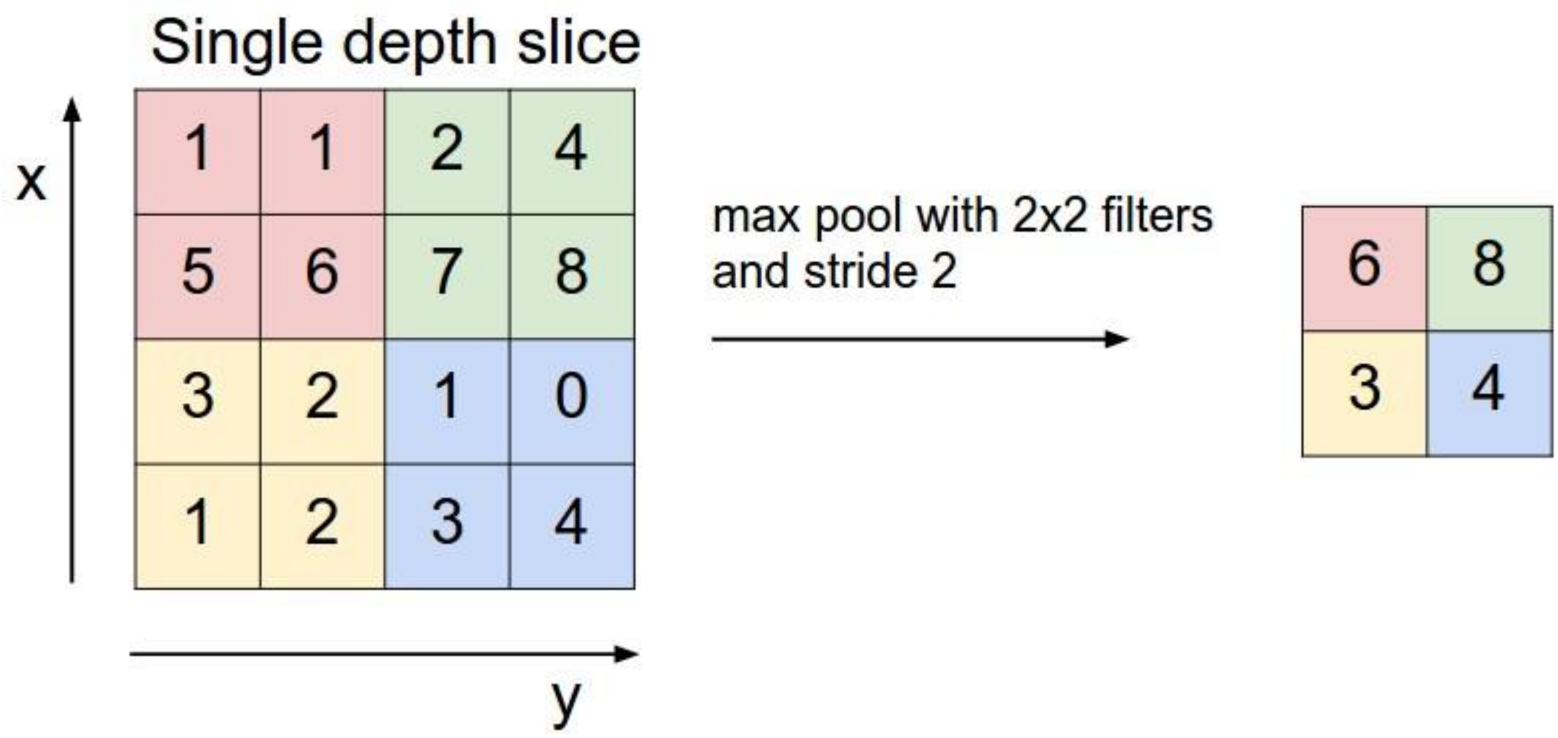
CNN - Convolution



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

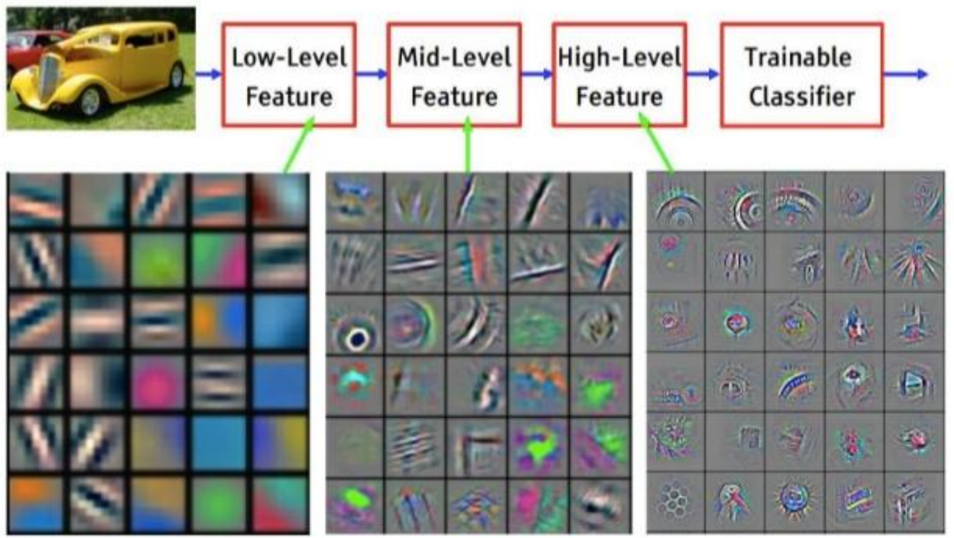
http://neuralnetworksanddeeplearning.com/images/tikz44.png

CNN - Max pooling



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

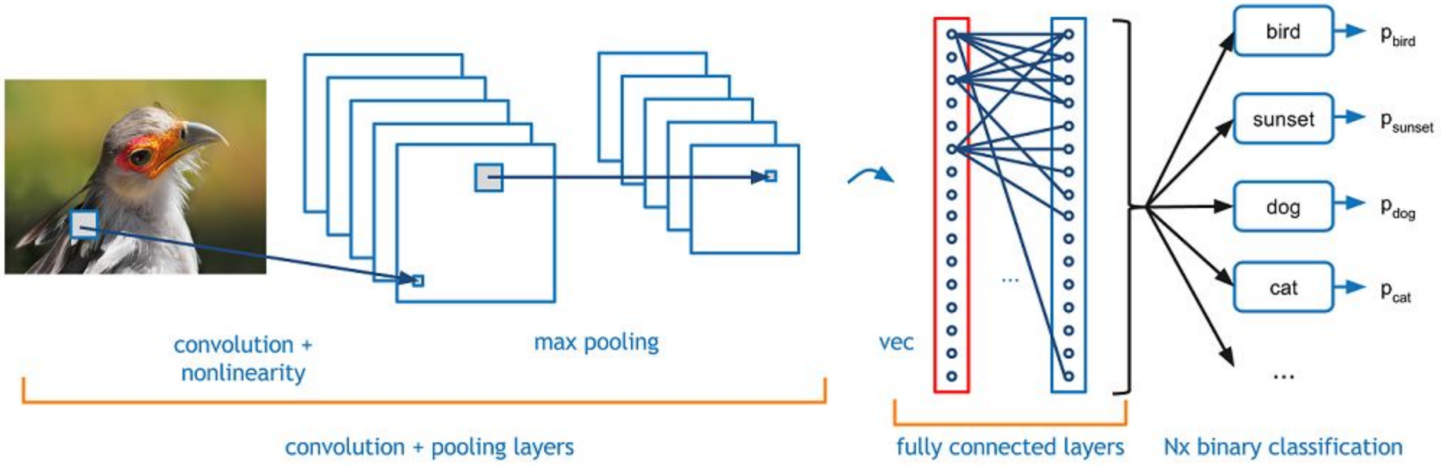
CNN - What it sees



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

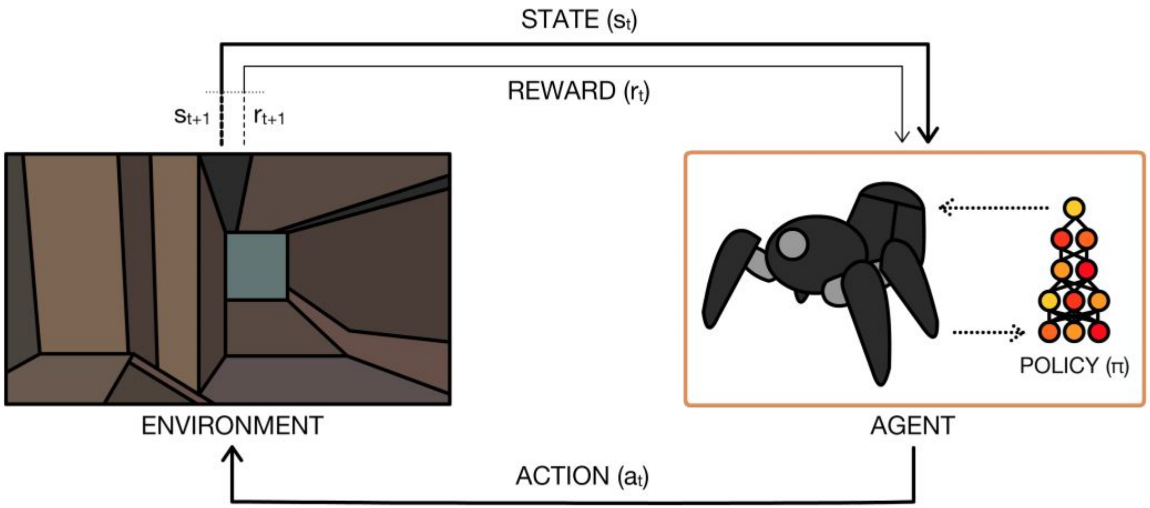
adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks

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

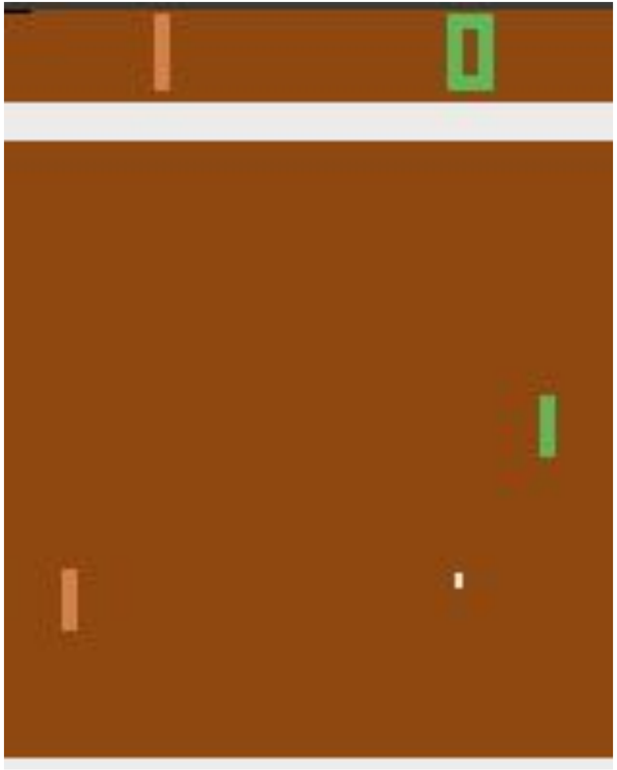
Differences between standard reinforcement learning and DRL:

- **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)

if **π** = neural network

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

DRL:

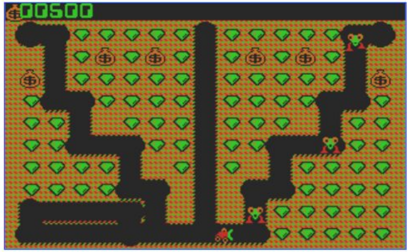
π = deep neural network for **feature extraction and policy function**

=> can automatically find compact low-dimensional representation

RL vs DRL - 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:

Image = 210x160 8-bit RGB

|S|

$= 256^{210*160*3}$

$\approx 4 * 10^{242750}$

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**
  Clipped to the interval [-1, 1] to make it work for different games.
- 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 $\boldsymbol{\pi}$

DRL - Instability

Causes of instability:

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

- **Feedback loop** when updating network
  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** when updating network
  - **target network** = copy of main network
  - use **target network** in loss function
  - allow network updates to accumulate
  - copy main network over target network periodically

DRL - Preprocessing step

Some more preprocessing before passing to the network:

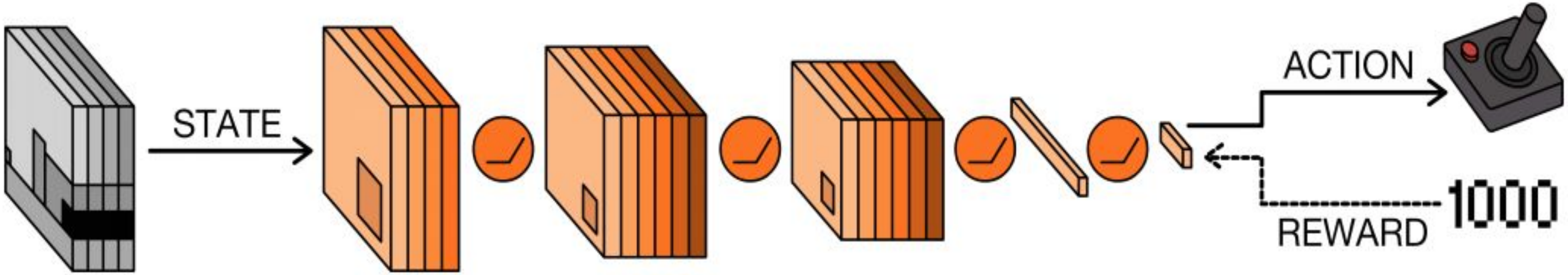
1. Stack **4 consecutive states** into an 84 x 84 x 4 array
   This is **1 sample**
2. Do this 32 times (batch size 32) for random samples
3. Feed this to the neural network

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 $\left(\phi_j, a_j, r_j, \phi_{j+1}\right)$ from D

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

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

synchronise with target network

    Every C steps reset $\hat{Q} = Q$

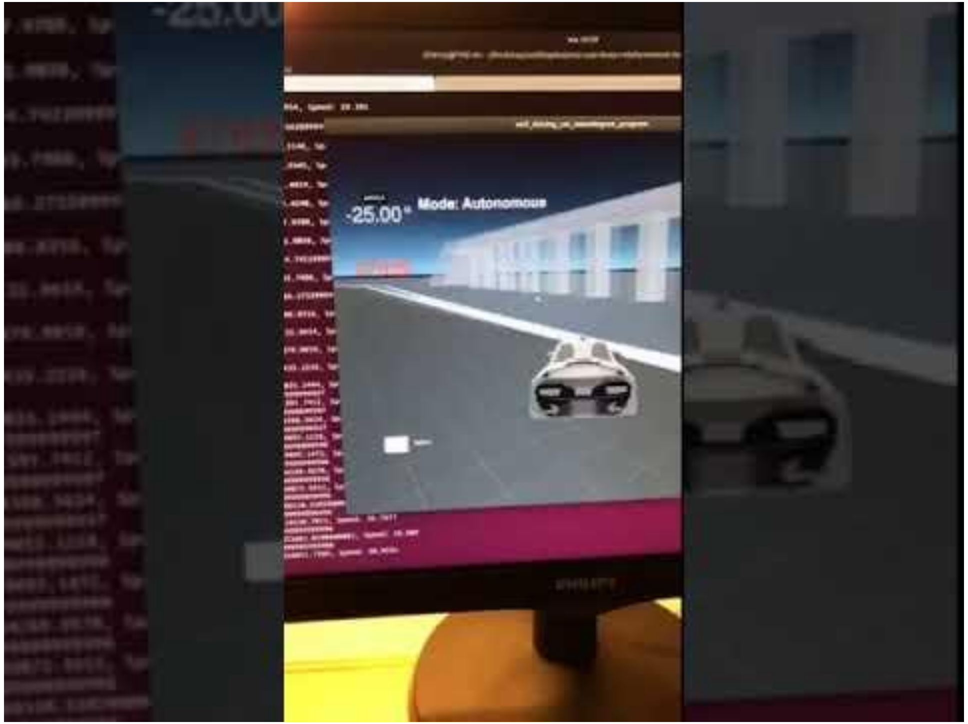
  End For

End For

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

Demo!

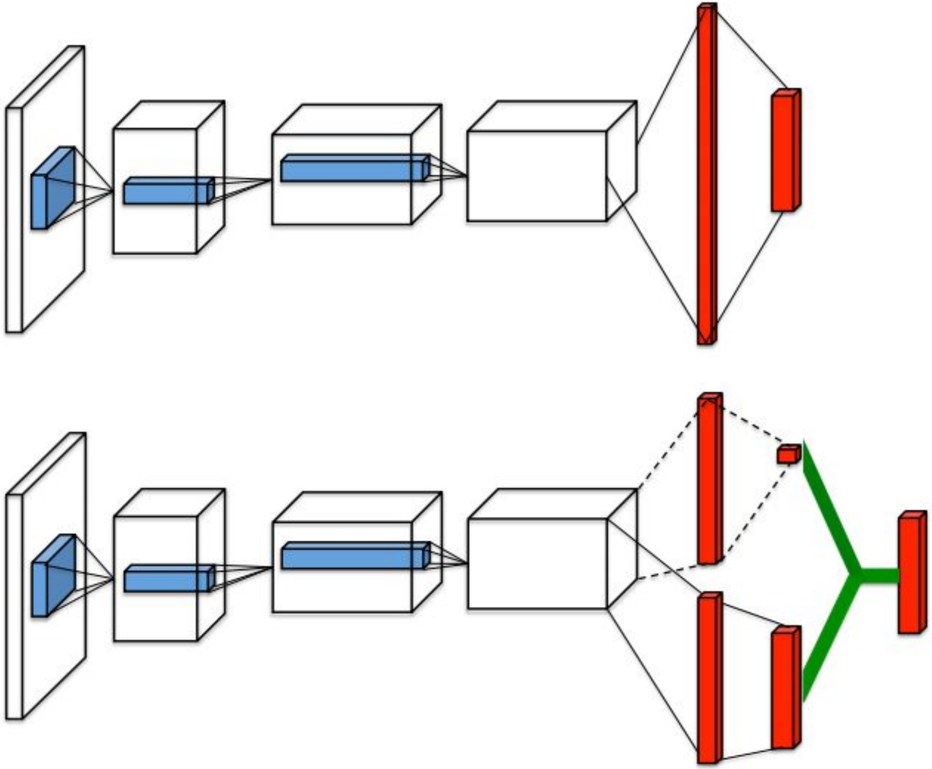
Reward function

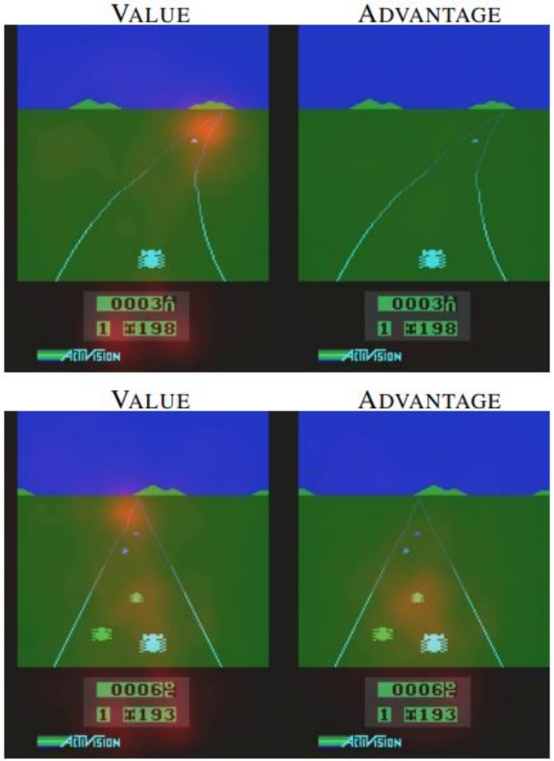


DRL - More methods

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

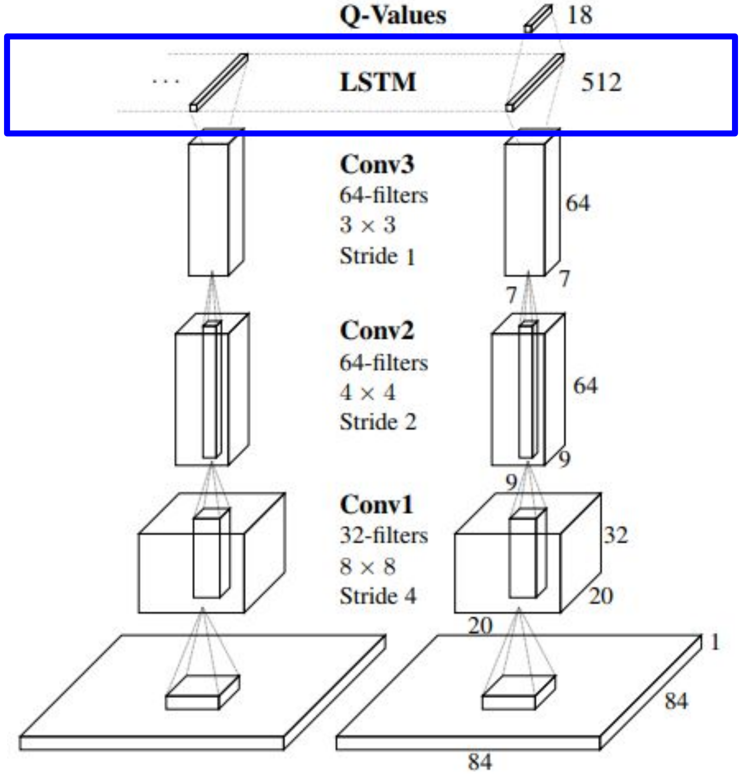
DRL - Dueling Network Architectures





Dueling Network Architectures for Deep Reinforcement Learning, [arXiv:1511.06581](https://arxiv.org/abs/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

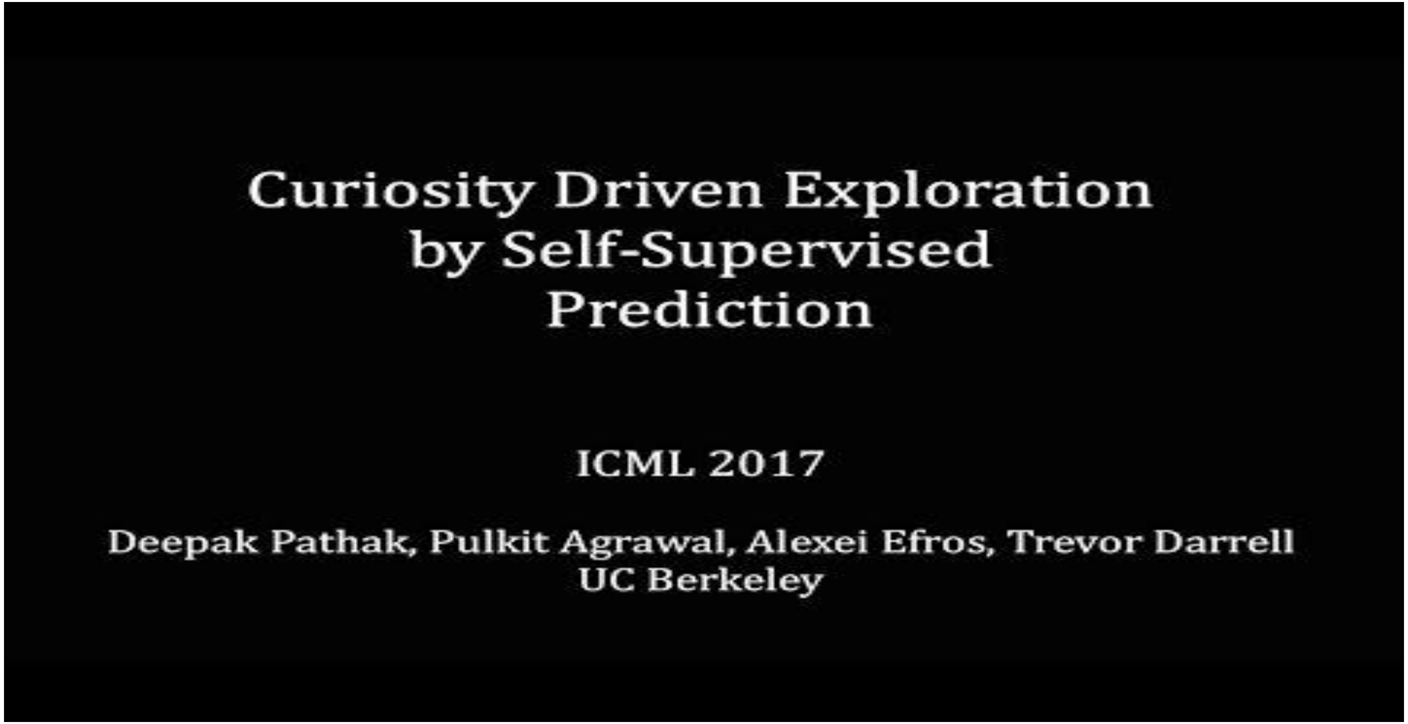
Curiosity reinforcement learning - How it's done

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