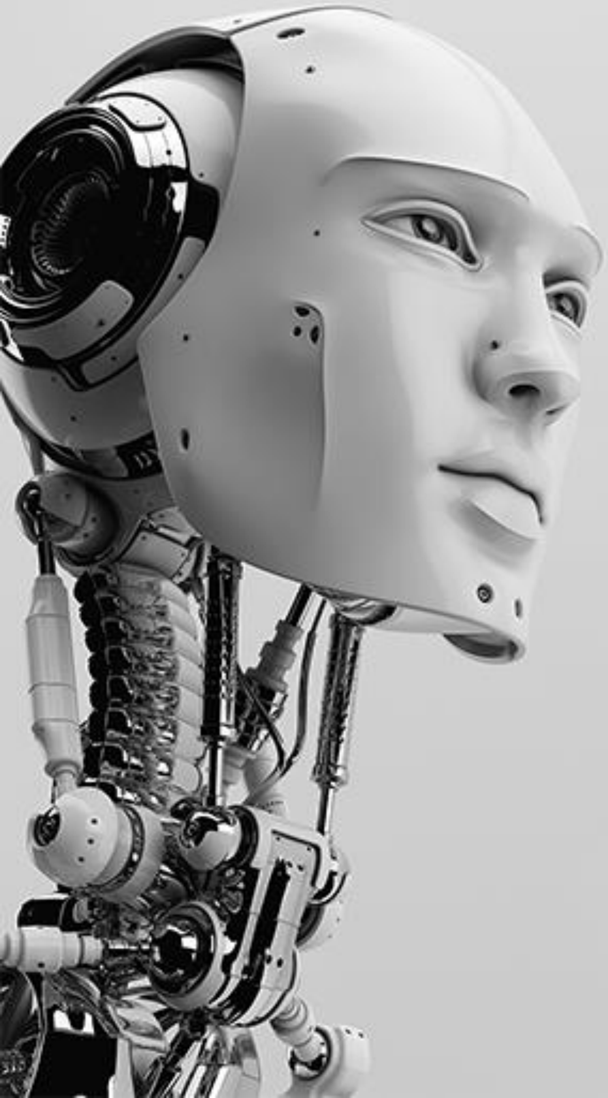


Introduction to Reinforcement Learning

Chris DeBellis

June 10, 2019

www.linkedin.com/in/chrisdebellis

My Journey to AI and Deep Learning

Georgia Tech College of Engineering

IBM

YAHOO! SEARCH

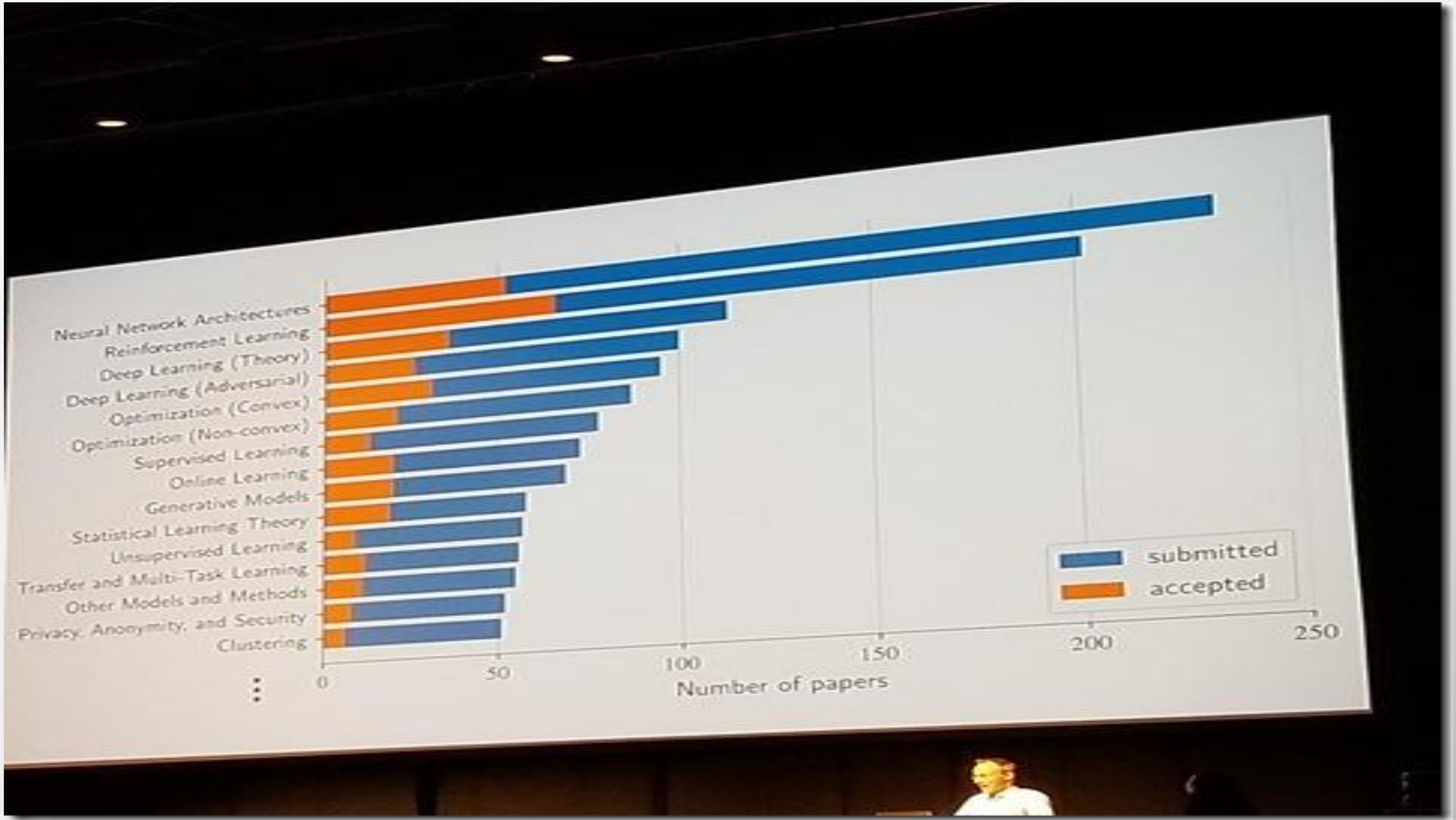
Georgia State University

Honeywell
THE POWER OF CONNECTED

rAI relationalAI

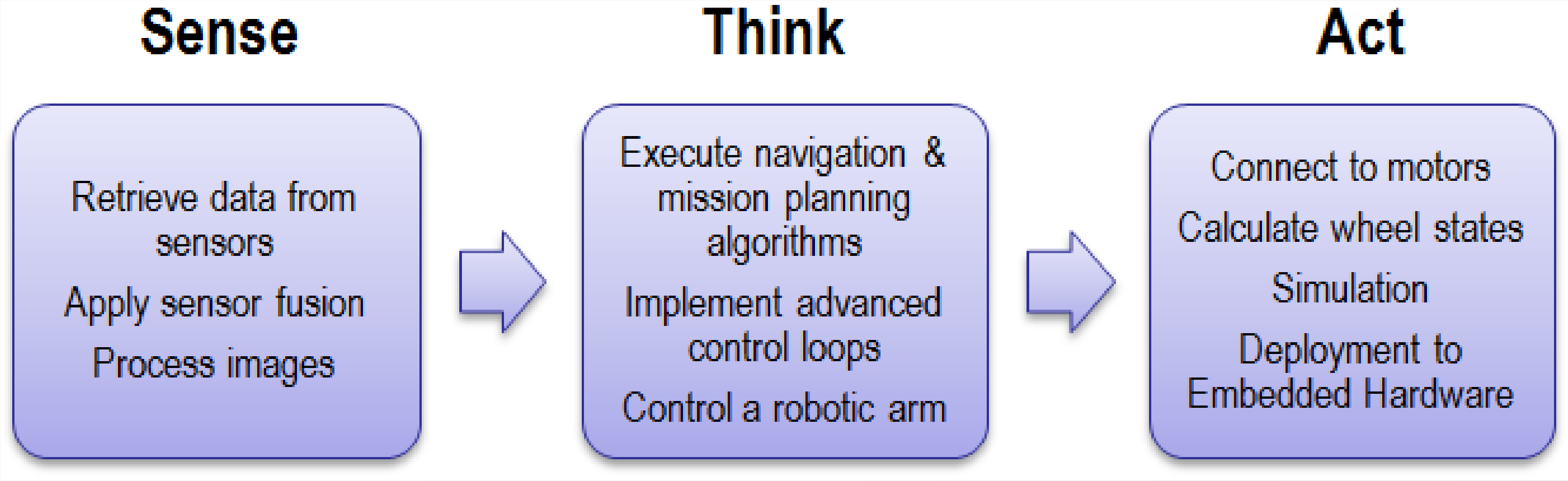
www.linkedin.com/in/chrisdebellis
https://changelog.com/practicalai/11

Reinforcement Learning Is a Hot Research Area



Source: The 35th [International Conference on Machine Learning](#) (ICML) - 2018

Deep Learning Trends for Robotic Functions

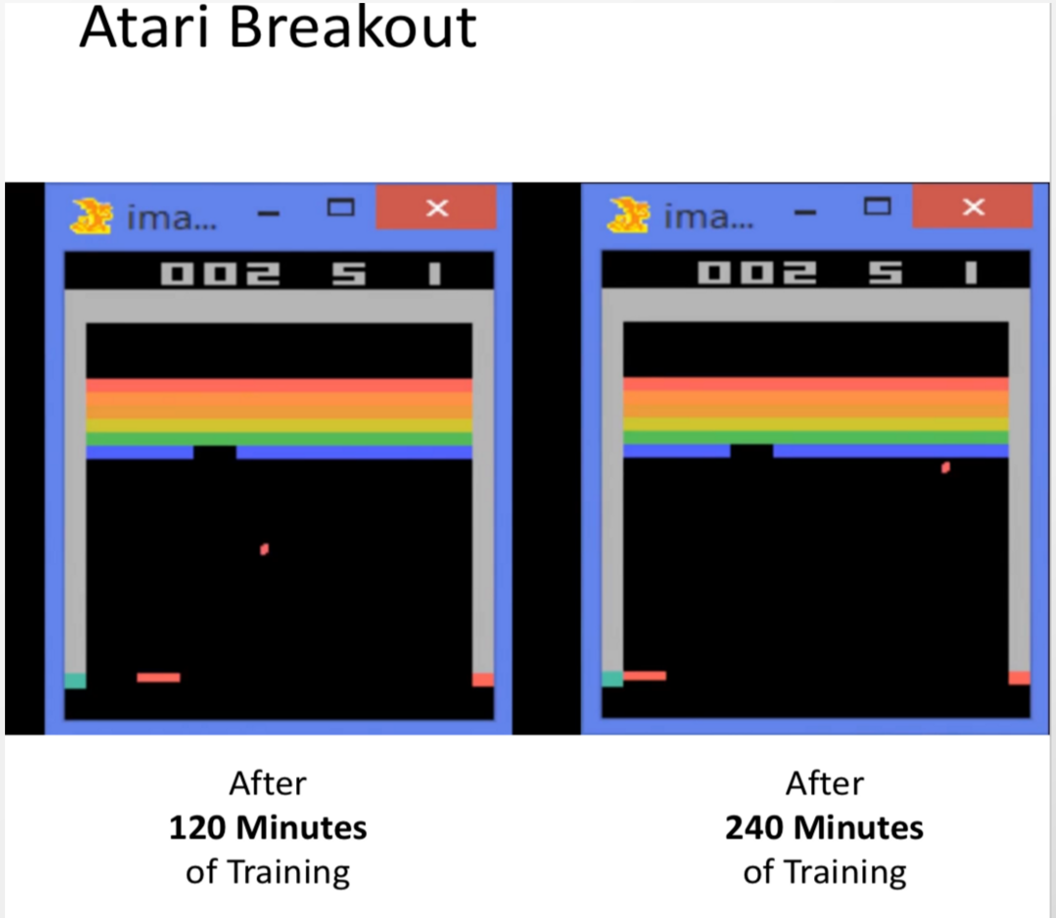


Perception Models (scene understanding and sensor fusion)

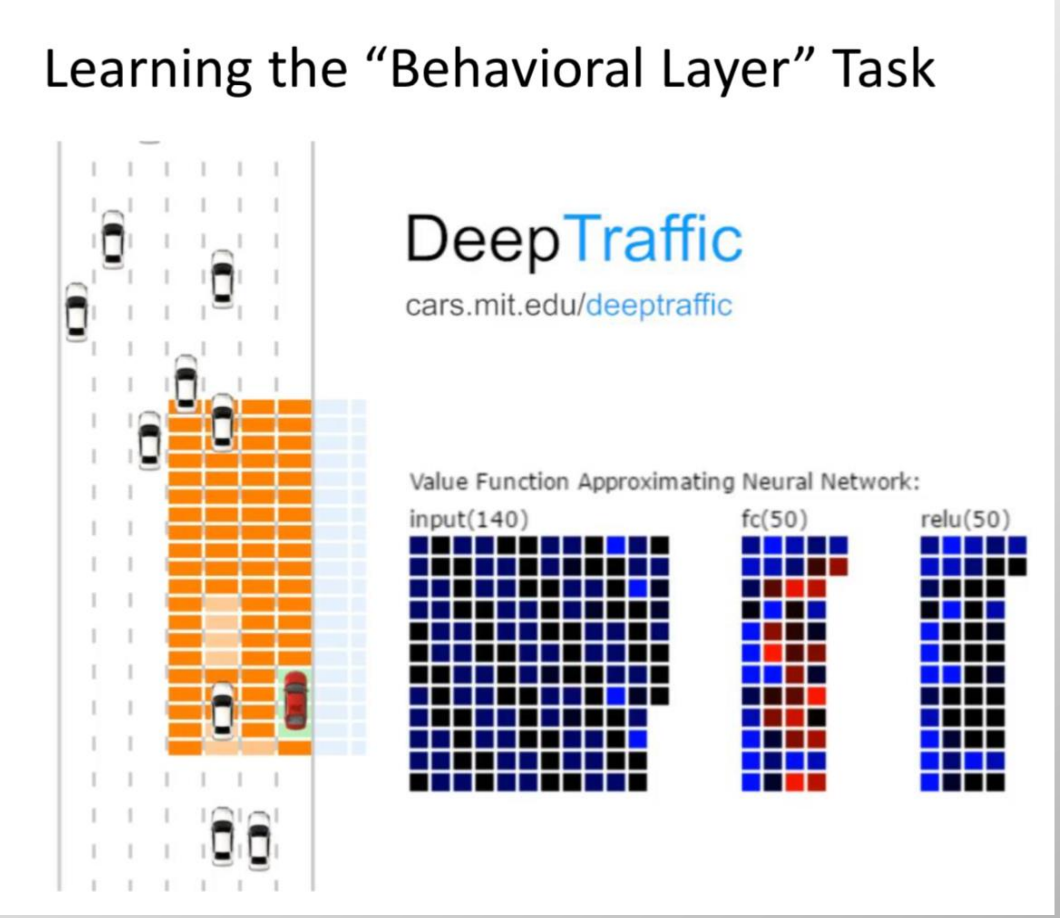
Reinforcement Learning Models (choose best action)

Data Collection for Model Training

Reinforcement Learning Use Cases

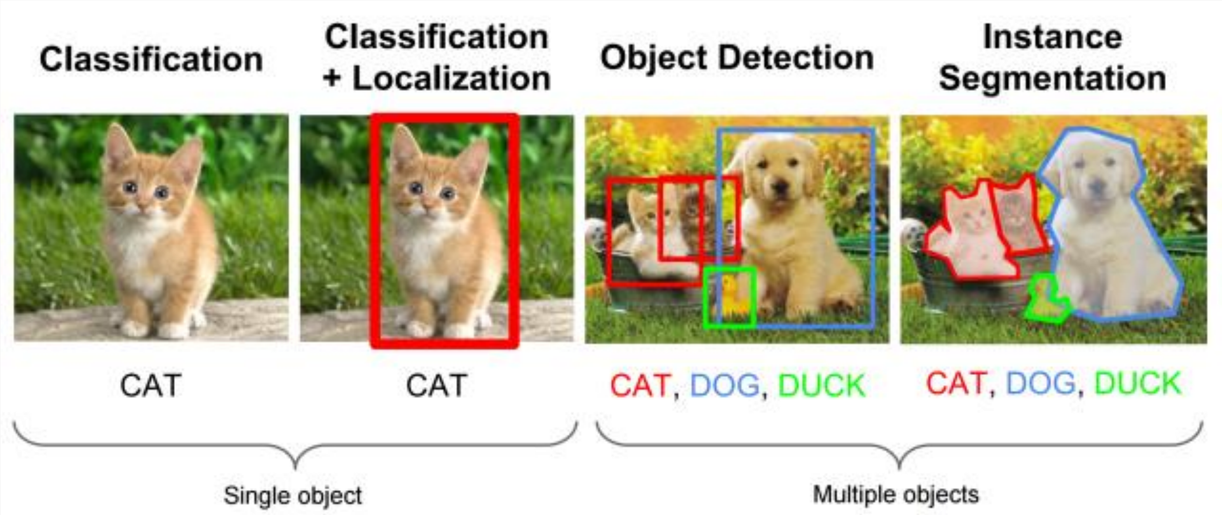


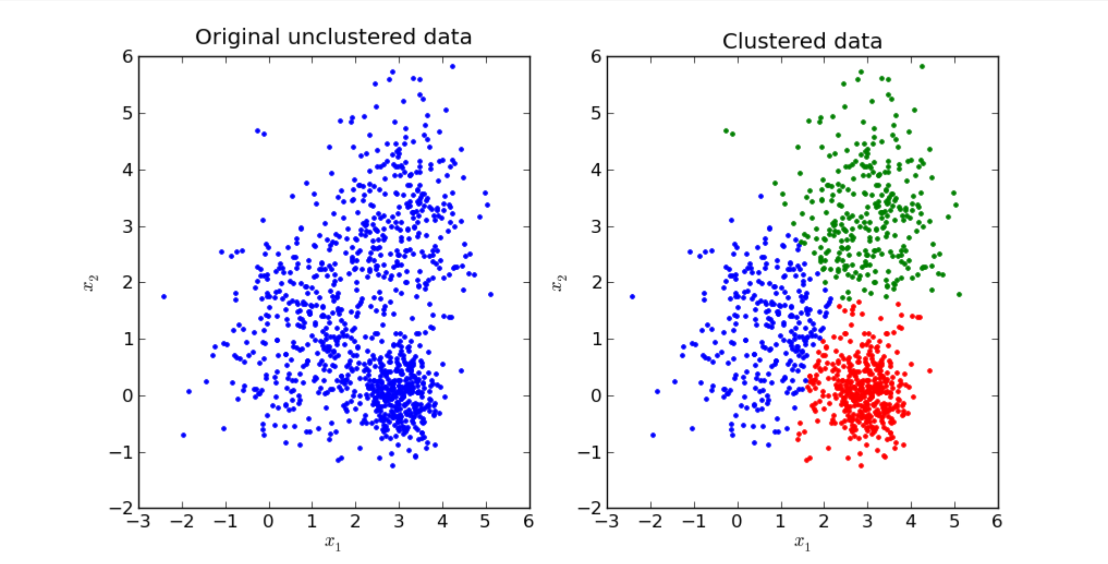
Games – Atari, Doom, Go, etc.



Autonomous Systems – self driving vehicles, robotics, etc.

Machine Learning Approaches





Supervised Learning

- Classification
- Regression
- Deep Learning

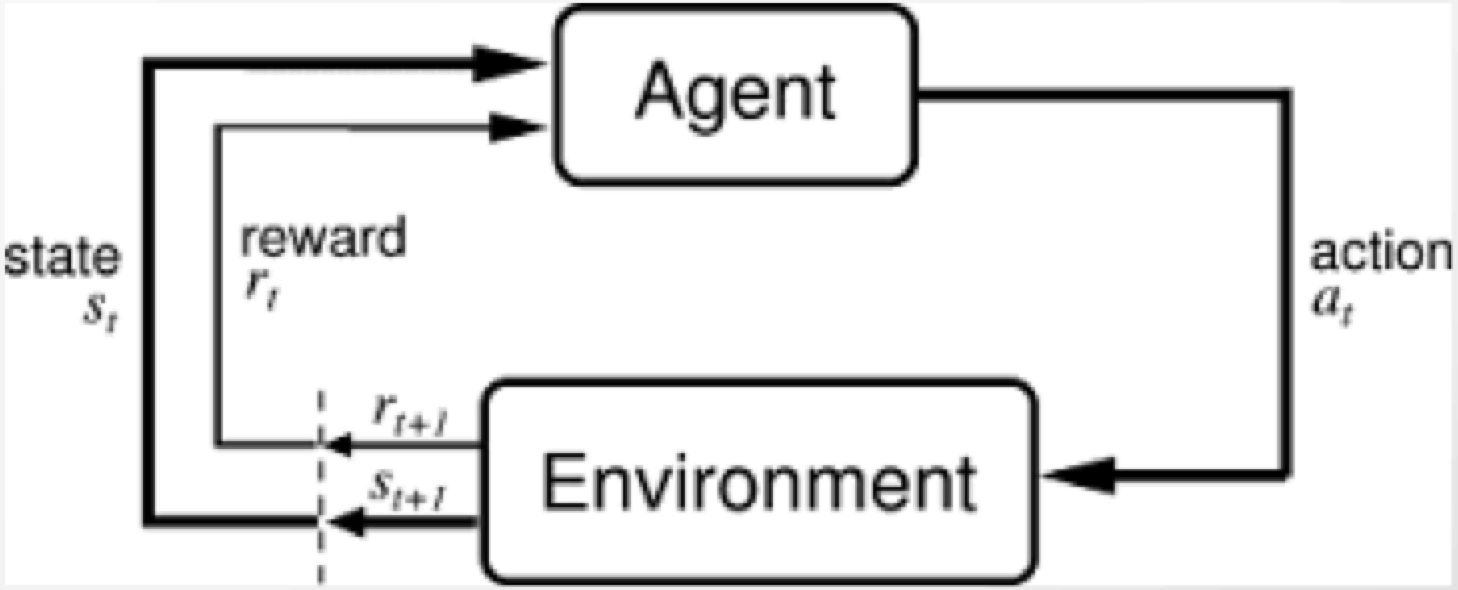
Requires ground truth (i.e. labeled data)

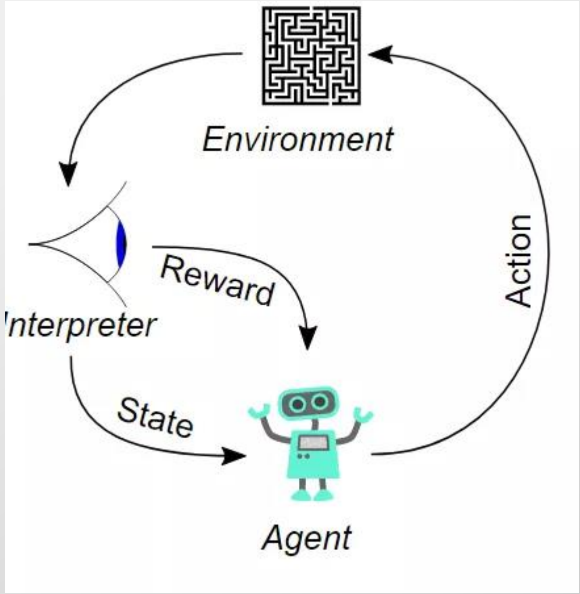
Unsupervised Learning

- (e.g. Clustering)

Group by similarity of features
No ground truth needed, but cannot measure accuracy, however can visualize it

Reinforcement Learning = Experiential Learning





Source: Wikipedia

- Agent (robot, game bot, etc.) operates in an environment
- Agent takes actions, which results in a reward (or penalty) and a new state
- Maximize the rewards (short or long term)
- May or may not use deep learning (probably does except for simple environments)

Reinforcement Learning

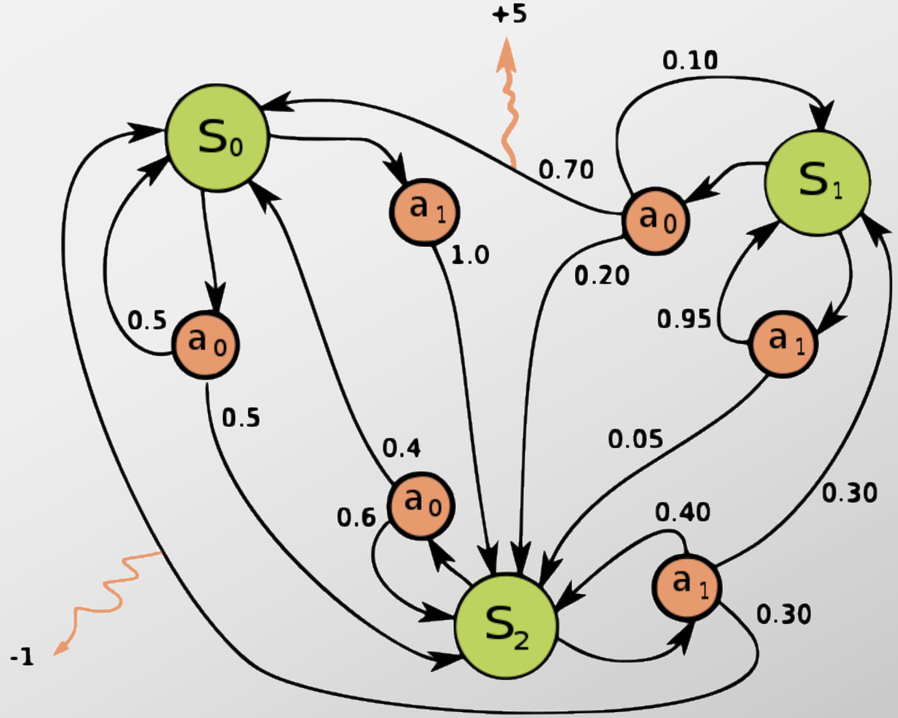
Markov Decision Process

S = set of all states
A = set of all actions
P = probability of taking action A when in state S
R = immediate reward for taking an action
Model = transition probabilities for the environment

Can be implemented as a table, array, list, etc. of [state][action] pairs

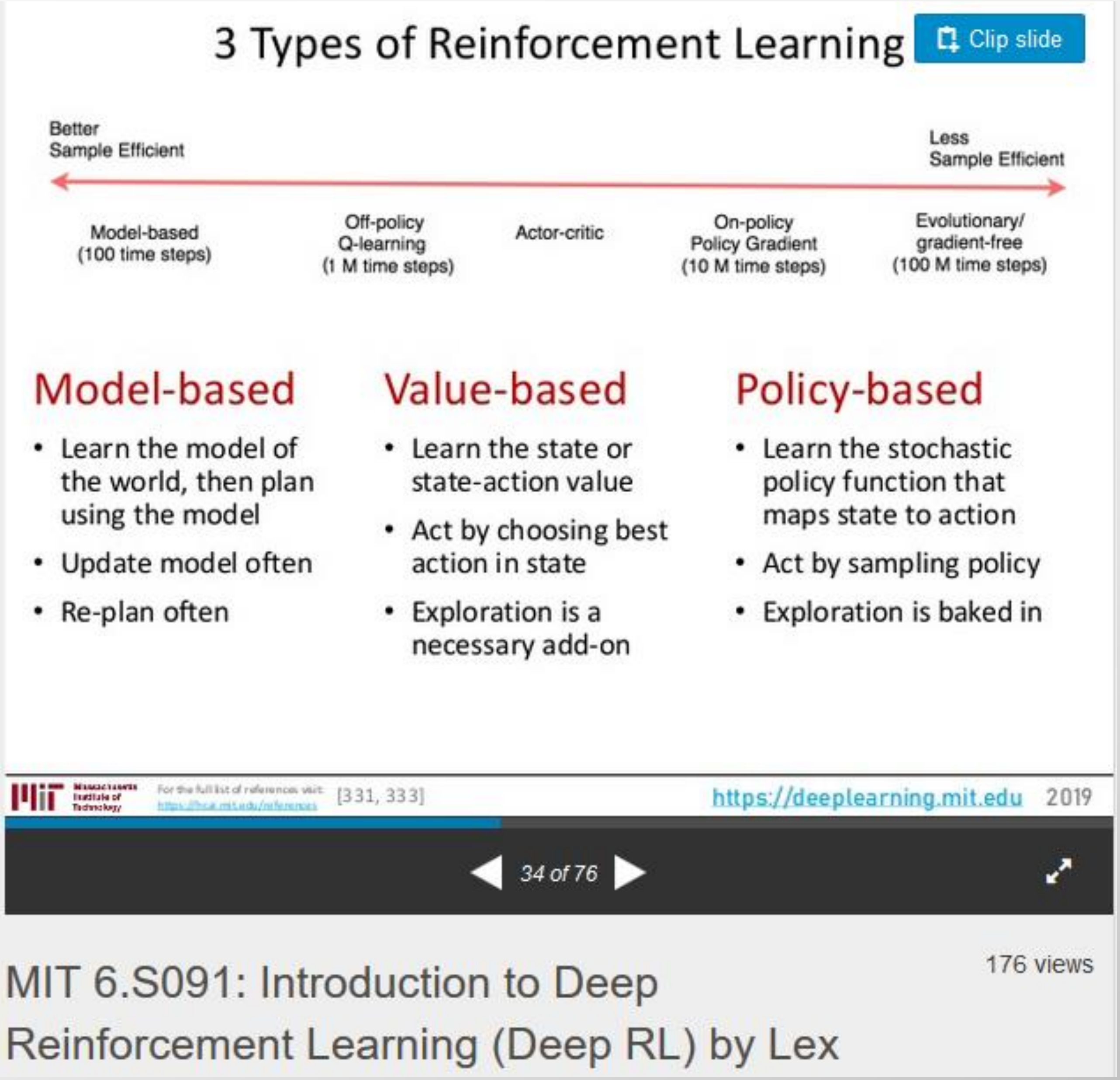
S_0 A_0
S_0 A_1
…
S_0 A_N

S_1 A_0
S_1 A_1
…
S_K A_0
S_K A_1
…
S_K A_N

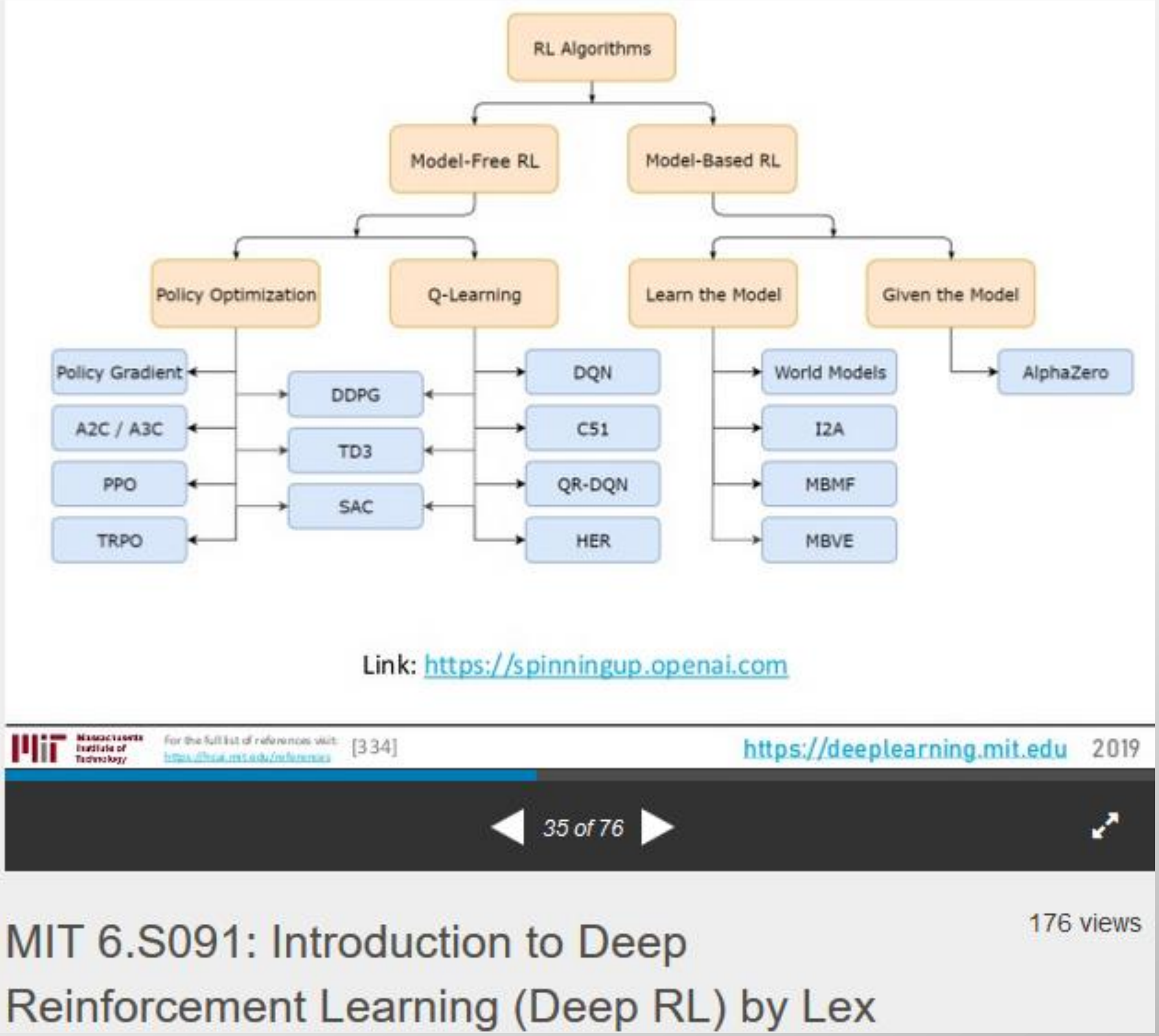


For this model, the Q Table would contain 6 values (3 states * 2 actions)

Types of RL


3 Types of Reinforcement Learning
Clip slide
Better
Sample Efficient
Less
Sample Efficient
Model-based
(100 time steps)
Off-policy
Q-learning
(1 M time steps)
Actor-critic
On-policy
Policy Gradient
(10 M time steps)
Evolutionary/
gradient-free
(100 M time steps)
Model-based
• Learn the model of the world, then plan using the model
• Update model often
• Re-plan often
Value-based
• Learn the state or state-action value
• Act by choosing best action in state
• Exploration is a necessary add-on
Policy-based
• Learn the stochastic policy function that maps state to action
• Act by sampling policy
• Exploration is baked in
[331, 333]
https://deeplearning.mit.edu
2019
34 of 76
MIT 6.S091: Introduction to Deep Reinforcement Learning (Deep RL) by Lex
176 views

Types of RL


RL Algorithms
Model-Free RL
Model-Based RL
Policy Optimization
Q-Learning
Learn the Model
Given the Model
Policy Gradient
A2C / A3C
PPO
TRPO
DDPG
TD3
SAC
DQN
C51
QR-DQN
HER
World Models
I2A
MBMF
MBVE
AlphaZero
Link: https://spinningup.openai.com
[334]
https://deeplearning.mit.edu 2019
35 of 76
176 views
MIT 6.S091: Introduction to Deep
Reinforcement Learning (Deep RL) by Lex

On-Line vs. Off-Line Learning

- On-line Learning
  - Put the robot / agent in the environment to learn
  - Is the environment safe? Can the robot take a fatal action?
  - May be slow to learn
    - Robots are generally slow moving.
    - Resets are probably also slow.
- Off-line Learning
  - Learn using a simulator to simulate the environment
  - Transfer the learned policy to the agent and let the agent run in the real environment
  - Difficult to build a simulator that accurately reflects the physics of the real world
  - There will likely be a delta between the learned best policy of the simulated environment and the actual best policy for the real environment.

Q-Learning

- V = value (utility / usefulness of being in a given state)
  - Closer to a large reward is more valuable
- Q(*s, a*) = Q-value is the value of taking action *a* at a given state *s*
- Q learning = use Monte Carlo simulation to learn the (*s, a*) values

  Also can infer the transition probabilities
- Policy (∏) maps best action to take for every state
  - By learning Q we infer the policy (don't directly learn the environment / model ➔ model free)
- Learn by iterating (Monte Carlo Simulation)

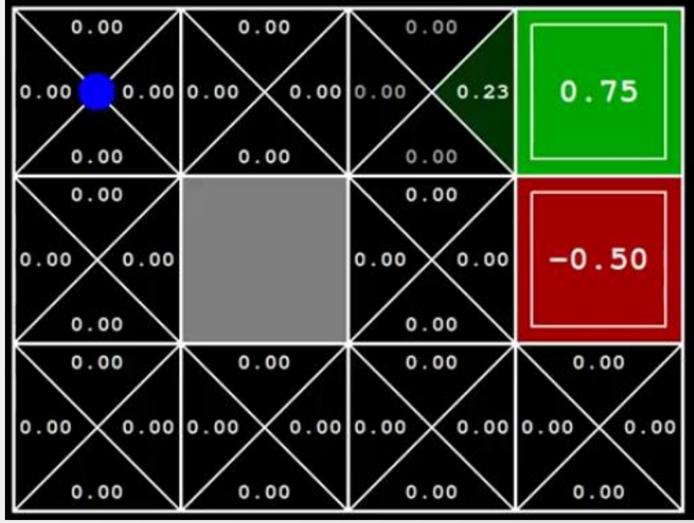
Bellman Equation

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

- Only one state-action pair is updated per action
- Referred to as Bellman Backup

https://skymind.ai/images/wiki/q_learning_equation.png

Q Learning Demo – Grid World



- Agent (blue dot) moves around the grid
  - Only 4 possible actions – move up, down, left, right
- The only states are the position of the robot in 3x4 grid
- 2 possible termination states

- Q table needed is small (< 48 values for the 12 states x 4 actions)

- Only one Q-value updates when you take an action and move to new state
- Q values converge eventually BUT you need sufficient samples
  - Note changes to Q values in bottom left @0:49 for left and up actions
- Nondeterministic model / environment
  - with some probability your agent moves in the wrong direction (robot fails to complete the action and ends up in unexpected state) – refer to video @0:11

Grid World Demo

https://www.youtube.com/watch?v=AMnW-OsOcl8

alpha (learning rate)	0.5	
gamma (discount rate)	0.9	
Reward	0	
Current Q	Next Q	New Q
0	0.5	0.23
0.23	0.75	0.45
0	0.23	0.10
		0.00
Exit Condition		
0	1	0.50
0.5	1	0.75
0.75	1	0.88

Q Learning Concepts

- What is the value of a state?

  See end of video

  $V(s) = Max_a Q(s,a)$

What action do we take?

  $a(s) = argmax_a\ Q(s,a)$

What is the optimal policy Π*?

  At any state, take the action that yields greatest Q value (total reward according to Bellman equation)

Rewards Affect Learned Behavior

- How can we encourage the agent to finish as quickly as possible?
- What value should the reward be?

- If the reward is -1, what would the agent do when it was in the bottom right state?

- What if we want to encourage survival in a game?

- But what would the agent do if it is at grid coordiante (1,3) and the reward is +1?

$\mathcal{E}$-Greedy (Epsilon Greedy) Algorithm

- Exploration vs Exploitation

- During training:
  - With some probability (1-ε) take the optimal action known at that time (exploitation)
  - Else with probability ε take some other action (exploration)
    - Several methods exist for determining which other action to take

- Why do we need exploration?
- What should we do in production?

Follow the learned policy Π^* → No exploration!

Q-Learning Example – OpenAI Gym Taxi

https://learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/

6 Actions

- south
- north
- east
- west
- pickup
- dropoff

500 States

25 grid locations
4 destinations / pickup-up points
5 possible passenger locations
(4 pickup points + 1 inside taxi)

How large does the Q table need to be?

How do we differentiate each state?
(taxi row, taxi col, passenger loc, dest)

Reward=-1 or -10 or +20. Why -1?

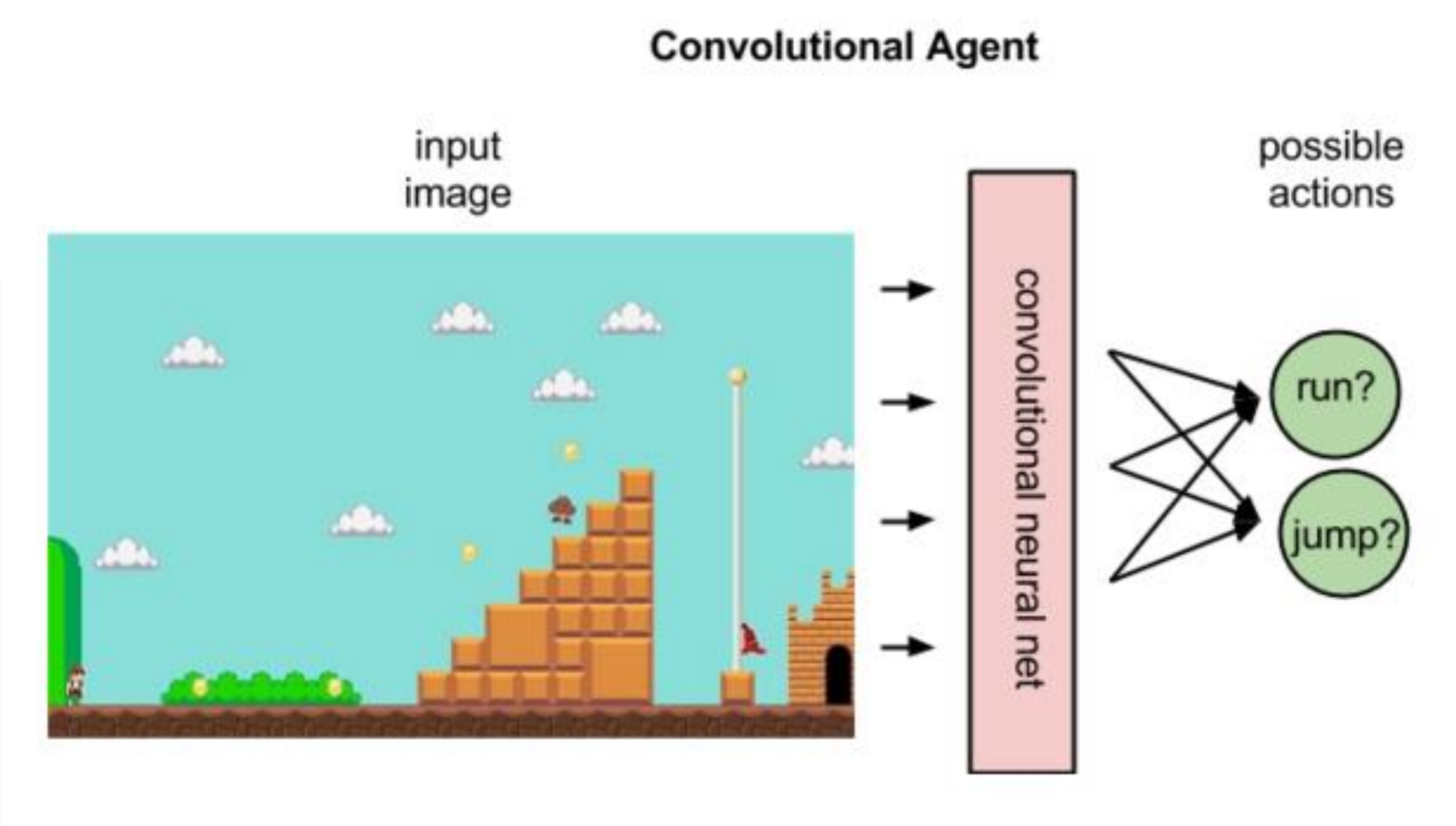
Pickup passenger at Y and drop off at R

Problems with Q-Learning

- Large State Space
  - Needs huge amount of memory for all the (s,a) pairs
  - Need to visit all of the states and take each action multiple times to ensure convergence → takes too long
  - Need to discretize continuous values (velocity, position, rotation, etc.)

- How many states are there in:
  - A tic-tac-toe game? 3^9 = 19,683 including invalid states (upper bound)
  - A game of chess? Estimated to be ~7.7×10^{45}
  - A self driving car? Depends on level of accuracy of velocity, position, etc.
    More states if you round velocity to nearest tenth vs MPH

Deep Q-Learning to the Rescue (Kinda)



Use a neural network to map states to actions rather than build a table for every state.

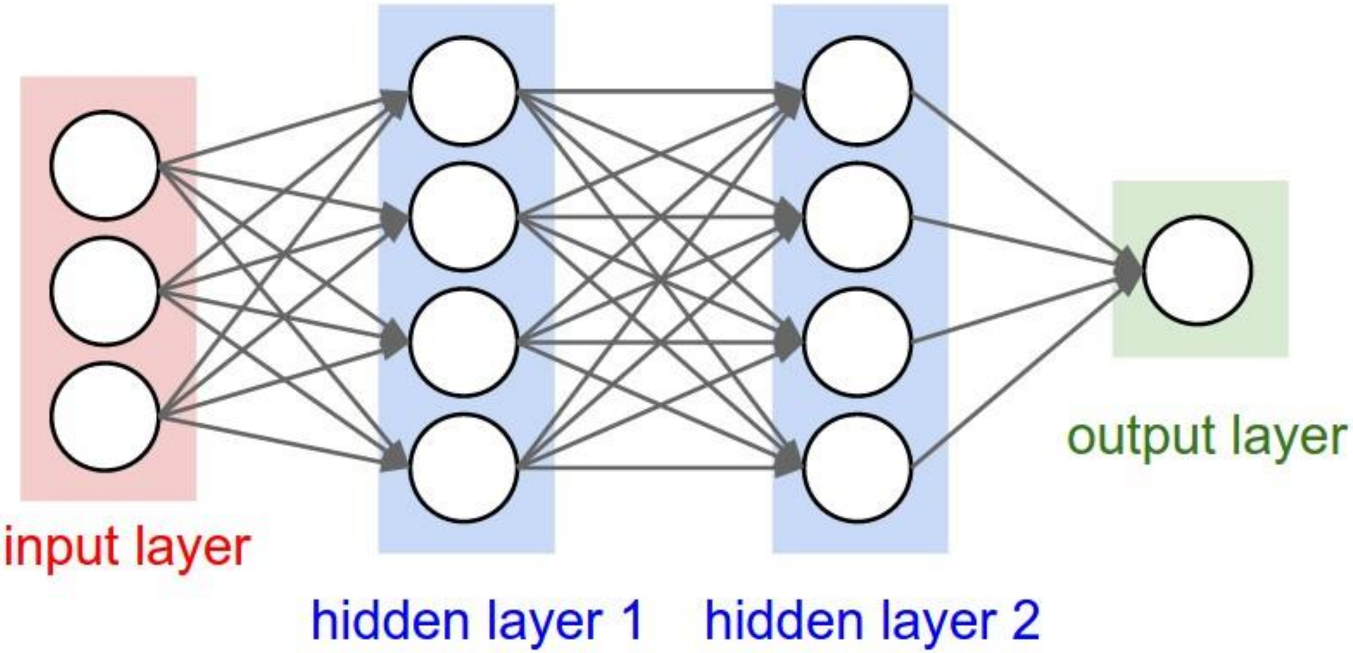
Useful for:

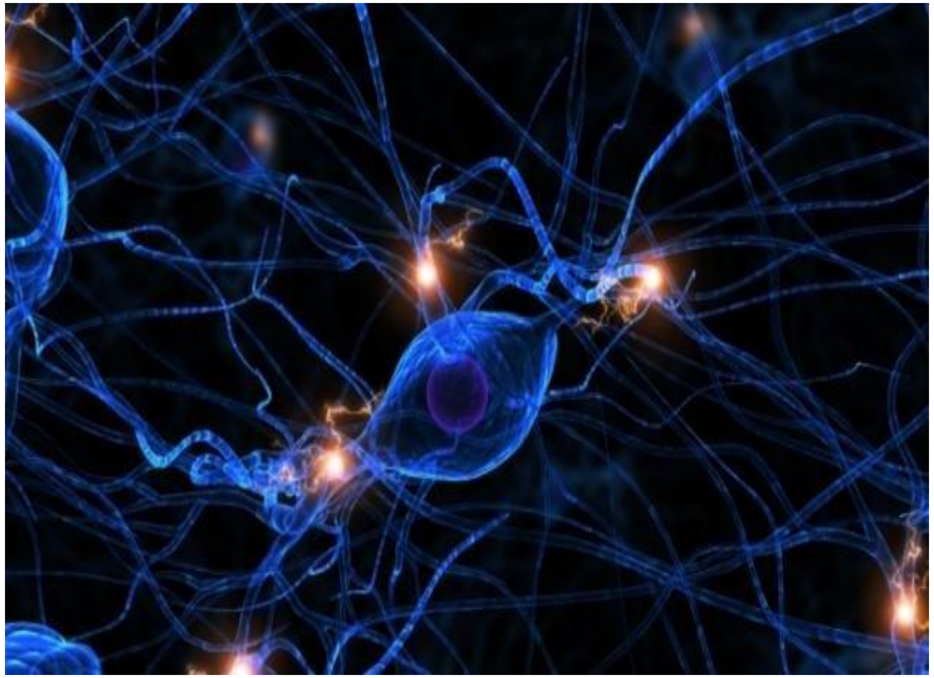
- Very large state space where it is impractical to visit every (s,a) pair
- Continuous state space (avoids the need to discretize states)

Neural network is a universal approximation function

https://skymind.ai/wiki/deep-reinforcement-learning

Feed Forward Neural Network (MLP)





https://nutricaobrasil.wordpress.com/esclerose-neuronios-cerebro-20110608-size-62-2/

- Layers contain nodes that perform mathematical functions on a set of inputs
- Activation functions (represented as circles in the image) bound the output range and add non-linearity
- Outputs of one layer are inputs to the next layer (feed forward)
- Deep networks have more layers
- Motivation derived from our understanding of human brain function

Deep Q-Network (DQN)

What do we need to do supervised learning?
- Ground truth so we can calculate the error of the predictions
- Objective function (loss) to minimize

$$L_i(\theta_i) = \mathbb{E}_{s,a,s',r\sim D}\left(\underbrace{r + \gamma \max_{a'} Q(s', a'; \theta_i^-)}_{\text{target}} - Q(s, a; \theta_i)\right)^2$$

New predicted value (target) – original Q value (before taking the action)

Note: this equation used for L_2-norm loss with MSE. Could use L_1-norm loss with MAE instead (Absolute value of the difference).

Problems Implementing the Neural Network

1. Need iid (independent, identically distributed) samples
   Sequential play inherently makes samples correlated
   Network will otherwise forget past learning and optimize for newest samples
2. Ground truth Q changes with each iteration and target depends on Q

   → Chasing (optimizing) a non-stationary target

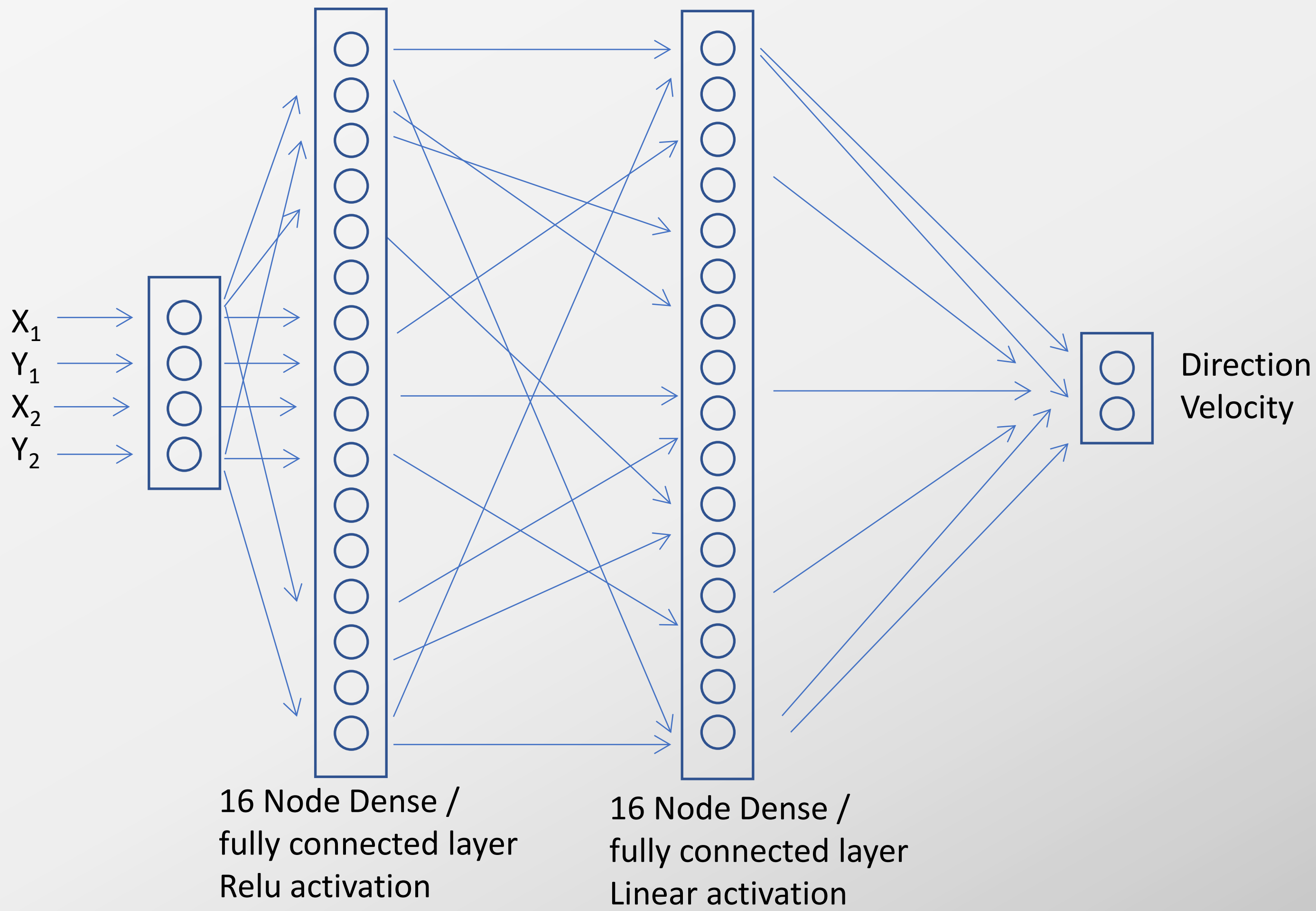
Experience replay buffer

- Save all (s,a,r,s') values in memory
- Train the network on randomly selected samples from the buffer in batches
  - Eliminates sequential correlation
  - Ground truth is static for that batch so neural net can converge

CartPole Demo

- https://www.analyticsvidhya.com/blog/2017/01/introduction-to-reinforcement-learning-implementation/
- Video https://youtu.be/XiigTGKZfks

DQN Architecture for CartPole Example



Questions