Reinforcement Learning

Lab #4: TD-learning and Neural Networks

Daniil Arapov

Test questions for today:

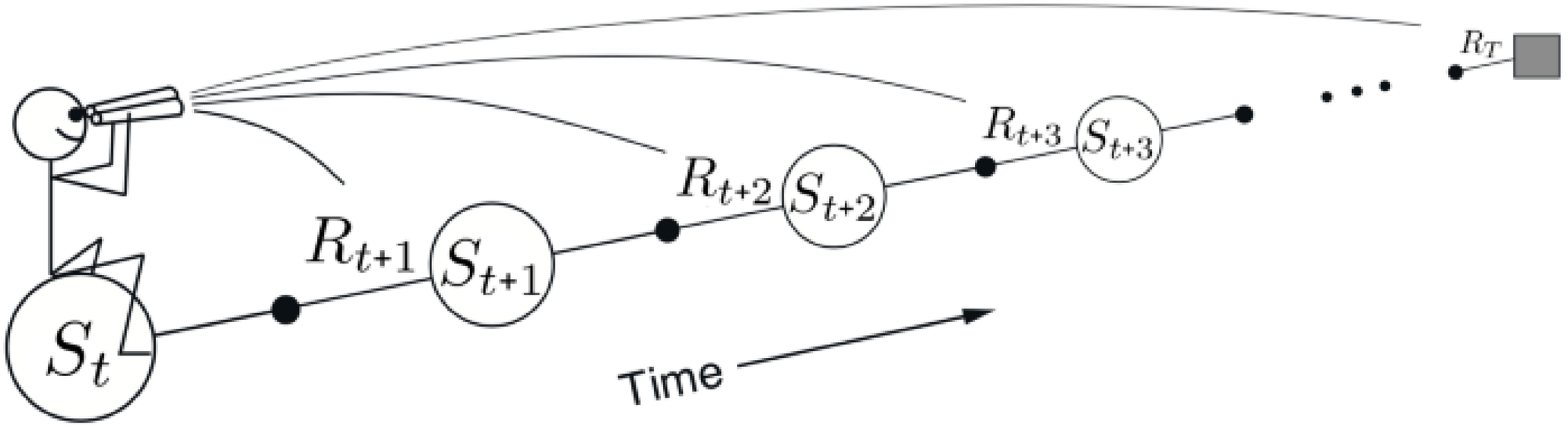
1. What is the difference between on-policy and off-policy learning?
2. Write down SARSA formula.
3. Write down 3 function approximators known to you.
4. How ***policy evaluation*** and ***policy improvement*** steps differ for 1) fully observable MDP and 2) model-free scenarios?

Topic #1

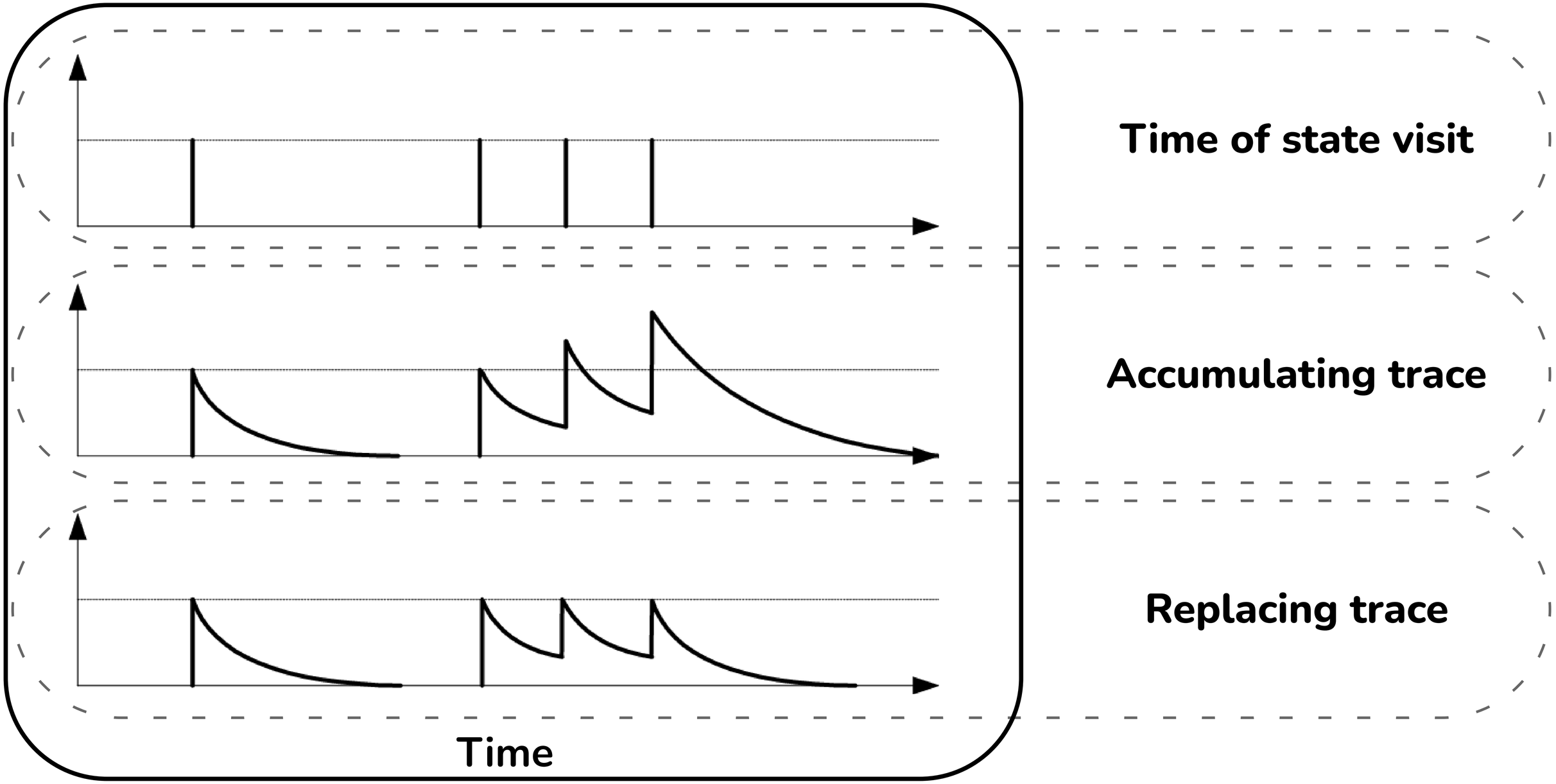
TD-lambda

Forward-view TD

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$$



Eligibility traces

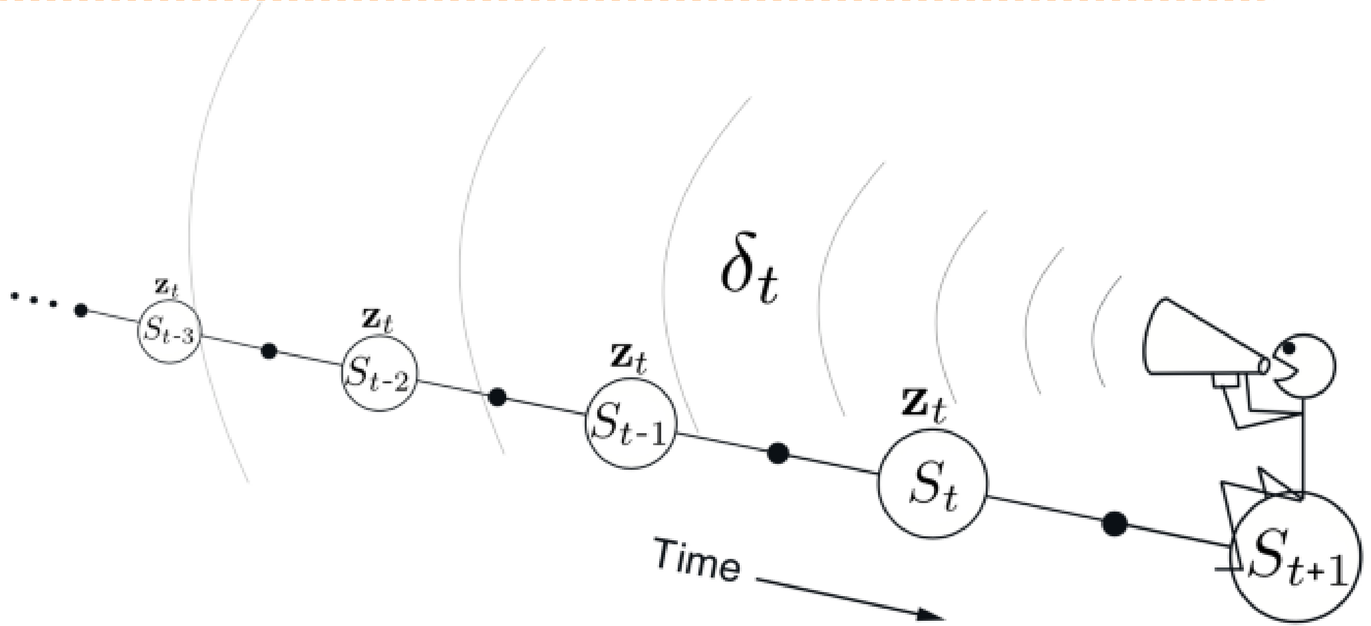


Backward-view TD

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

$$V(s) \leftarrow V(s) + \alpha \delta_t E_t(s)$$

$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbf{1}(S_t = s)$$



Sarsa(λ) Algorithm

Initialize $Q(s,a)$ arbitrarily and $e(s,a) = 0$, for all s,a

Repeat (for each episode) :

Initialize s,a

Repeat (for each step of episode) :

Take action a, observe r,s'

Choose a' from s' using policy derived from Q (e.g. $?$ - greedy)

$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$

$e(s,a) \leftarrow e(s,a) + 1$

For all s,a :

$Q(s,a) \leftarrow Q(s,a) + \alpha\delta e(s,a)$

$e(s,a) \leftarrow \gamma\lambda e(s,a)$

$s \leftarrow s'; a \leftarrow a'$

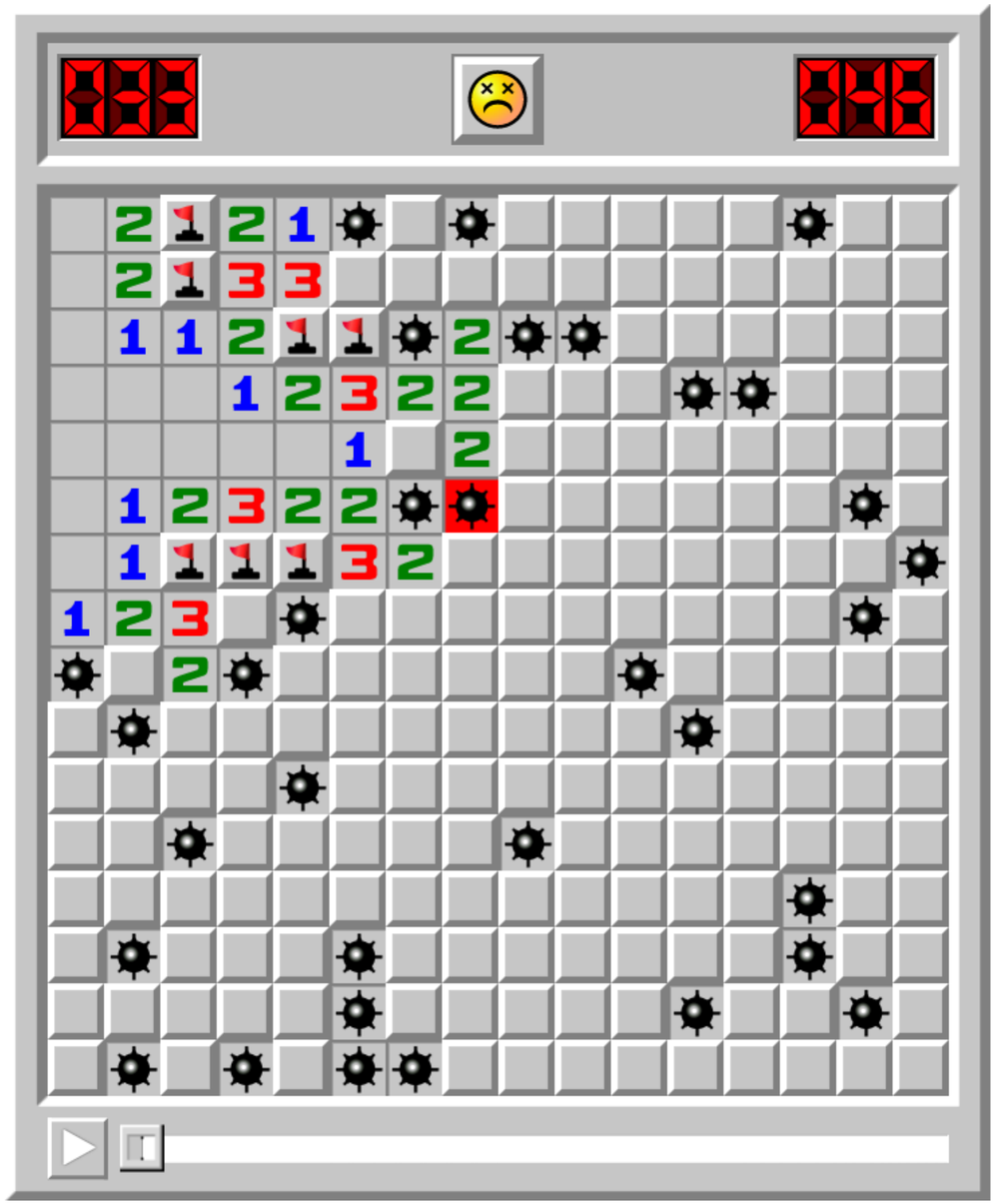
Until s is terminal

https://towardsdatascience.com/eligibility-traces-in-reinforcement-learning-a6b458c019d6

Topic #2

Neural Networks

Variable environments



Variable environments

Minesweeper

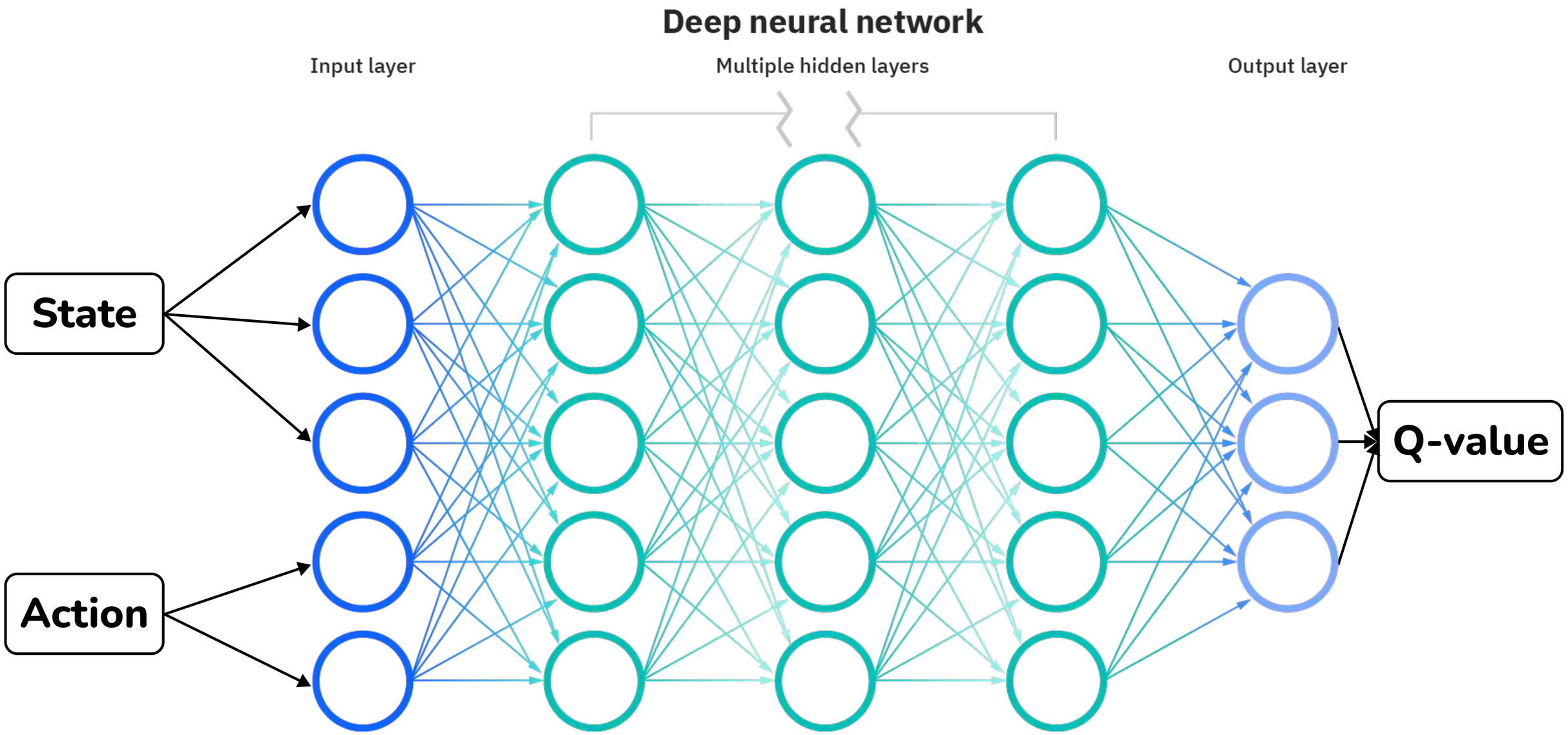
- Varying size of the field
- Varying number of mines
- Varying placement of mines

We want to create an agent capable of playing Minesweeper using any field.

For that, we will need to create a model approximating Q(S, a).

P.s. do we really need a function approximation for Minesweeper? :)

Neural Networks

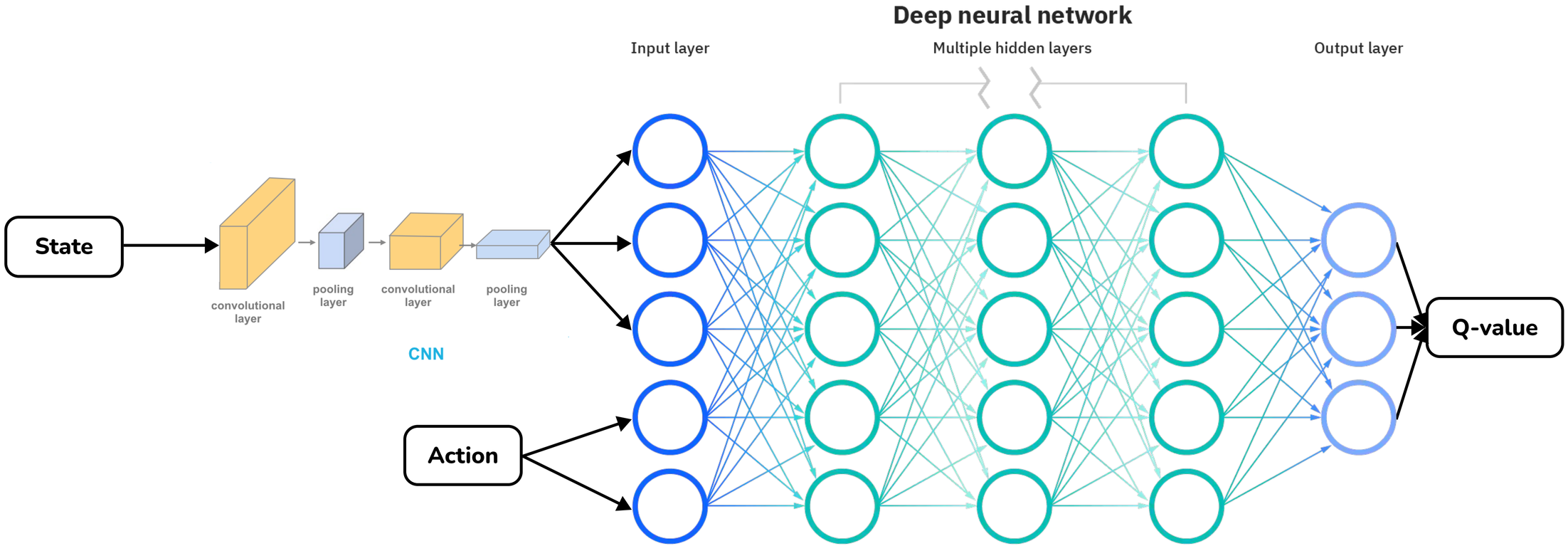


https://www.ibm.com/cloud/learn/neural-networks

Multi-Layer Perceptrones: Limitations

1. Fixed input size

2. Awful for image/signal processing

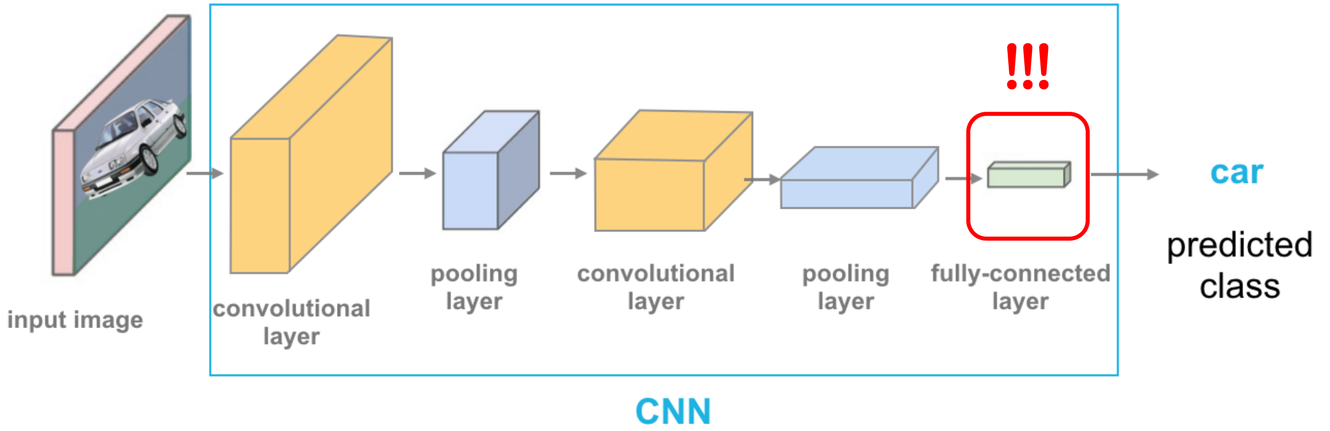


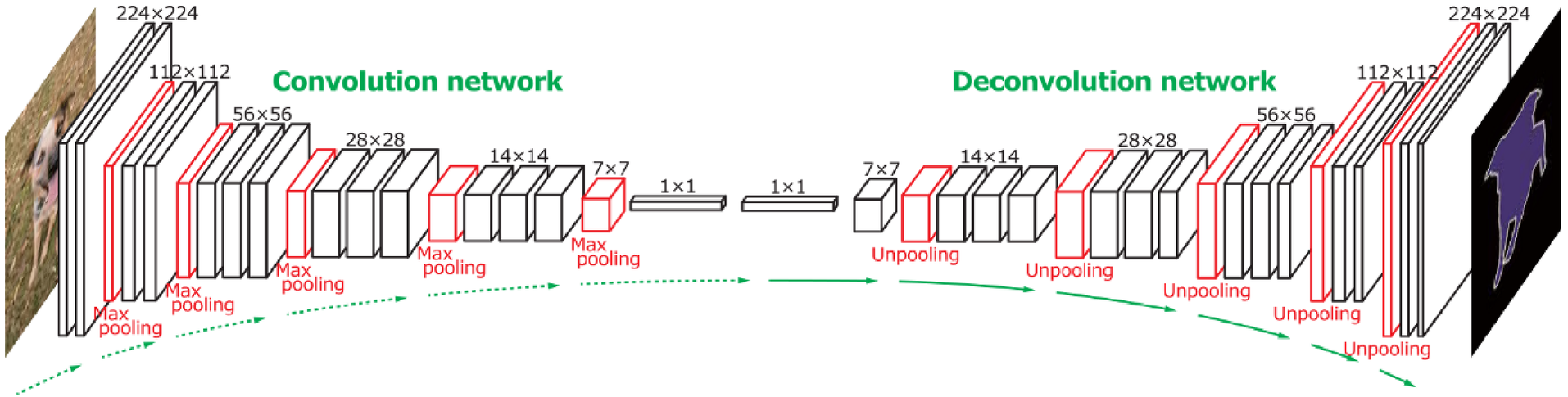
Convolutional Neural Networks

1. Cool for image/signal processing

2. Fixed input size???

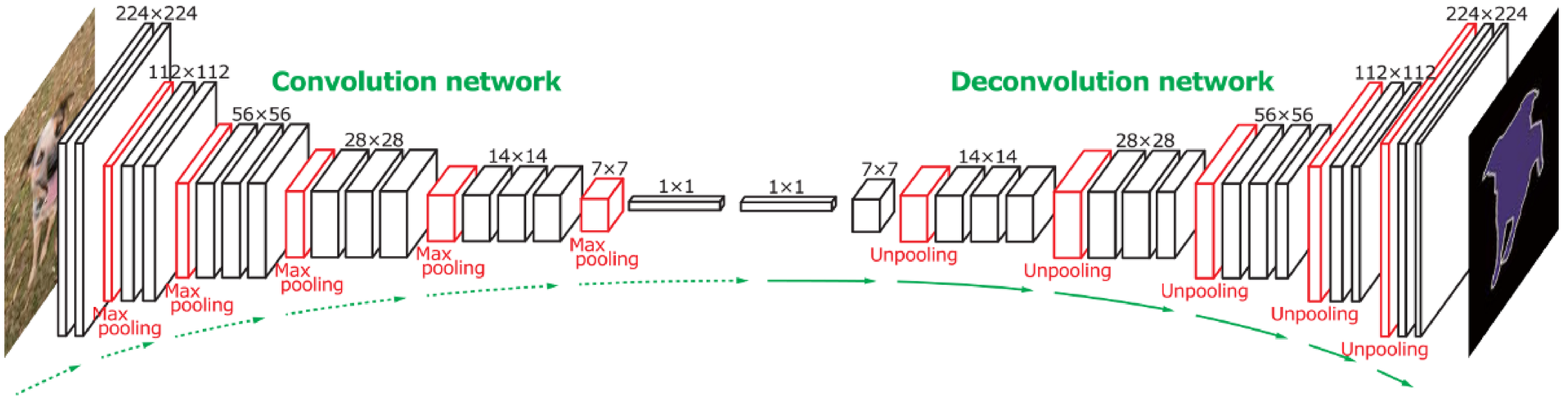
Convolutional Neural Networks





Fully convolutional networks work with varying width/height

Convolutional Neural Networks



+ global average pooling =

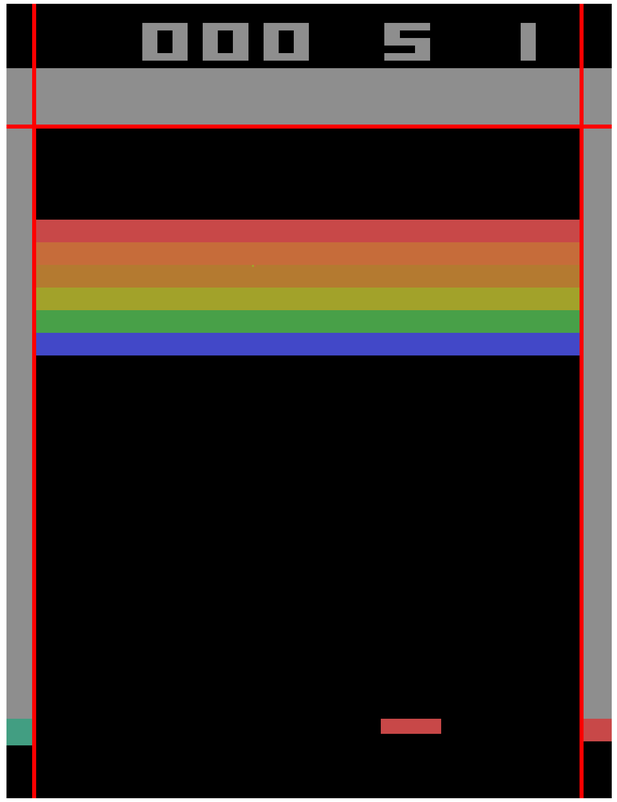
size-invariant prediction

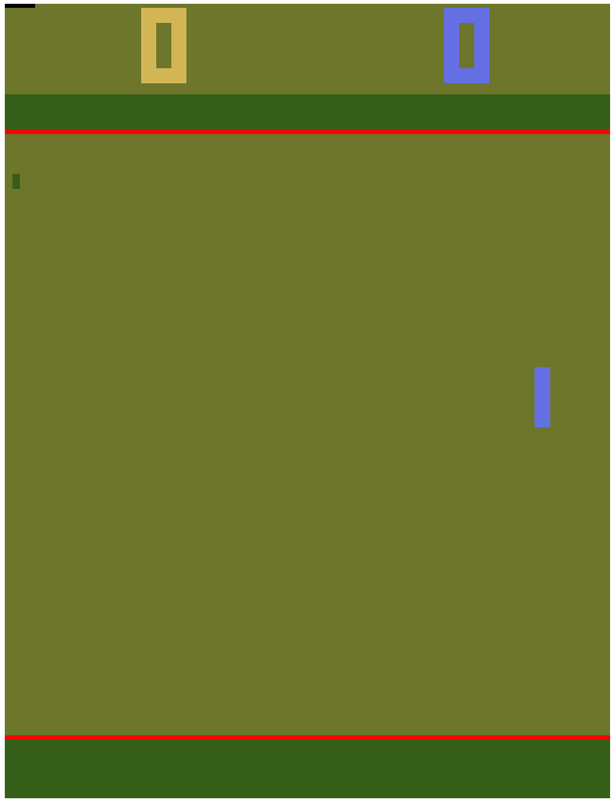
Fully convolutional networks work with varying width/height

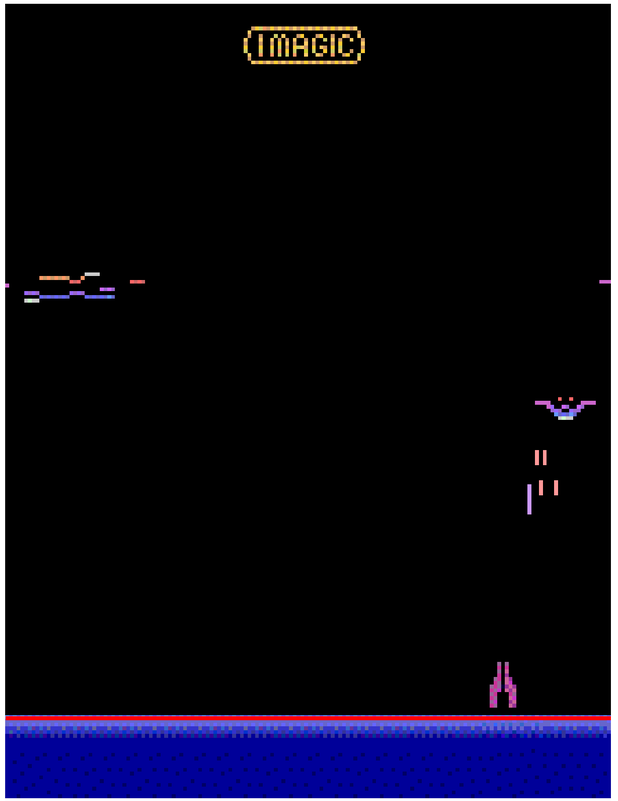
Topic #3

Games with visual representation

Atari







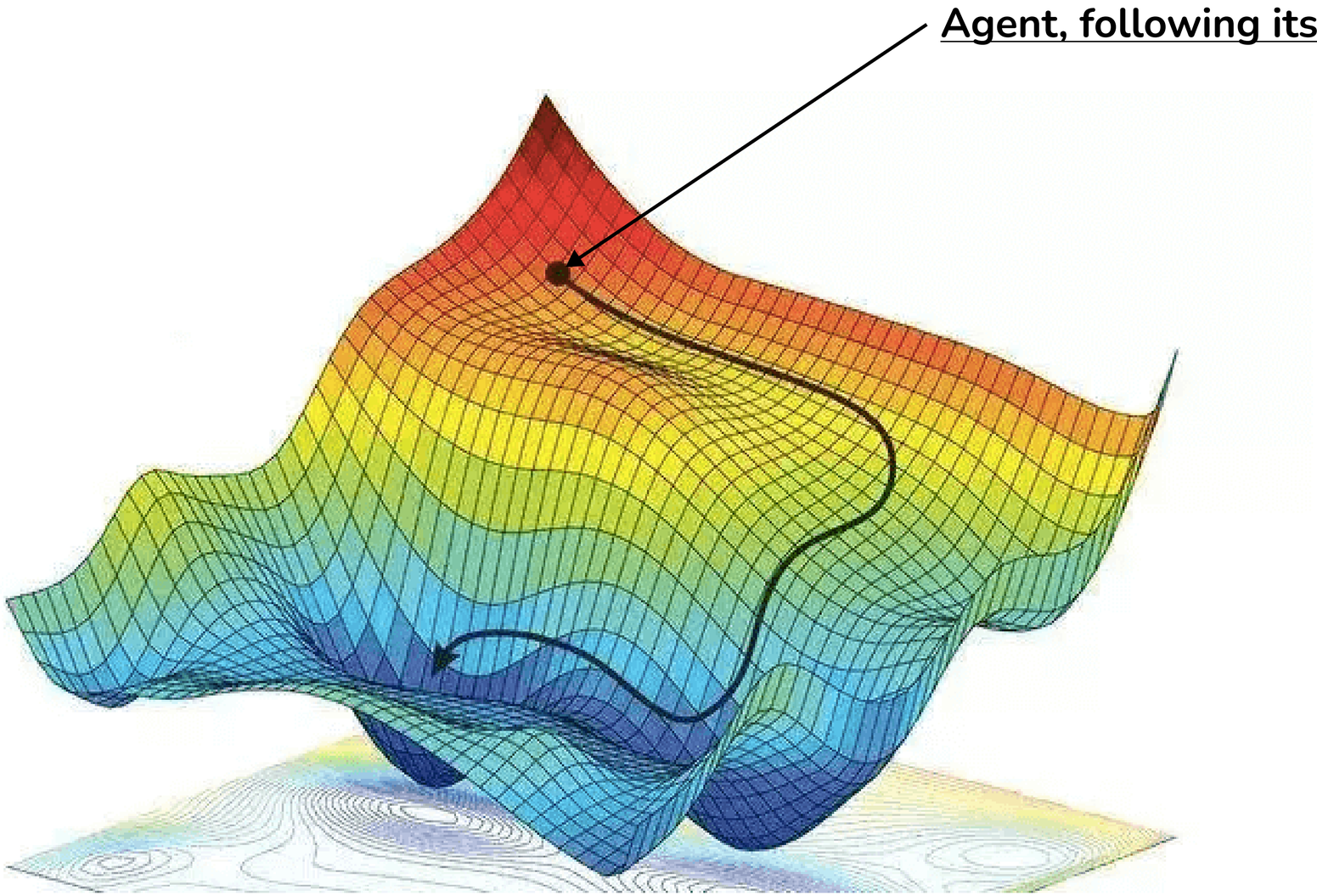

PRODUCTION
DeepMind
StarCraft II
DEMONSTRATION
Workers: 0/16
Workers: 1/3
4:38
REPLAY
Catalyst LE
DeepMind
AlphaStar
37 /47
200 +671
76 +268
15
22
SUPPLY
MINERALS
GAS
WORKERS
ARMY
WEAPONS & ARMOR
LiquidMaNa
42 /55
145 +783
136 +291
26
15
ORACLE
0
69
31
Kills: 7


OpenAI (Bot)
4

Topic #4

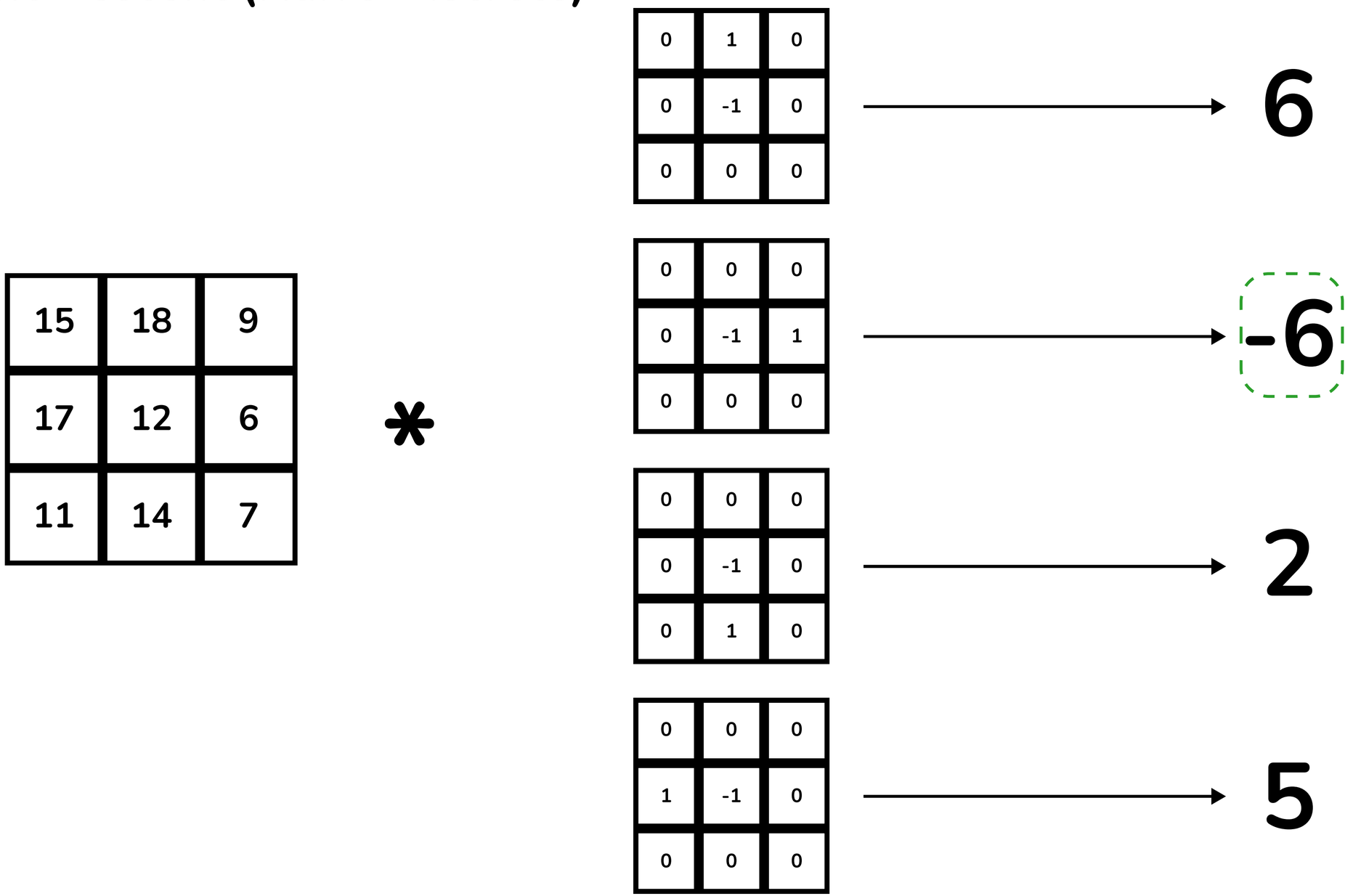
Optimization

Gradient Descent



https://easyai.tech/en/ai-definition/gradient-descent/

Gradient Descent (Naive Discrete)

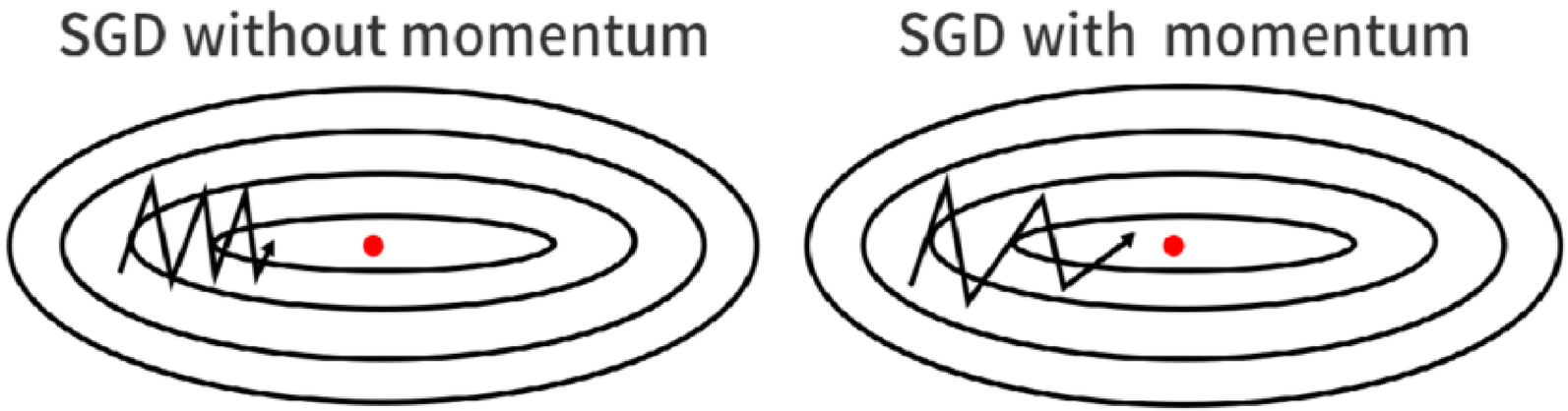


Gradient Descent

- Continuous

- Optimizers may have an internal state:

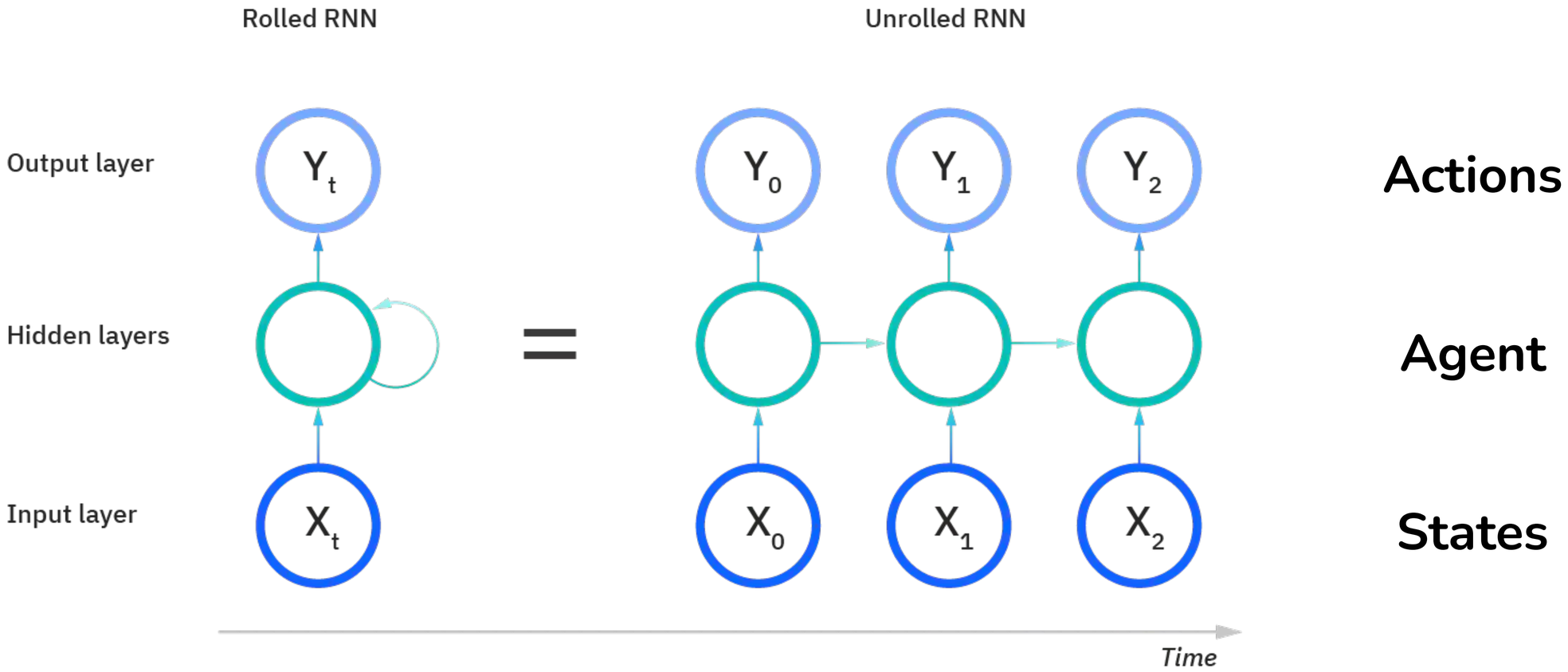
$$W_t = W_{t-1} - V_t$$
$$V_t = \beta V_{t-1} + (1-\beta)\nabla_W L(X, y; W)$$



Topic #5

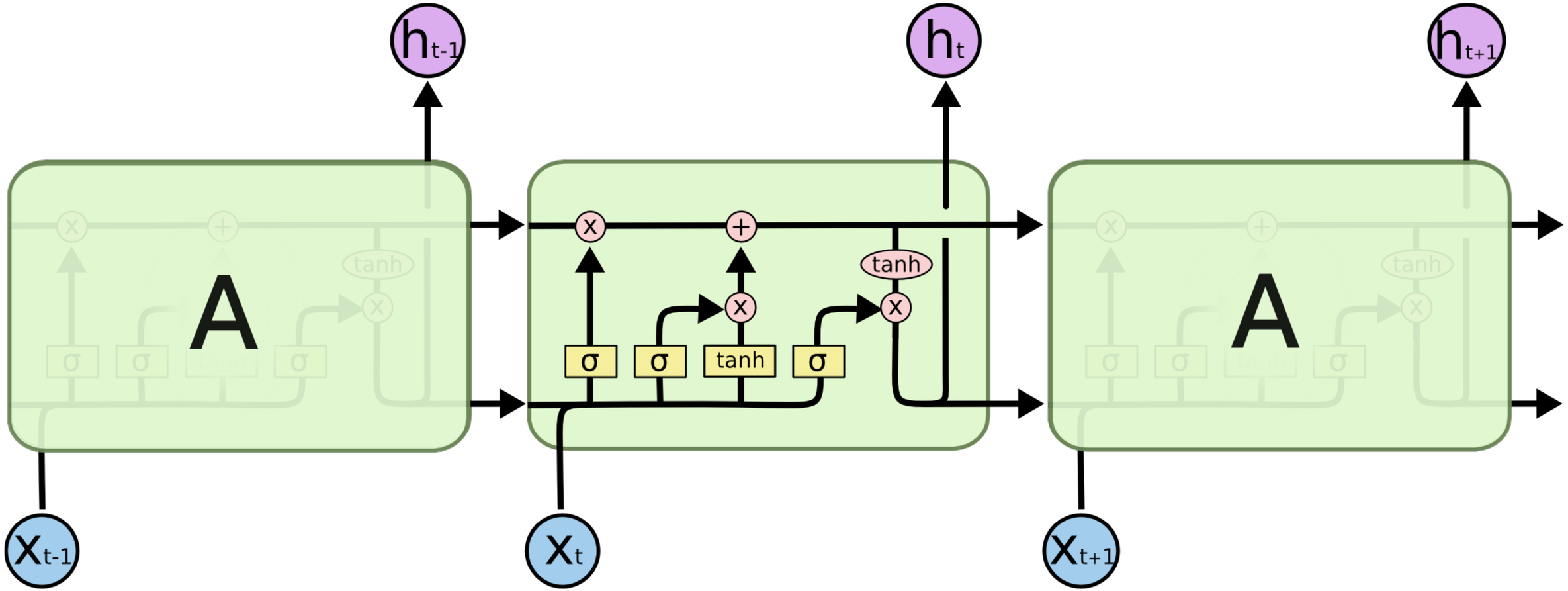
Reccurent Neural Networks
Long-Short Term Memory

Recurrent Neural Networks



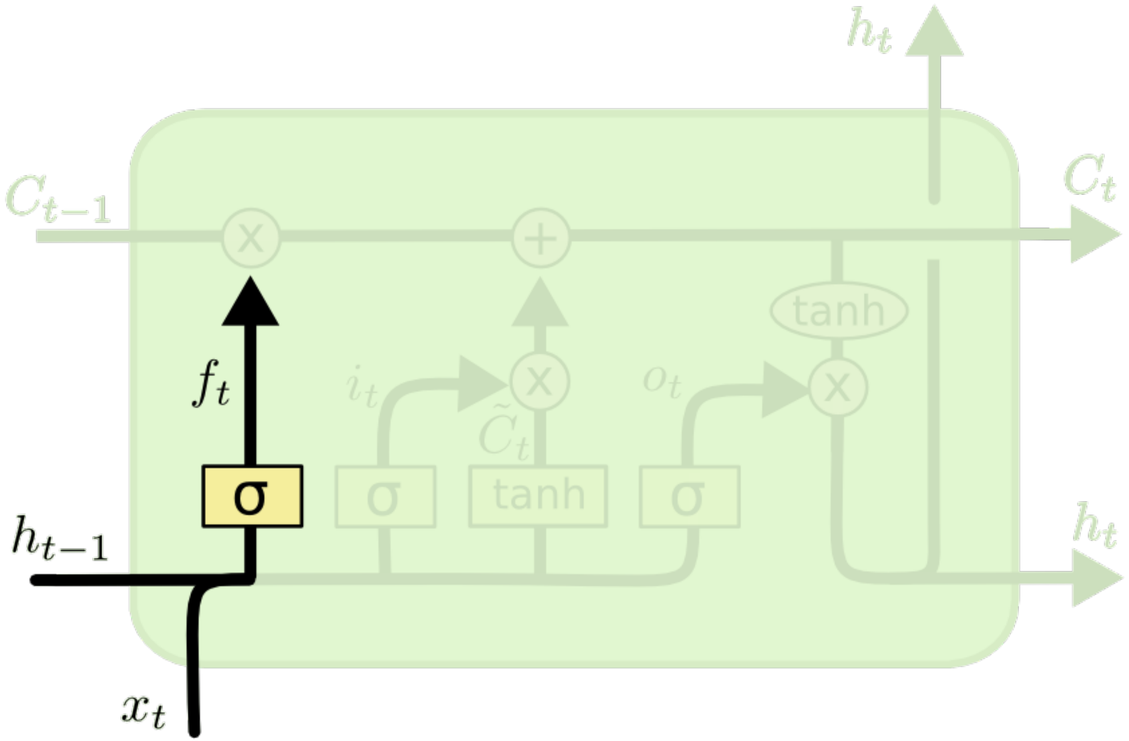
$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$$

Long-Short Term Memory



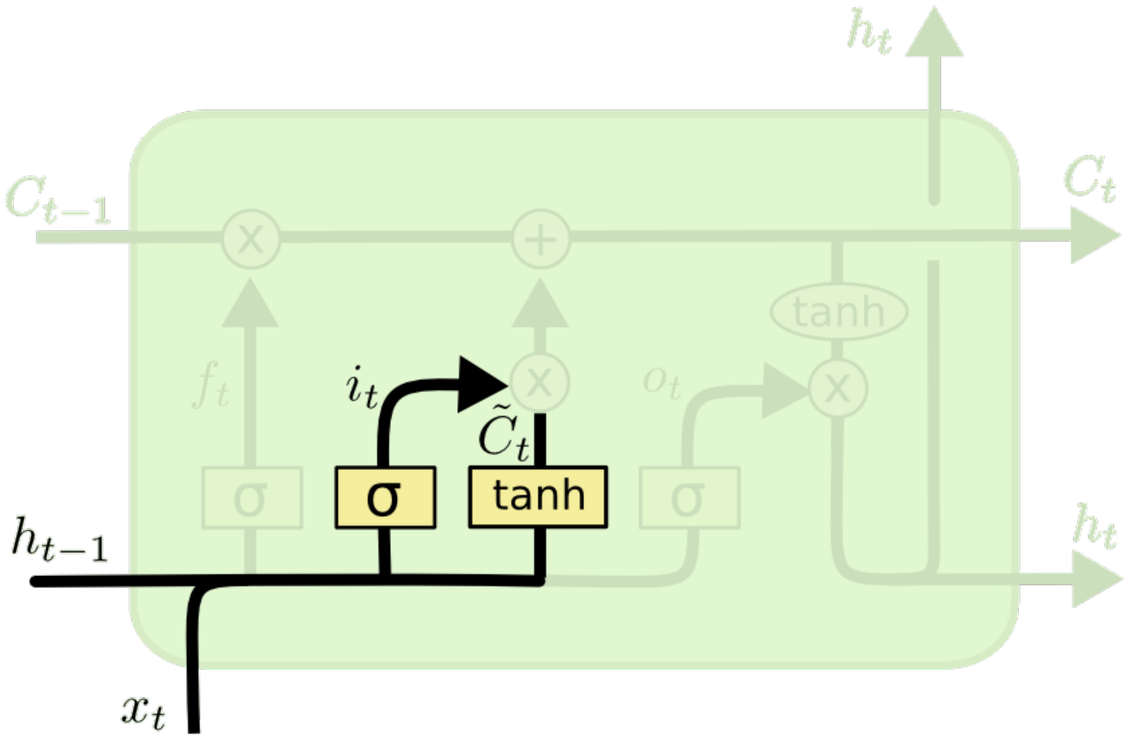
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long-Short Term Memory (forget gate)



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

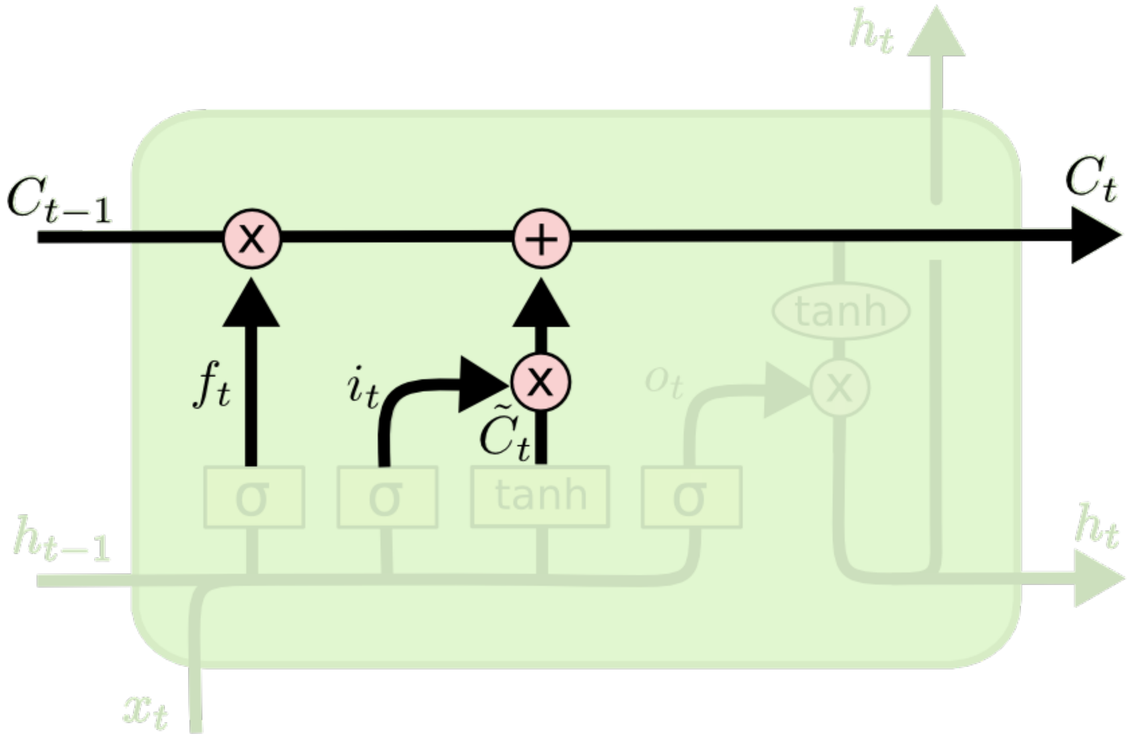
Long-Short Term Memory (input gate)



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

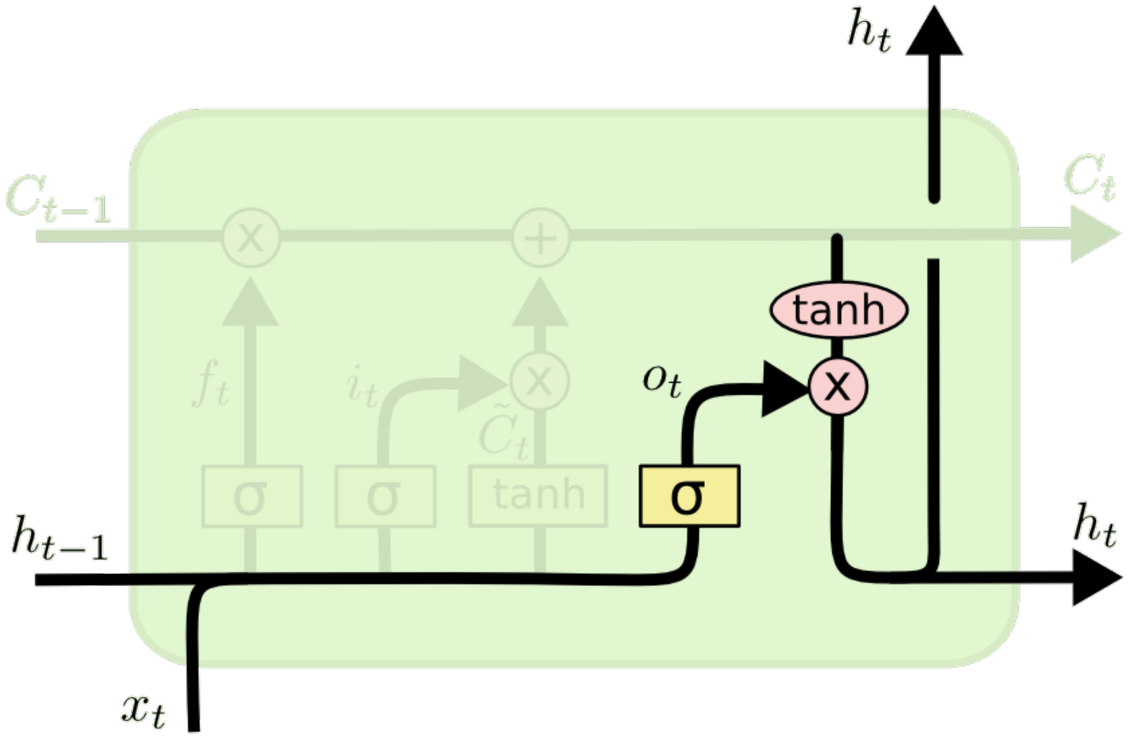
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Long-Short Term Memory (update)



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Long-Short Term Memory (output)



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$