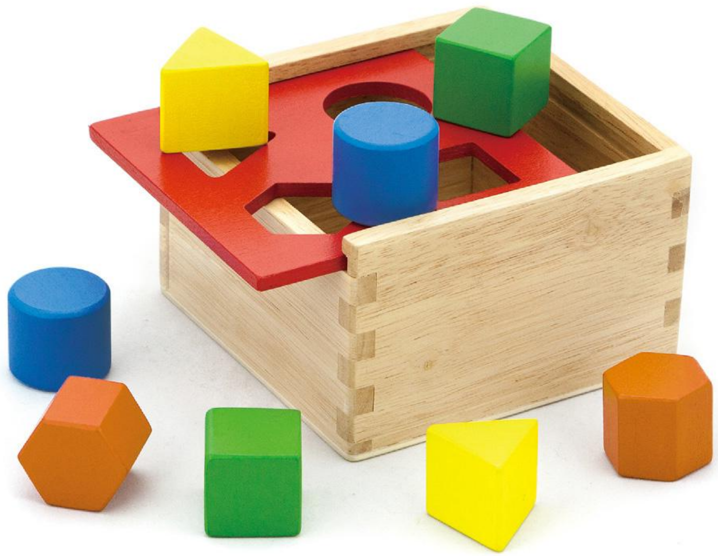
What Would it Take to Train an Agent to Play with a Shape-Sorter?



Feryal Behbahani

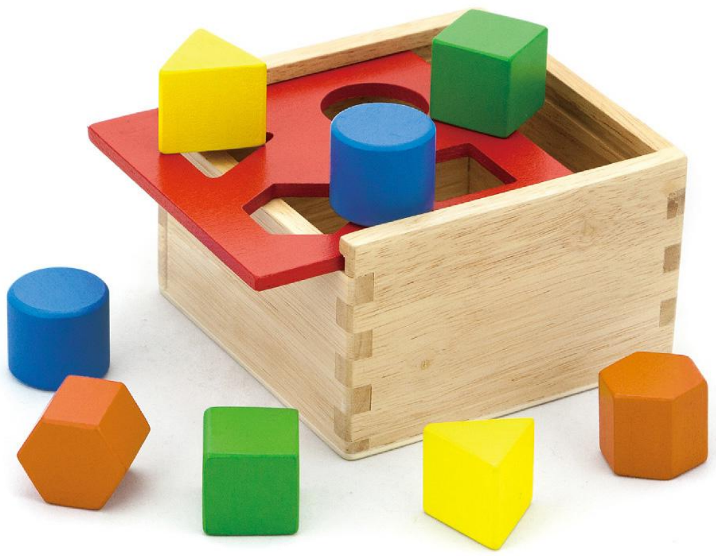
Imperial College London

Shape sorter?

- Simple children toy: **put shapes in the correct holes**
- Trivial for adults
- Yet children cannot fully solve until 2 years old (!)

Requirements

- Recognize different shapes
- Grasp objects and manipulate them
- Understand the task and how to succeed
- Mentally / physically rotate shapes into position
- Move precisely to fit object into hole



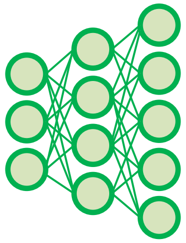
How to do it?

- Classical robotic control pipeline approach

Observations → State estimation → Modeling & prediction → Planning → Low-level control → controls

- Deep robotic end-to-end learning

End-to-end learning

Observations …………  ………… controls

Using simulations as a proxy

- How many samples do we need to train a good behaviour?
  - Real robot/car: stuck to real time speed
  - MuJoCo simulator: up to 10000x real time

Udacity car simulator

Real Jaco arm

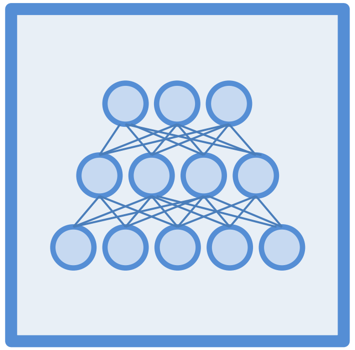
MuJoCo simulation

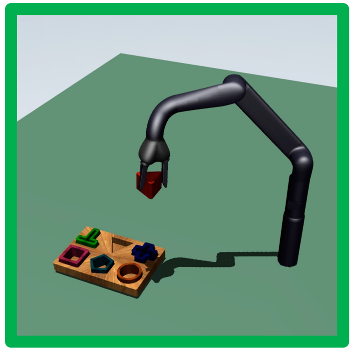
Finger tracking with CyberGlove synced with 3D reconstruction in MuJoCo

[Todorov et al., 2012 & Behbahani et al., 2016]

Deep Reinforcement Learning for control

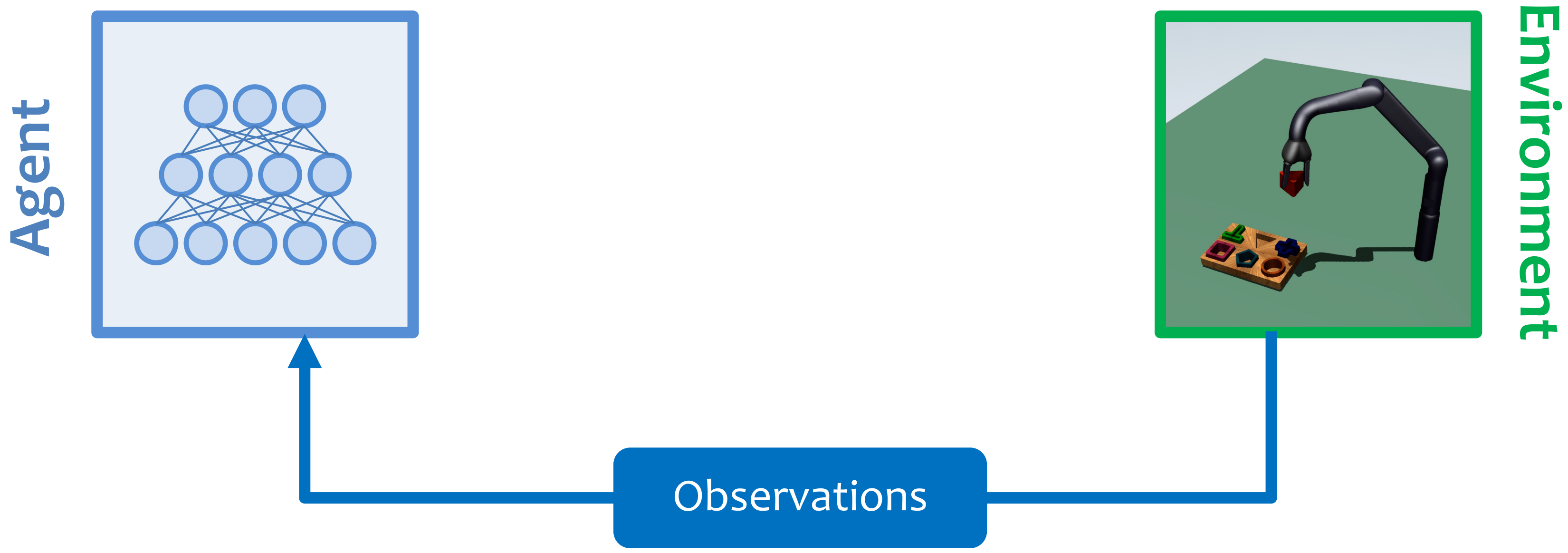
Agent



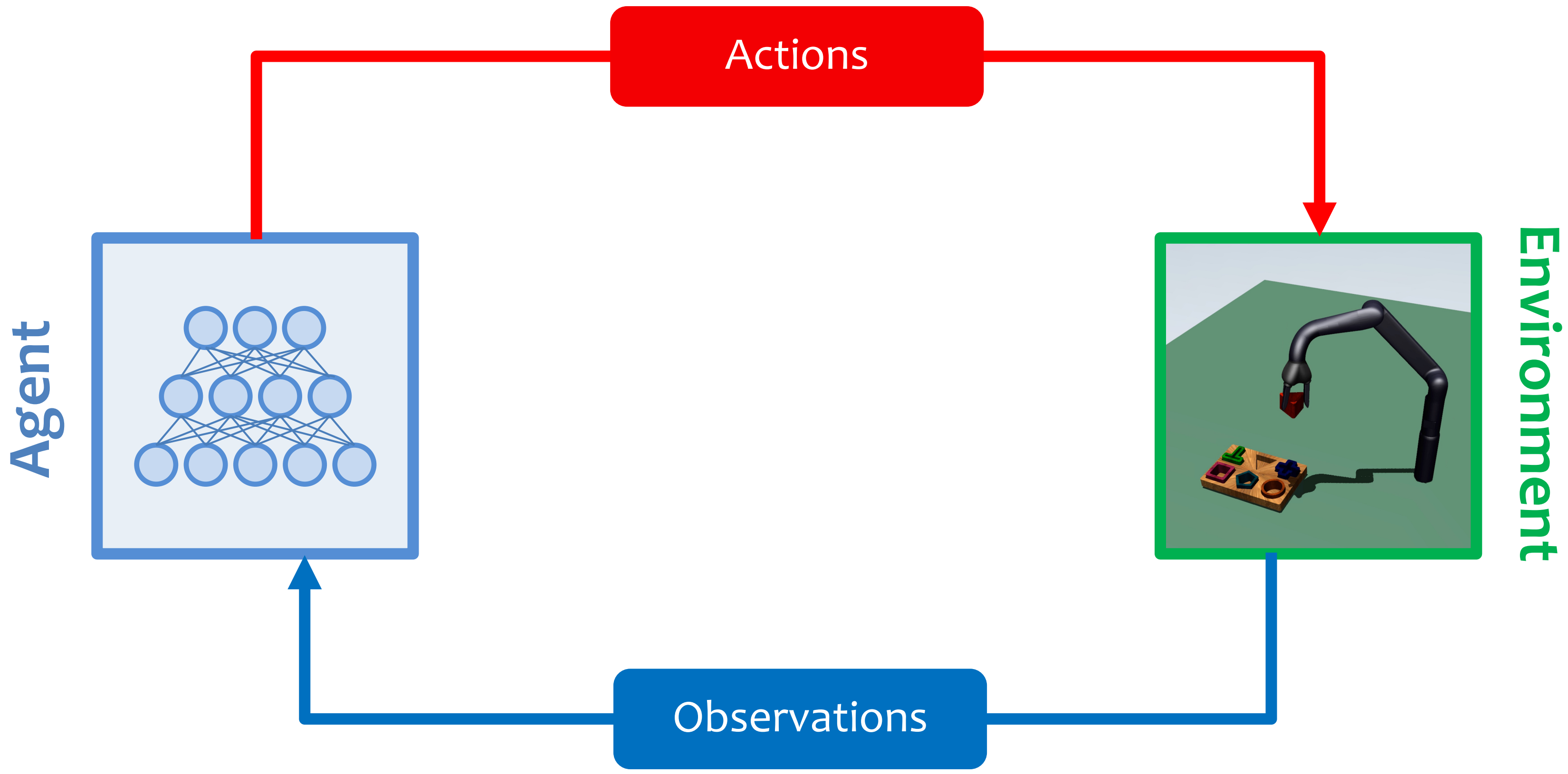


Environment

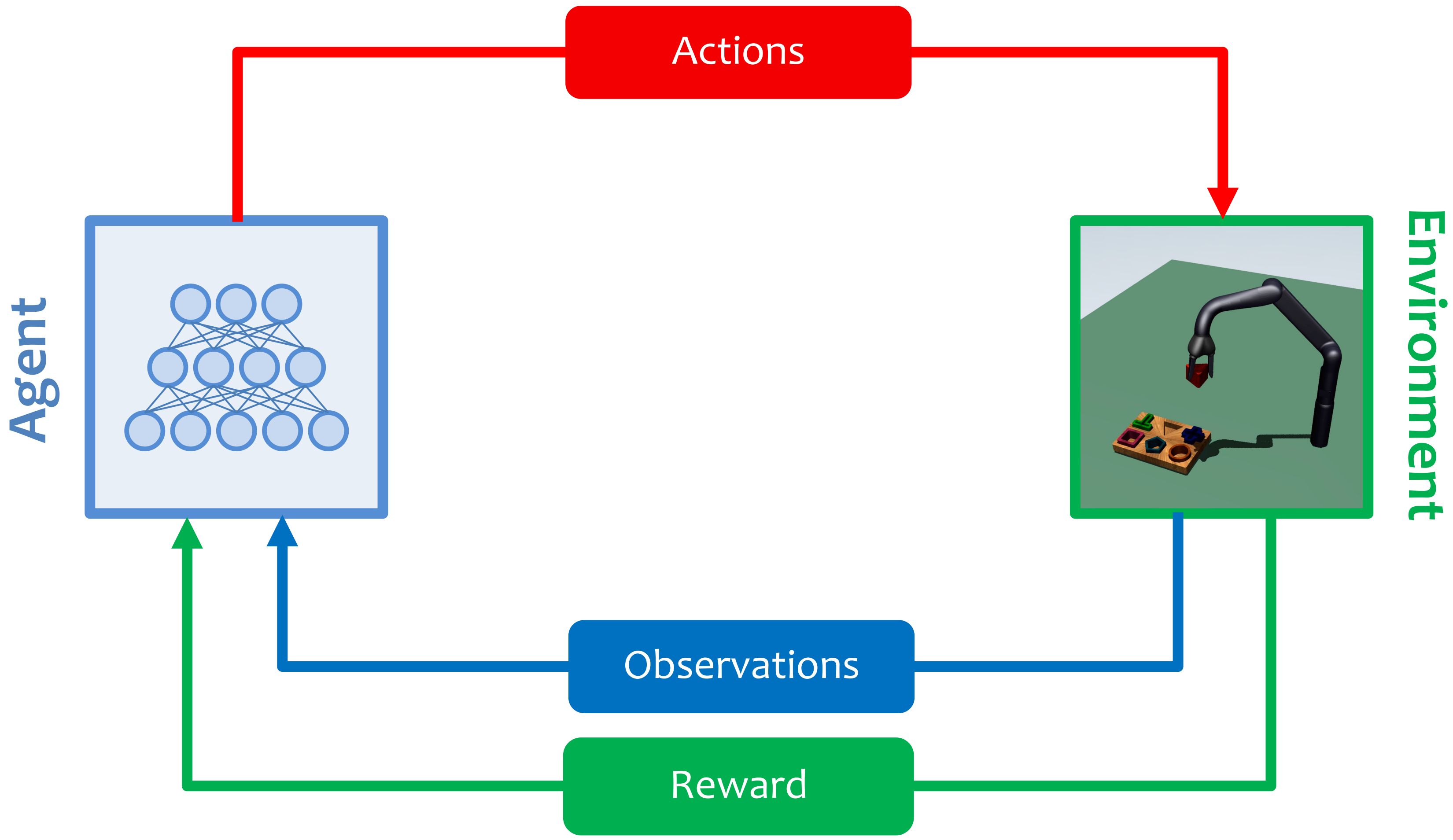
Deep Reinforcement Learning for control



Deep Reinforcement Learning for control

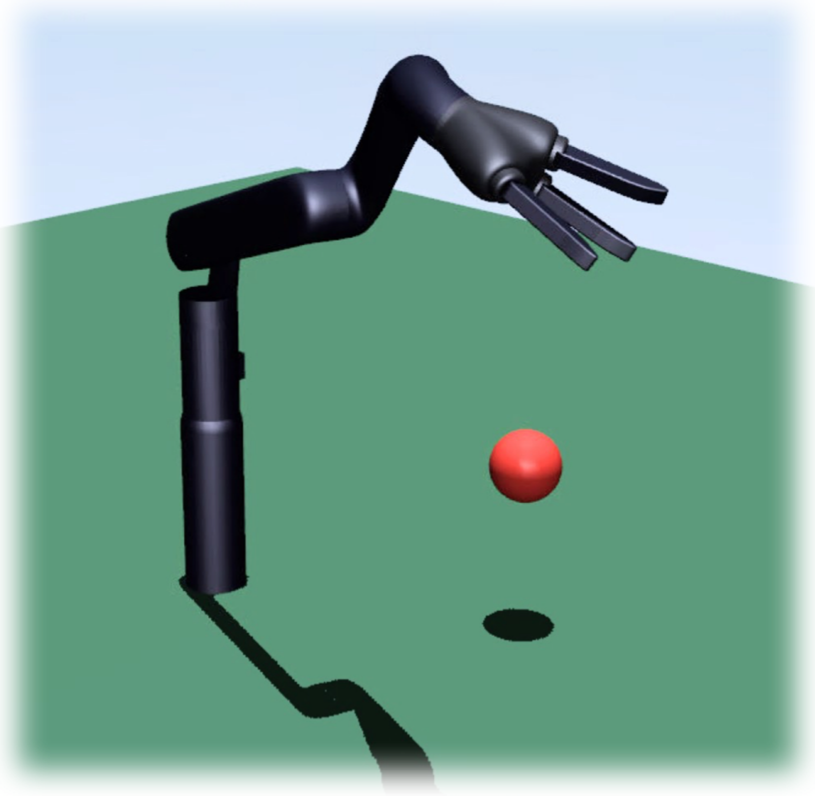


Deep Reinforcement Learning for control



Learning to reach

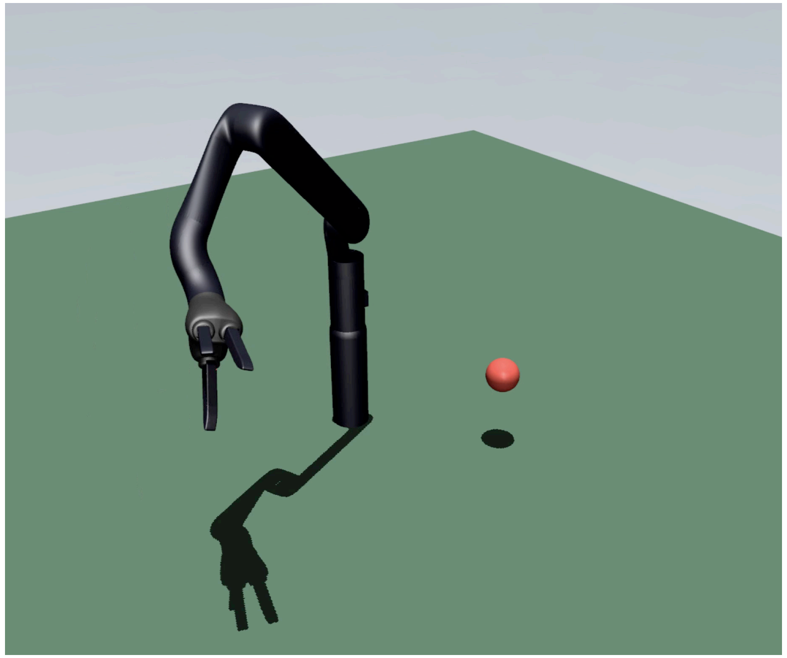
- Let's first try to reach to a target and grasp it.
- Should be able to do this regardless of object location

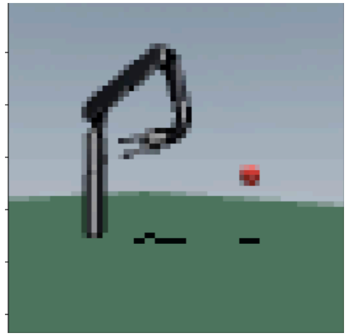


Task and setup

- **Reach red target**
  - Reward of 1 if target inside hand
  - Random position each episode
    40 x 40 x 40 cm
- **Observation space:**
  - Two camera views
- **Action space:**
  - Joint velocities
    9 actuators, 5 possible velocities

Random agent

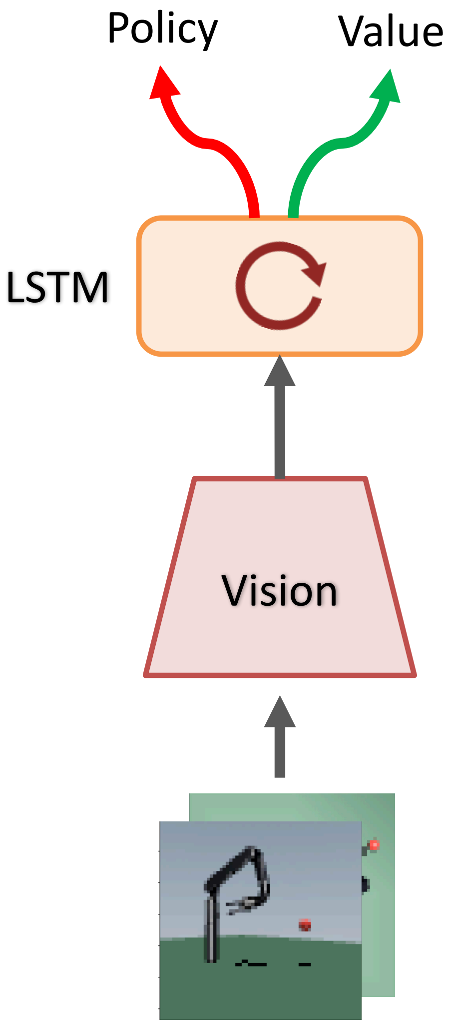




View 1

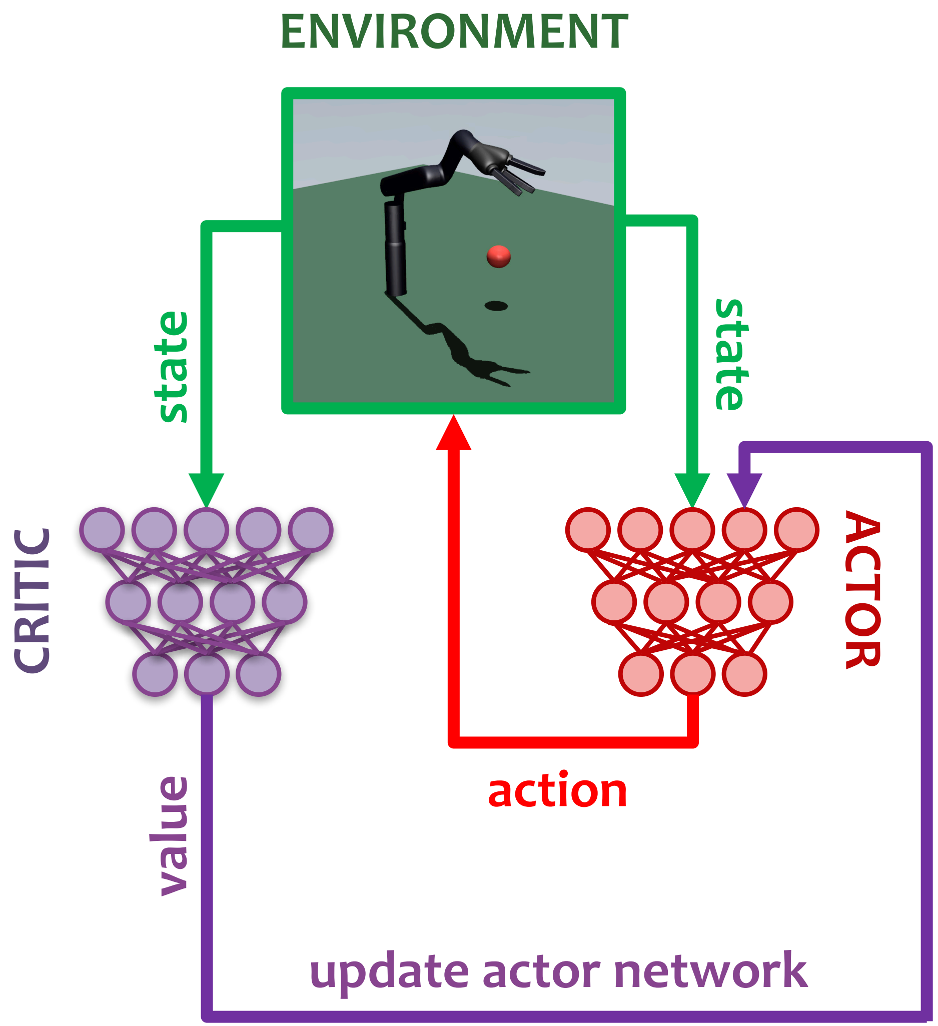
View 2

Agent architecture



- **Inputs:**
  - 64 x 64 x 6 channels
- **Vision**
  - ConvNet 2 layers
  - ReLU activations
- **LSTM** (recurrent core)
  - 128 units
- **Policy**
  - Softmax per actuator (5 values)
- **Value**
  - Linear layer to scalar

Asynchronous Advantage Actor-Critic (A3C)



Agent acts for T timesteps (e.g., T=100)

For each timestep t, compute

$$\hat{R}_t = r_t + \gamma r_{t+1} + \ldots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-t}\hat{V}(s_T)$$

$$\hat{A}_t = \hat{R}_t - \hat{V}(s_t)$$

Compute loss gradient:

$$g = \nabla_\theta \sum_{t=1}^{T} -\log \pi(a_t|s_t)\hat{A}_t + \left(\hat{V}(s_t) - \hat{R}_t\right)^2$$

Plug g into a stochastic gradient descent optimiser (e.g. RMSprop)

Multiple workers interact with their own environments and send gradient updates asynchronously

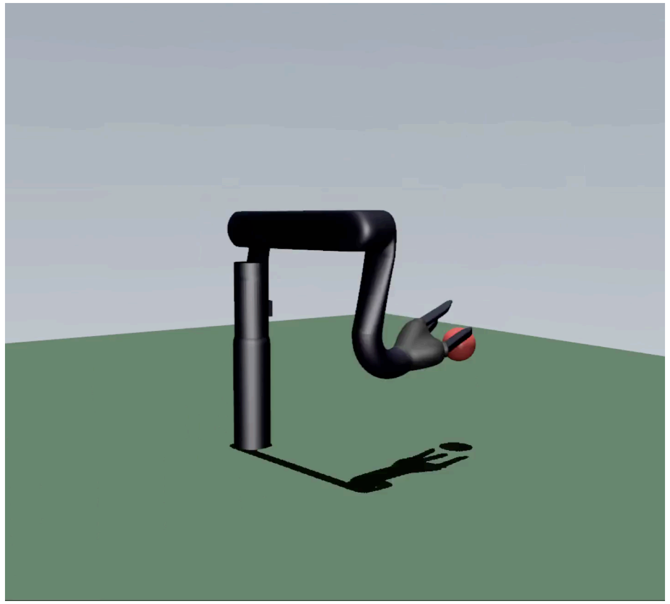
This helps with robustness and experience diversity

[Mnih et al, 2016, Rusu et al., 2016]

Results

- Successfully learns to reach to all target locations with sparse rewards
  ~6 million training steps

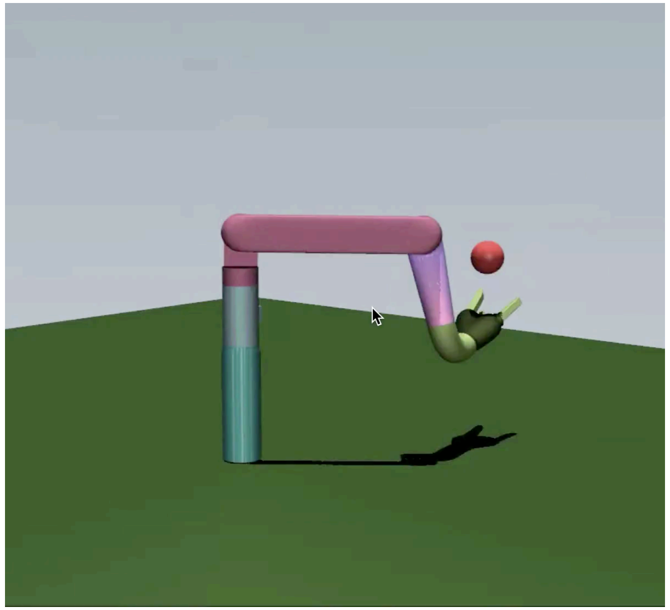
Camera side views



After ~6 million training step

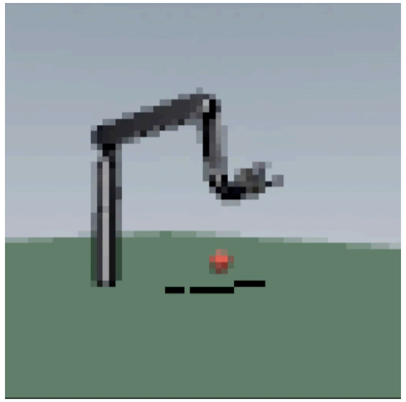
Each episode can last up to 100 steps

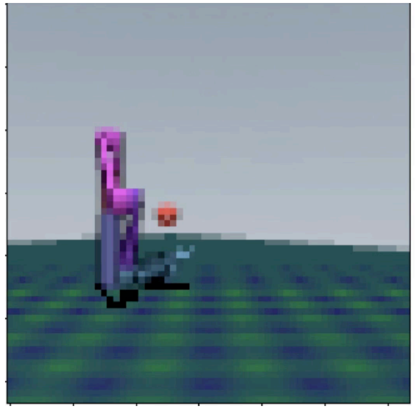
When learned ~7 steps



Domain randomisation

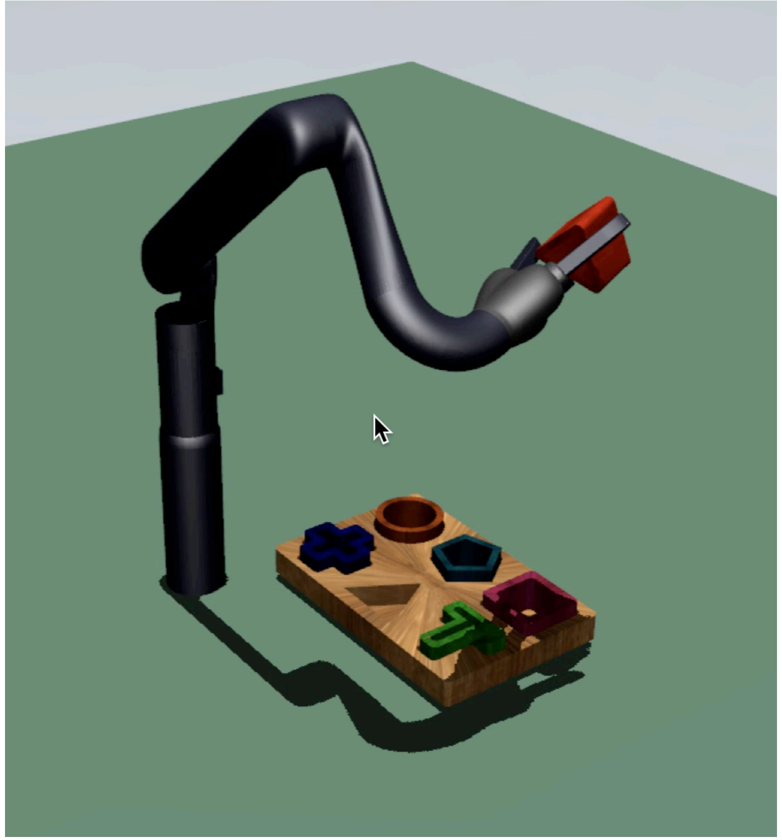
for robustness in transfer to real world

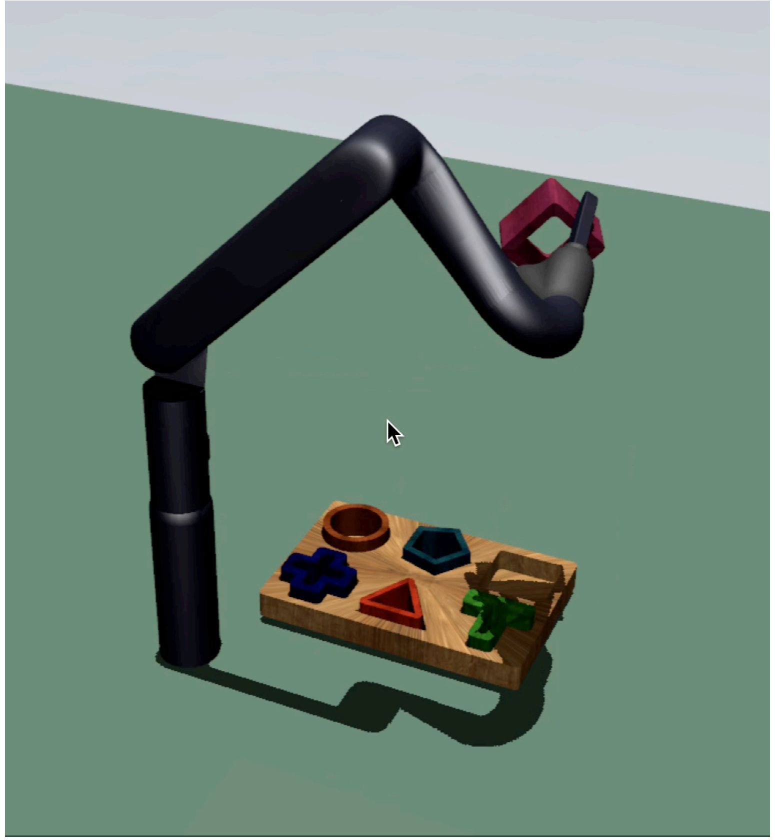




Place shape into its correct position

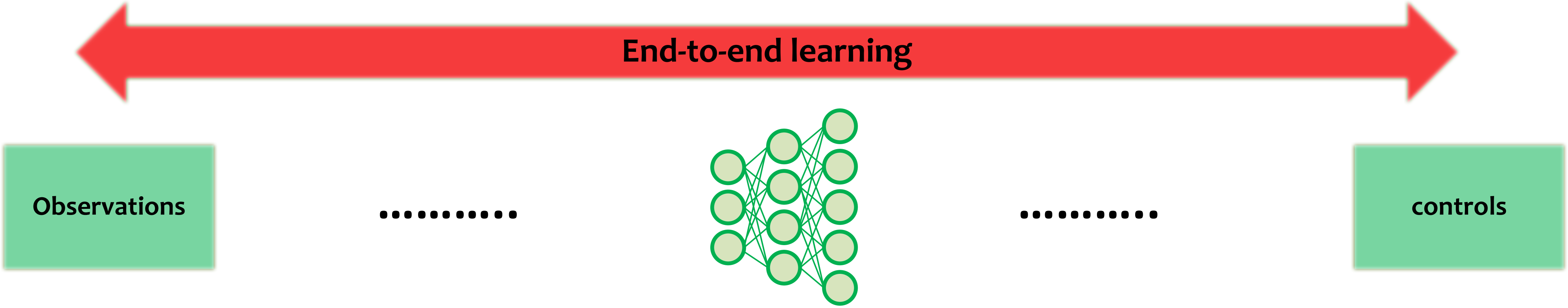
- Tries to place object in correct place but struggles to fit in





Deep RL end-to-end limitations

- Reward function definition is more of an art than science!
- Very sample inefficient
- Learning vision from scratch every time
- Policy does not transfer effectively to slightly different situations (e.g. move target by a few centimeters)



A great recent overview of DRL methods →

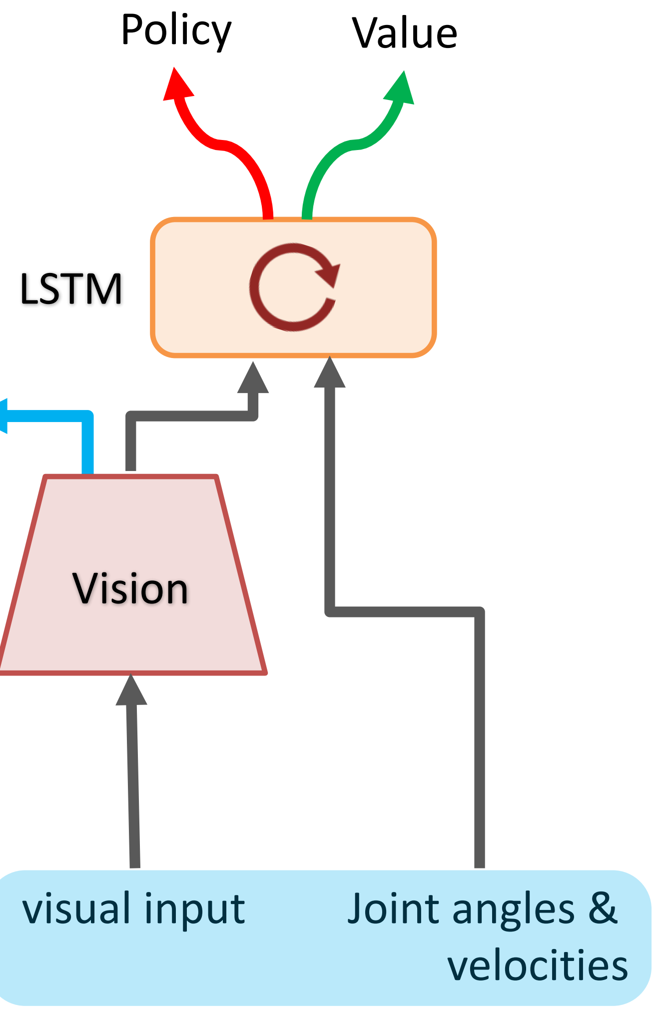
A Brief Survey of Deep Reinforcement Learning

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, Anil Anthony Bharath

Possible solutions

- **Learning with auxiliary information**

  Leverage extra information in simulation, forcing the agent to make sense of the geometry of what it sees. This accelerate and stabilises reinforcement learning



Auxiliary task:
Predict auxiliary Information: e.g. depth

Auxiliary input
Leverage information available only within simulation and learn to cope without them

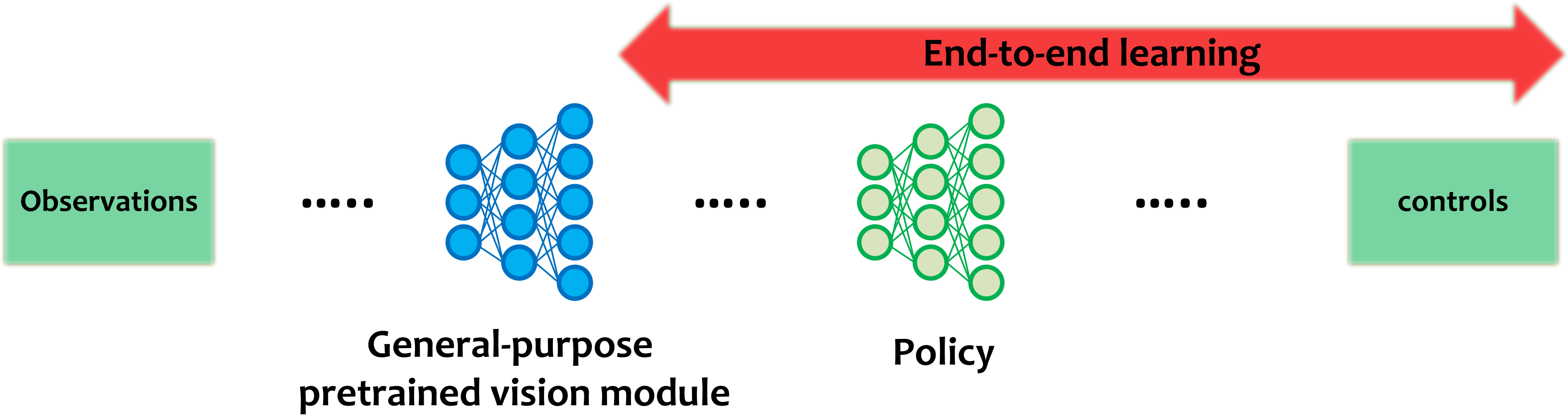
[e.g. Levine et al, 2016 & Mirowski et al., 2016]

Possible solutions

Separating learning vision from the control problem

Avoid learning vision every time, focus on the task at hand

Requires a “general” vision module, useful on many possible tasks.



Learn robust and transferable vision module

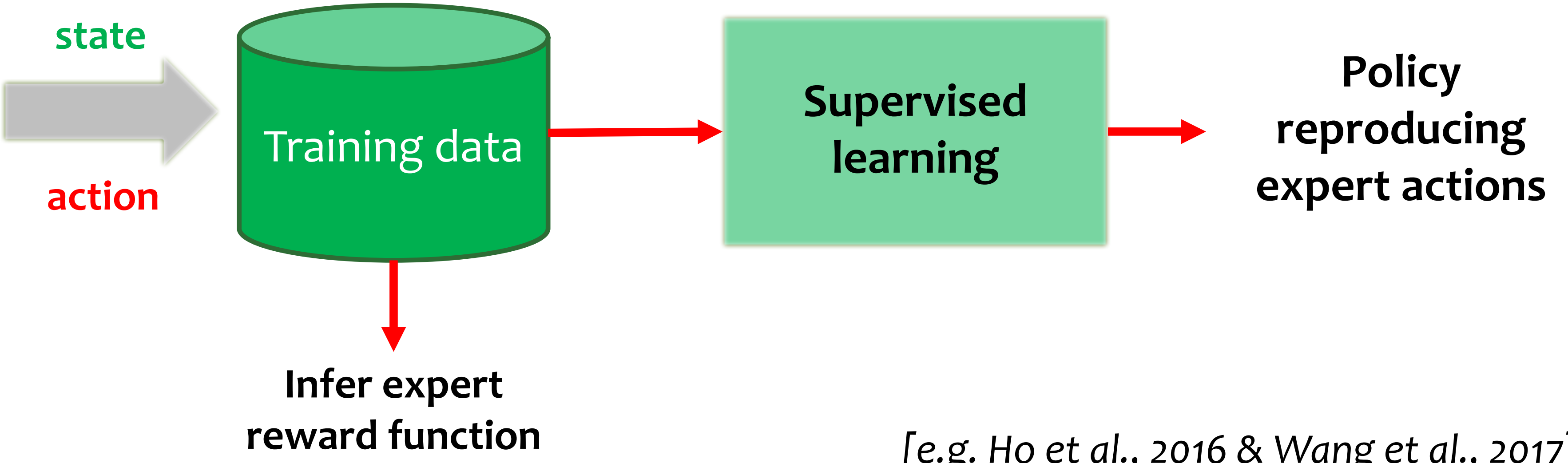
e.g. *[Higgins et al. 2017 & Finn et al. 2017]*

Possible solutions

- **Learning from Demonstrations**

  Imitation Learning: Directly copy the expert (e.g. supervised learning)

  Inverse RL: First infer what the expert is trying to do (learn its reward function **r**), then learn your own optimal policy to achieve it using RL.



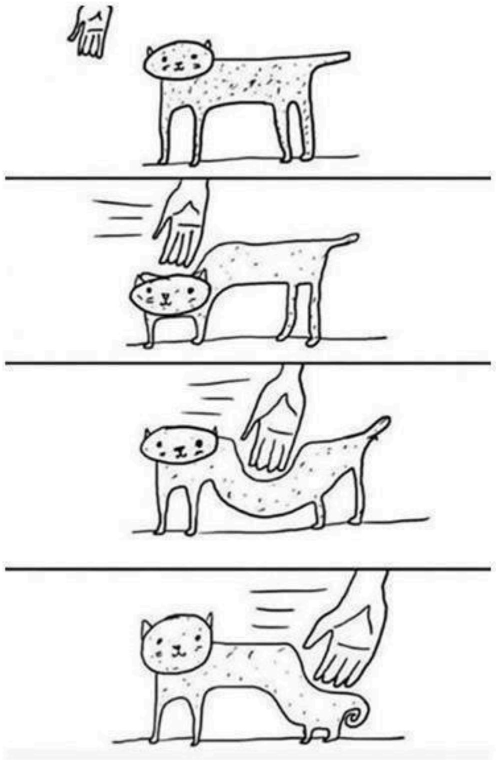
[e.g. Ho et al., 2016 & Wang et al., 2017]

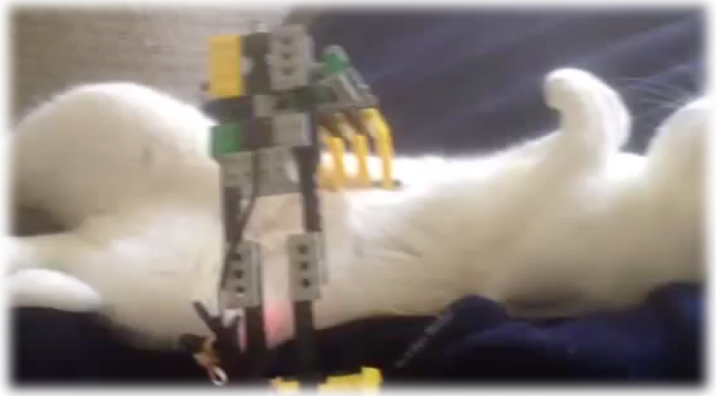
Possible solutions

- **Learning from Demonstrations**

  Imitation Learning: Directly copy the expert (e.g. supervised learning)

  Inverse RL: First infer what the expert is trying to do (learn its reward function **r**), then learn your own optimal policy to achieve it using RL.



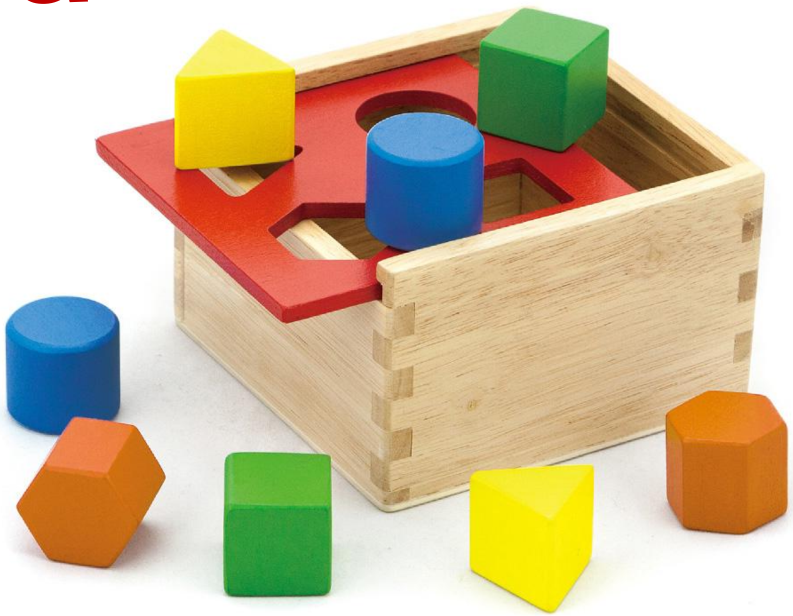


YouTube **World's first cat-petting robotic arm!**

Modelling for deformable objects is challenging!

Current simulators fail to capture full variability of deformable objects and even small differences can break the robot!

Thank you



Imperial College
London

Dr Anil
Bharath

Kai
Arulkumaran

Feryal Behbahani

Morpheus Labs

feryal.github.io @feryalmp @feryal feryal@morpheuslabs.co.uk