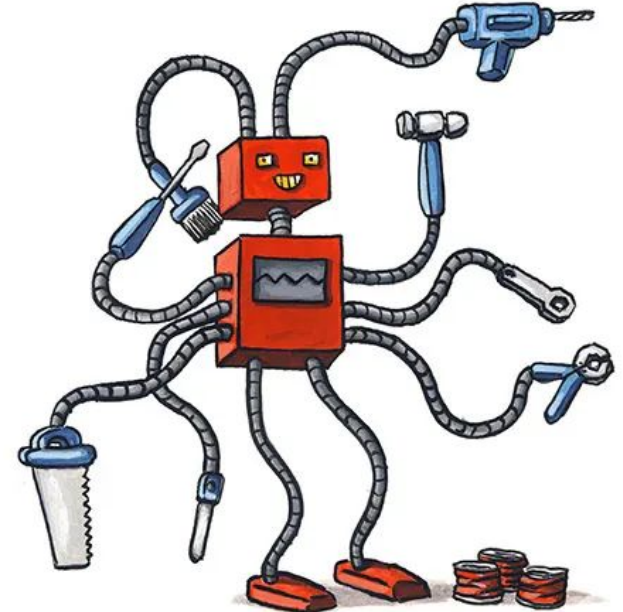


Playing ATARI Games with Deep Reinforcement Learning

Dr. Svetlin Penkov

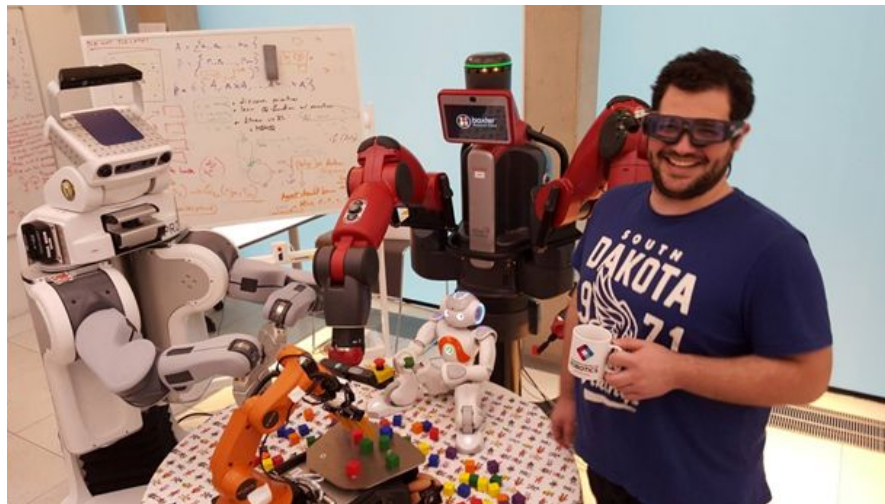
About me

Dreaming about building intelligent robots since the age of 6...



by [Becky Barnicoat](#)

About me



PhD in Robotics & AI



THE UNIVERSITY
of EDINBURGH

FIVE
AI

Research Scientist & Team Lead



About me

Robots should learn to program themselves...



- Team of world experts in AI and robotics
- Design, develop and deploy AI based solutions in **challenging domains**
- Research new **state-of-the-art AI methods**

Agenda

www.github.com/svepe/atari-dqn-workshop

10:30 - 11:20 Introduction to RL and OpenAI gym

Break

11:30 - 12:20 Introduction to deep neural networks and Chainer

Break

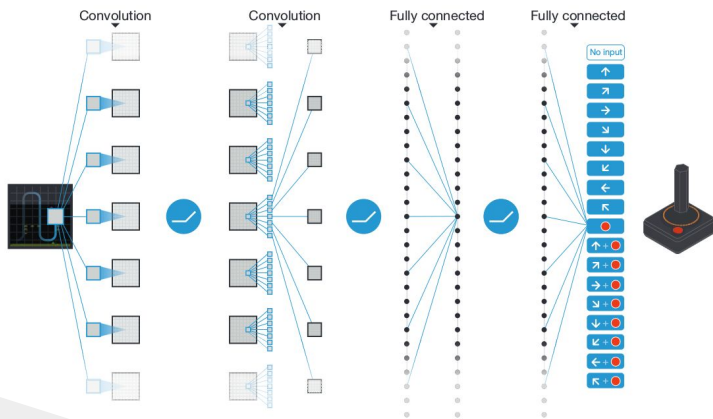
12:30 - 13:20 Implementing a Deep Q-learning Network (DQN)
agent to play ATARI games

Goal LETTER

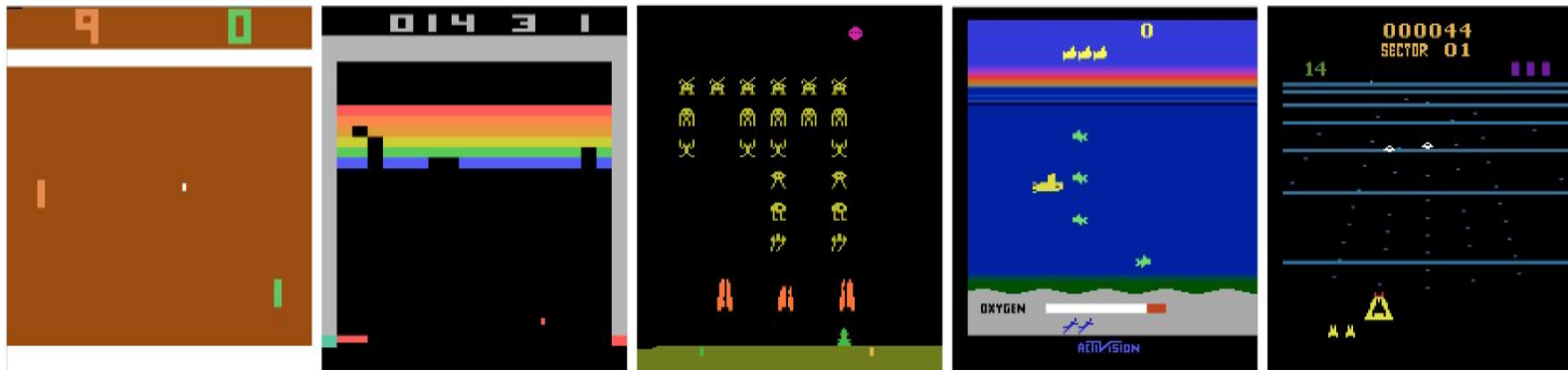
doi:10.1038/nature14236

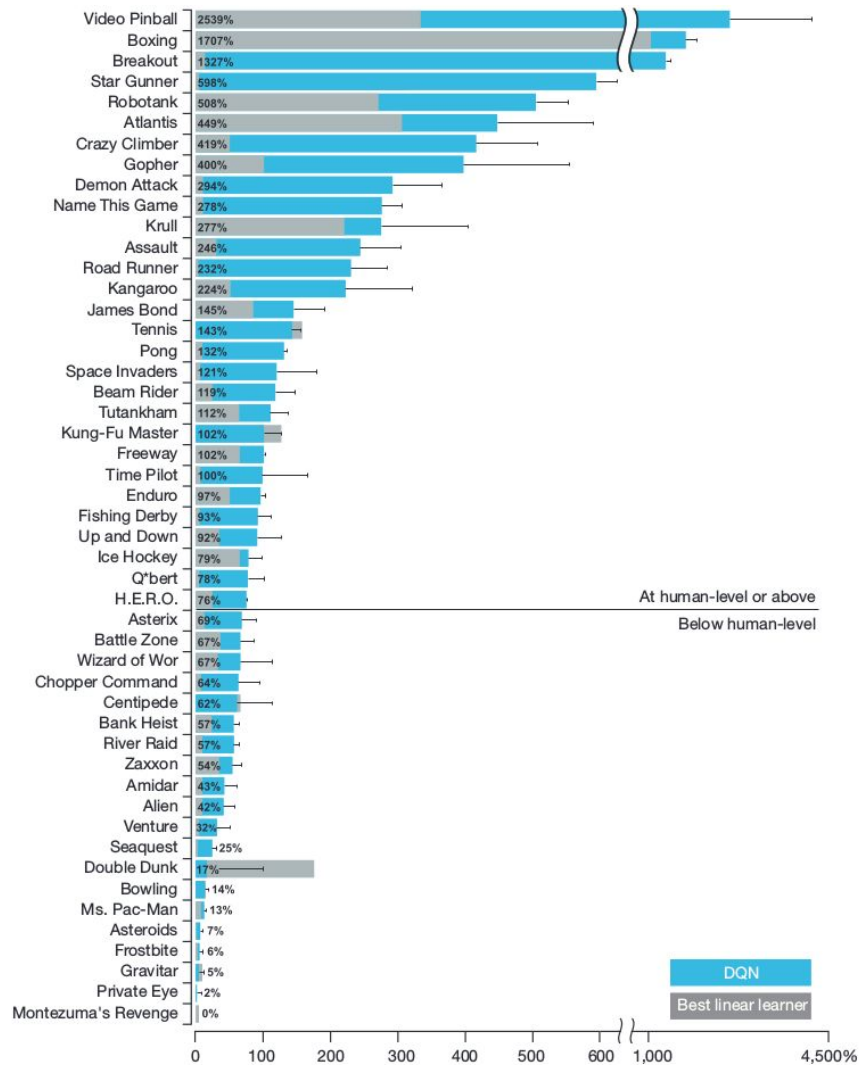
Human-level control through deep reinforcement learning

Volodymyr Mnih^{1*}, Koray Kavukcuoglu^{1*}, David Silver^{1*}, Andrei A. Rusu¹, Joel Veness¹, Marc G. Bellemare¹, Alex Graves¹, Martin Riedmiller¹, Andreas K. Fiedjeland¹, Georg Ostrovski¹, Stig Petersen¹, Charles Beattie¹, Amir Sadik¹, Ioannis Antonoglou¹, Helen King¹, Dharshan Kumaran¹, Daan Wierstra¹, Shane Legg¹ & Demis Hassabis¹

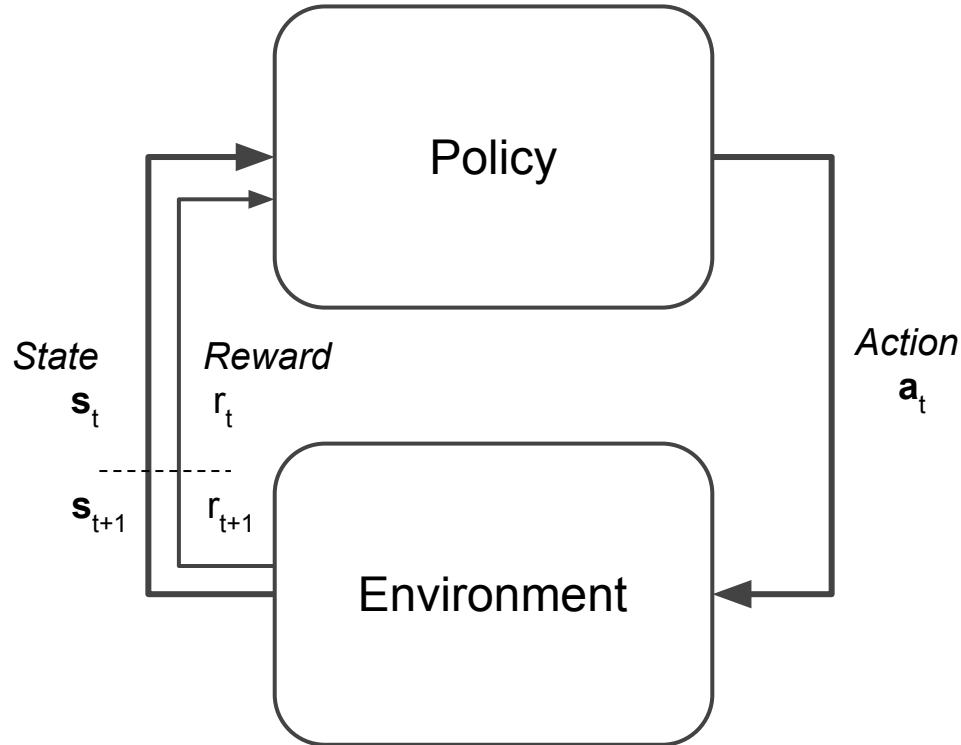


Playing ATARI Games

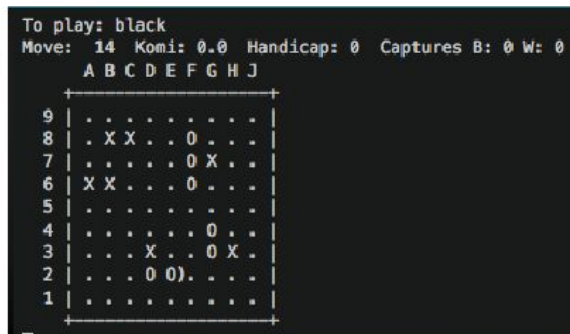
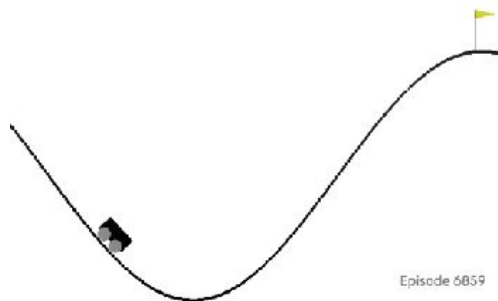




Reinforcement Learning 101



OpenAI Gym



Total length of input instance: 4, step: 3

Observation Tape : BBA

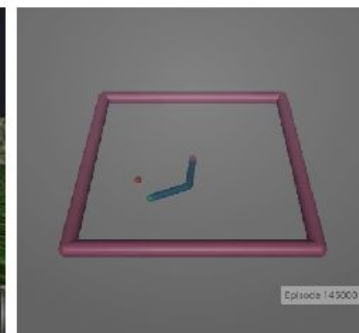
Output Tape : BA

Targets : BA

Current reward : 1.000

Cumulative reward : 2.010

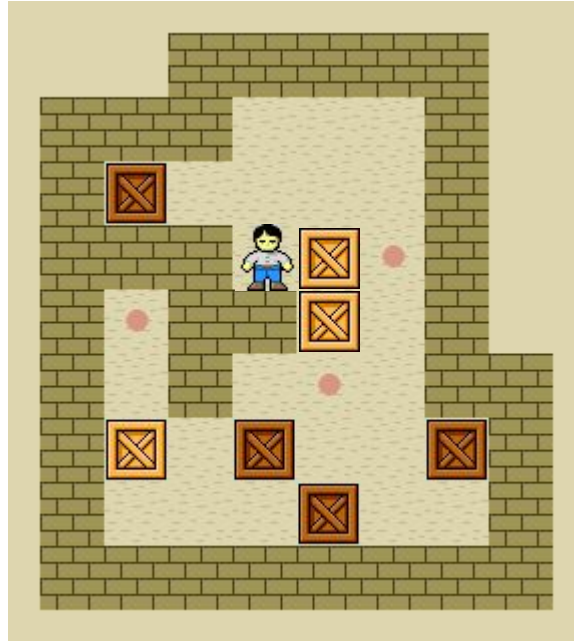
Action : Tuple(move over input
write to the ou
prediction: A)



OpenAI Gym

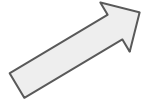


Sokoban Example



Episode 1

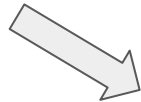
$\mathbf{s}_0 \in \mathbf{S}$



$\mathbf{a}_1 \in \{\text{move}_1, \text{move}_2, \text{move}_3, \text{move}_4\} = \mathbf{A}$

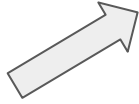
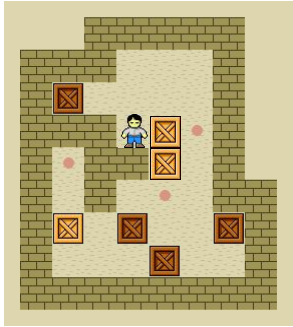


$\mathbf{a}_1 = ? \mid \mathbf{s}_0$

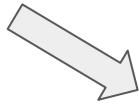
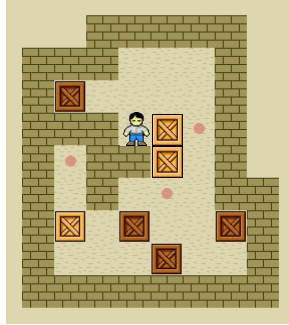
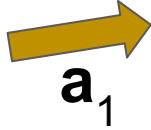


Episode 1

$s_0 \in \mathcal{S}$

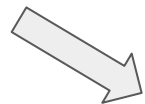
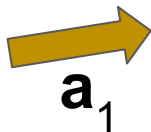
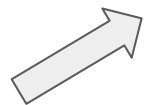
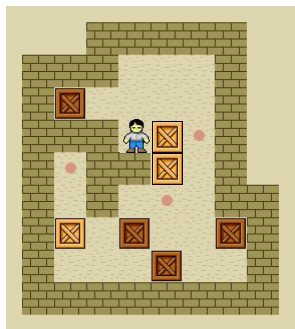


$s_1 \in \mathcal{S}, r_1 = 0$

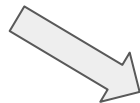
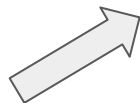
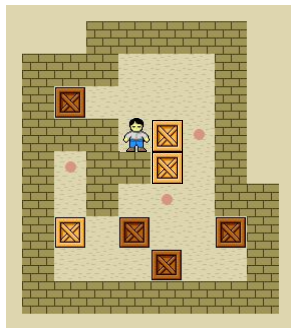


Episode 1

$s_0 \in \mathcal{S}$

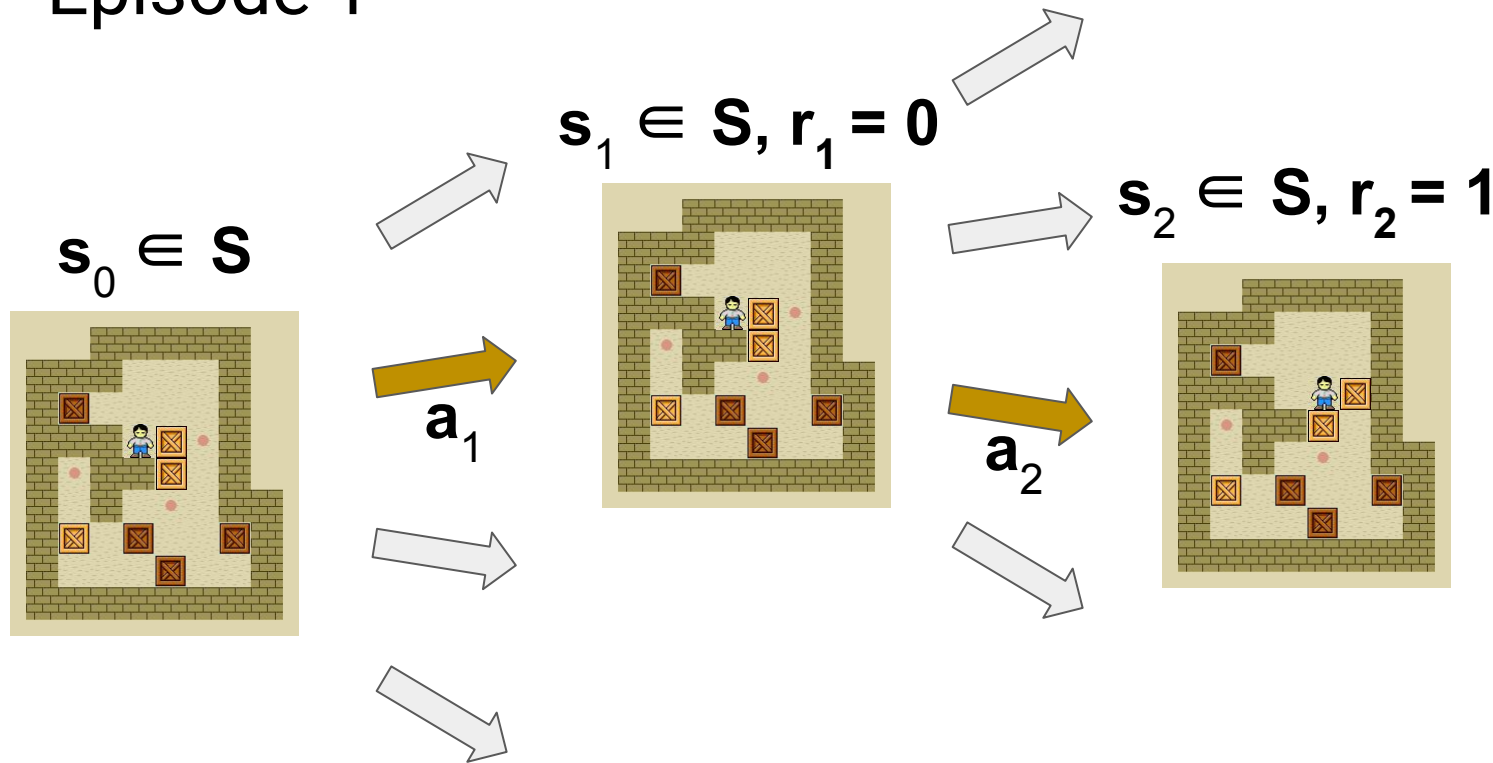


$s_1 \in \mathcal{S}, r_1 = 0$

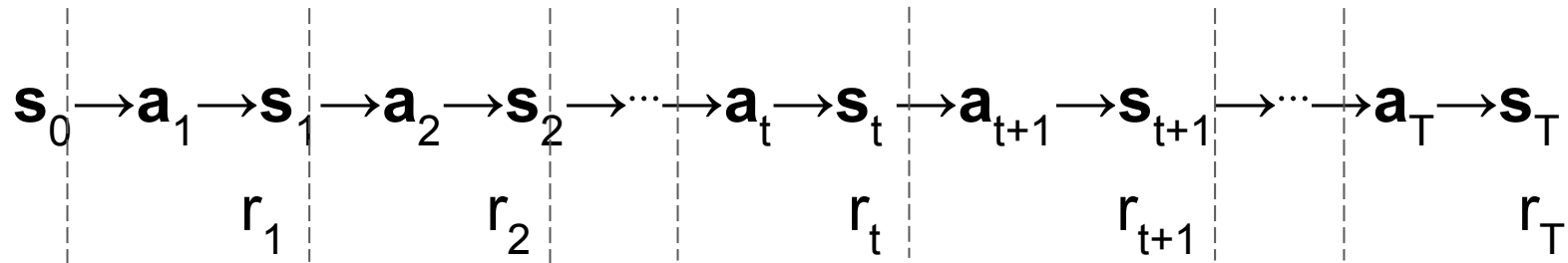


$a_2 = ? \mid s_1$

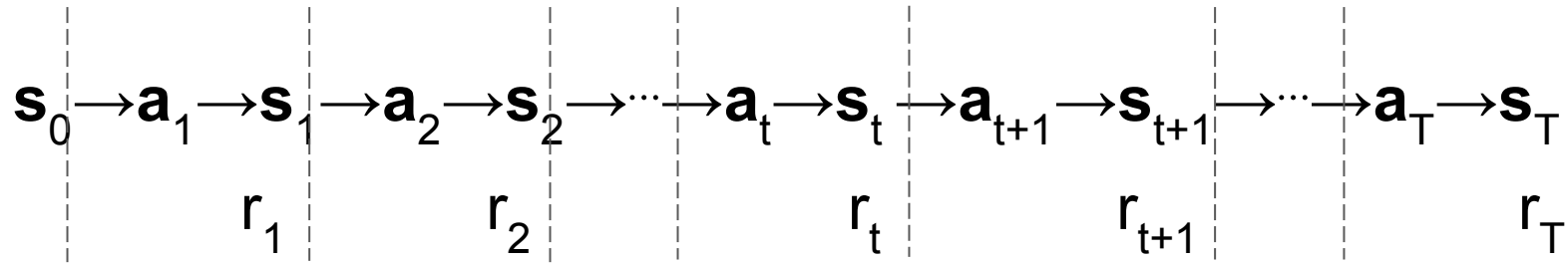
Episode 1



Episode Trace

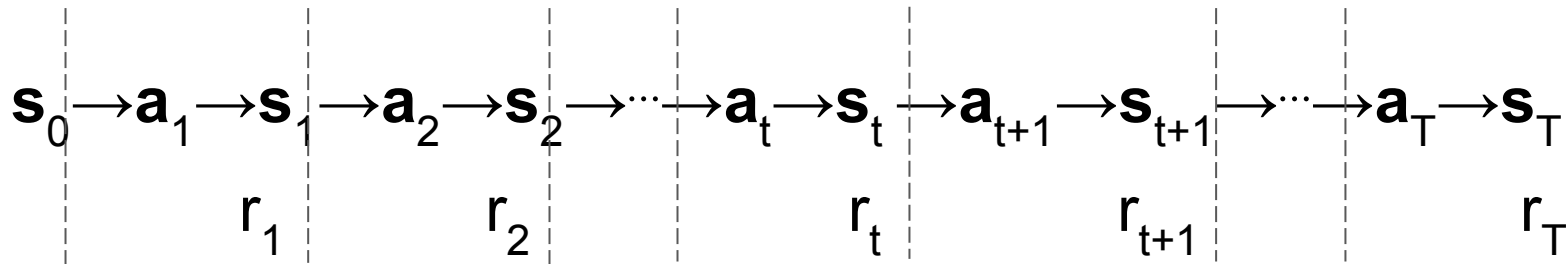


State Value



$$V(s_t) = r_t + r_{t+1} + r_{t+2} + \dots + r_T$$

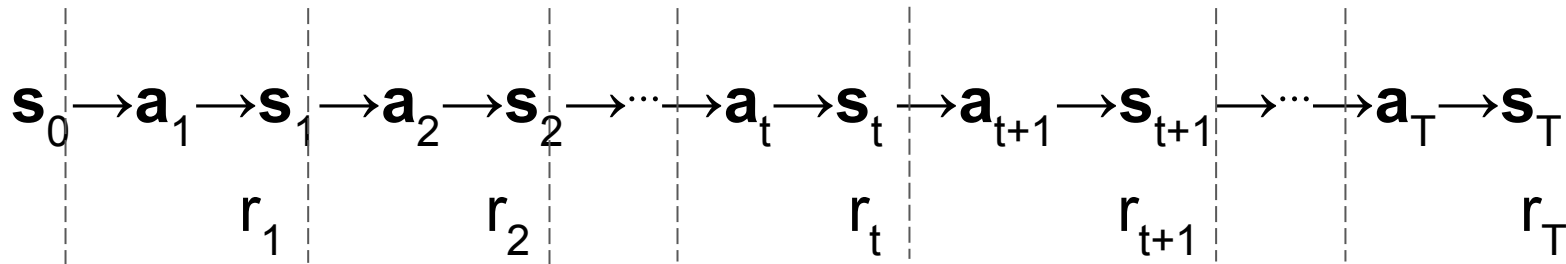
Discounted State Value



$$V(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \dots + \gamma^{T-t} r_T$$

$\gamma \in [0, 1)$ - discount factor

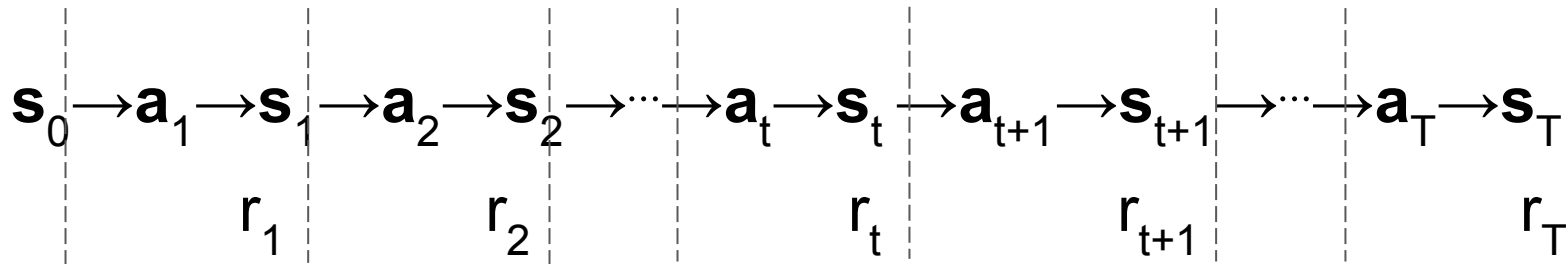
RL Objective



Choose $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T$ such that we maximise

$$\mathbf{V}(\mathbf{s}_1) = r_1 + \gamma r_2 + \gamma^2 r_3 \dots + \gamma^T r_T$$

Temporal Difference (TD-learning)

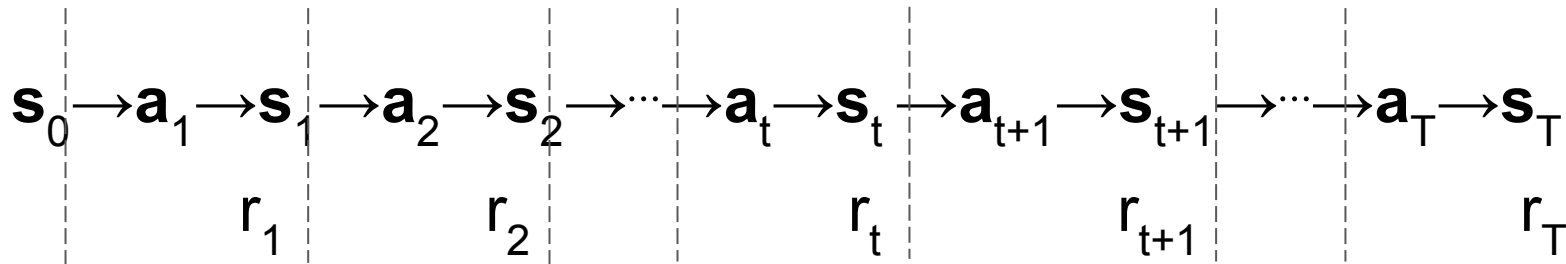


$$\mathbf{V}(\mathbf{s}_1) = r_1 + \gamma r_2 + \gamma^2 r_3 \dots + \gamma^T r_T$$

$$= r_1 + \gamma(r_2 + \gamma^1 r_3 \dots + \gamma^{T-1} r_T)$$

$$= r_1 + \gamma \mathbf{V}(\mathbf{s}_2)$$

Temporal Difference (TD-learning)

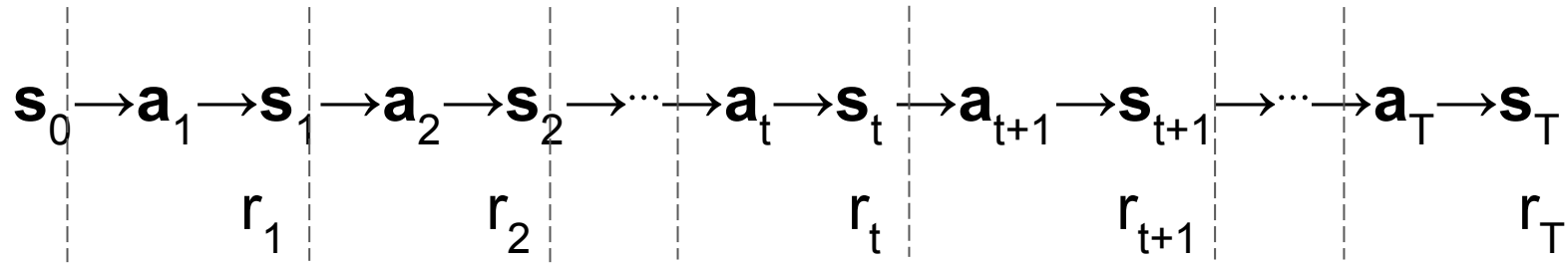


$$\mathbf{V}(\mathbf{s}_1) = r_1 + \gamma \mathbf{V}(\mathbf{s}_2)$$

\Downarrow

$$r_1 + \gamma \mathbf{V}(\mathbf{s}_2) - \mathbf{V}(\mathbf{s}_1) = 0$$

Temporal Difference (TD-learning)

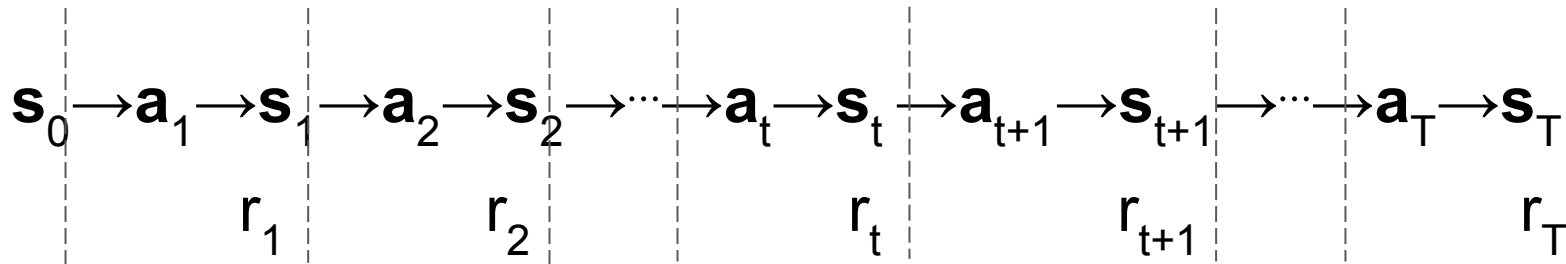


$$\mathbf{V}(\mathbf{s}_t) = r_t + \gamma \mathbf{V}(\mathbf{s}_{t+1})$$

\Downarrow

$$r_t + \gamma \mathbf{V}(\mathbf{s}_{t+1}) - \mathbf{V}(\mathbf{s}_t) = 0$$

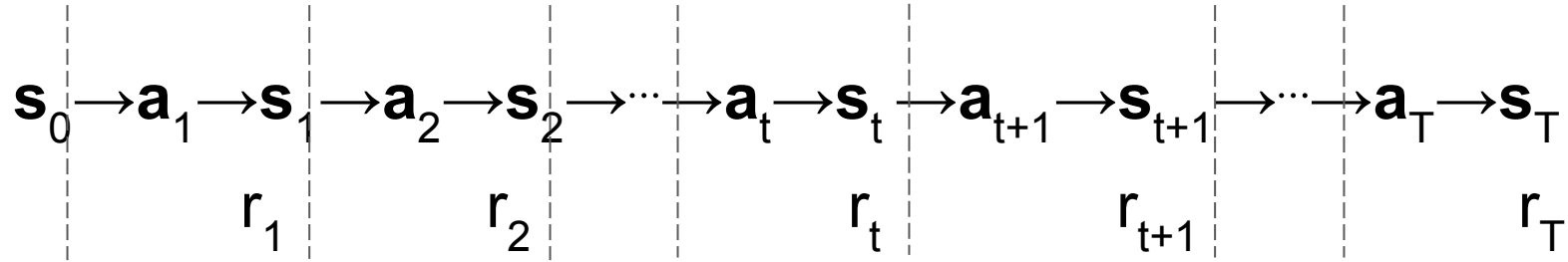
Temporal Difference (TD-learning)



$$r_t + \gamma \mathbf{V}(\mathbf{s}_{t+1}) - \mathbf{V}(\mathbf{s}_t) = 0 \text{ for any } (\mathbf{s}_t, r_t, \mathbf{s}_{t+1})$$

Choose $\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_T$ such that... but how?

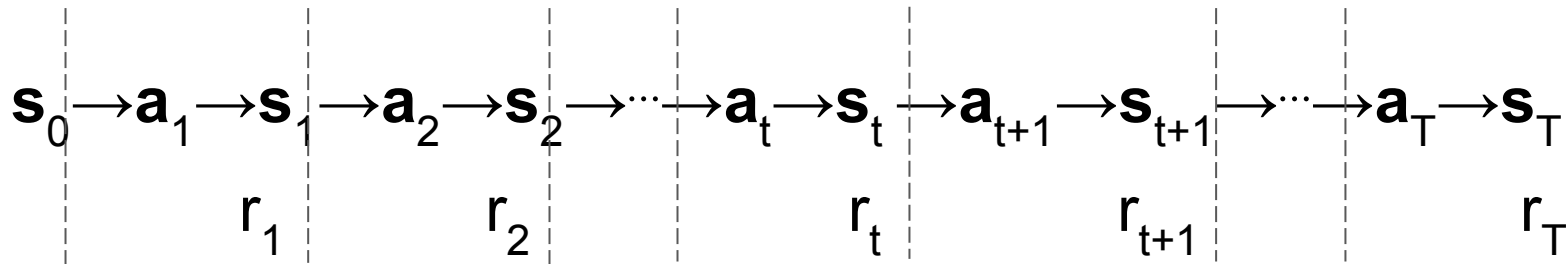
Action Value Function (Q-value)



$Q(s, a)$ - the value of choosing a in state s

$a^* = \operatorname{argmax}_a Q(s, a)$ - the best action in state s

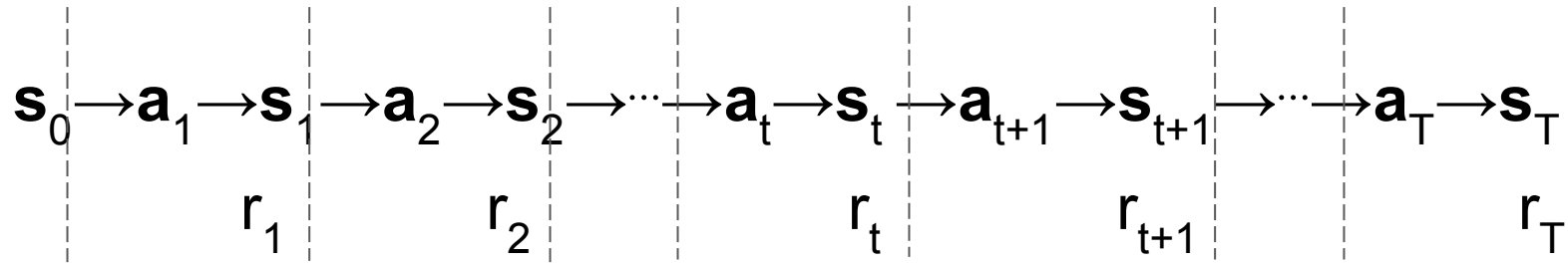
Action Value Function (Q-value)



$$\mathbf{V}(\mathbf{s}) = \max_a \mathbf{Q}(\mathbf{s}, \mathbf{a})$$

$$r_t + \gamma \mathbf{V}(\mathbf{s}_{t+1}) - \mathbf{V}(\mathbf{s}_t) = 0 \quad \text{for all } (\mathbf{s}_t, r_t, \mathbf{s}_{t+1})$$

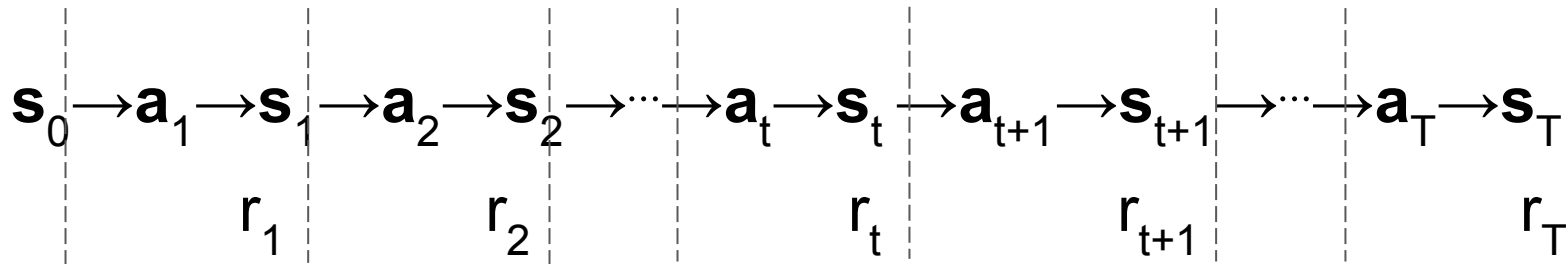
Action Value Function (Q-value)



$$V(s) = \max_a Q(s, a)$$

$$r_t + \gamma \max_a Q(s_{t+1}, a) - \max_a Q(s_t, a) = 0 \text{ for all } (s_t, r_t, s_{t+1})$$

Action Value Function (Q-value)



$$r_t + \gamma \max_a Q(s_{t+1}, a) - \boxed{\max_a Q(s_t, a)} = 0 \quad \text{for all } (s_t, r_t, s_{t+1})$$

Find the best action a_t for state s_t

Q-learning

$$r_t + \gamma \max_a \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}) - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) = 0 \text{ for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

Q-learning (annoying details)

$$r_t + \gamma \max_a Q(\mathbf{s}_{t+1}, \mathbf{a}) - Q(\mathbf{s}_t, \mathbf{a}_t) = 0 \quad \text{for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

If \mathbf{s}_t is terminal there is no \mathbf{s}_{t+1}

Q-learning

$$y_t - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) = 0 \text{ for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

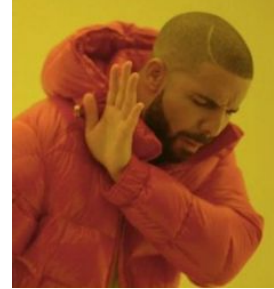
$$y_t = \begin{cases} r_t + \gamma \max_{\mathbf{a}} \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}) & \text{if } \mathbf{s}_t \text{ not terminal} \\ r_t & \text{otherwise} \end{cases}$$

Gathering Experience



What is the Q-value function?

$$Q: S \times A \rightarrow R$$



```
def Q(s: State, a: Action) -> float:
```

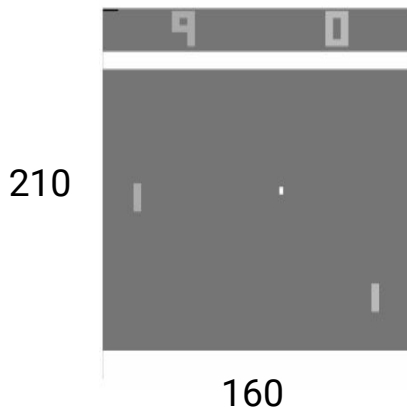
Tabular Q-learning

```
def Q(s: State, a: Action) -> float:
    return q[s, a]
```

.... for all $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$

Learning from Images

```
def Q(s: int, a: int) -> float:
    return q[s, a]
```



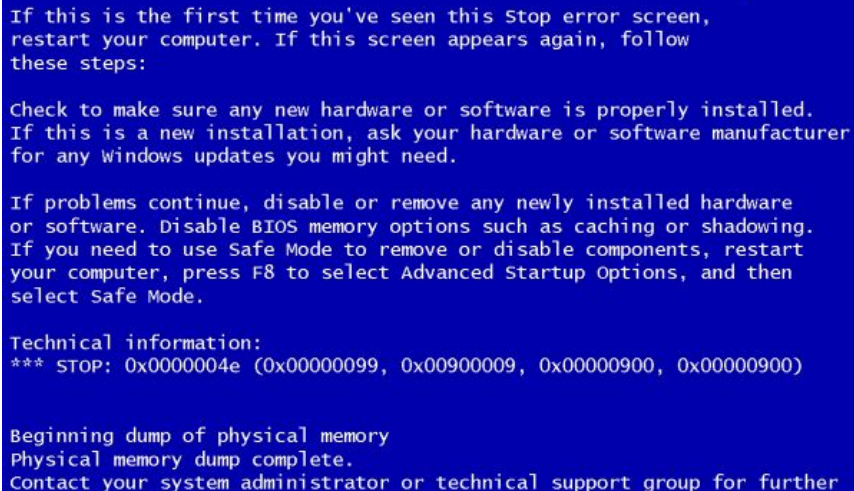
$255^{210 \times 160}$ states

left, right, up, down, fire, NOOP

6 actions

Learning from Images

```
q = np.zeros((255**33600, 6))
```



If this is the first time you've seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to make sure any new hardware or software is properly installed. If this is a new installation, ask your hardware or software manufacturer for any Windows updates you might need.

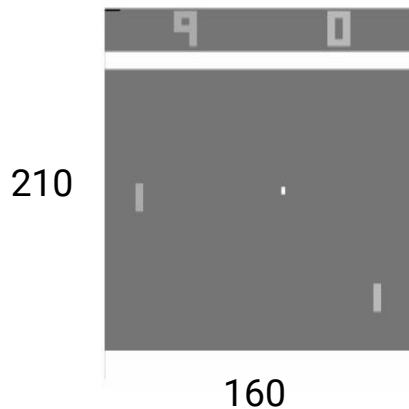
If problems continue, disable or remove any newly installed hardware or software. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Technical information:
*** STOP: 0x0000004e (0x00000099, 0x00900009, 0x00000900, 0x00000900)

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further

Learning from Images

```
def Q(s: int, a: int) -> float:
    ???
```



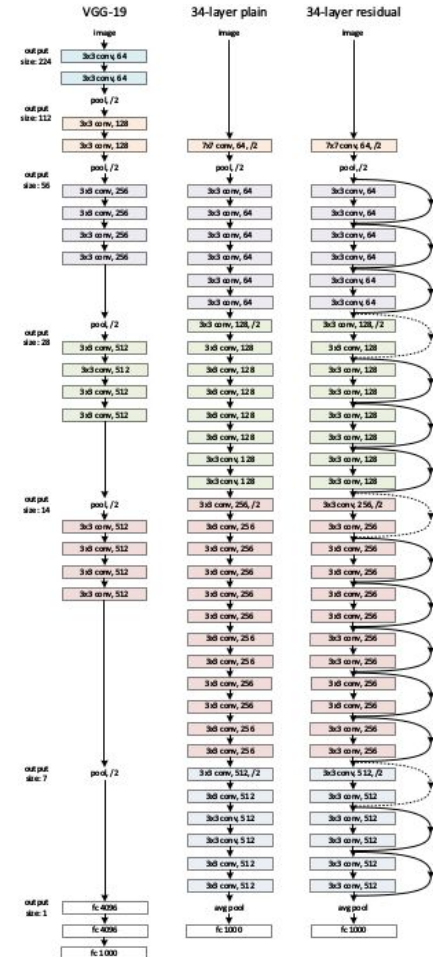
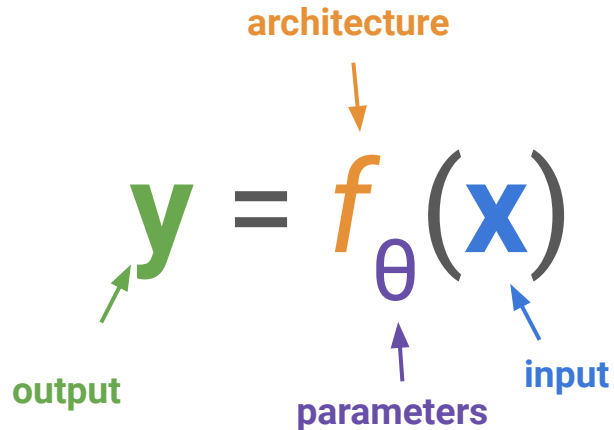
$255^{210 \times 160}$ states

left, right, up, down, fire, NOOP

6 actions

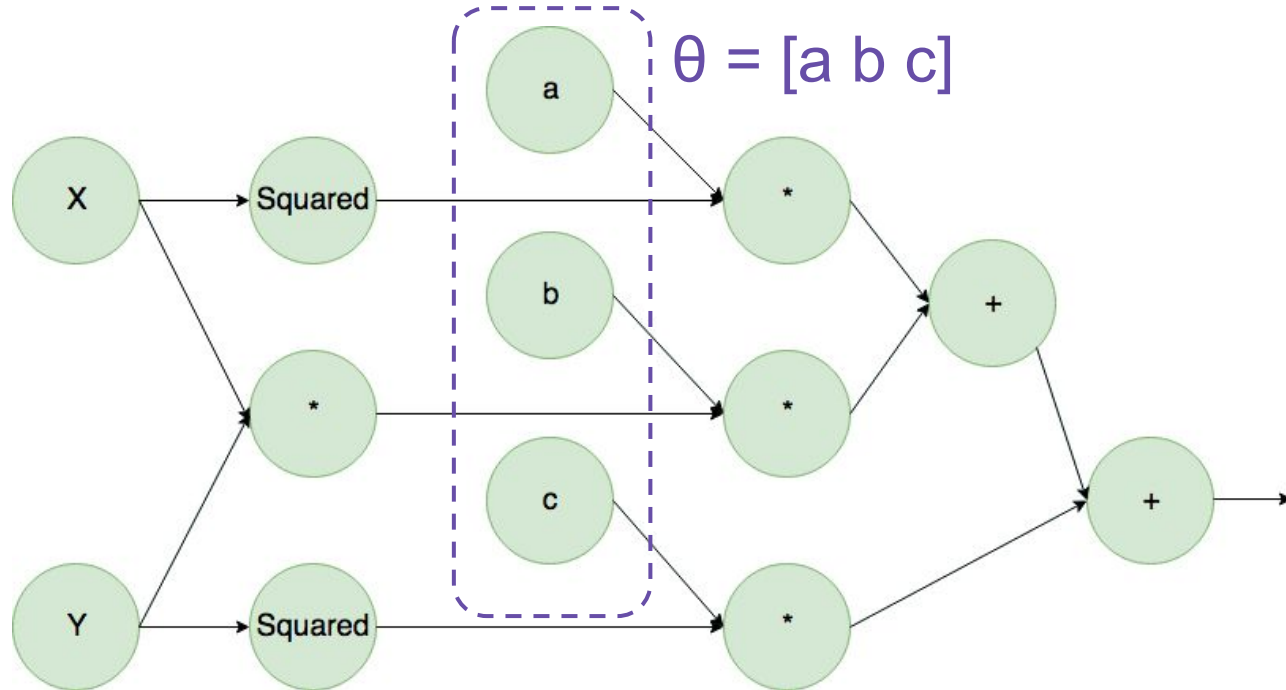
Deep Neural Networks

- Universal function approximators
- Lots of linear algebra run on GPU
- Typically gigabytes of parameters



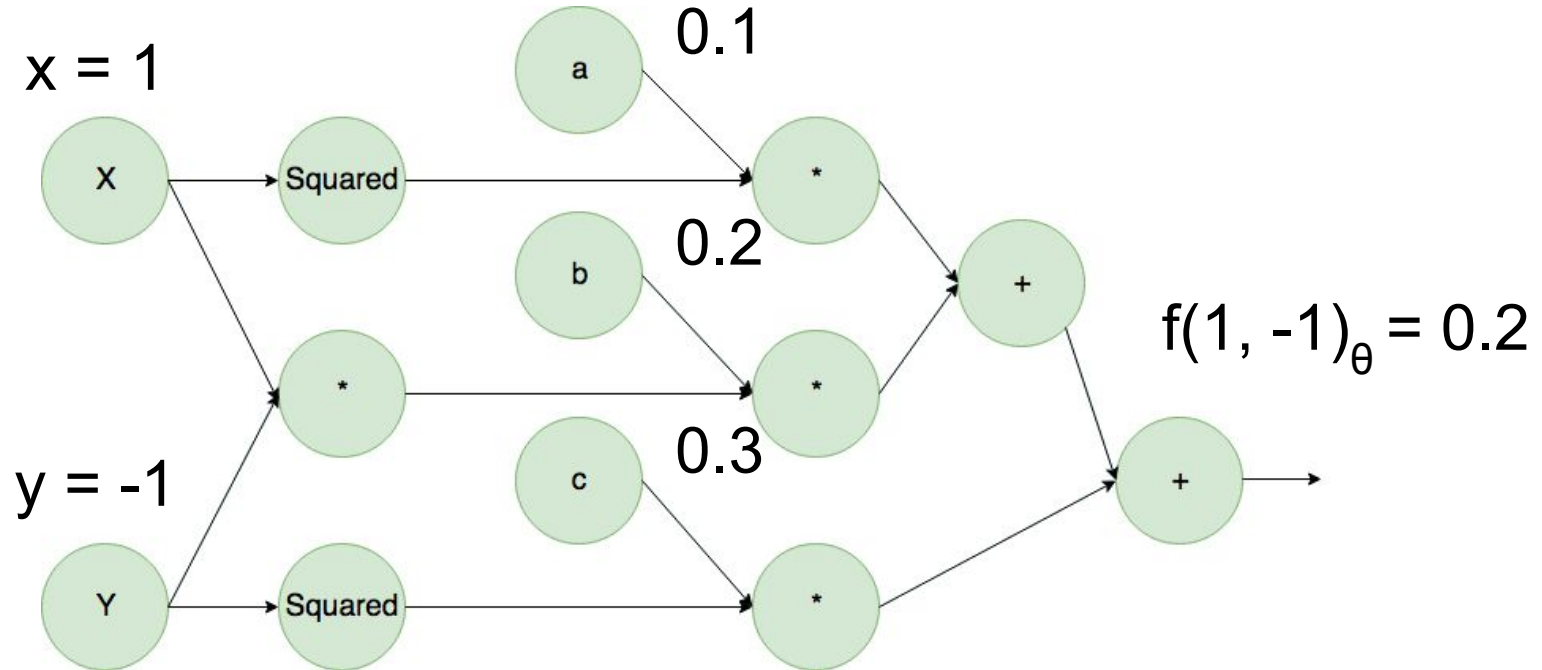
Neural Networks as Computational Graph

$$f(x, y)_{\theta} = ax^2 + bxy + cy^2$$



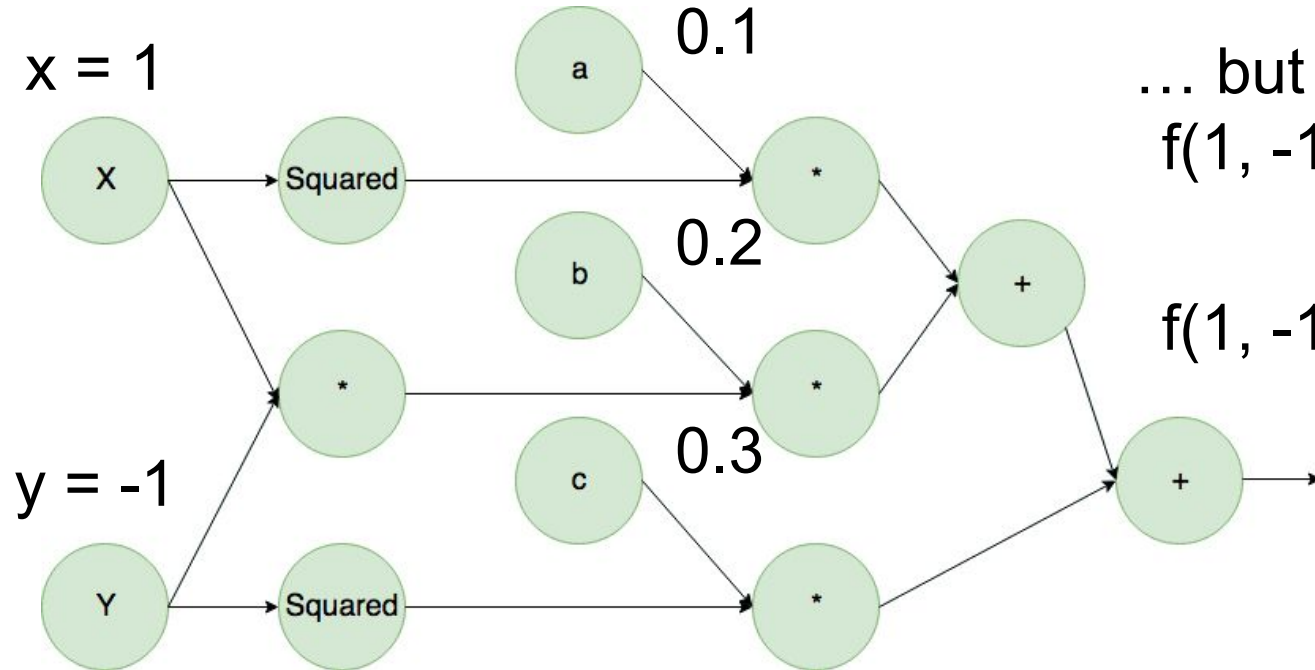
Example

$$f(x, y)_{\theta} = 0.1x^2 + 0.2xy + 0.3y^2$$



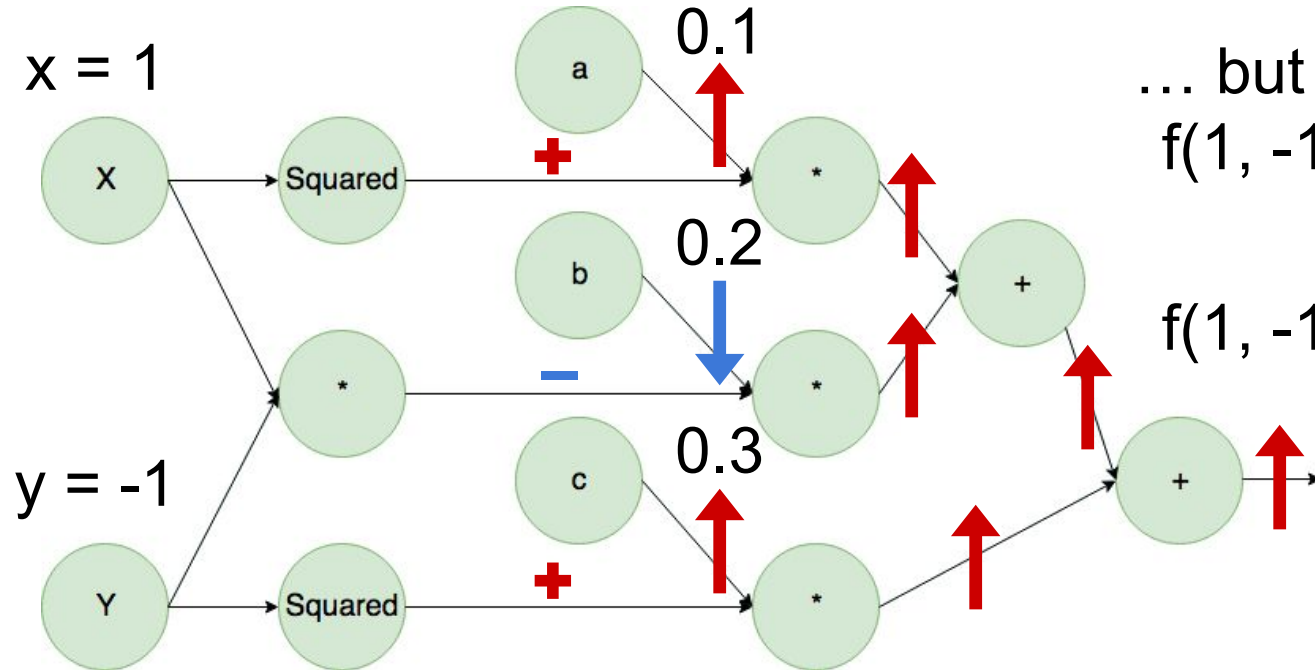
Example

$$f(x, y)_{\theta} = 0.1x^2 + 0.2xy + 0.3y^2$$



Automatic Differentiation

$$f(x, y)_{\theta} = 0.1x^2 + 0.2xy + 0.3y^2$$



... but we want
 $f(1, -1)_{\theta} = 0.5$

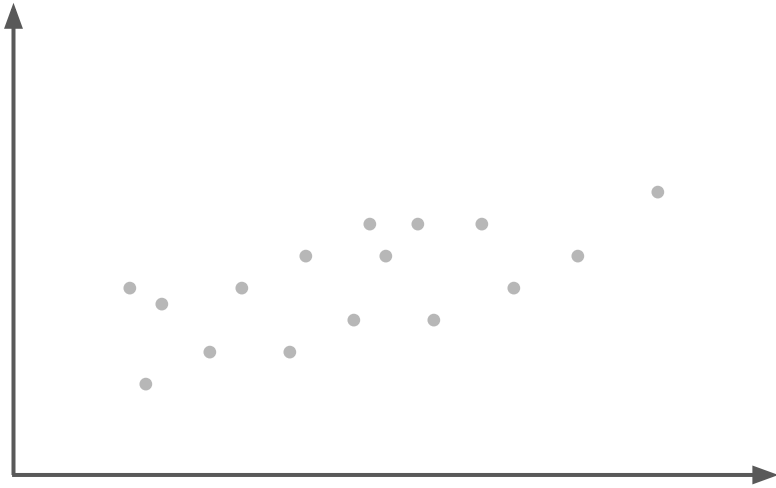
$f(1, -1)_{\theta} = 0.2$

Automatic Differentiation

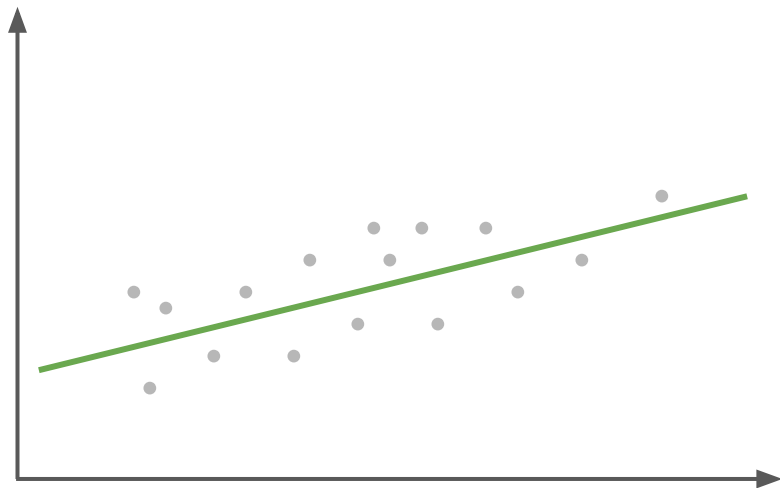
- Automatic differentiation packages calculate gradients automatically
- Deep learning packages are essentially fancy automatic differentiation libraries

Linear Regression Example

Data: $\{(x_i, y_i)\}_{i=1}^N$



Linear Regression Example



Data: $\{(x_i, y_i)\}_{i=1}^N$

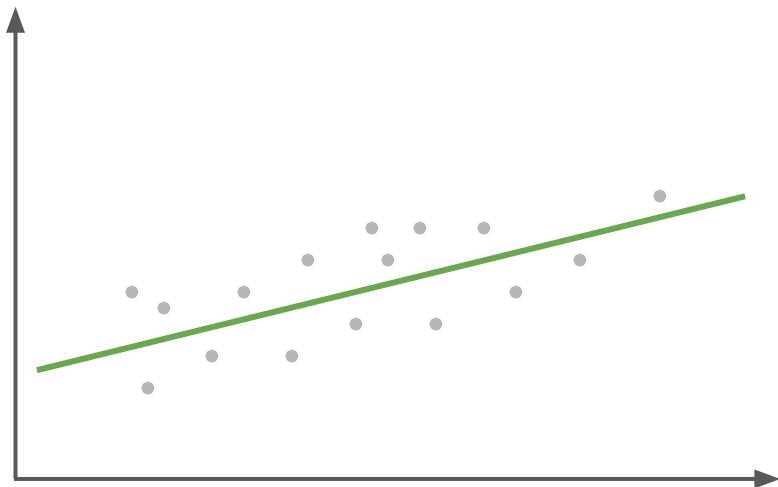
2D line:

$$y = w_2 x + w_1$$

Linear algebra notation:

$$y = \mathbf{w}^T \mathbf{x}$$

Linear Regression Example



Data: $\{(x_i, y_i)\}_{i=1}^N$

2D line:

$$y = w_2 x + w_1$$

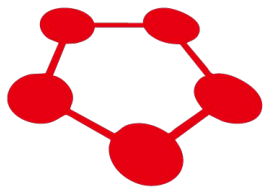
Linear algebra notation:

$$y = \mathbf{w}^T \mathbf{x}$$

Loss:

$$L_{\mathbf{w}}(D) = (1 / N) \sum_{i=1}^N (y_i - \mathbf{w}^T \mathbf{x}_i)^2$$

Chainer & CuPy

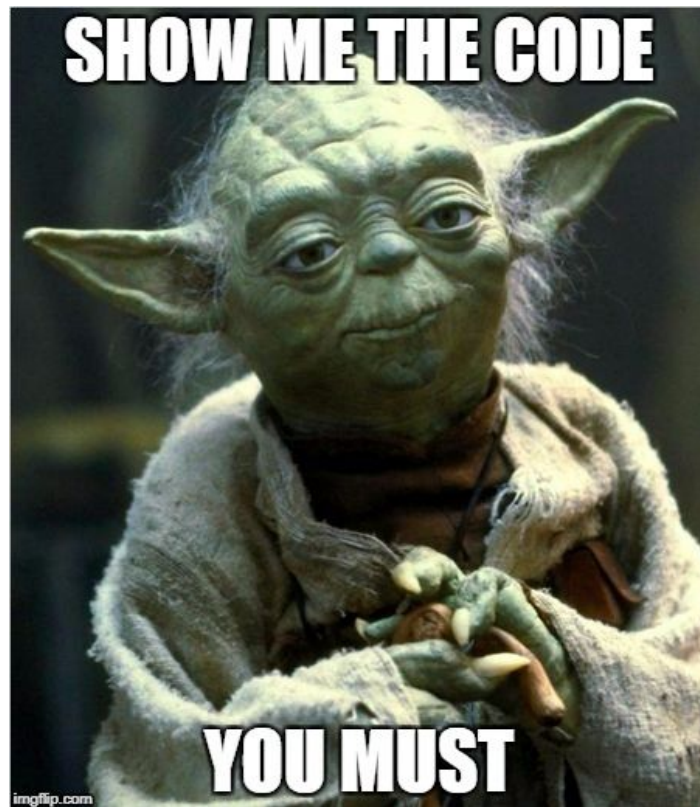


Chainer



CuPy

Chainer & CuPy



Convolutional Neural Networks (CNNs)

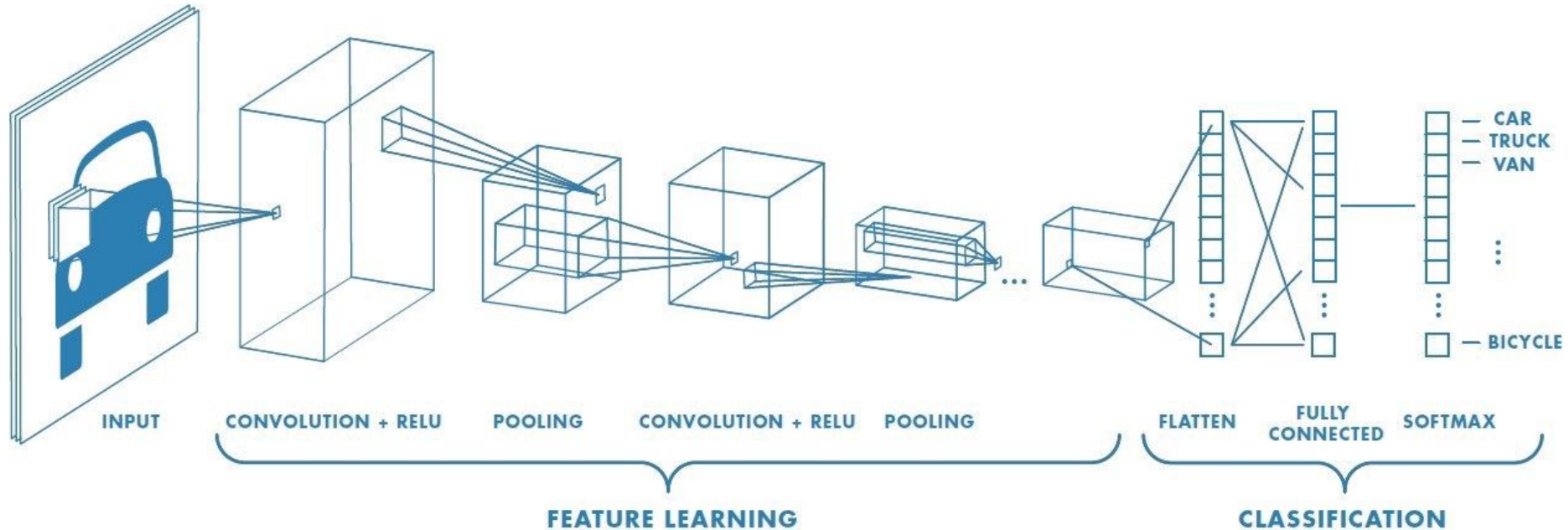
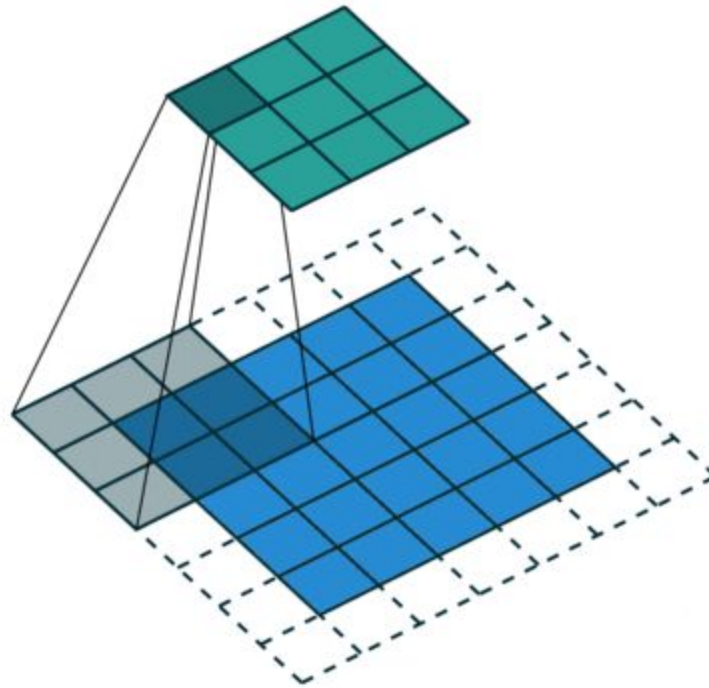
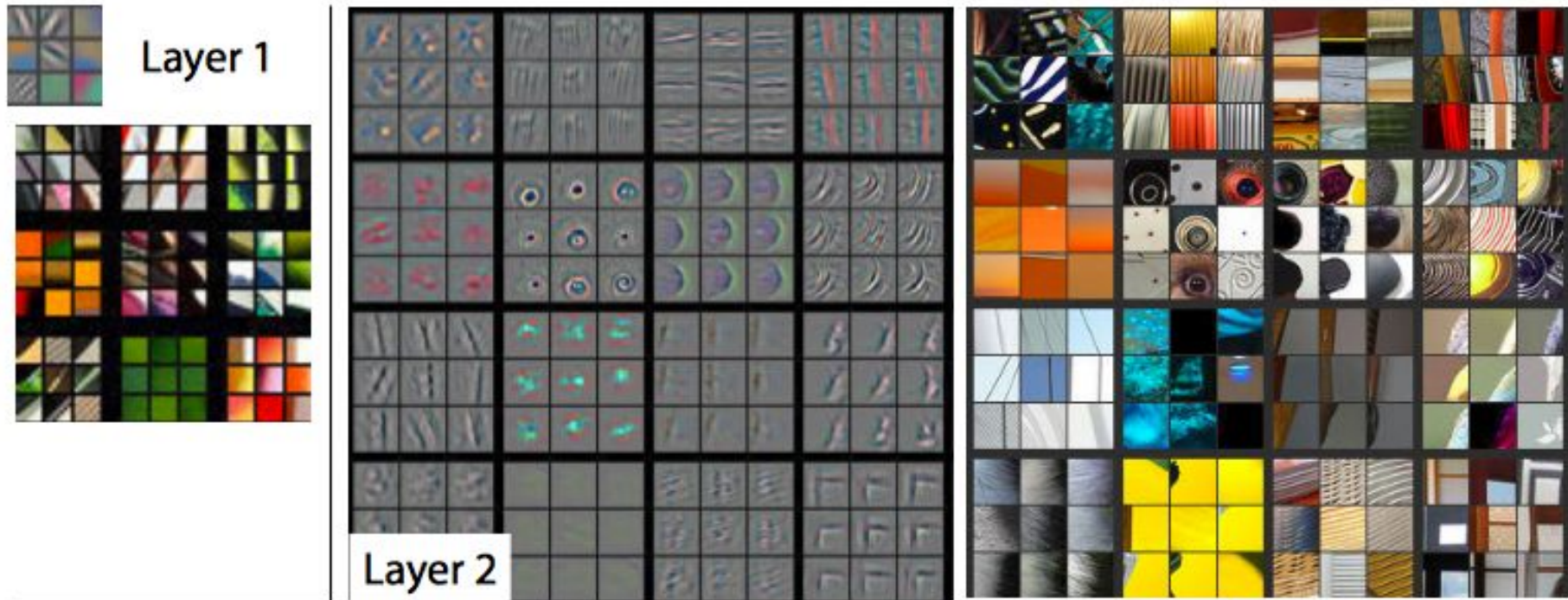


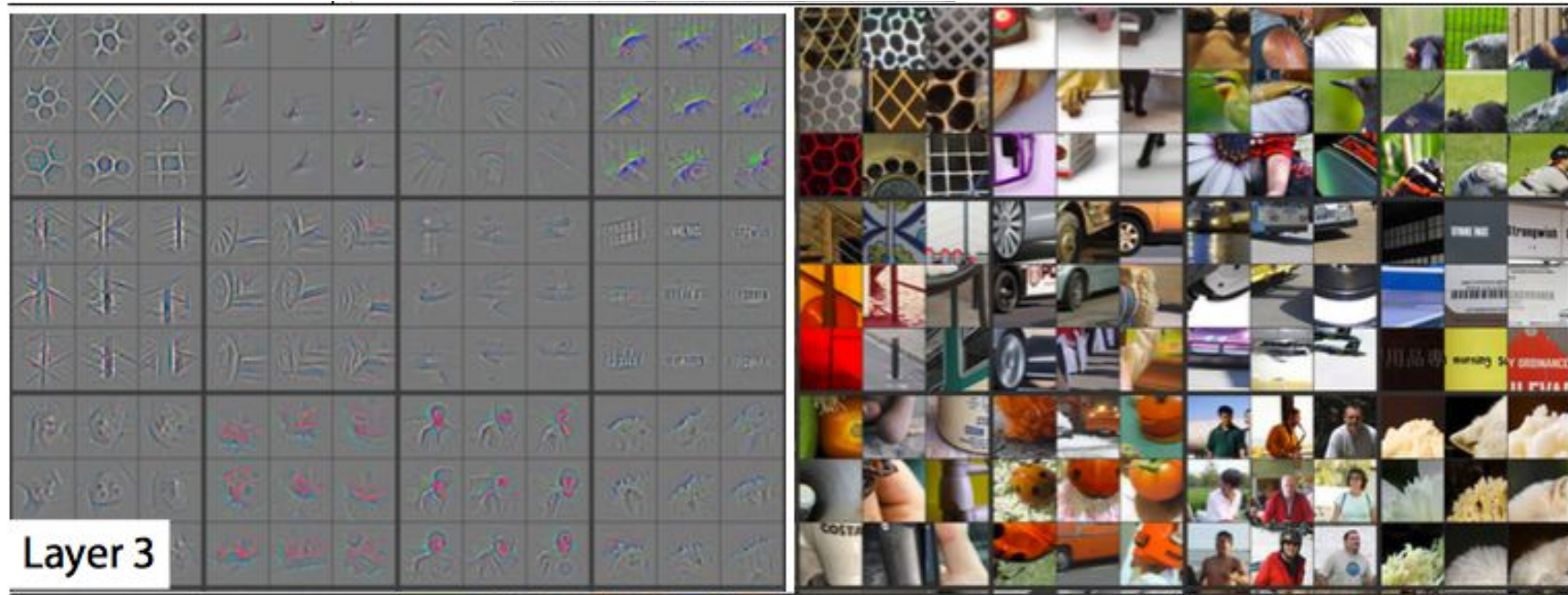
Image Convolution



Why CNNs?



Why CNNs?



Why CNNs?

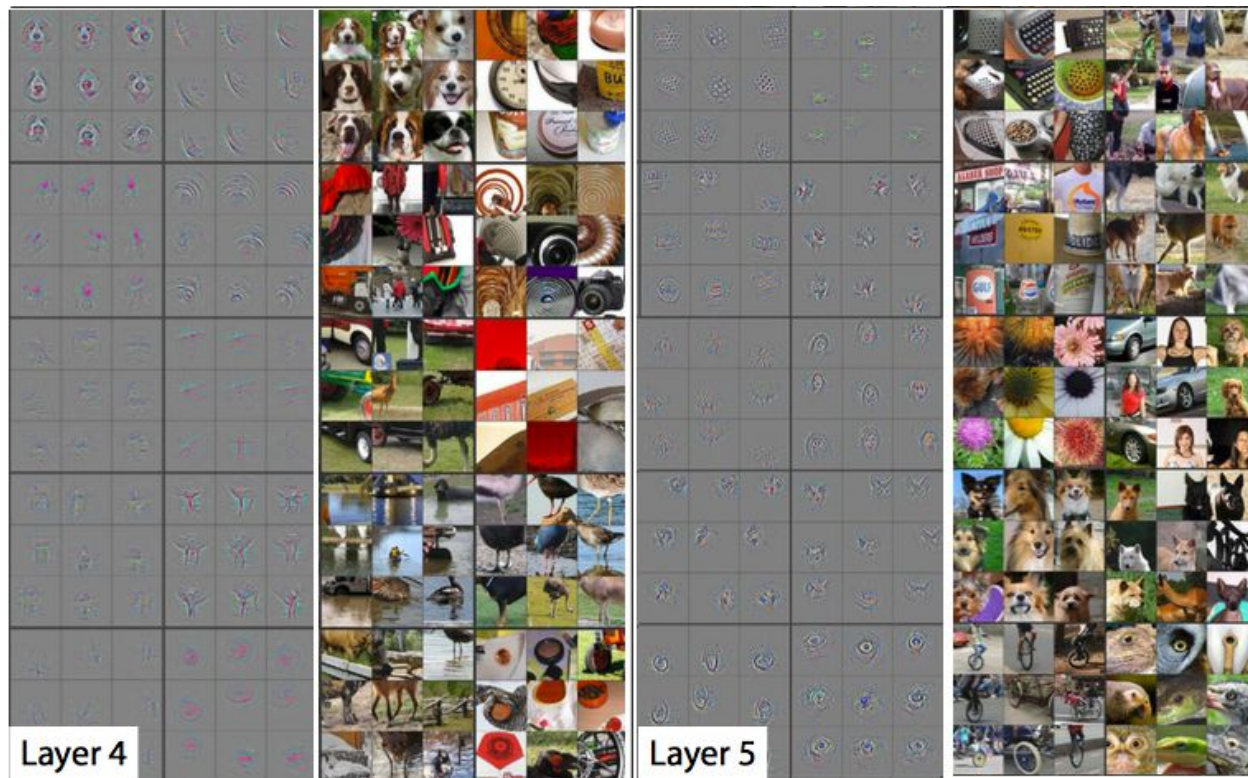
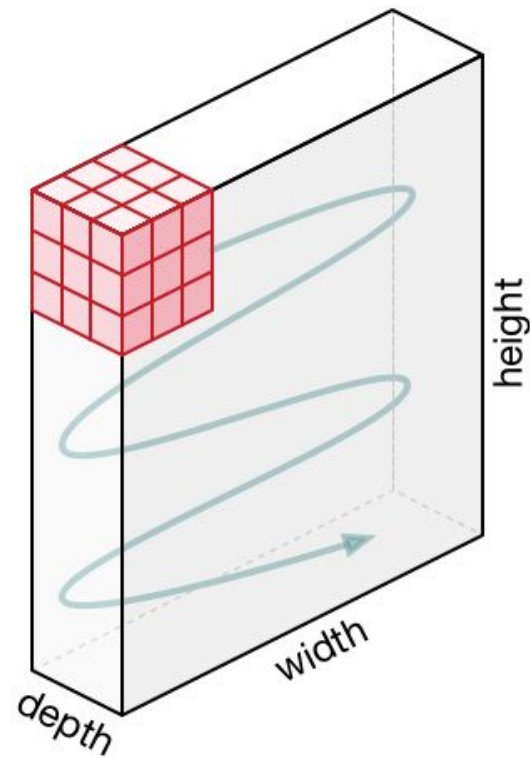
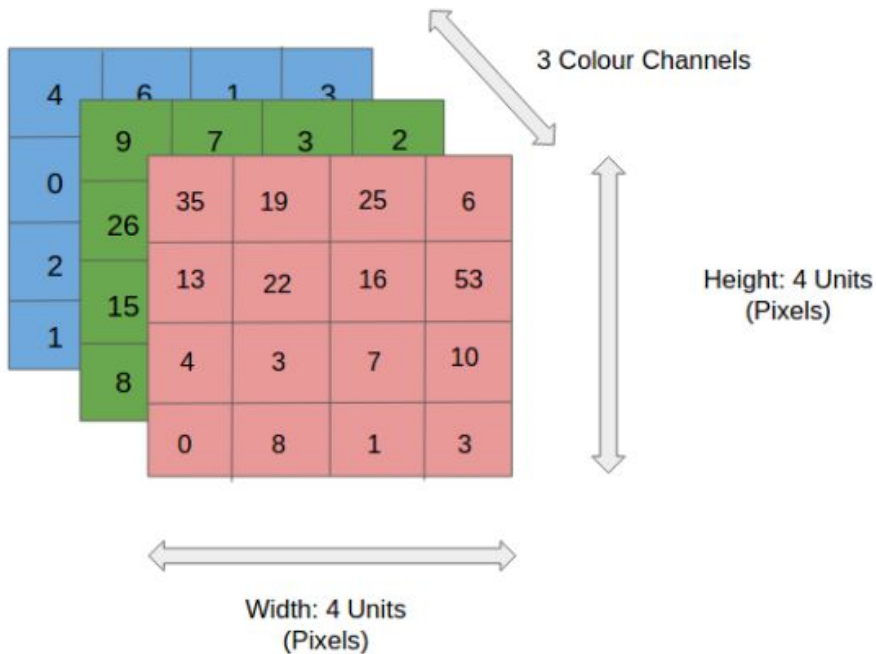


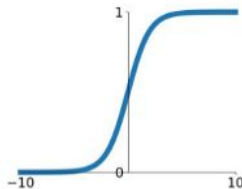
Image Convolution Kernels



Activation Functions

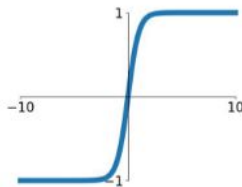
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



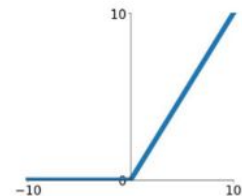
tanh

$$\tanh(x)$$



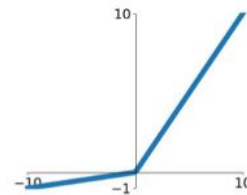
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

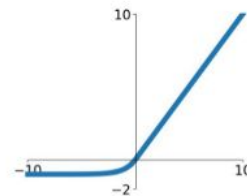


Maxout

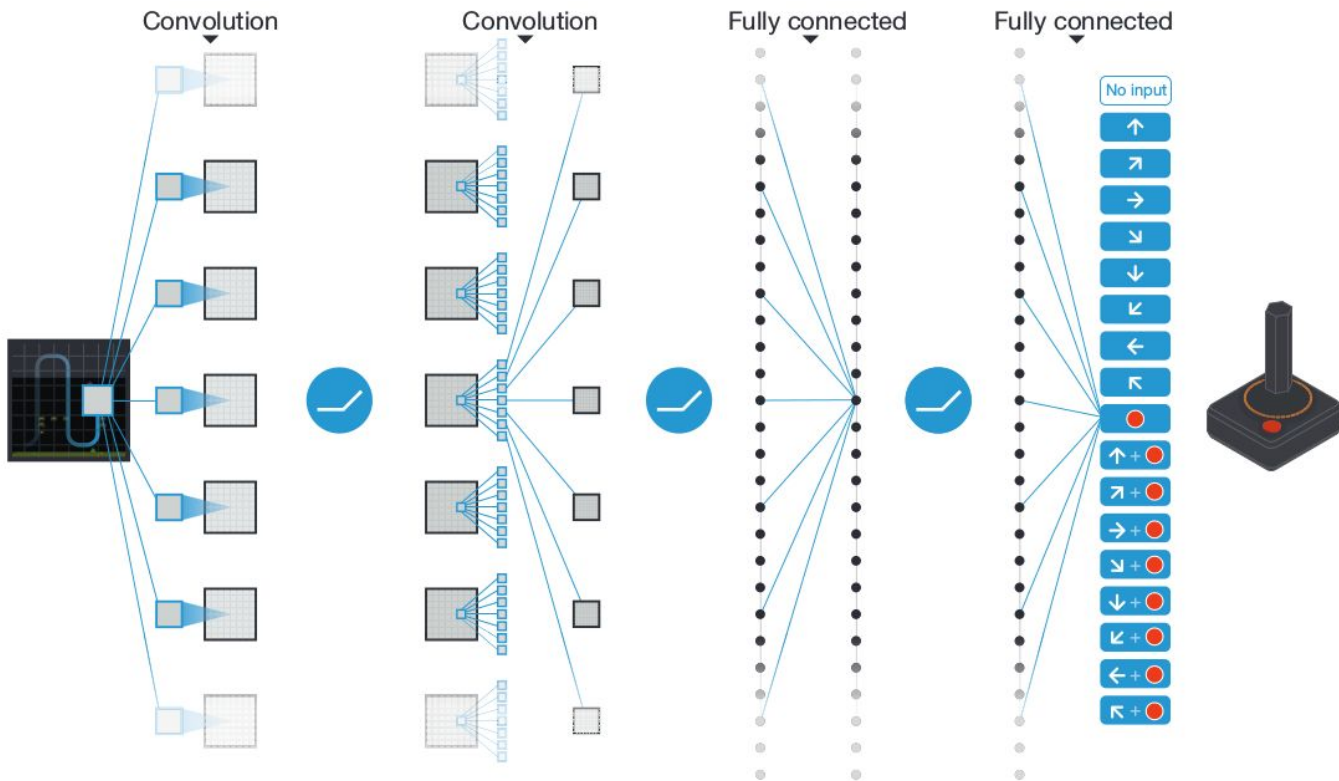
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

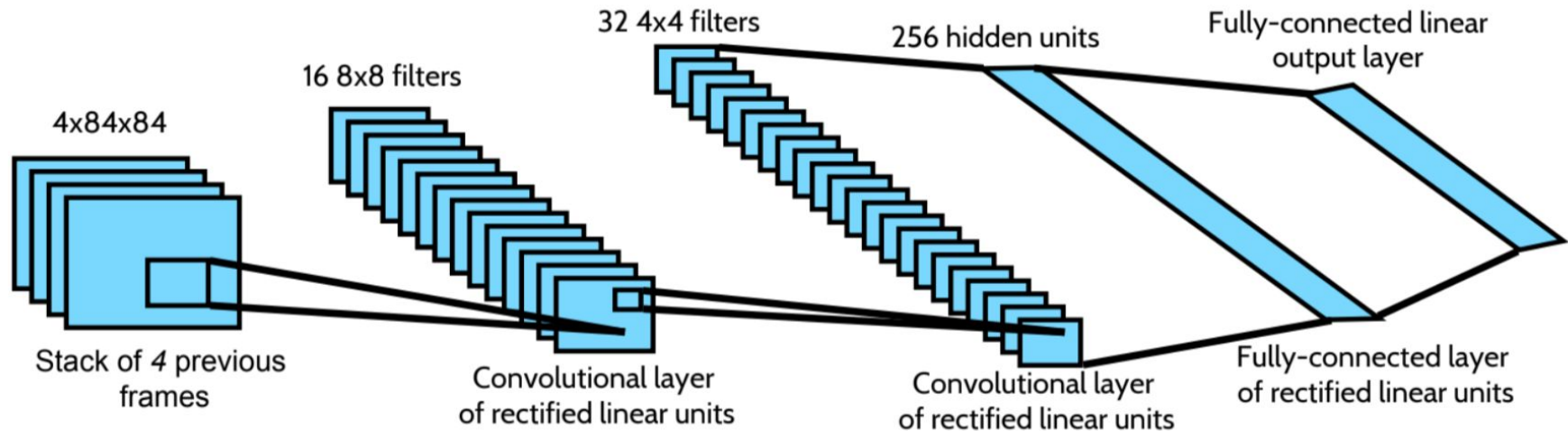
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Deep Q-value Network (DQN)



Deep Q-value Network (DQN)



Implementing DQN



Putting Everything Together

Q-learning

$$y_t - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) = 0 \text{ for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

$$y_t = \begin{cases} r_t + \gamma \max_{\mathbf{a}} \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}) & \text{if } \mathbf{s}_t \text{ not terminal} \\ r_t & \text{otherwise} \end{cases}$$

Q-learning

$$r_t + \gamma \max_a \mathbf{Q}(\mathbf{s}_{t+1}, \mathbf{a}) - \mathbf{Q}(\mathbf{s}_t, \mathbf{a}_t) = 0 \quad \text{for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

Q-learning (annoying details cont'd)

$$r_t + \gamma \left[\max_a Q(\mathbf{s}_{t+1}, \mathbf{a}) - Q(\mathbf{s}_t, \mathbf{a}_t) \right] = 0 \quad \text{for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

If we update **Q** every timestep there is too much noise

Q-learning (annoying details cont'd)

$$r_t + \gamma \max_a Q_{\theta^-}(s_{t+1}, a) - Q_{\theta}(s_t, a_t) = 0 \text{ for all } (s_t, a_t, r_t, s_{t+1})$$

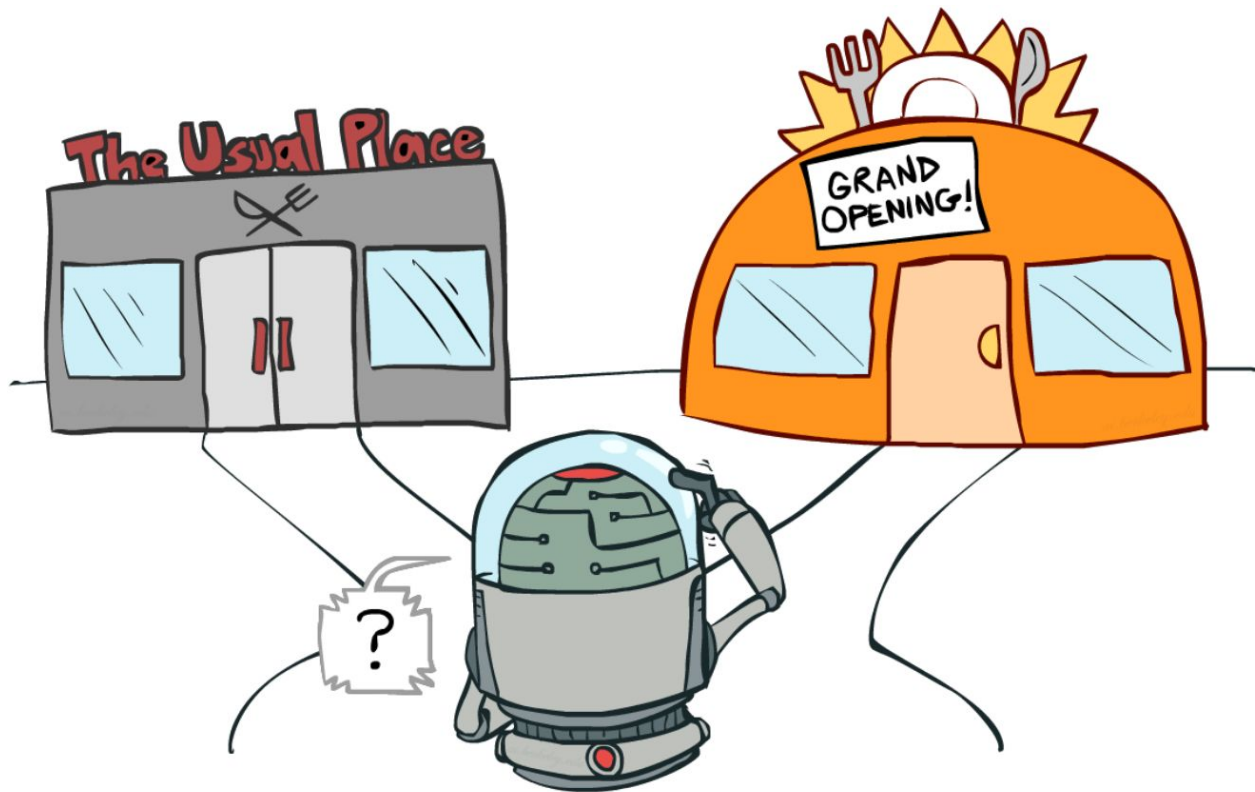
- Update Q_{θ} every timestep
- Periodically Set $Q_{\theta^-} = Q_{\theta}$

Q-learning

$$y_t - \mathbf{Q}_\theta(\mathbf{s}_t, \mathbf{a}_t) = 0 \text{ for all } (\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$$

$$y_t = \begin{cases} r_t + \gamma \max_{\mathbf{a}} \mathbf{Q}_\theta(\mathbf{s}_{t+1}, \mathbf{a}) & \text{if } \mathbf{s}_t \text{ not terminal} \\ r_t & \text{otherwise} \end{cases}$$

Exploration vs. Exploitation



ϵ -greedy Policy

- Choose random action with probability ϵ
- Full exploration $\epsilon = 1.0$
- Full exploitation $\epsilon = 0.0$

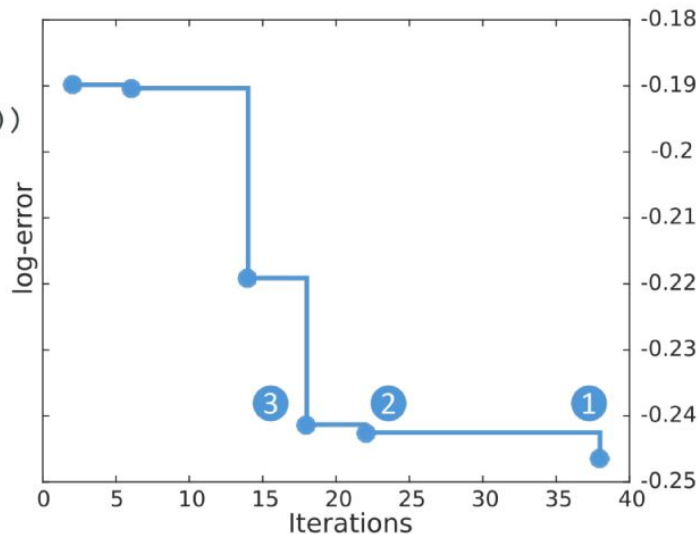
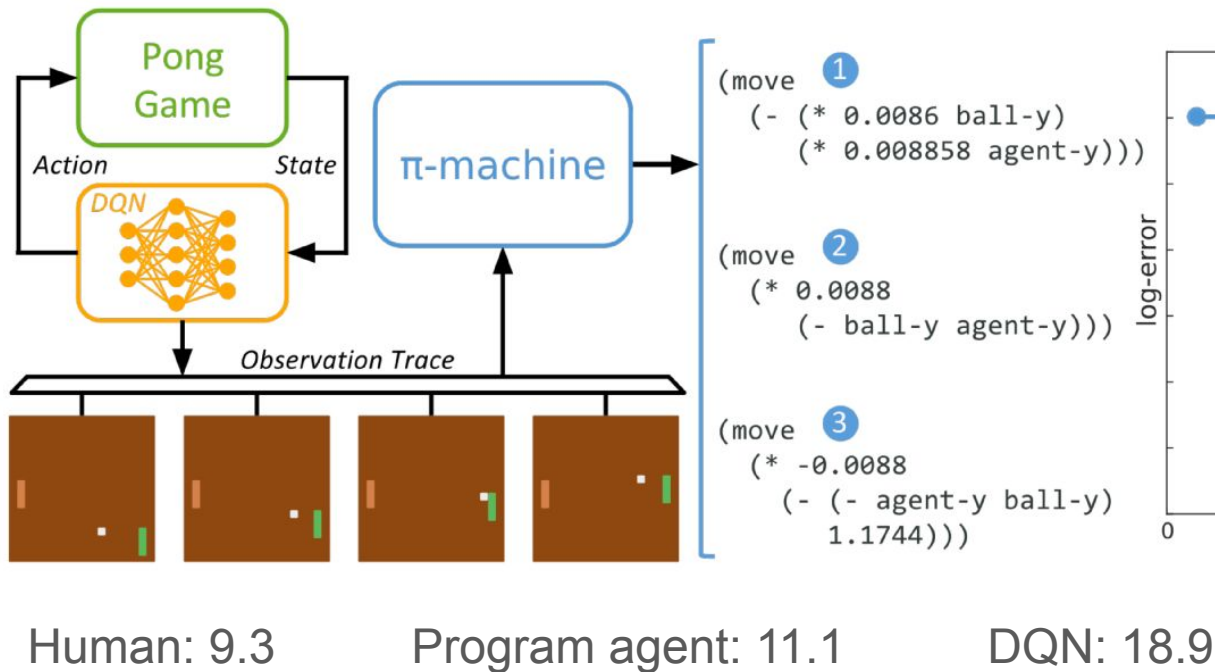
Training

- ~24 hours on i7 laptop with GTX1050
- Can be taken down to ~1 hour with a beefy machine and a few optimisations
- Deep RL is notorious for irreproducibility
- Sometimes multiple runs are needed to get a working agent

Playing Better Than Human



Deep Q-learning Network



Starting soon...



Be the first one to get an invite to apply!

sciro.ai