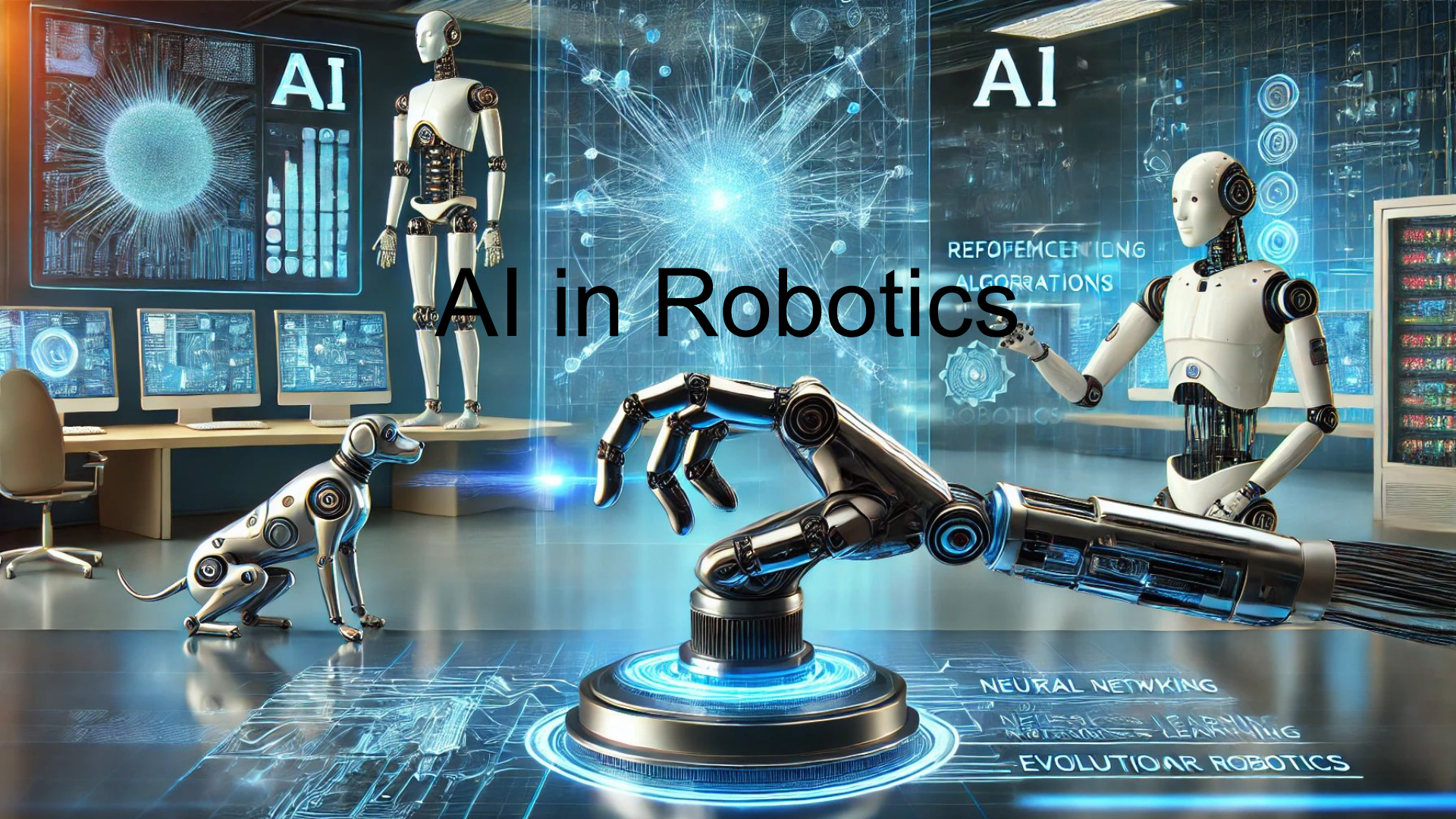




AI

AI

AI in Robotics

RECOGNITION
ALGORITHMS



ROBOTICS

NEURAL NETWORKING

DEEP LEARNING

EVOLUTIONARY ROBOTICS

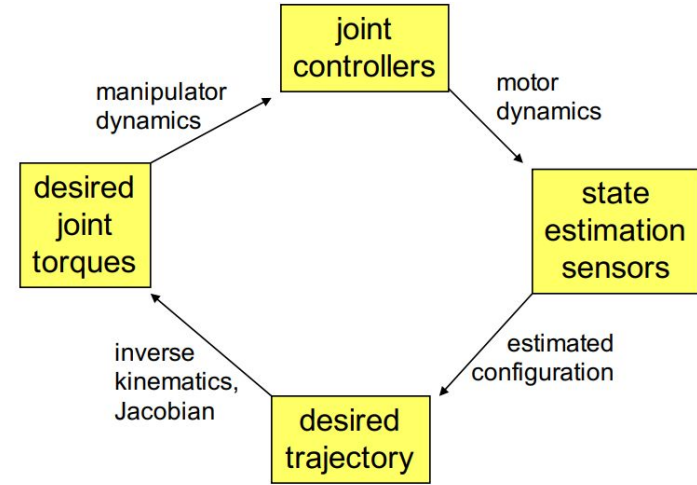
Overview of lecture

- Applications and motivation
- Sensing, planning and control
- AI for robot control:
Reinforcement learning
- AI for robot design:
Evolutionary robotics
- Reality gap



In this course

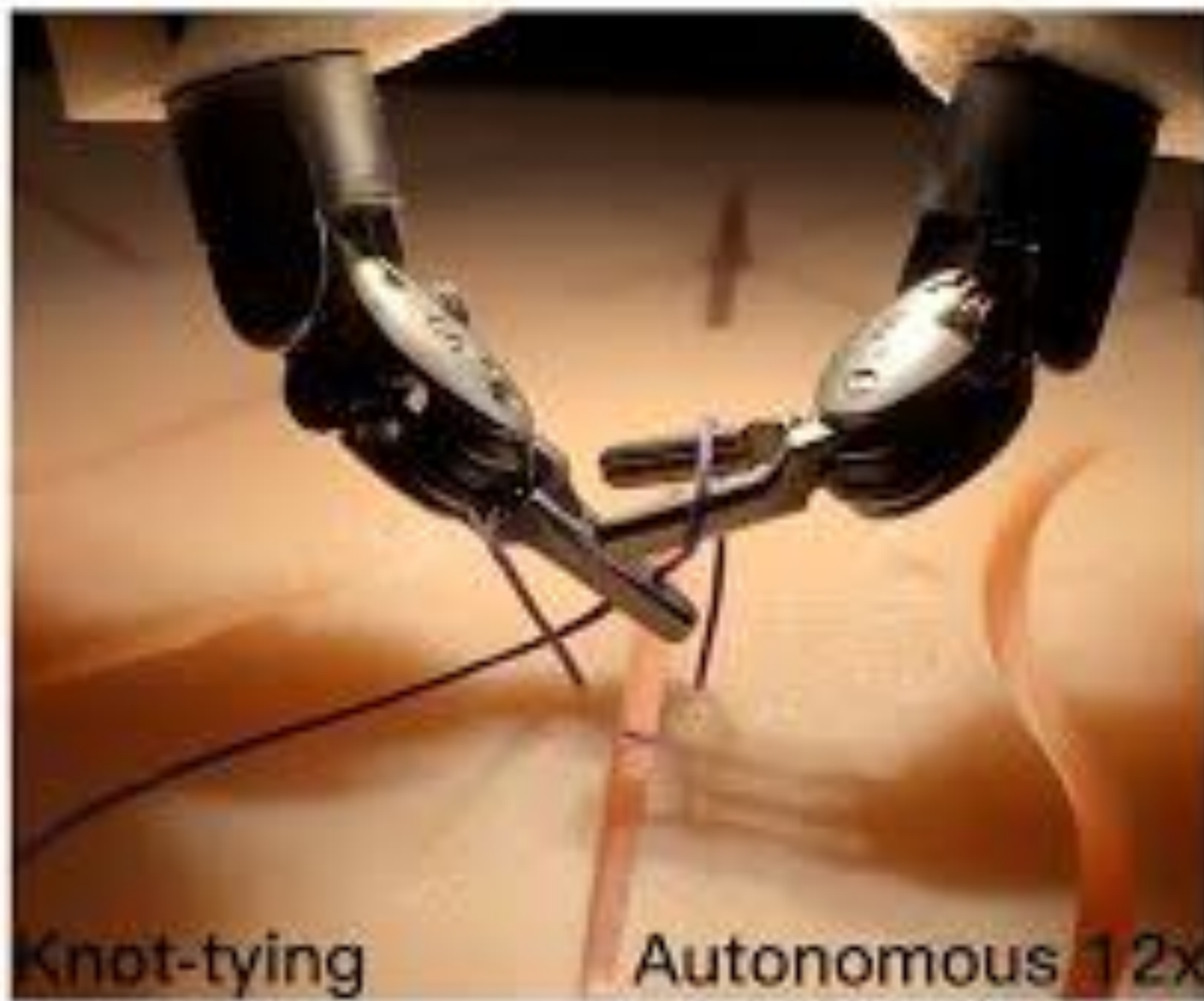
- Kinematics
- Control theory
- ...



- These are applied for **fixed** control mechanisms in **predictable** environments

1X END-TO-END AUTONOMY
MARCH 2024 UPDATE





Knot-tying

Autonomous 12x





HUMANOID ROBOTS 2024 ///



HD Atlas

Boston Dynamics



NEO

1x



GR-1

Fourier



Figure

01



Phoenix

Sanctuary AI



Apollo

ApptroNIK



Digit

Agility



Atlas

Boston Dynamics



H1

Unitree



Optimus Gen 2

Tesla



150cm / 4'11"

160cm / 5'3"

165cm / 5'5"

167cm / 5'6"

170cm / 5'7v"

172cm / 5'8"

175cm / 5'9"

175cm / 5'9"

180cm / 5'11"

180cm / 5'11"

6.0'

5.0'

4.0'

3.0'

2.0'

1.0'

Human

🕒 103lb | 47kg

66lb | 30kg

121lb | 55kg

132lb | 60kg

154lb | 70kg

159lb | 72kg

146lb | 64kg

🛡️ 180lb | 80kg

103lb | 47kg

🍀 103lb | 47kg

180lb | 81kg

🏃 5mph | 8km/h

7.4mph | 12kmh

5mph | 8kmh

2.6mph | 4.3kmh

3mph | 5kmh

7.4mph | 12kmh

3.3mph | 5.4kmh

🏆 11mph | 18kmh

5mph | 8kmh

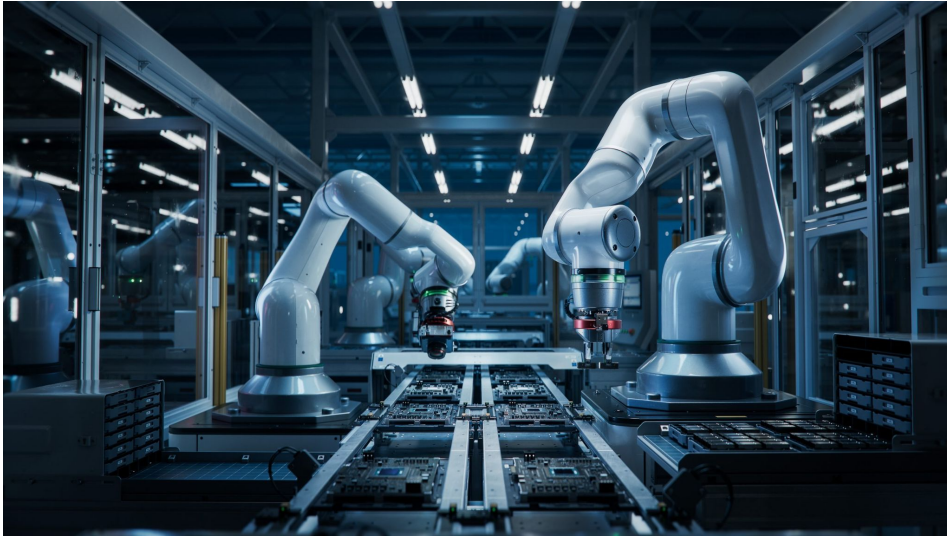
8mph | 13kmh

SOURCE: WWW.LIFEARCHITECT.AI/HUMANOIDS/



MADE VISUAL

Why do we want AI in robotics?

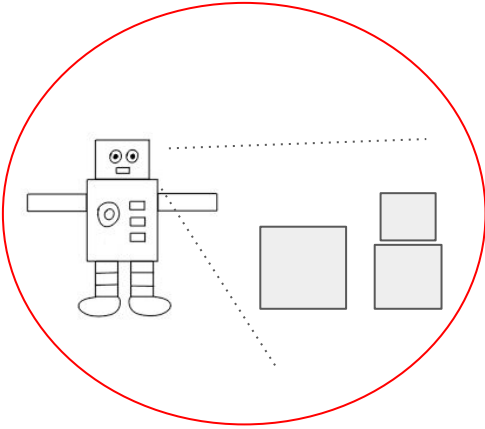


Predictable environments: Can easily create required control mechanisms

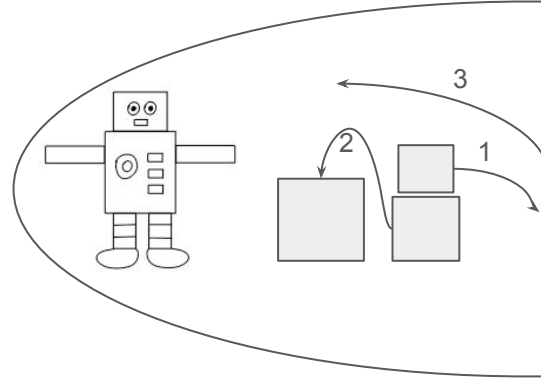


Unpredictable environments: Need to react to unpredictable events

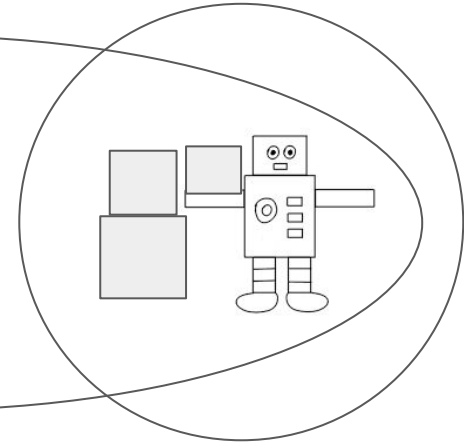
Sensing



Planning



Control



Combine

Use preset control
mechanisms

OR

Use AI

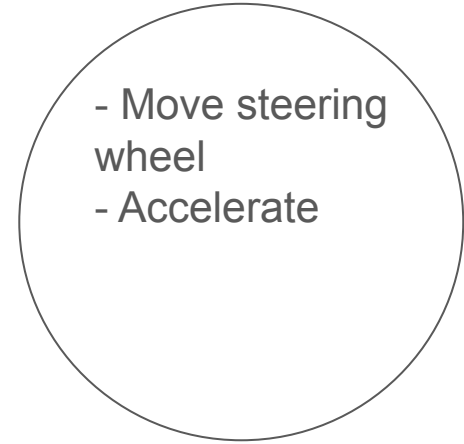
Sensing



Planning

1. Wait for traffic to clear
2. Turn left

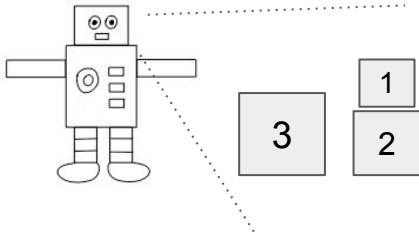
Control



Use preset control mechanisms
OR
Use AI

Sensing

Sensing is important to **detect features**



Sensing overview

Example sensors

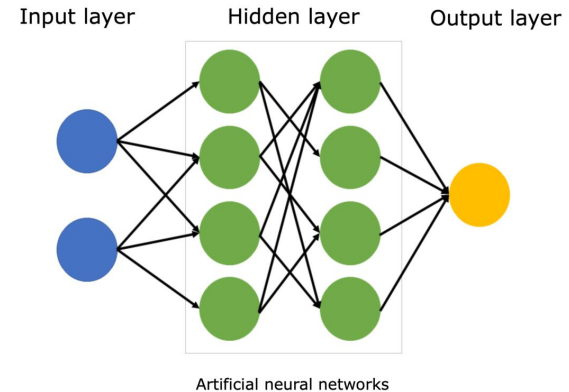
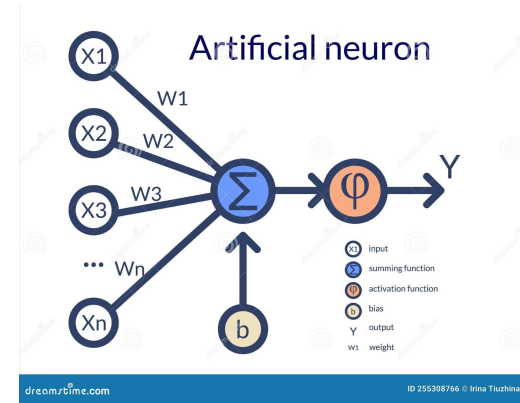
- Motor encoders
- IMU - Inertial Measurement Unit
- Vision (camera)
- Contact and force

Example techniques

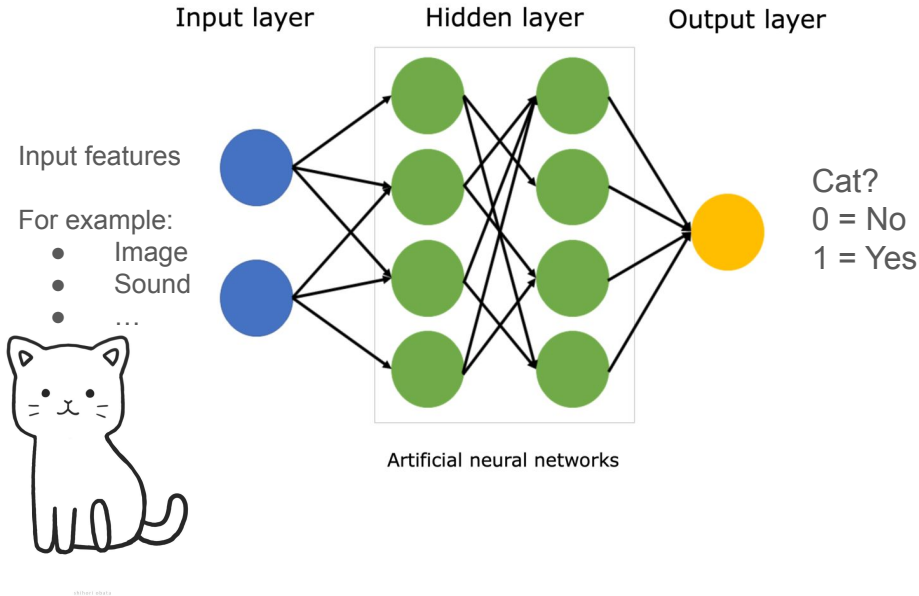
- Convolutional Neural Networks
- Recurrent Neural Networks
- Time Series
- LSTMs

Sensing - Neural Networks

- Neural Networks (NNs) are **machine learning** models inspired by the human brain, used for **recognizing patterns**, **making predictions**, and **learning from data**.
- Consists of:
 - Neurons: Nodes to process and pass information
 - Weights: Strength of each connection
 - Activation functions: Calculate neuron activation
 - Input Layer: Receives data
 - Hidden Layers: Applies abstraction
 - Output Layer: Produces the final result



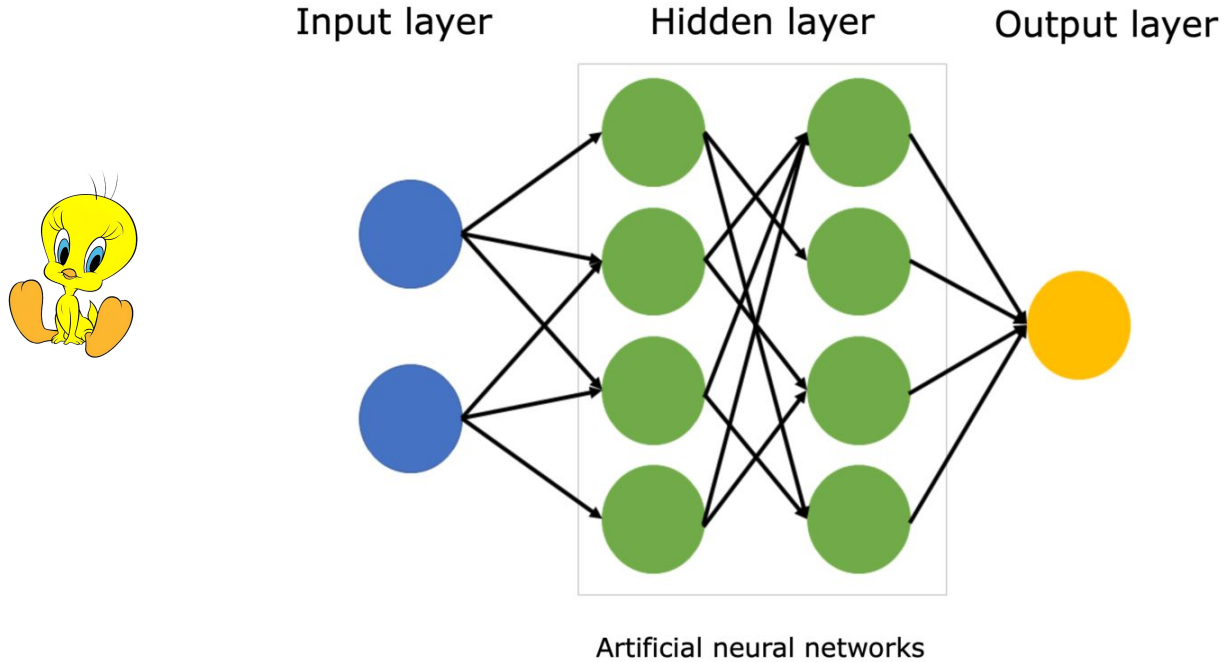
Sensing - Neural Network



Is the prediction correct?

- Yes - GREAT!
Nothing to be done
- No - Update network **weights** using **Backpropagation**

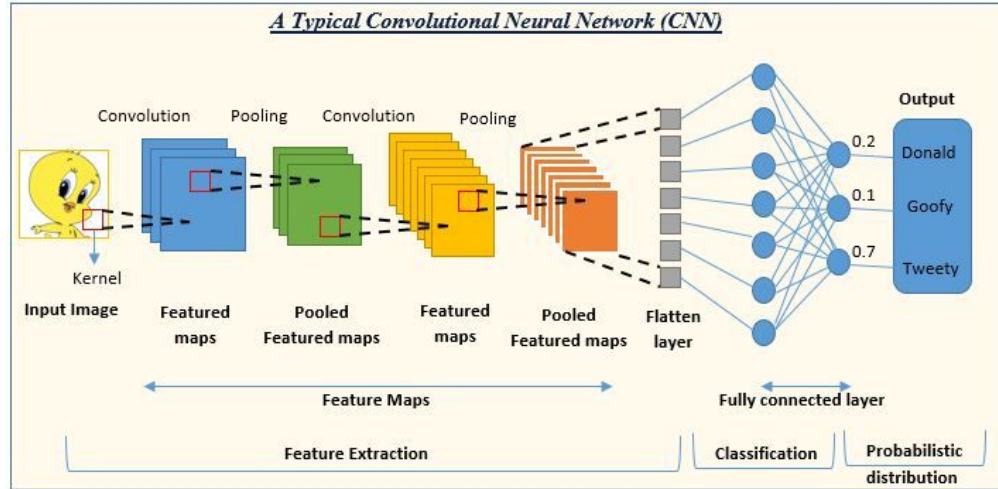
Sensing - Convolutional Neural Networks



How to use an image as input for an ANN?

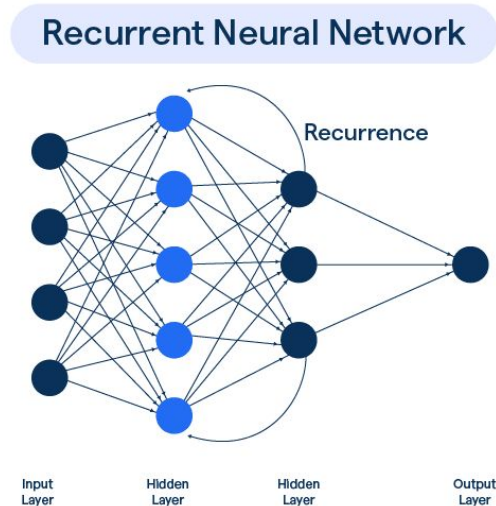
Sensing - Convolutional Neural Networks

- Designed to deal with images
- Reduces number of parameters

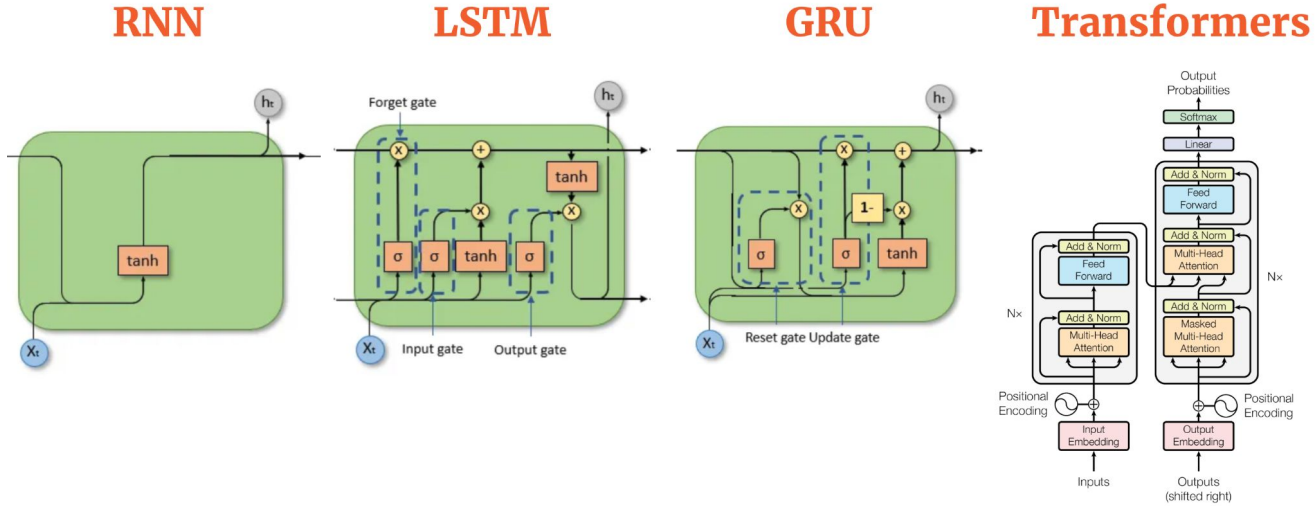


Sensing - Recurrent Neural Network

- Designed for **sequential data**
- Maintains a **memory**
- Used for:
 - Time-series
 - Natural Language Processing
 - Robot control sequences



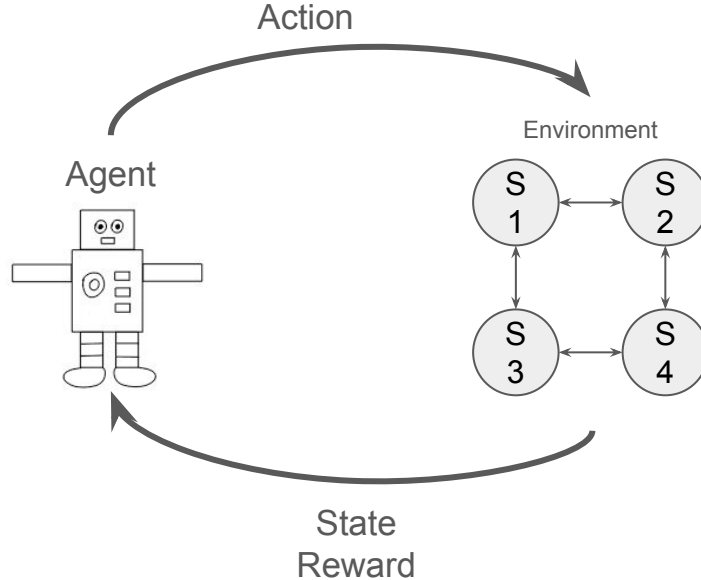
Sensing - Recurrent Neural Network



<https://aiml.com/compare-the-different-sequence-models-rnn-lstm-gru-and-transformers/>

AI for robot control: Reinforcement learning

Reinforcement learning



The agent chooses according to the current state:

- Action

The environment returns according to State-Action pair:

- Next state
- Reward

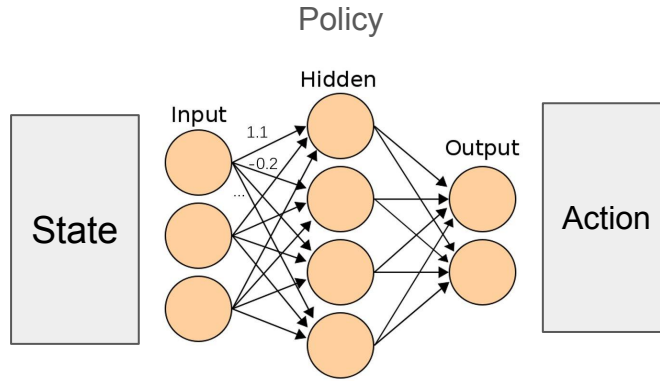
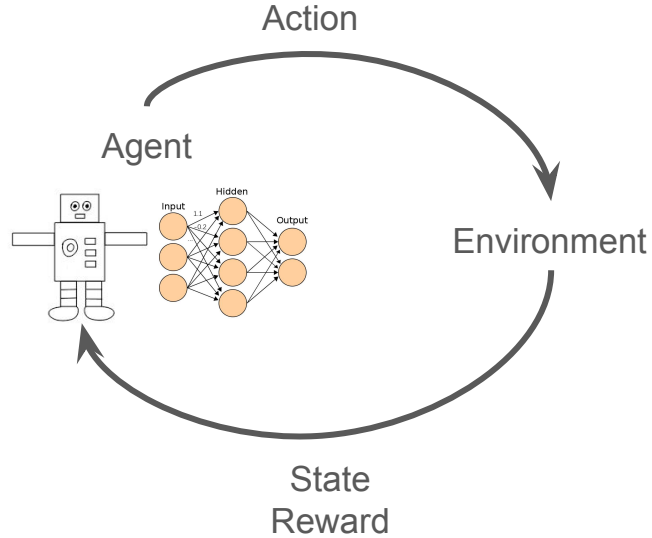
Goal is to find a **policy** for the **agent** to **maximize reward**.

Reinforcement learning

Important terminology:

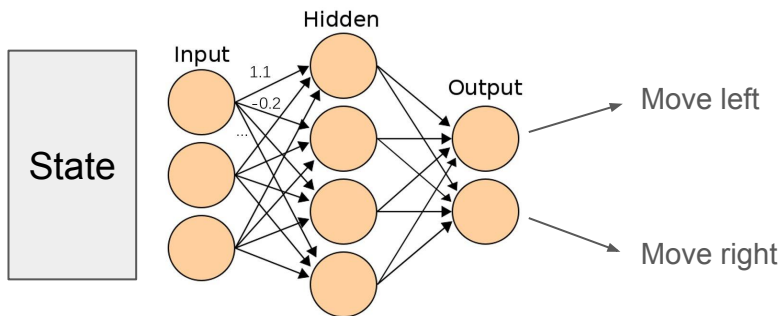
- Agent: Model/Robot interacting with the environment
- State: Perception of the agent in the environment
- Action: Behaviour of agent to transition from one state to another
- Reward: Positive/Negative outcome based on action done in state
- Policy: Plan of actions based on states to maximize reward
- Value: Expected reward at state

Deep reinforcement learning



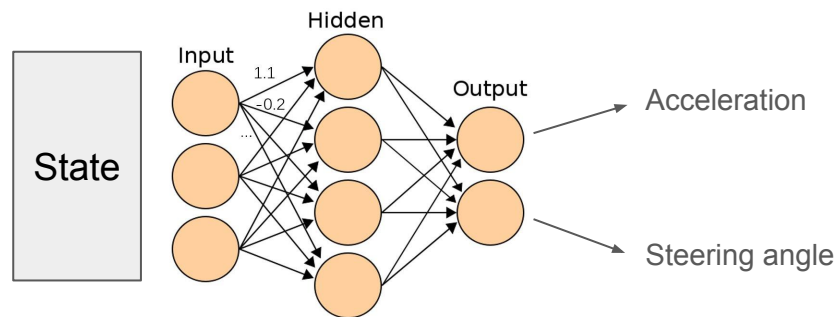
Deep reinforcement learning

Discrete actions



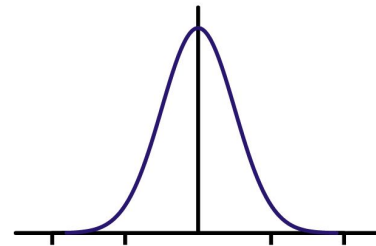
Actions are the direct output of the model

Continuous actions

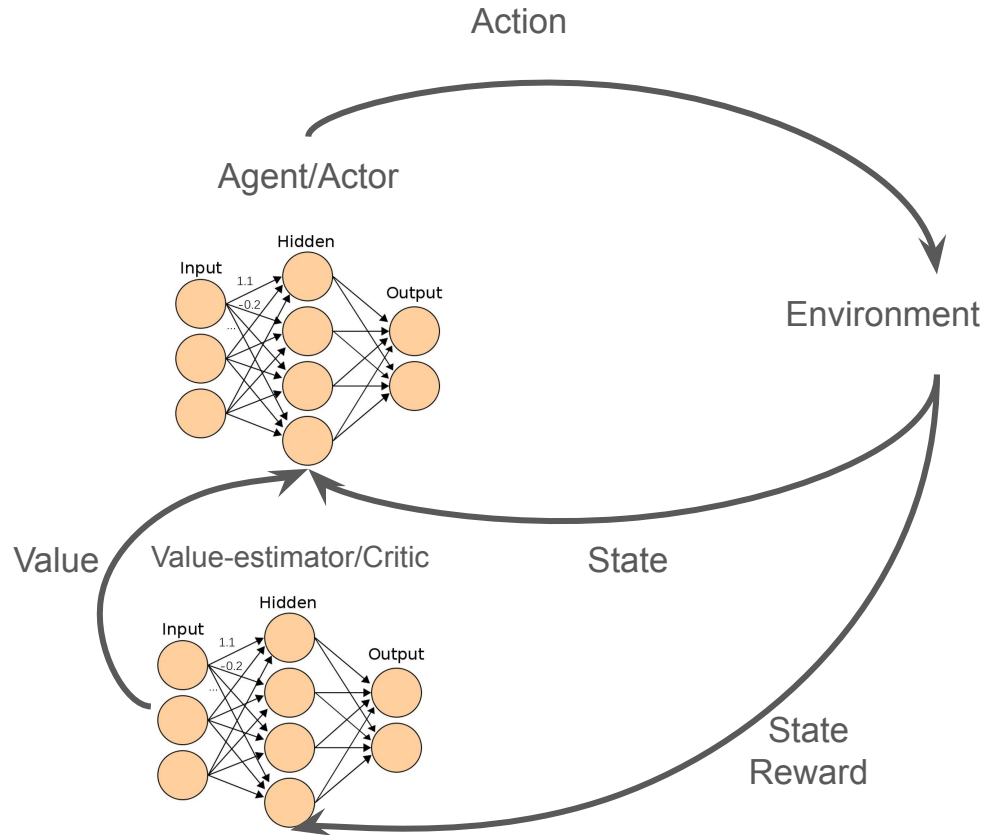


Often a stochastic policy is used

- The model returns Mean and Standard deviation for action
- Improves stability and exploration



Deep reinforcement learning - Actor-critic network

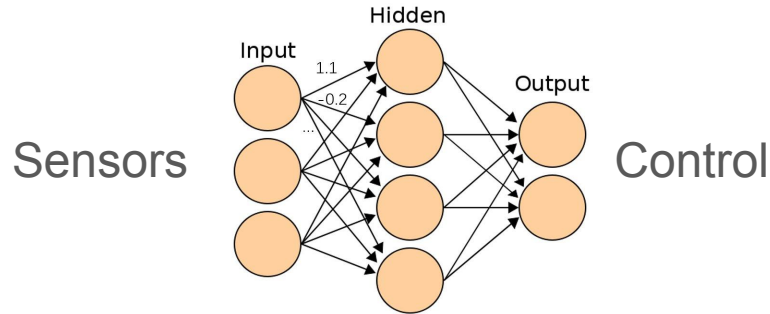


Add Critic network as state-**value** estimator

- Improves stability
- Improves sample efficiency

Deep RL in Robotics - Sense and control

One policy for all actions

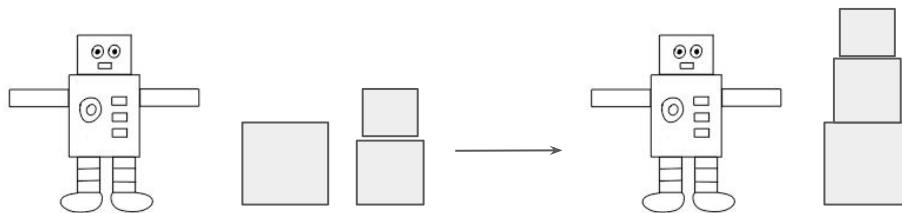


- Skip planning and act directly on sensing
- For example: the humanoid robot previously shown in this lecture

Deep RL in Robotics - Issues

Issues for Deep RL in Robotics:

- Costly (trial and error)
- What is the reward function?
 - Sparse rewards
 - Reward hacking - Robot finds a way to get high reward with unwanted actions



Imitation learning

Use expert demonstrations as ground truth

Collect data

- Expert demonstrations
- Results in State-Action pairs for the robot to use
- Examples
 - Doctor in surgical operations
 - Car drivers

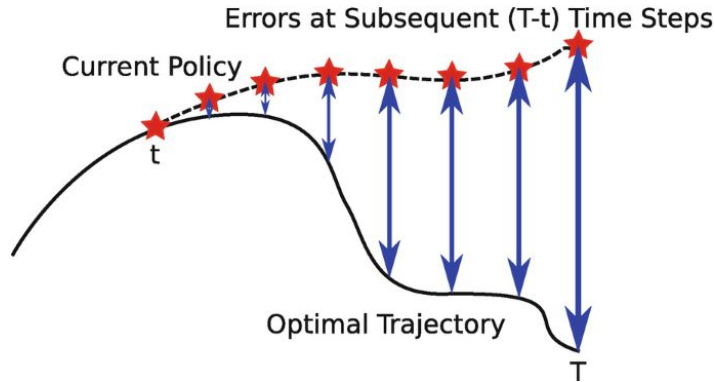
Learn policy

- Use State-Action pairs to optimize policy of robot
- Supervised Learning
- Combine with Reinforcement learning

Execute policy

- Robot performs task autonomously
- Fine-tuning to improve performance

Imitation learning - Compounding errors



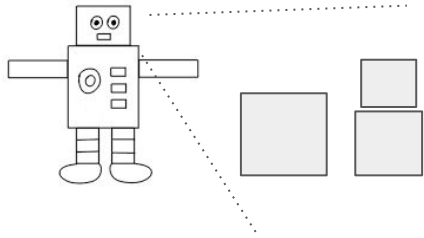
Possible solution:

1. Collect data during policy training
2. Use expert to relabel new data
3. Add new data to dataset
4. Repeat

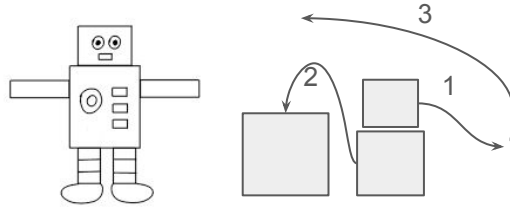
Issue: Time consuming to relabel new data using expert

Hierarchical reinforcement learning

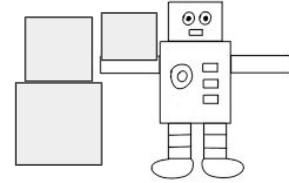
Sensing



Planning

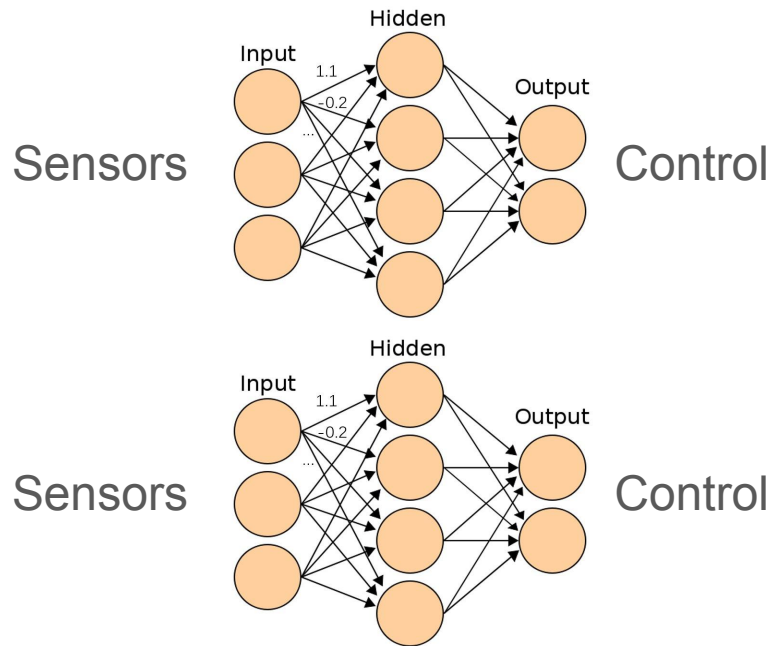


Control

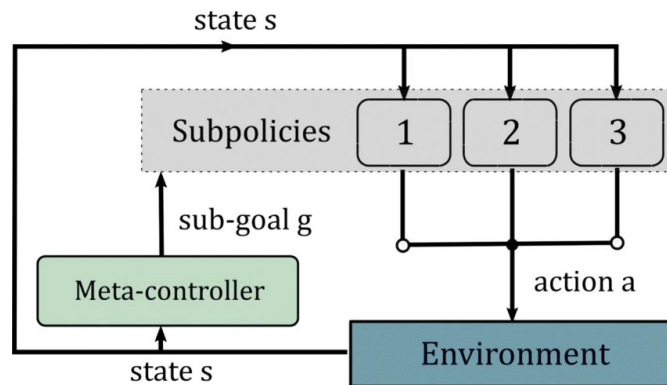


Hierarchical reinforcement learning

One policy for each action



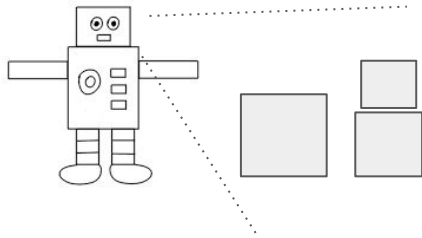
- Create policies for all known actions
- Plan actions using meta-controller



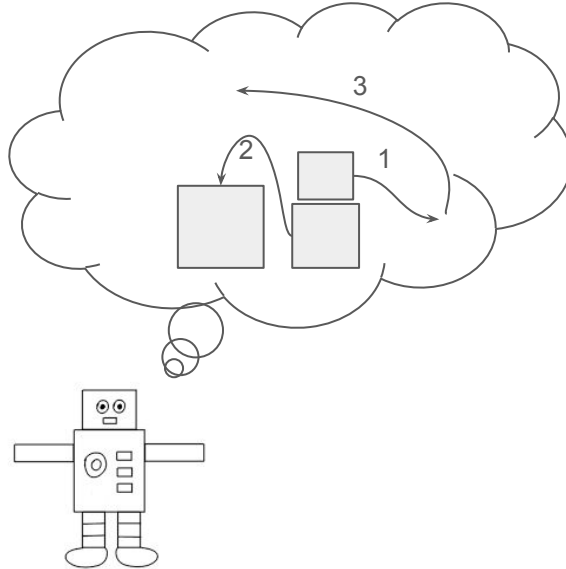
Model predictive control

- Look into the future and base actions on that

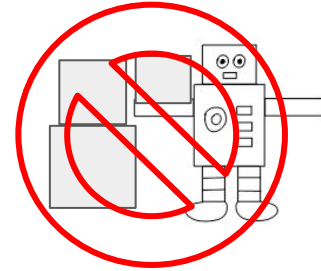
Sense

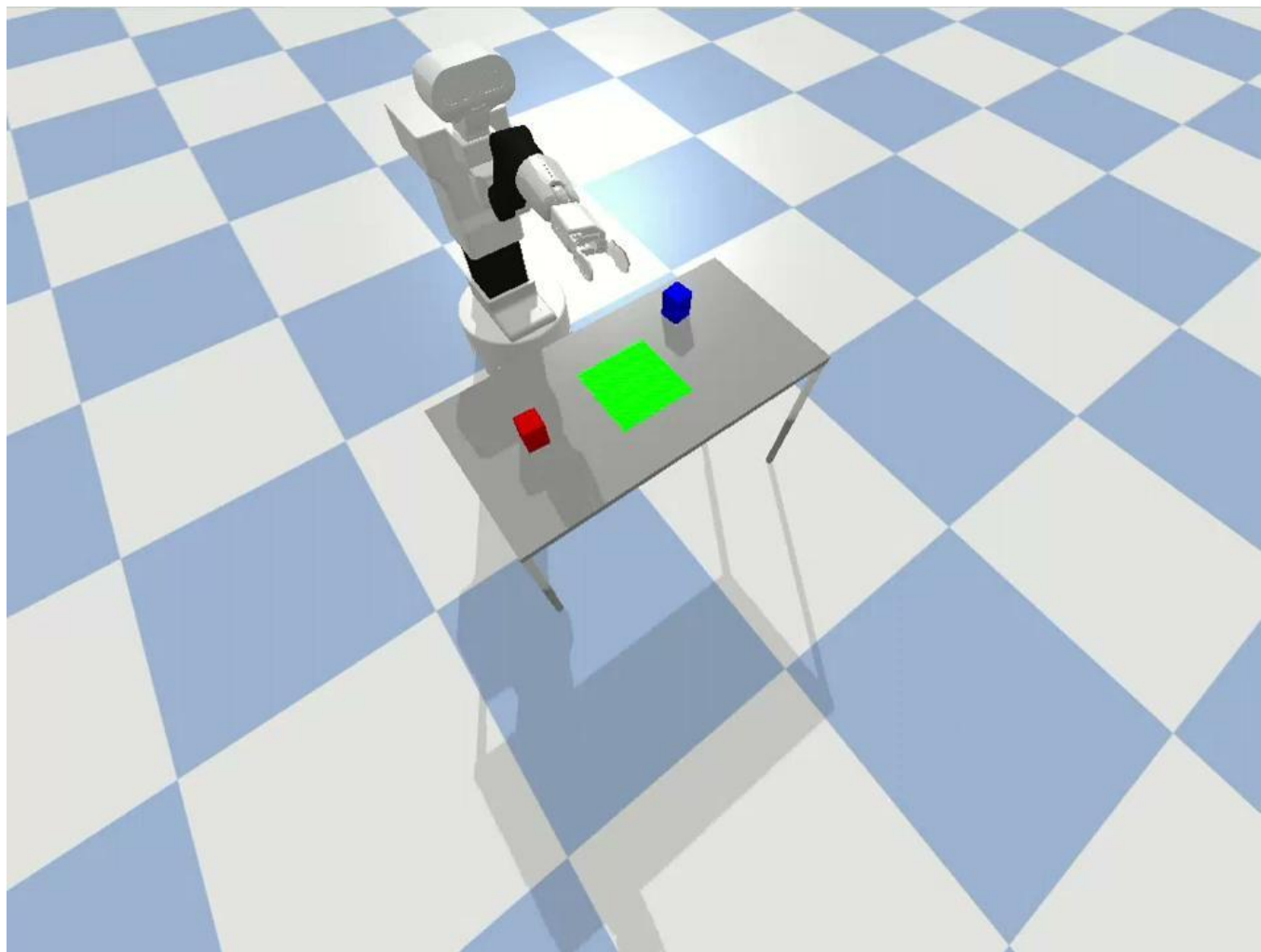


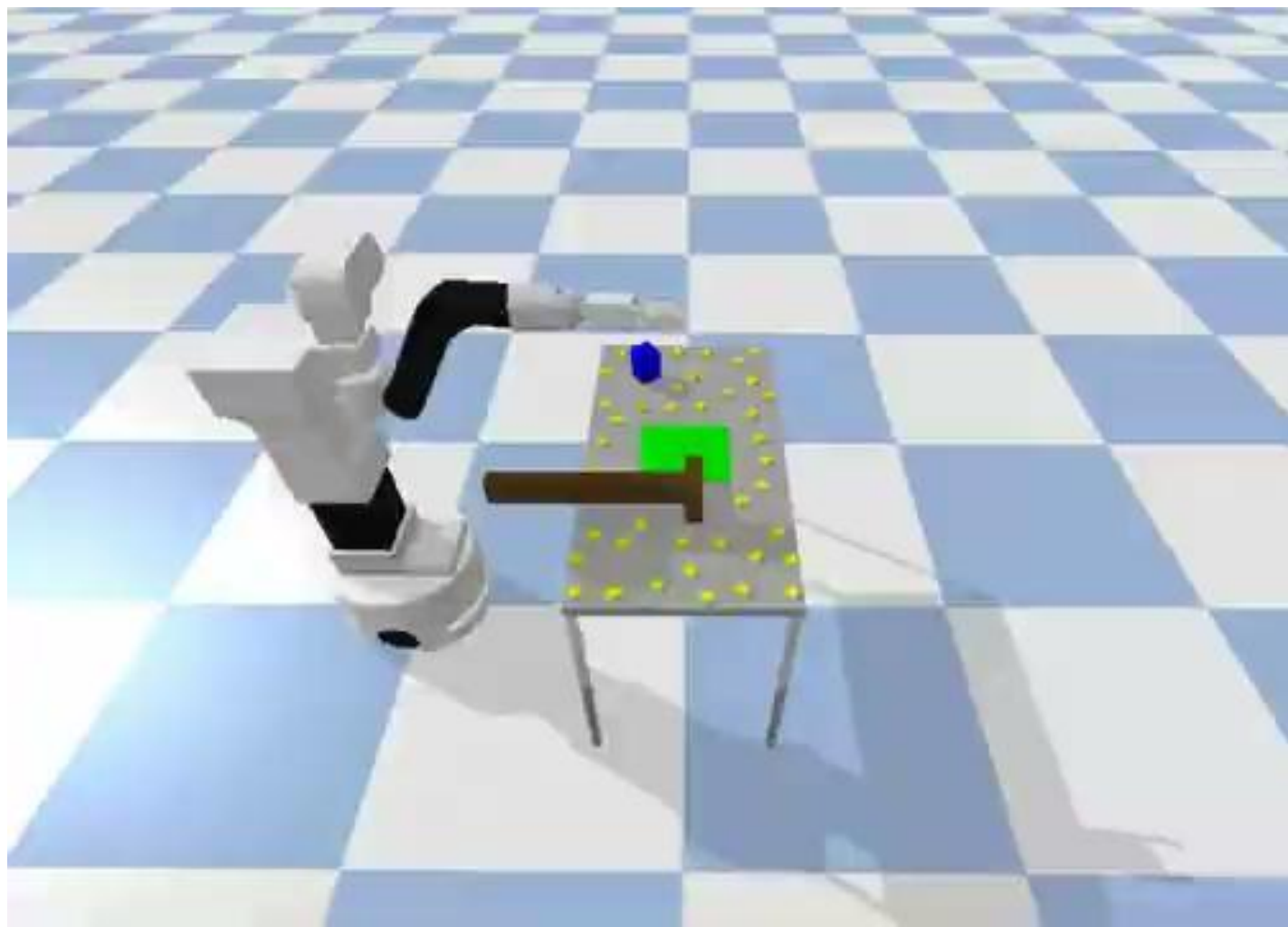
Simulate
actions

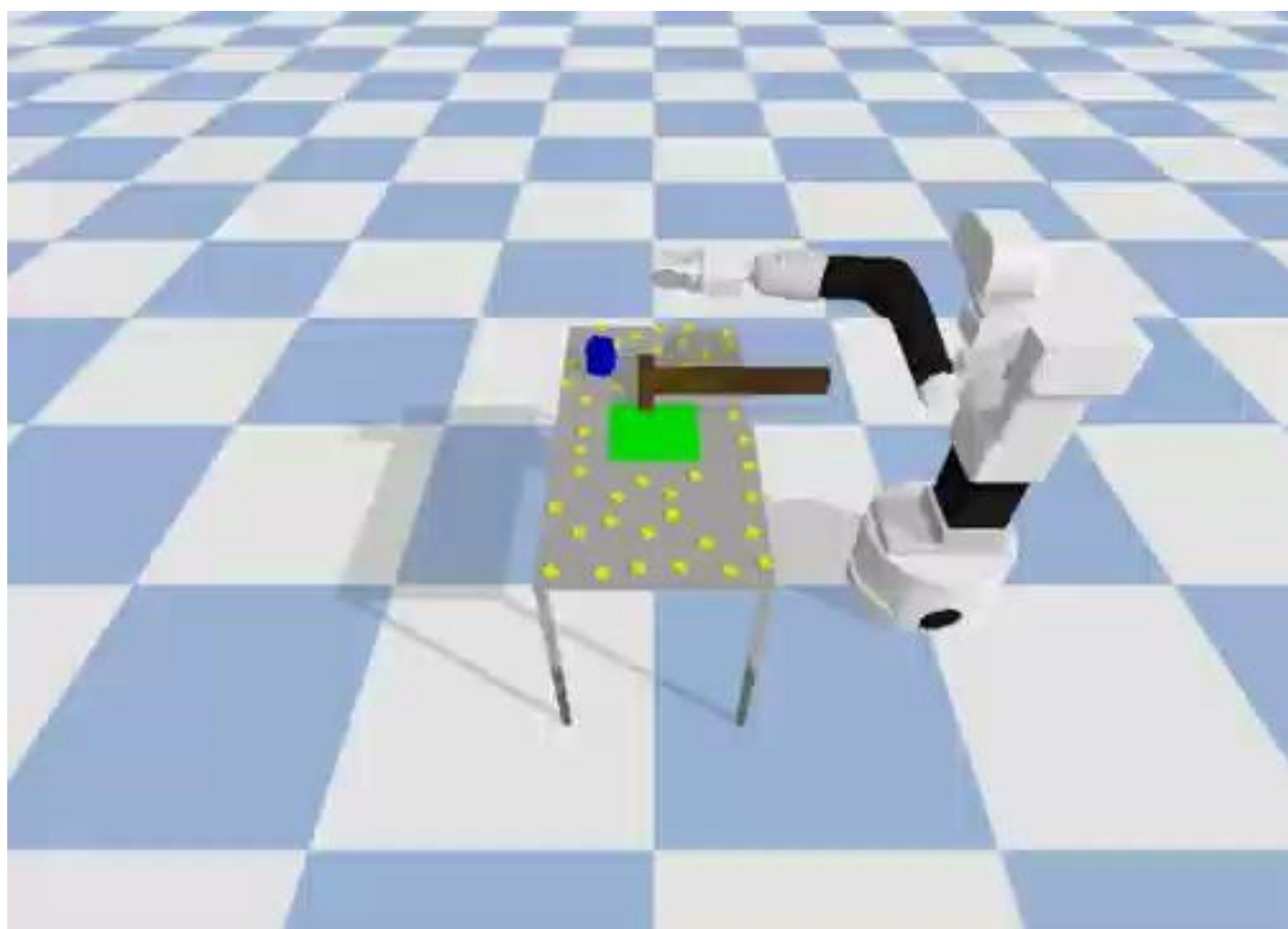


Act







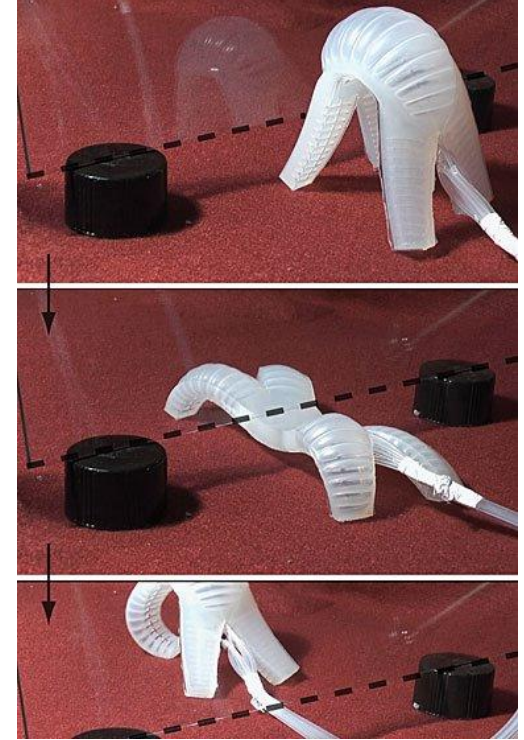
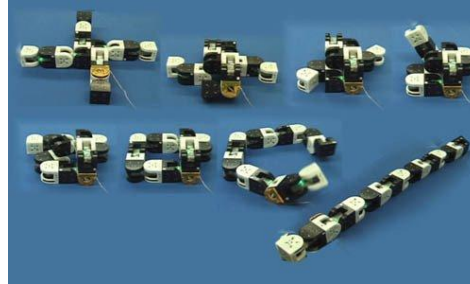


AI for robot design: Evolutionary robotics

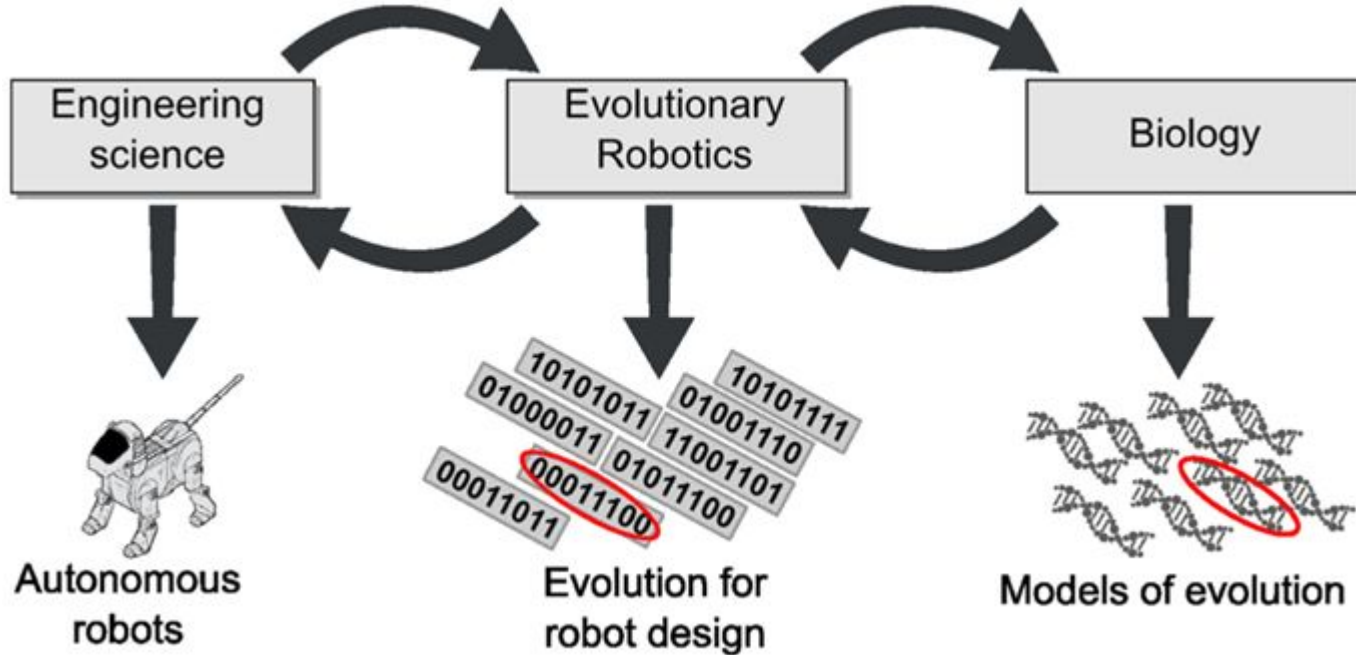


Evolutionary Robotics: Motivation

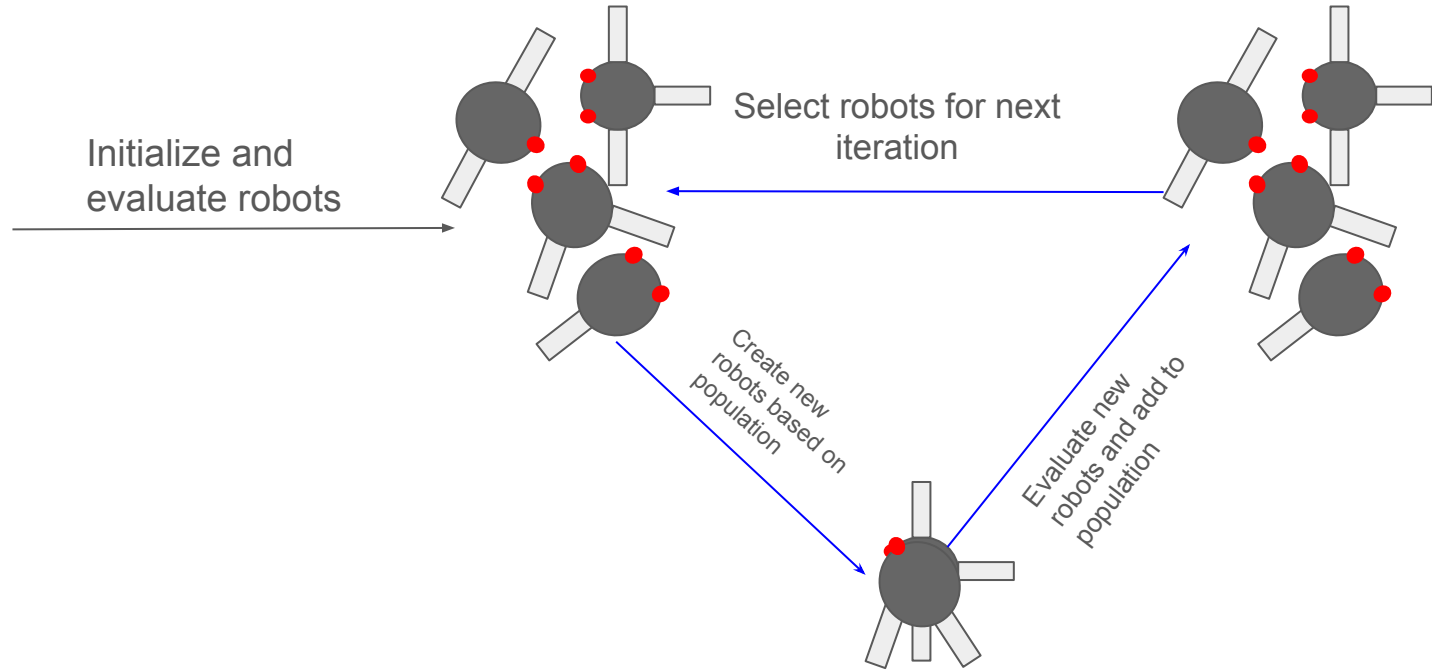
- Adaptability
- Unknown environments
- If evolution created intelligence, can Artificial Evolution create Artificial Intelligence?



Evolutionary Robotics: Motivation



Evolutionary Robotics: How does it work?



Evolutionary Robotics: How does it work?

Genotype

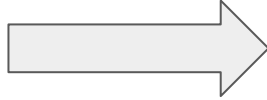


Used for search operators

Phenotype

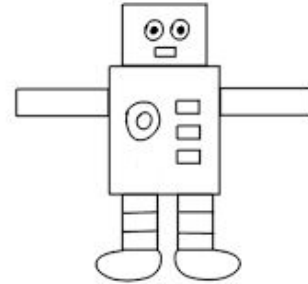


Used for evaluation

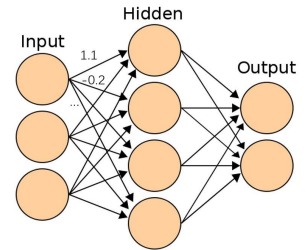


A robot has two components to optimize

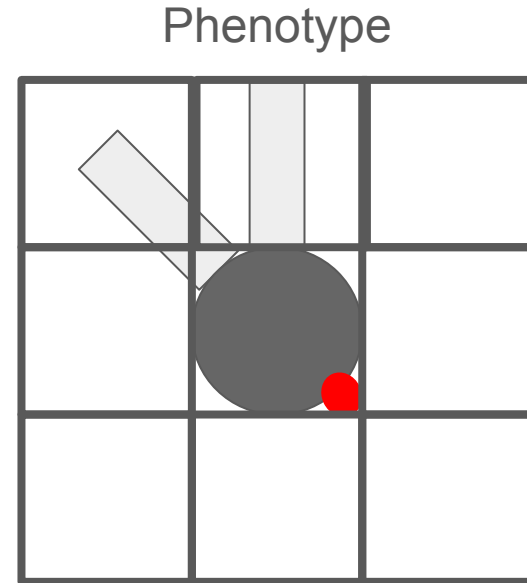
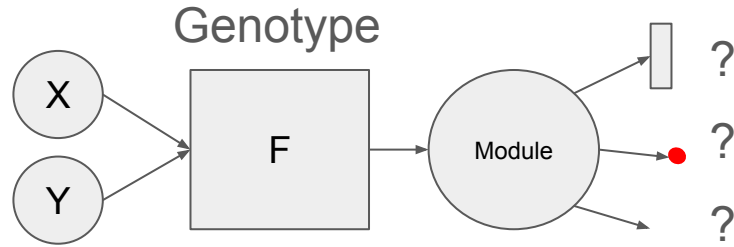
Body



Control



How does it work? - Representation Body/Morphology



Indirect encoding: Not reversible

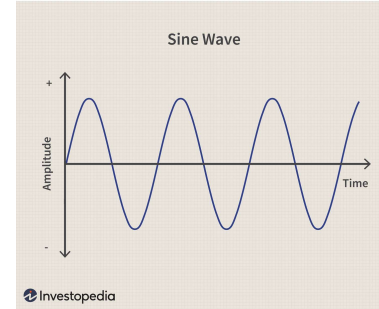
How does it work? - Representation Brain/Controller

Genotype

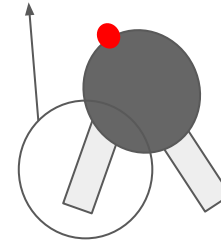
List of parameters for Sine Wave:

$A=0.4$, $P=92^\circ$, $O=0.4$, $F=4.0$

Phenotype

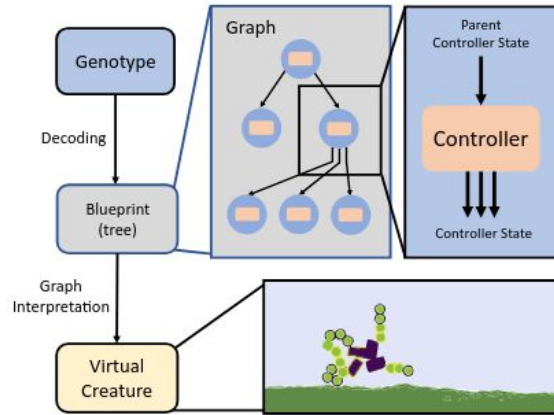


$$\Theta = A * \sin(\phi + P) + O$$
$$\phi = \phi + \Delta\phi * F$$



Direct encoding: Reversible

Encoding



<https://www.youtube.com/watch?v=dE1L48bbgAo>



(a) Direct encoding PCO



(b) Direct encoding P-RNN



(c) Indirect encoding SW



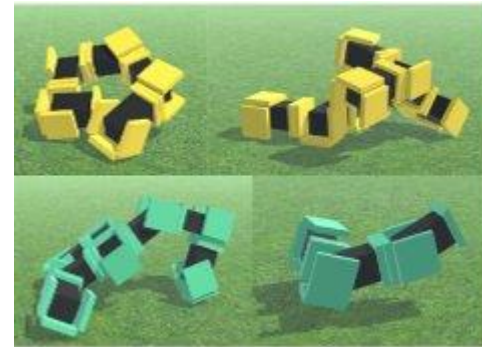
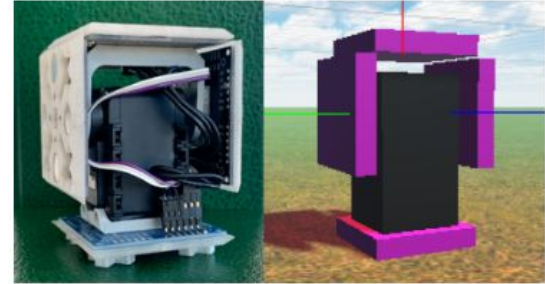
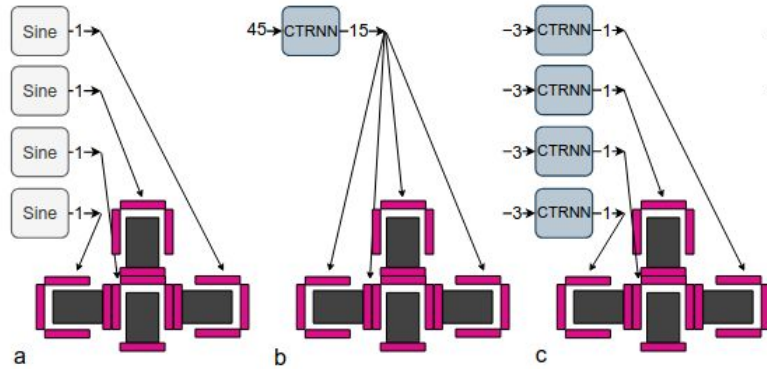
(d) Indirect encoding PCO



(e) Indirect encoding RNN

Veenstra, F., Szorkovszky, A., & Glette, K. (2023). *Decentralized Control and Morphological Evolution of 2D Virtual Creatures*.

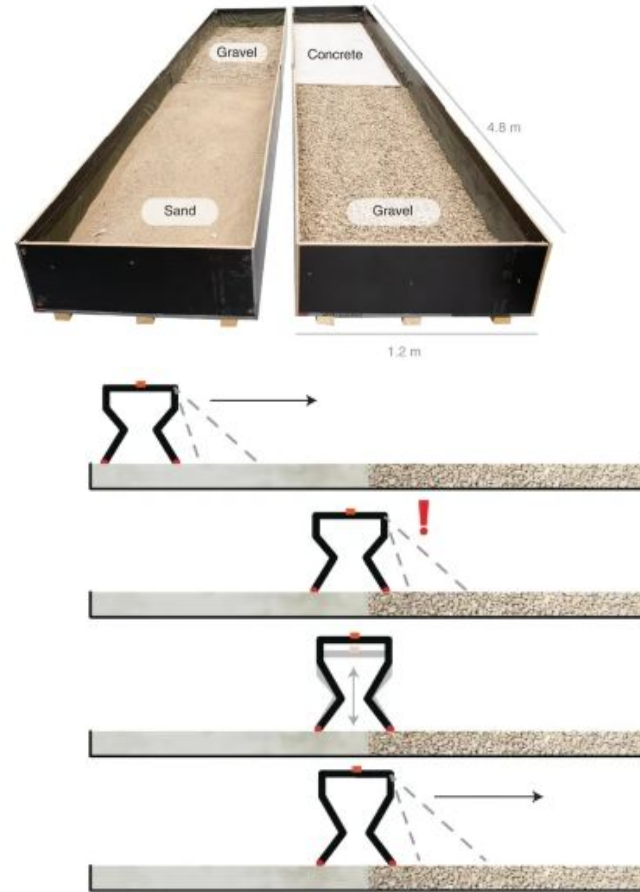
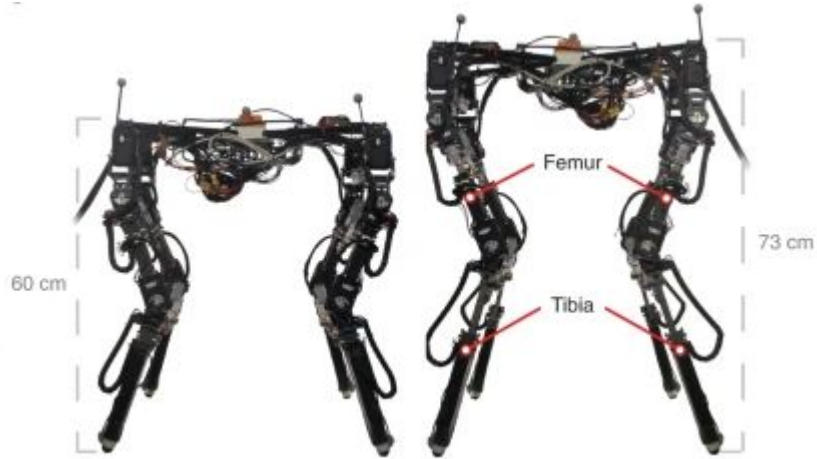
Type of controller



<https://www.mn.uio.no/ifi/english/research/groups/robin/research-projects/cocomo/modbots.html>

Kvalsund, M, Glette, K., & Veenstra, F. (2022). *Centralized and Decentralized Control in Modular Robots and Their Effect on Morphology.*

Real-world evolution

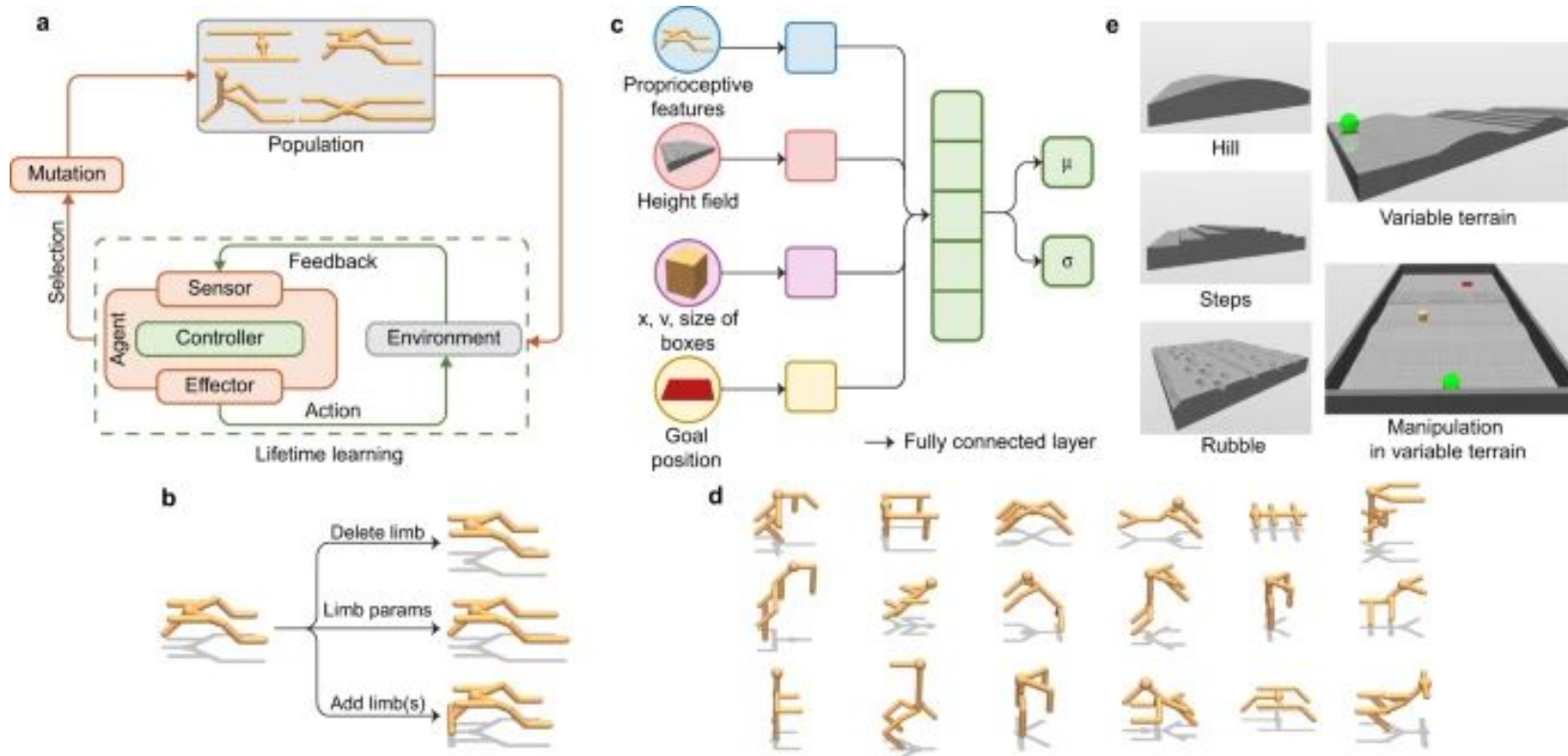


Nygaard, T. F., Martin, C. P., Torresen, J., Glette, K., & Howard, D. (2021).
Real-world embodied AI through a morphologically adaptive quadruped robot

A Morphologically Adaptive Quadruped in the Wild



Combining Evolutionary robotics and Reinforcement learning



Reality gap

Reality gap

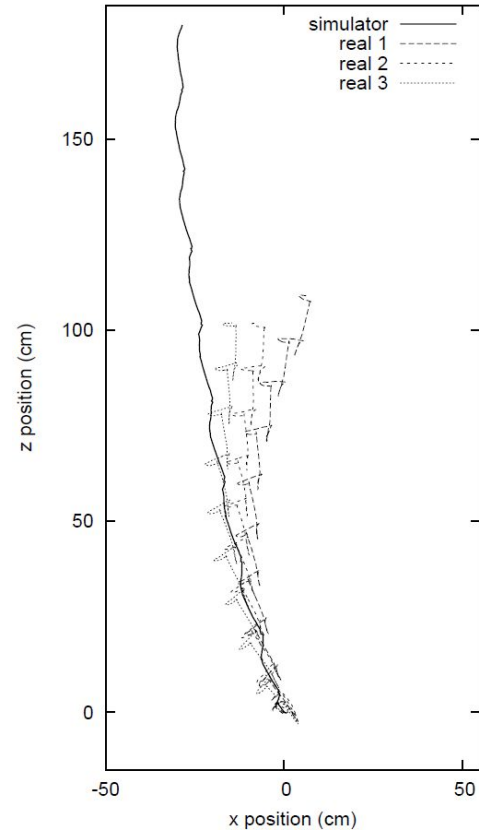
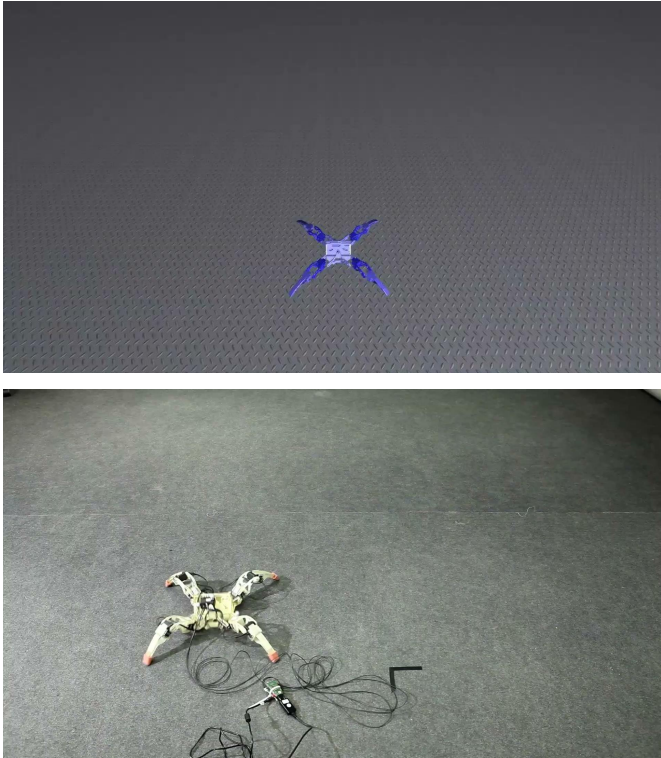
- Training robots in **real life** is **costly**
- **Simulations** can be used.
- Simulators are **fast** and can evaluate solutions **in parallel**

But...

- A simulator cannot capture all aspects of reality
- Evolved solutions may exploit features of the simulator not present in reality

The solutions evolved in simulation behave differently when applied to the real robot!

Reality gap



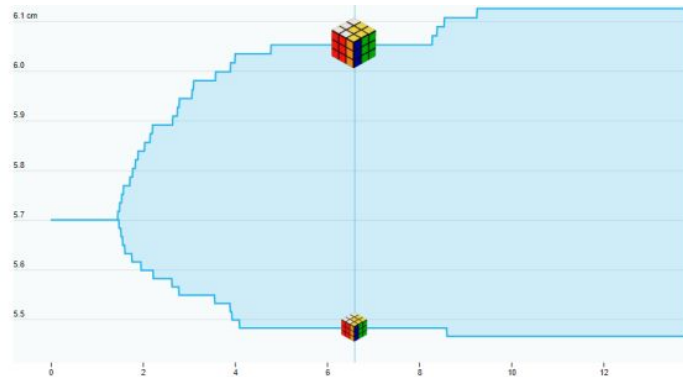
Glette, K., Johnsen, A. L., & Samuelsen, E. (2014). *Filling the reality gap: Using obstacles to promote robust gaits in evolutionary robotics.*

How to deal with the reality gap

1. Increase simulation fidelity
 - Manually: do more precise measurements, increase computation spent on solving physics equations
 - Automatically: measure deviation simulation-reality, auto-tune simulator for smaller deviation
2. Encourage robustness
 - Manually: E.g. Encourage slow, static movements, add noise
 - Automatically: Avoid solution types that transfer poorly
3. Online learning after deployment on real robot
 - Can use evolution, reinforcement learning, or other method

How to deal with reality gap

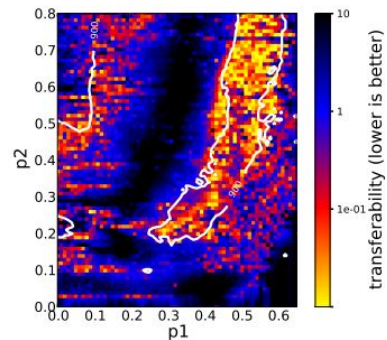
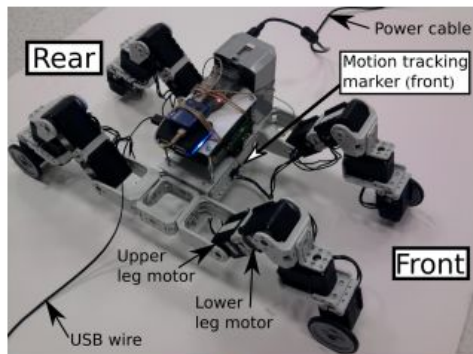
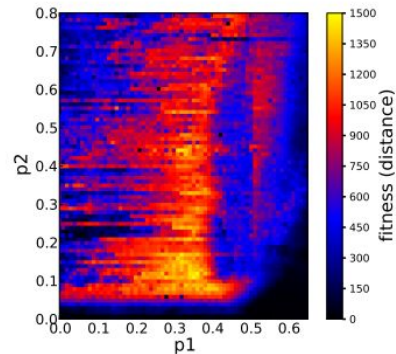
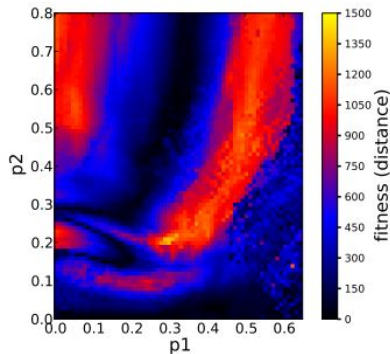
- Add noise to encourage robust behaviour
- Lots of computation!
 - 64 V100 GPUs and 920 32- core CPUs training for several months
 - 13 000 “years of experience”



[Solving Rubik's Cube with a robot hand | OpenAI](#)

How to deal with reality gap

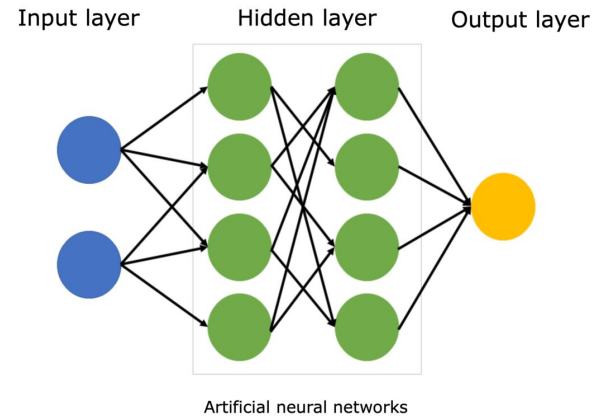
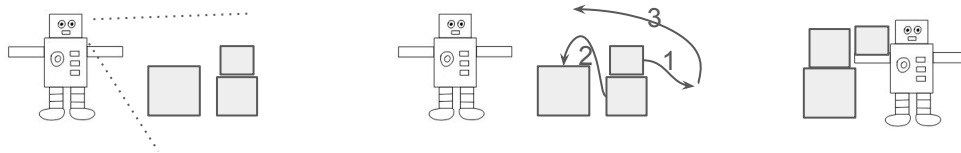
- Evaluate solutions in real life during optimization process
- Learn a “transferability function”: Score how well simulator matches reality



Mouret, J. B., Koos, S., & Doncieux, S. (2013). Crossing the reality gap: a short introduction to the transferability approach.

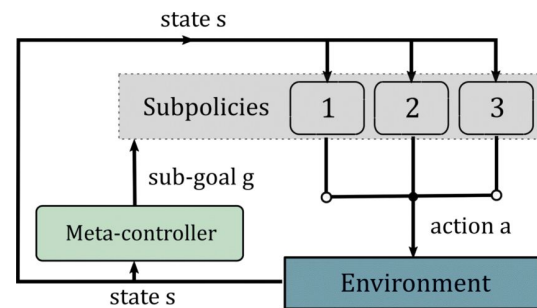
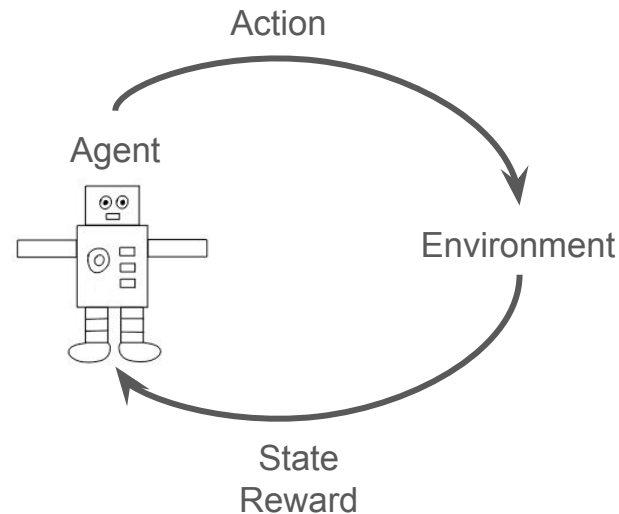
Summary of the lecture

- **Fixed** control mechanisms work well in **predictable** environments
- For **unpredictable environments** we can make use of **AI algorithms**
- AI robots **sense** the environment and interact accordingly
 - Common sensing techniques: Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks
 - Sense -> Plan -> Act



Summary of the lecture: Reinforcement learning

- Robot control is often done using Reinforcement Learning:
 - Trial and Error
 - Goal is to find a **policy** for the agent/robot to **maximize reward**.
 - With **Deep reinforcement learning**, we use a Neural Network as agent
- Reinforcement Learning applied to robots:
 - Imitation learning: Learn from experts
 - Hierarchical reinforcement learning: Split the problem into smaller sub-problems
 - Model predictive control: Look into the future by simulating actions
 - Other methods not covered in this lecture exist too



Summary of the lecture: Evolutionary robotics and the Reality gap

- Evolutionary robotics: Use evolutionary algorithms to design robots.
 - Combine Evolutionary robotics with Reinforcement learning: Use ER to make new robot designs, use RL to train them
- Reality gap: Simulations are faster but less accurate
 - The solutions evolved in simulation behave differently when applied to the real robot.
 - Solutions: Improve simulations, add noise for robustness, create a function that maps simulation performance to reality, test solutions in real life during optimization

