CS 175: Project in Artificial Intelligence

Winter 2025

Introduction

Roy Fox
Department of Computer Science
School of Information and Computer Sciences
University of California, Irvine

Today's lecture

Course overview

What is a project

What is reinforcement learning

Project ideas

Learning goals

Practice AI/ML

- Be creative about what problem to solve
- Get a feel for what's practical to solve and how
- Implement and debug a machine learning pipeline

Software Engineering

- Design and implement a complex software system
- Use modern software practices
- Experience collaborative software development

Presentation Skills

- "Sell" your ideas in writing, figures, and talk
- Present your project in a convincing manner
- Document and maintain a project website

Lectures and assignments

- Lectures in weeks 1 and 2
  - Overview of project expectations and ideas
  - Introduction of general principles of reinforcement learning (RL) in a nutshell
  - Many online resources; no discussion section
- Exercises due in weeks 3 and 4
  - Install one project platform
  - Implement and experiment with basic RL algorithm
- No exams

Project meetings and presentations

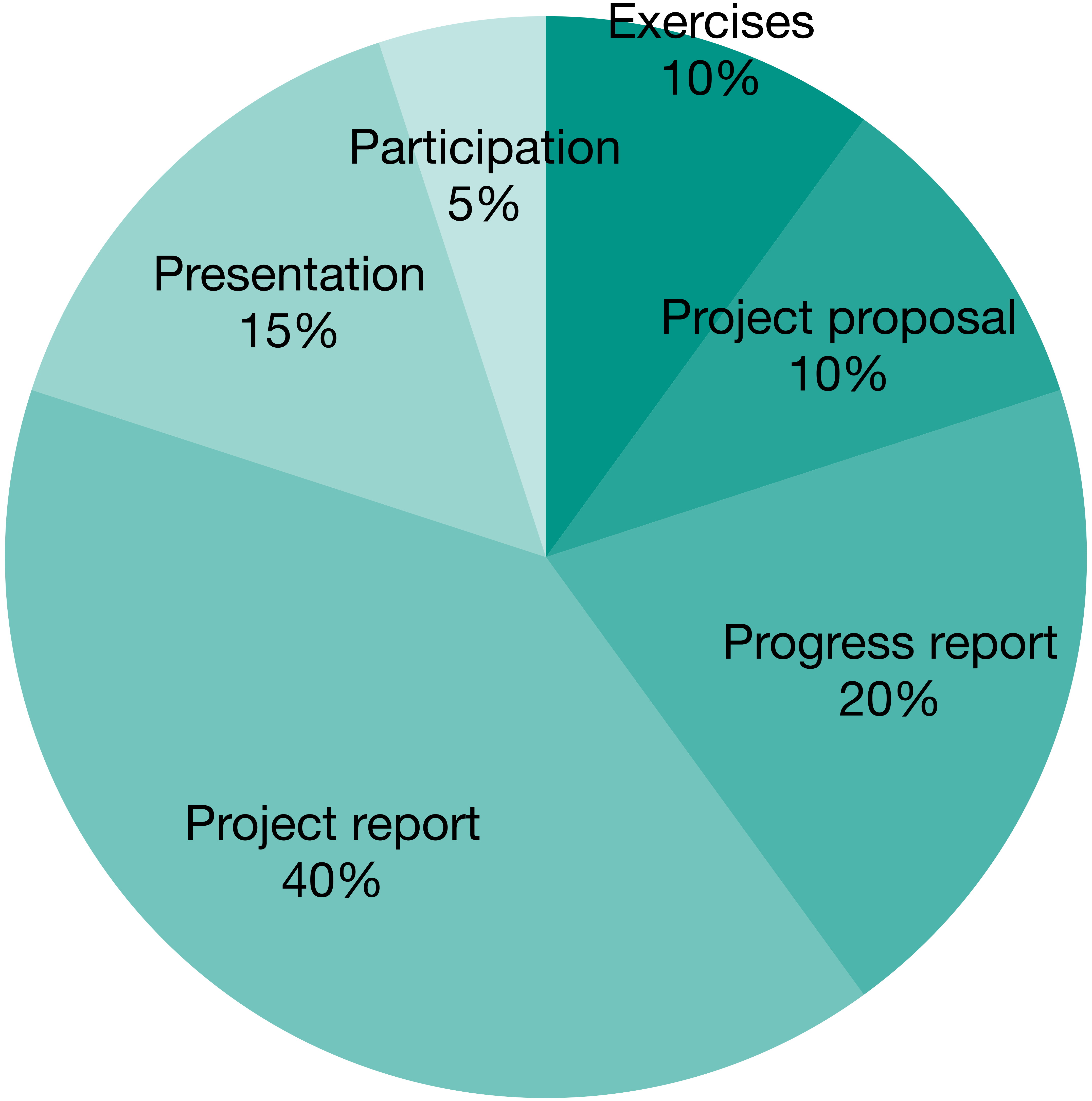
- Project timeline:
  - Week 3: team formation (3 students per team) + proposals
  - Continually: reading > thinking > implementation > experimentation > evaluation
  - Week 7: progress reports
  - Weeks 10 and 11: final reports, live presentations
- Project meetings:
  - Teams should meet regularly
  - Meet with course staff as often as you want; at least twice by weeks 5 and 9

Course logistics

- When: today and next Wednesday, 5pm–7:50
  - Week 11 presentations TBD
- Website: https://royf.org/crs/CS175/W25 ← Schedule! Resources!
- Forum: https://edstem.org/us/courses/71615
  - For announcements and questions (do not email)
- Exercise submission: https://www.gradescope.com/courses/945388
- Office hours: in-person or on zoom, more times available by request
  - TA: JB Lanier, office hours

Grading policy

- Exercises (weeks 3+4)
- Project proposal (week 3)
- Progress report (week 7)
- Project report (week 10)
- Presentation (week 11)
- Grace days:
  - Exercises: 3 days total per person
  - Project: 5 days total per team



How to participate

- Meetings
  - Show up prepared, ask questions, engage in discussion
- Forum
  - Ask questions if you have any, answer if you can
  - Post relevant useful links
  - Upvote useful posts
  - Give private feedback to staff
  - Logistics questions and comments appreciated, but substance counts
- Evaluations

Today's lecture

Course overview

What is a project

What is reinforcement learning

Project ideas

Project paradigms

- Application-driven
  - Identify a worthwhile task (or collection) and understand why it's hard
  - Use ~~any~~ means necessary to get an agent to (learn how to) perform the task(s)
    - Can be off-the-shelf methods, their adaptation, combination, or something new
- Method-driven
  - Study what makes a method good and/or make it better
  - Theory (analyze and prove), empirical science (measure), or engineering (build)
    - Show benefits on toy examples, simulations (simplified or not), or real domains

Application-driven projects

- Applications inform innovation
  - Can’t just define / assume / modify your way around challenges
  - Doesn’t mean you can’t choose your battles
    - Create stepping stones by simplifying hard problems
    - Know when to change approach, think outside the box, walk away, come back
- Bridging problems and solutions is key
  - Identify data, modeling assumptions, decompositions, pipelines, auxiliary tasks
  - May require domain knowledge, experimentation, adaptation

Method-driven projects

- Not all future applications need groundbreaking methods, but many do
- A method is measured by how it evaluates across tasks
  - Quantitatively and qualitatively
  - Benefit / applicability tradeoff
    - Narrower applicability is justified when benefits are large / value is high
    - Also matters: can you predict if a method is applicable / beneficial to a task?
  - But it’s not all about the technology: there’s science, art, education, recreation, …

Quantitative evaluation

- Expected rewards: may be what we really care about, or arbitrary
- Task success rate: may be what we really care about, or undefined
- Worst-case / safety violations
- Resource requirements
  - Sample complexity, expert supervision, learning / deployment compute, memory
- Compare to baselines / ablations
  - Don’t need to win on all / any metrics to be interesting
  - Show which aspects of the method matter for which aspects of the task

Qualitative evaluation

- Illustrate on toy examples
  - What does the solution look like? Is that expected? Desired?
  - Build intuition for the core task challenges and key method operation
  - How far can you push the method's benefits?
- What is the moonshot application(s)?
- Does the agent behavior exhibit interesting properties? Expected? Desired?
- Dirty laundry: what do failure modes look like? Any pattern?
  - Recommend when to use / avoid this approach? Detect failures? Future ideas?

FECs (Frequent Existential Crises)

- Is this project interesting? Significant? Impactful?
  - Why am I even doing this?
    - Why does anyone do anything?
- Is this task too hard? Too easy?
  - Is 7 weeks enough to make progress? Will the course staff be impressed?
- Am I using the right method? Right evaluation?
- Do I have enough data? Model size? Training time?
- Do I have a bug? 😱😱😱

Today's lecture

Course overview

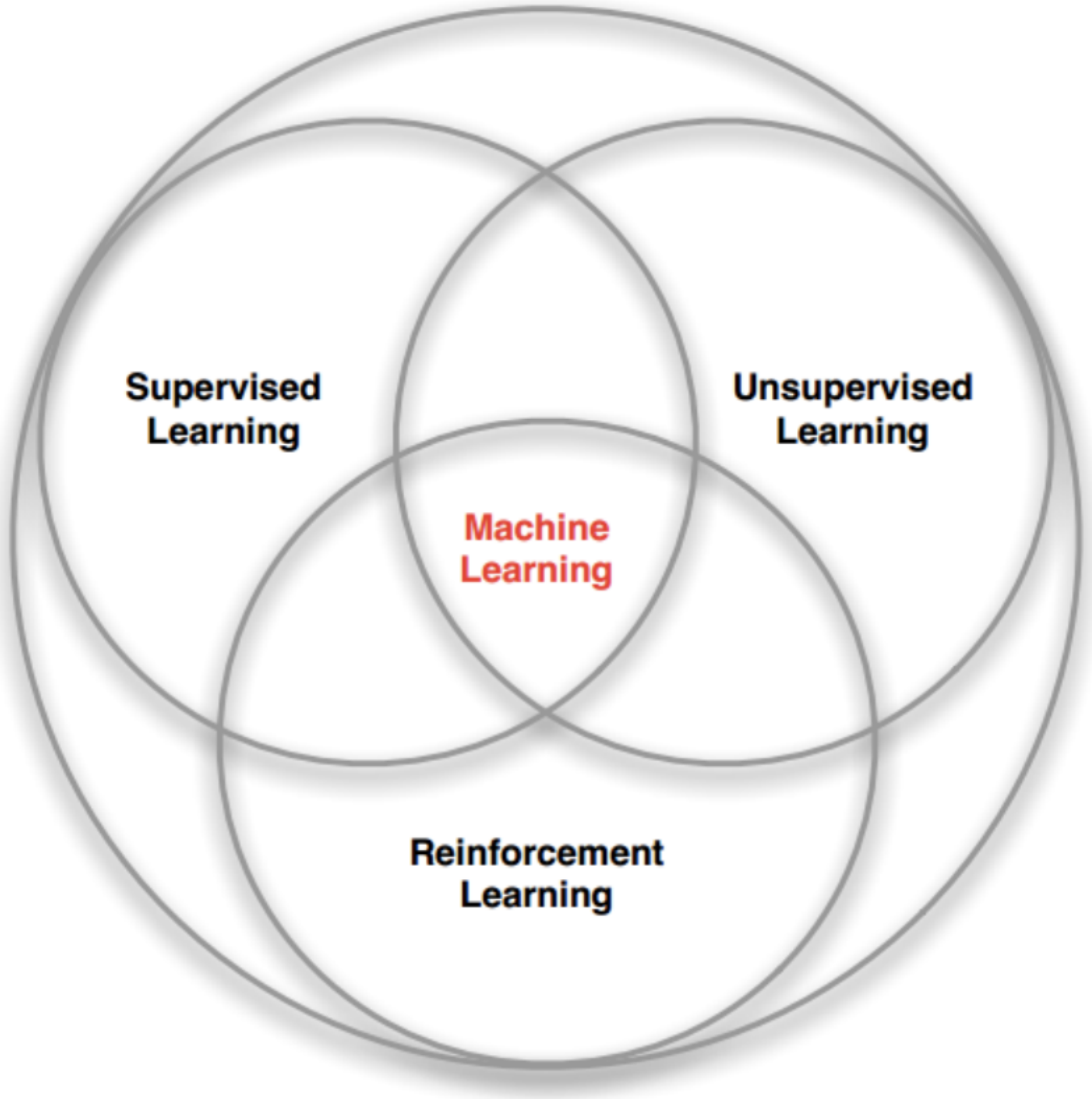
What is a project

What is reinforcement learning

Project ideas

RL ⊆ control learning ⊆ ML

- Reinforcement Learning = learning from reinforcement (rewards)
  - But it came to encompass many settings of learning to control
  - Distinguished by sequential decision making and learning
- Many consider RL a separate ML paradigm, but it can involve:
  - Supervised learning
  - Unsupervised learning
  - Active learning
  - Online learning



What is machine learning

- Can we build “intelligent” machines? Intelligence = good decision making

- Learning = taking in information to “know” more than you did before

- Machine learning = use data to make better decisions than before [Mitchell 1997]

- ML can help when other AI methods fail:

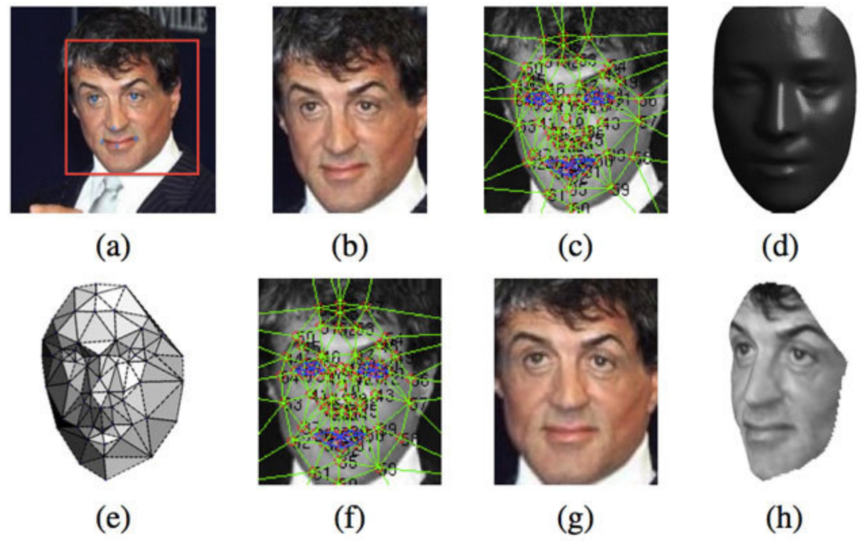
  - Experts are scarce

  - Rules / logic are hard to specify

  - Search space is too large

  - Models are unknown / hard to specify

Face recognition



Speech synthesis

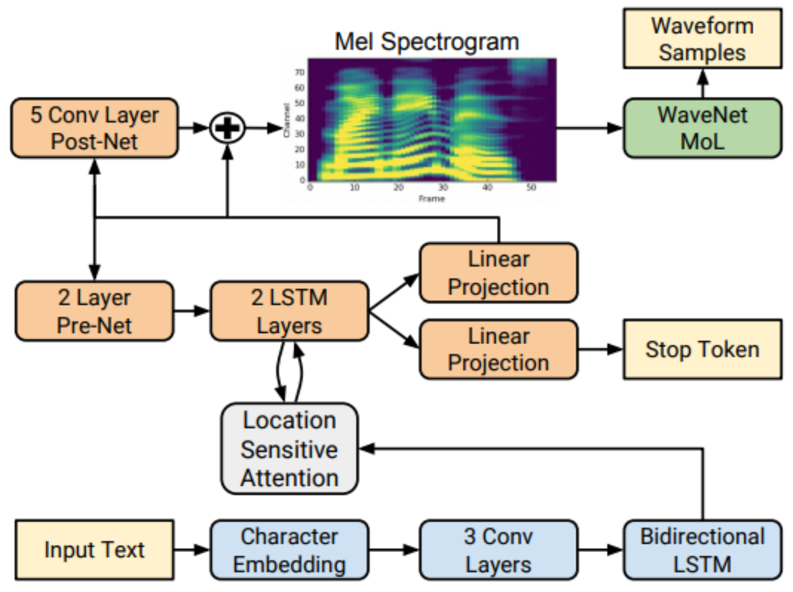
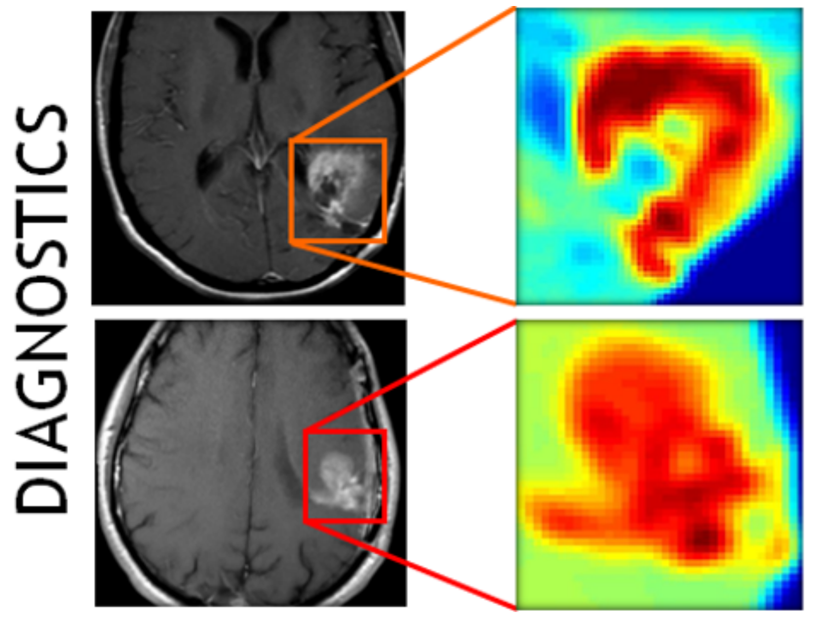


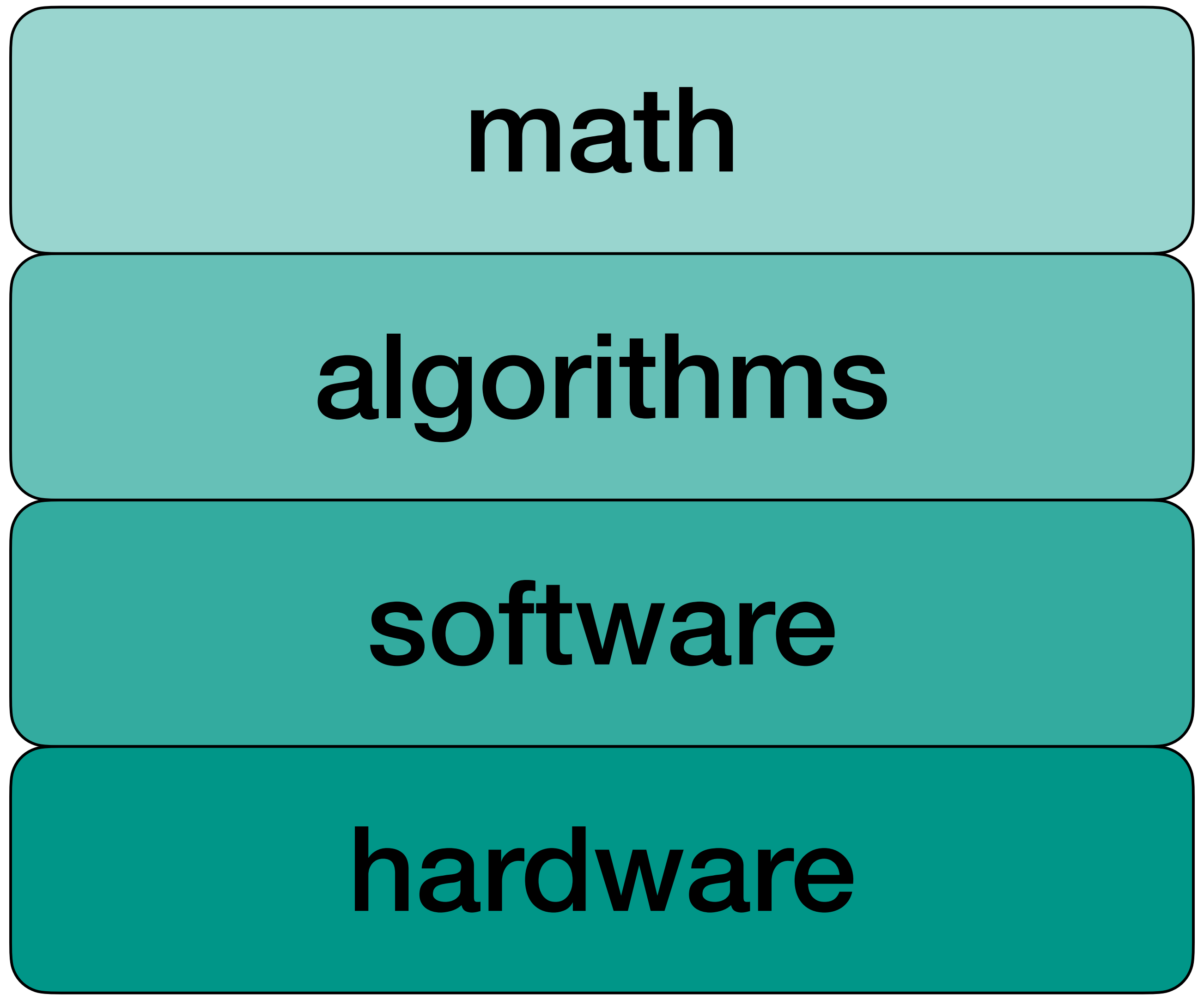
Fig. 1. Block diagram of the Tacotron 2 system architecture.

Medical diagnosis



[Taigman et al., 2014; Shen et al., 2018]

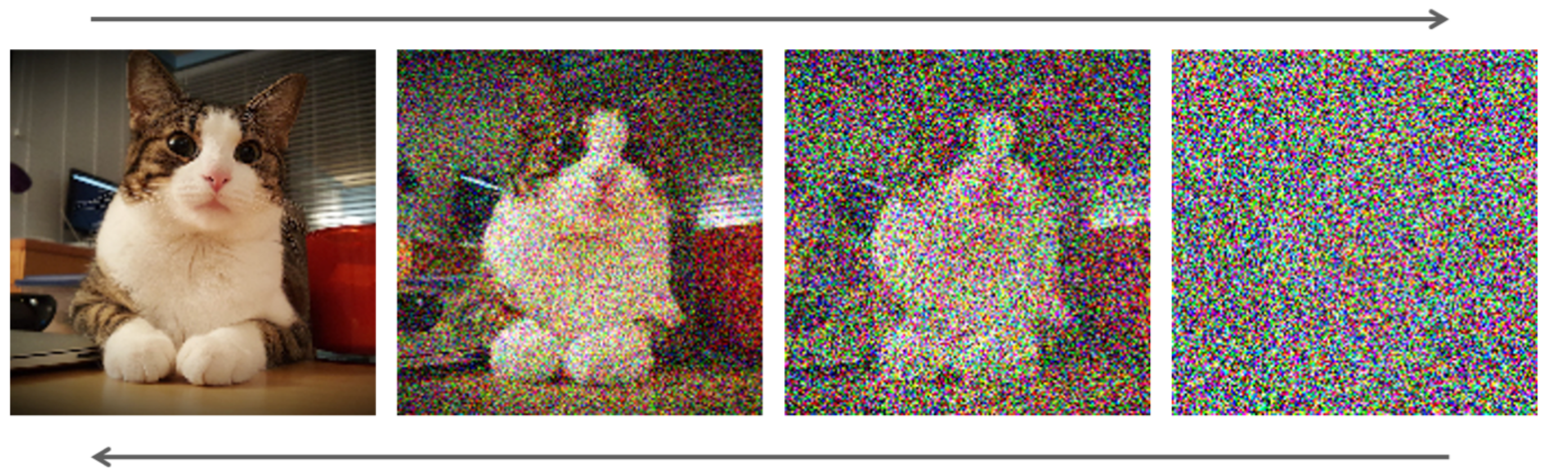
The ML stack



- Math: probability theory, (linear) algebra, computational learning theory
- Algorithms: ML algorithms, optimization, data structures
- Software: ML frameworks, databases, testing, deployment
- Hardware: cloud computing, distributed systems, cyber-physical systems

ML success stories

Image generation



Language generation

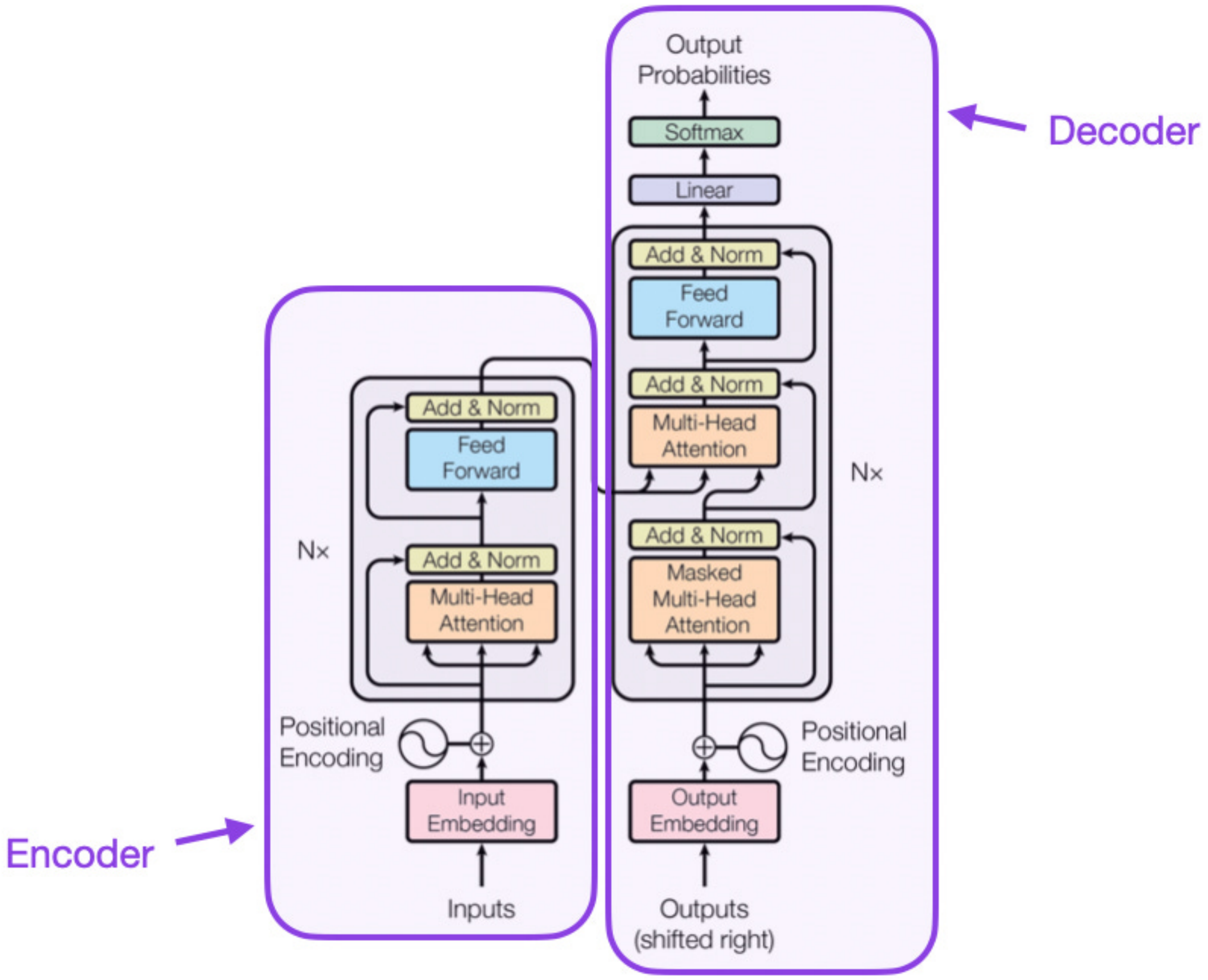
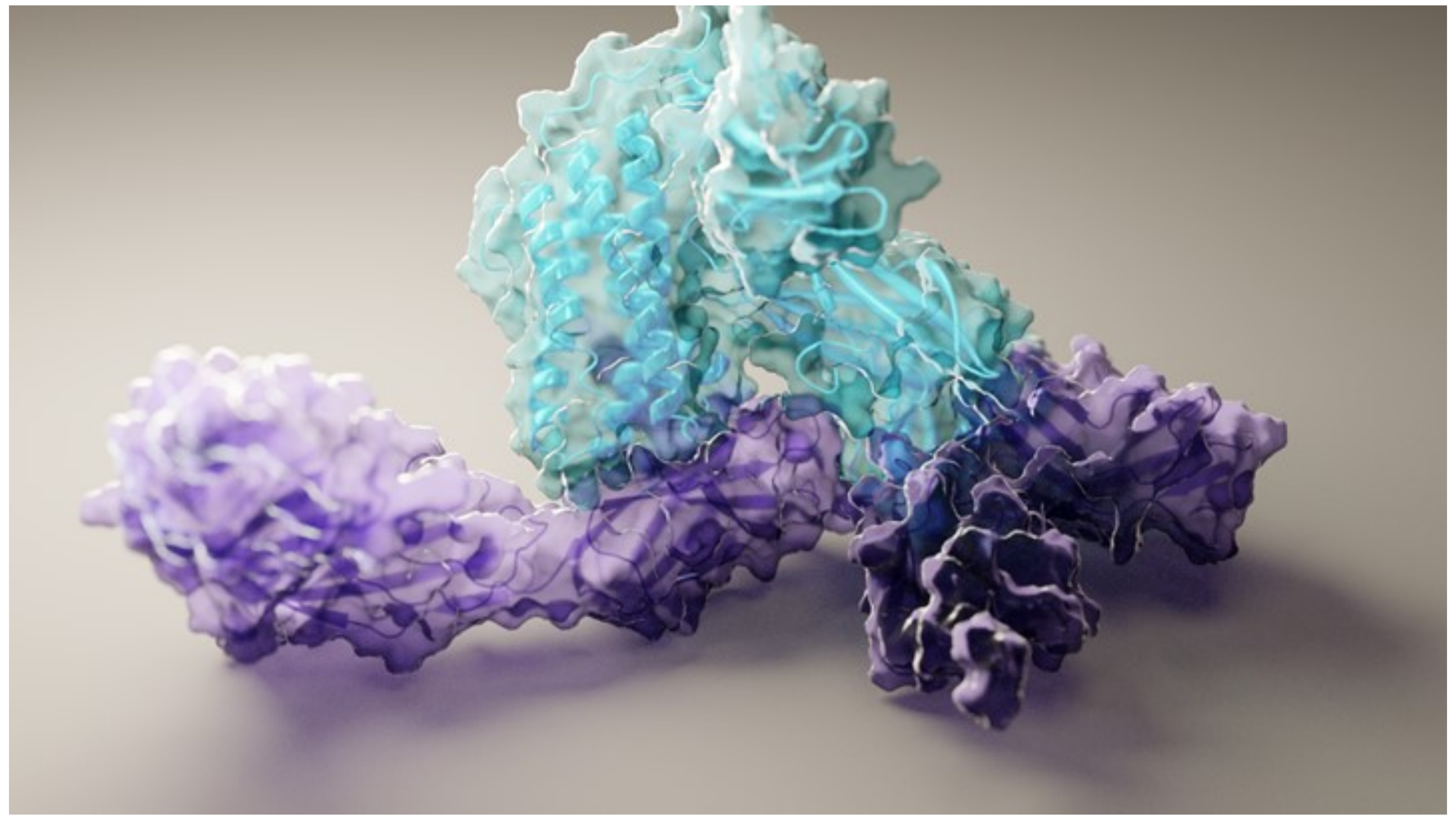


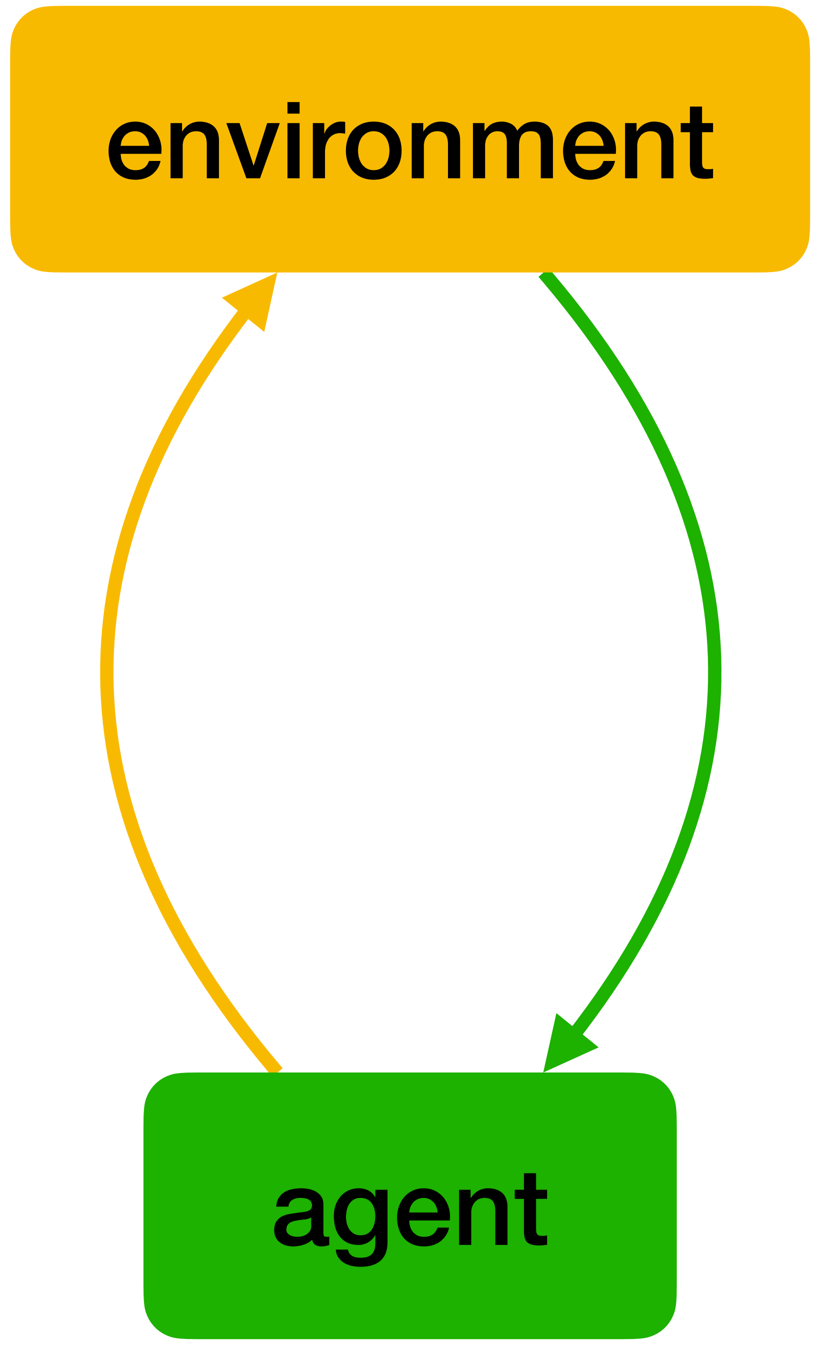
Figure 1: The Transformer - model architecture.

Protein folding



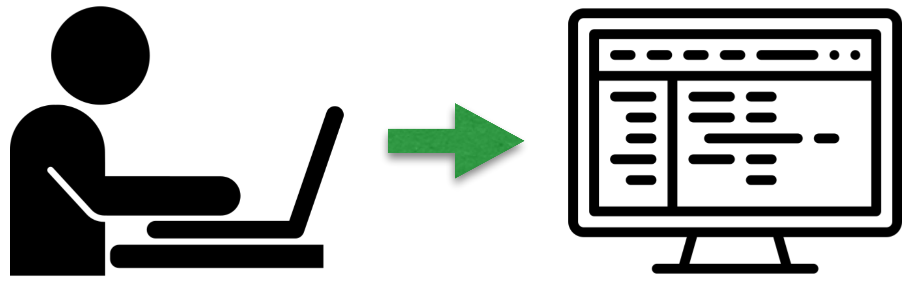
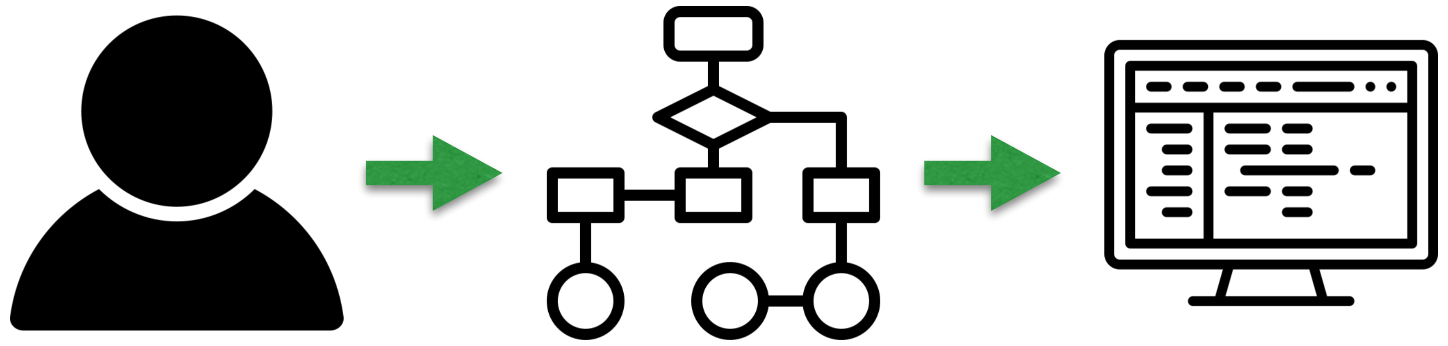
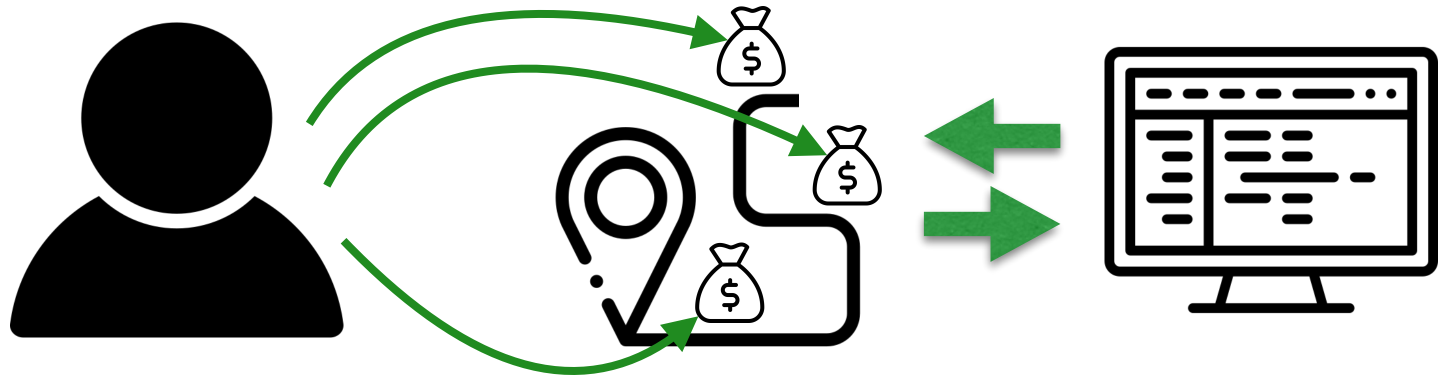
What is control learning (CL)?

- Intelligence appears in interaction with a complex system, not in isolation
  - An agent interacting with an environment
- Control = sequential decision making
  - Sense environment state s
  - Take action a
  - Repeat



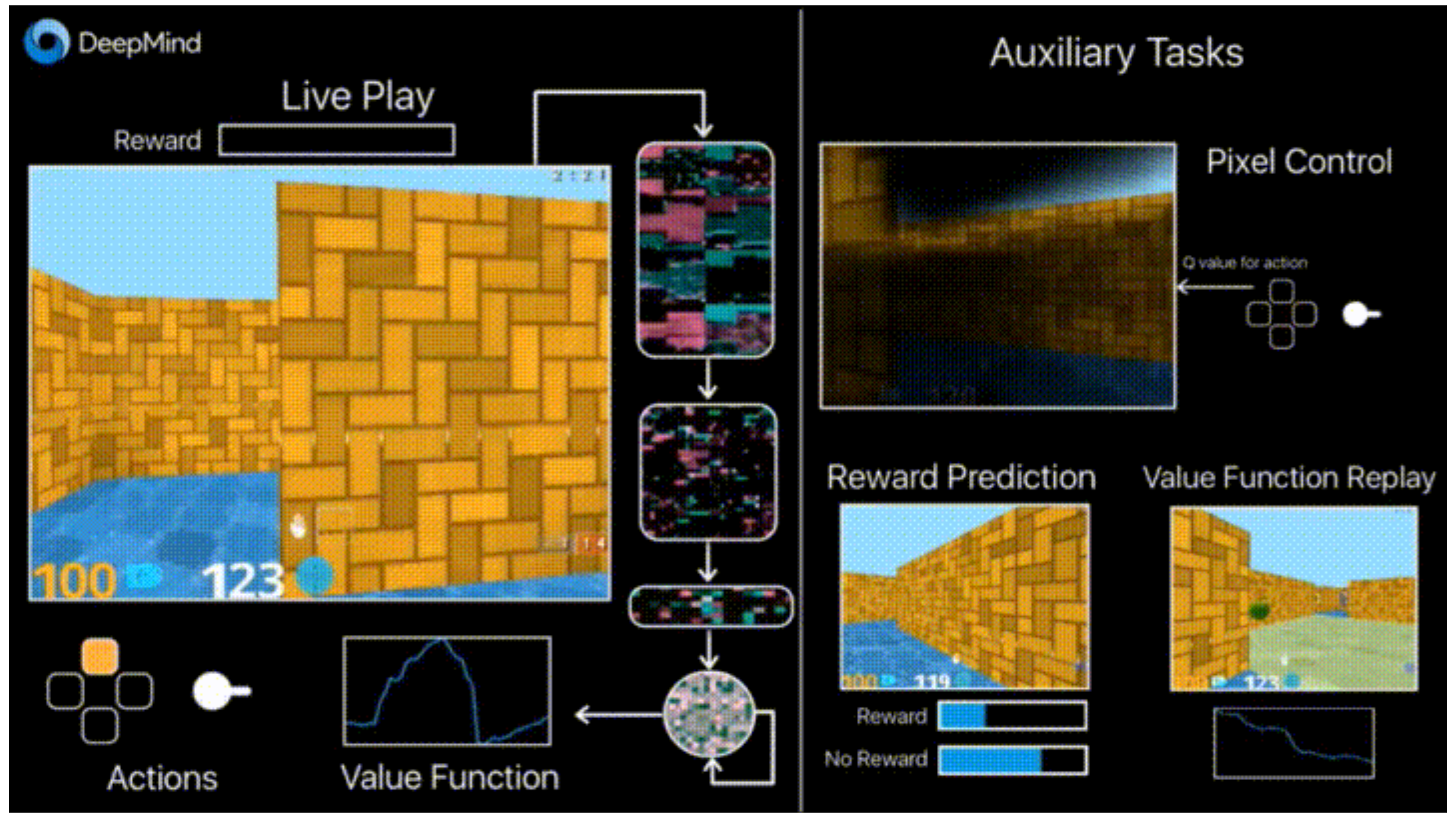
- Success can be measured by matching good actions — imitation learning (IL)
  - Or by accumulating high rewards $r(s, a)$ — reinforcement learning (RL)

Control preference elicitation

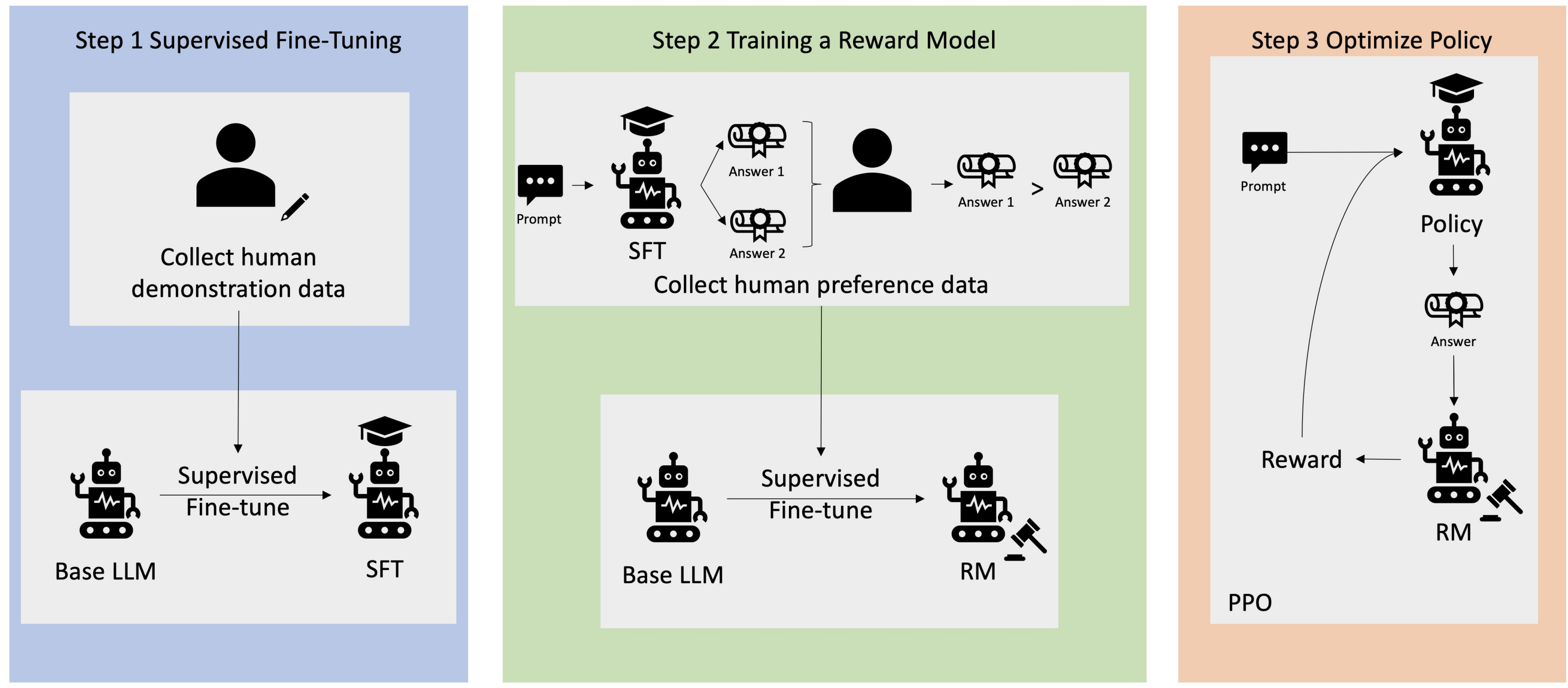
	Explicit	Implicit
"how"	**Programming** 	**Imitation Learning** 
"what"	**Instruction Following** 	**Reinforcement Learning** 

RL success stories

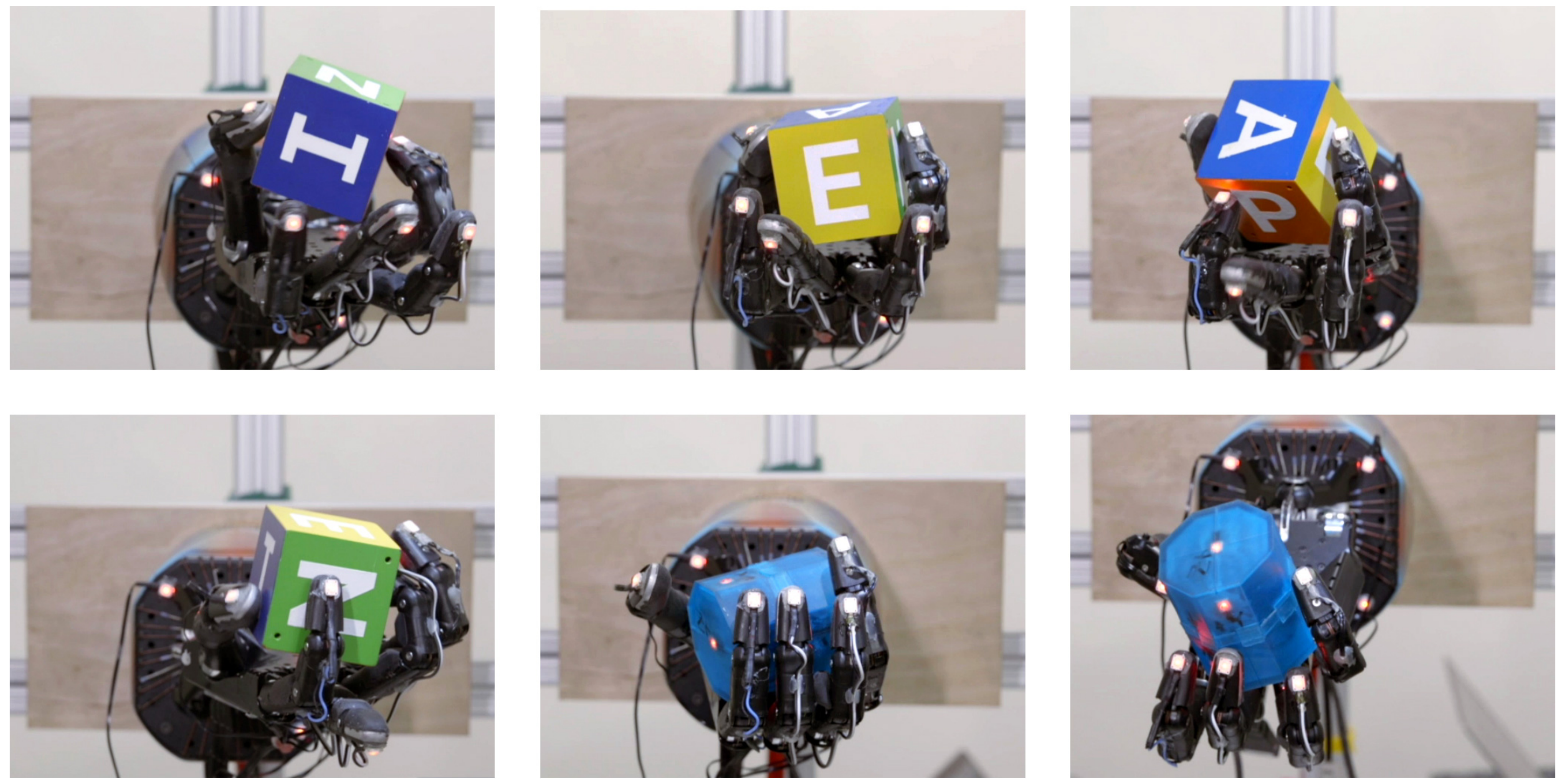
Spatial navigation



Generator fine-tuning



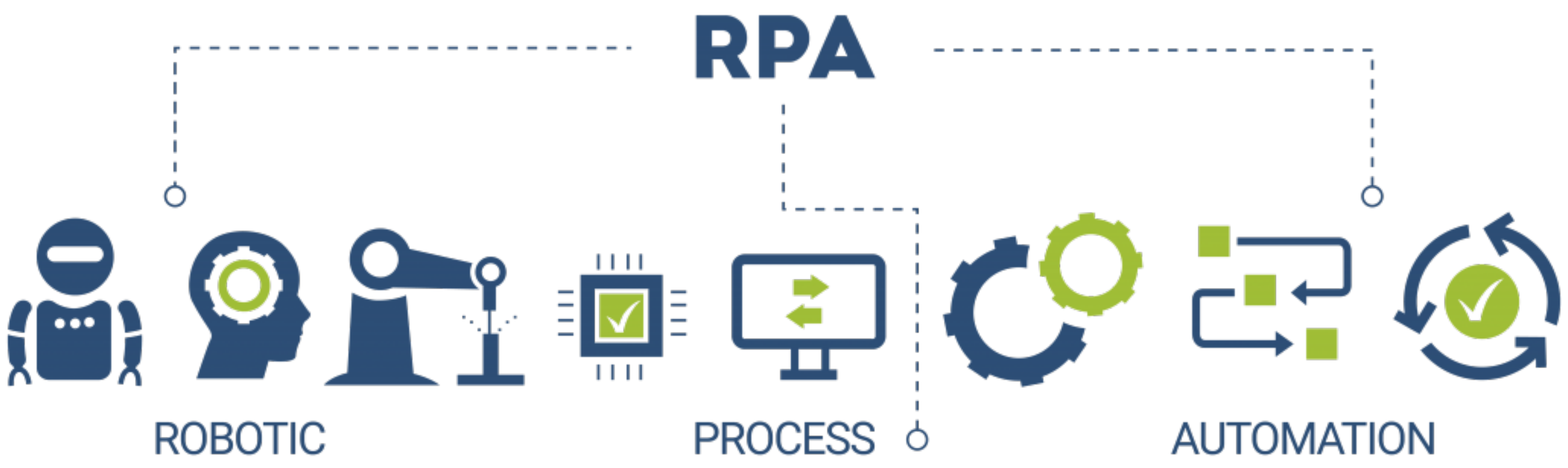
Dextrous manipulation



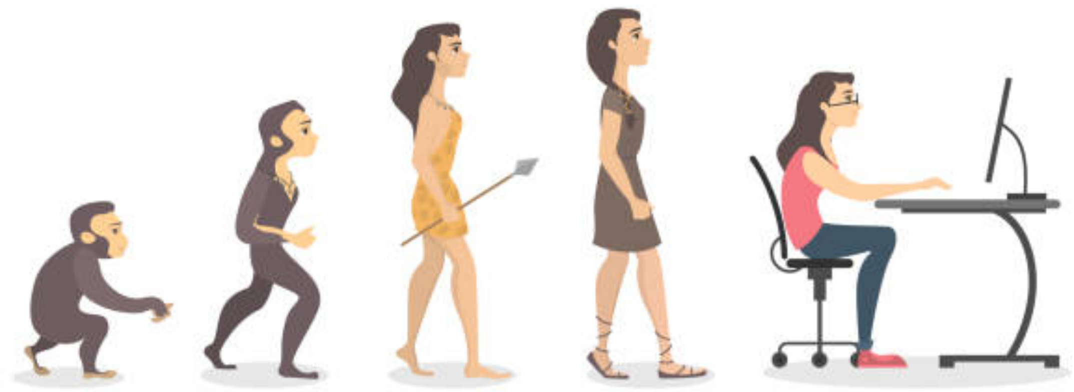
RL is ML... but special

- In RL, unlike supervised, no ground truth, only feedback (online learning)
- Exploration = the learner collects data by interaction
  - The agent decides on which states to train (active learning) — and test!
  - Cannot avoid some train–test mismatch
- Sequential decision making need to be coordinated
  - Optimization space is strewn with local optima
- A good policy may require memory
  - Agent state is latent → combine control and inference

Why is RL powerful?



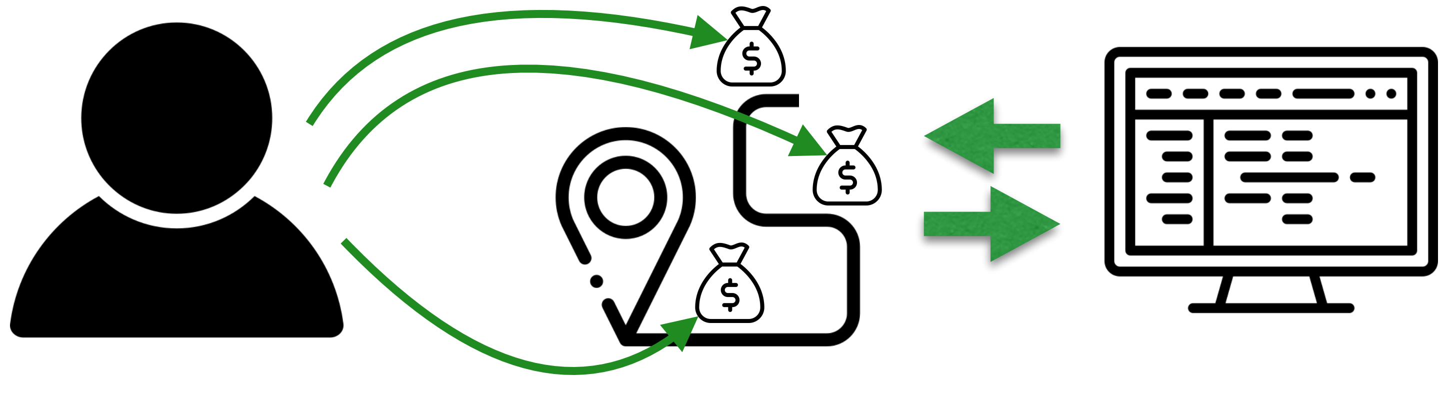
- Many (all?) problems can be formulated as control
  - But consider: is it sequential? multi-agent? a more specific structure?
- Active + online = very little supervision
  - Even incidental, like in evolution! Supervisor can be “surprised”



- More general CL: incorporate stronger supervision
  - Supervisor burden is a tradeoff between data amount ↔ informativeness

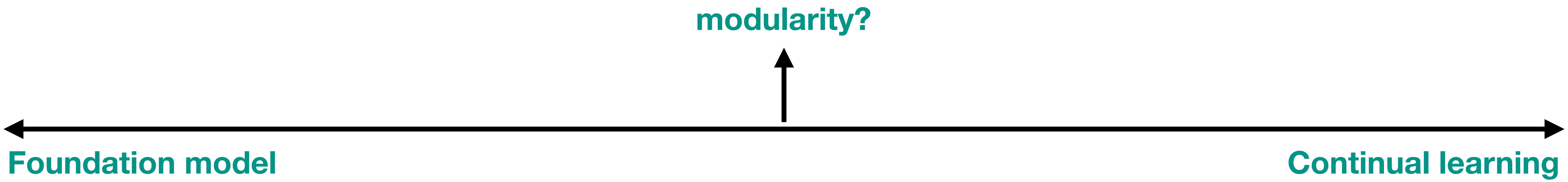
How is RL different?

online

	Provided correct output	Provided feedback
Teacher selects input (demo)	**IL from demonstrations** 	**Offline RL** 
Learner selects input (interact)	**IL from corrections** 	**RL** 

active

What would “solving” RL look like?

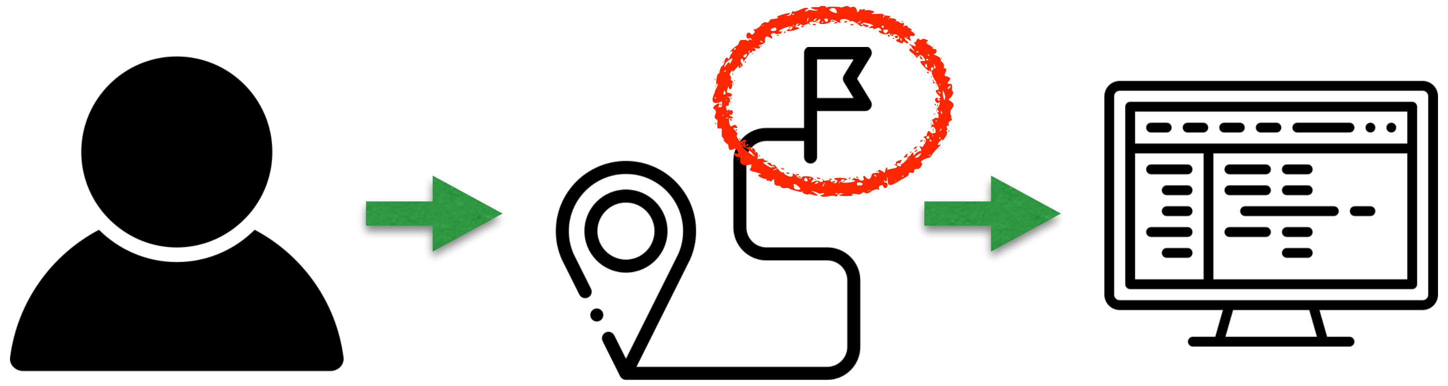
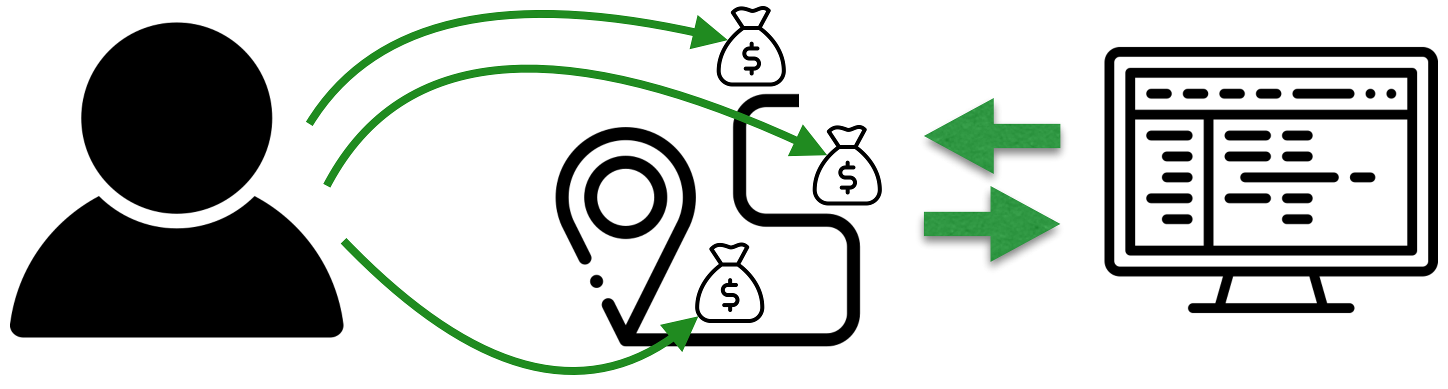


- Foundation model?
  - Large model
  - Huge amount of data
  - Centrally trained
  - Fine-tuned, built into pipelines
- Continual learning?
  - Flexible model
  - Ad-hoc data
  - Distributed learning
  - Mixed supervision, shared learning

The last ML frontier?

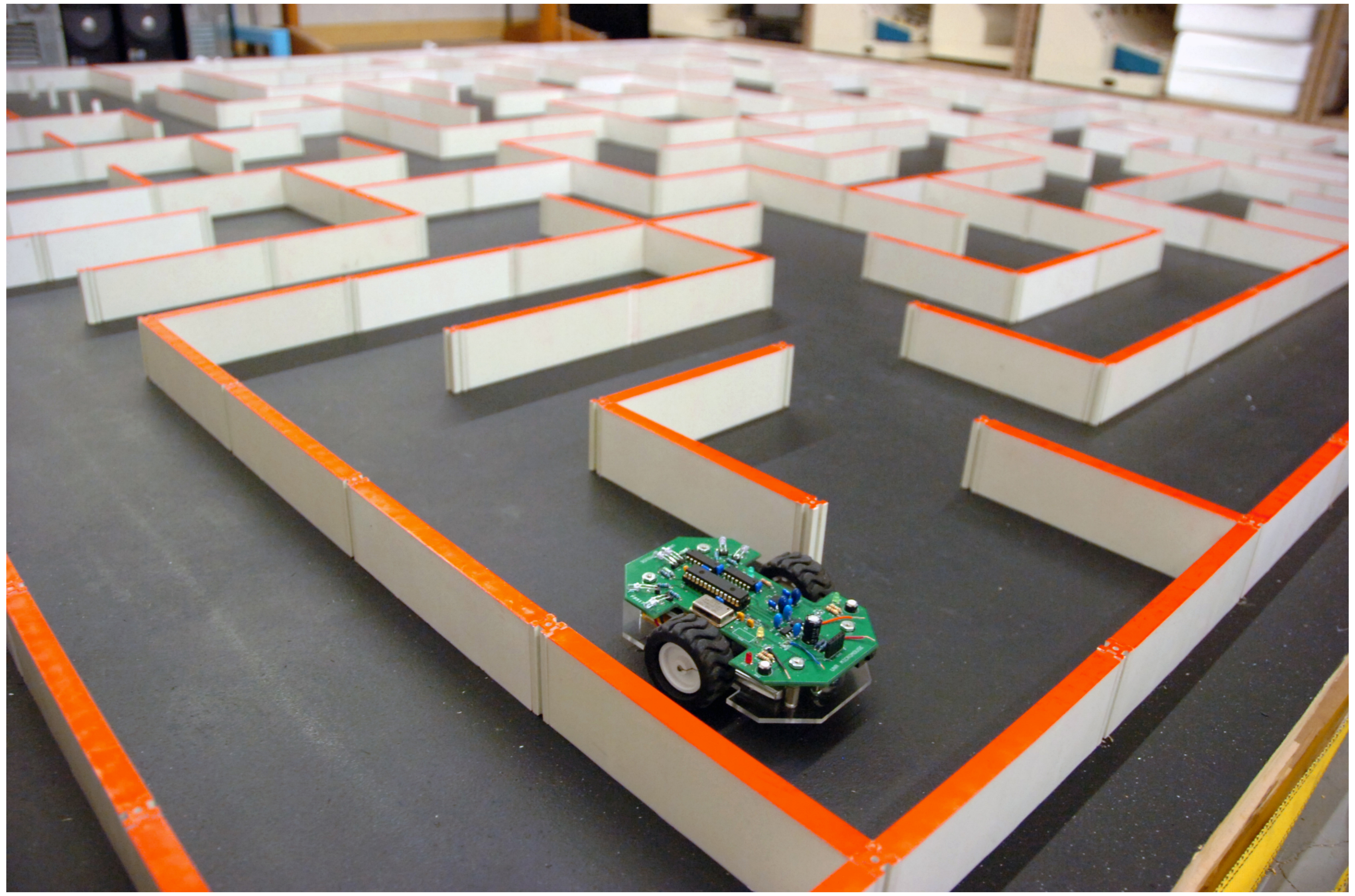
Why is RL hard?

- It's all about the data: amount and informativeness

	Provided correct output	Provided feedback
Teacher selects input (demo)	**IL from demonstrations**  **expert, train–test mismatch**	**Offline RL**  **extreme train–test mismatch**
Learner selects input (interact)	**IL from corrections**  **hard to give** **exploration**	**RL**  **weak signal, exploration**

After the break: Basic RL concepts

System state



$s_{t-1} \rightarrow s_t \rightarrow s_{t+1}$

System state

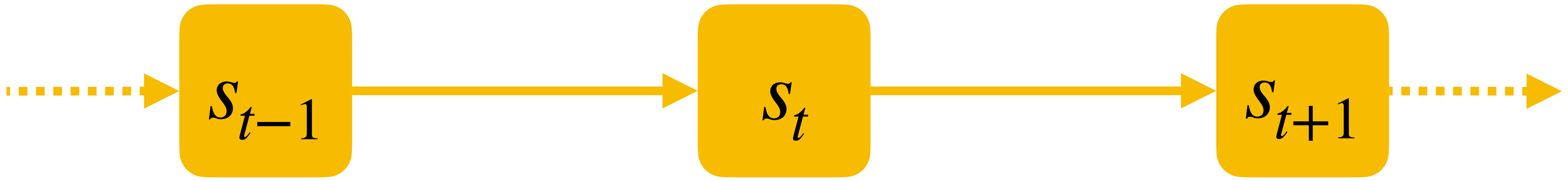
- Markov property: the future is independent of the past, given the present

$$p(s_{t+1}, s_{t+2}, \ldots \mid s_0, s_2, \ldots, s_t) = p(s_{t+1}, s_{t+2}, \ldots \mid s_t)$$

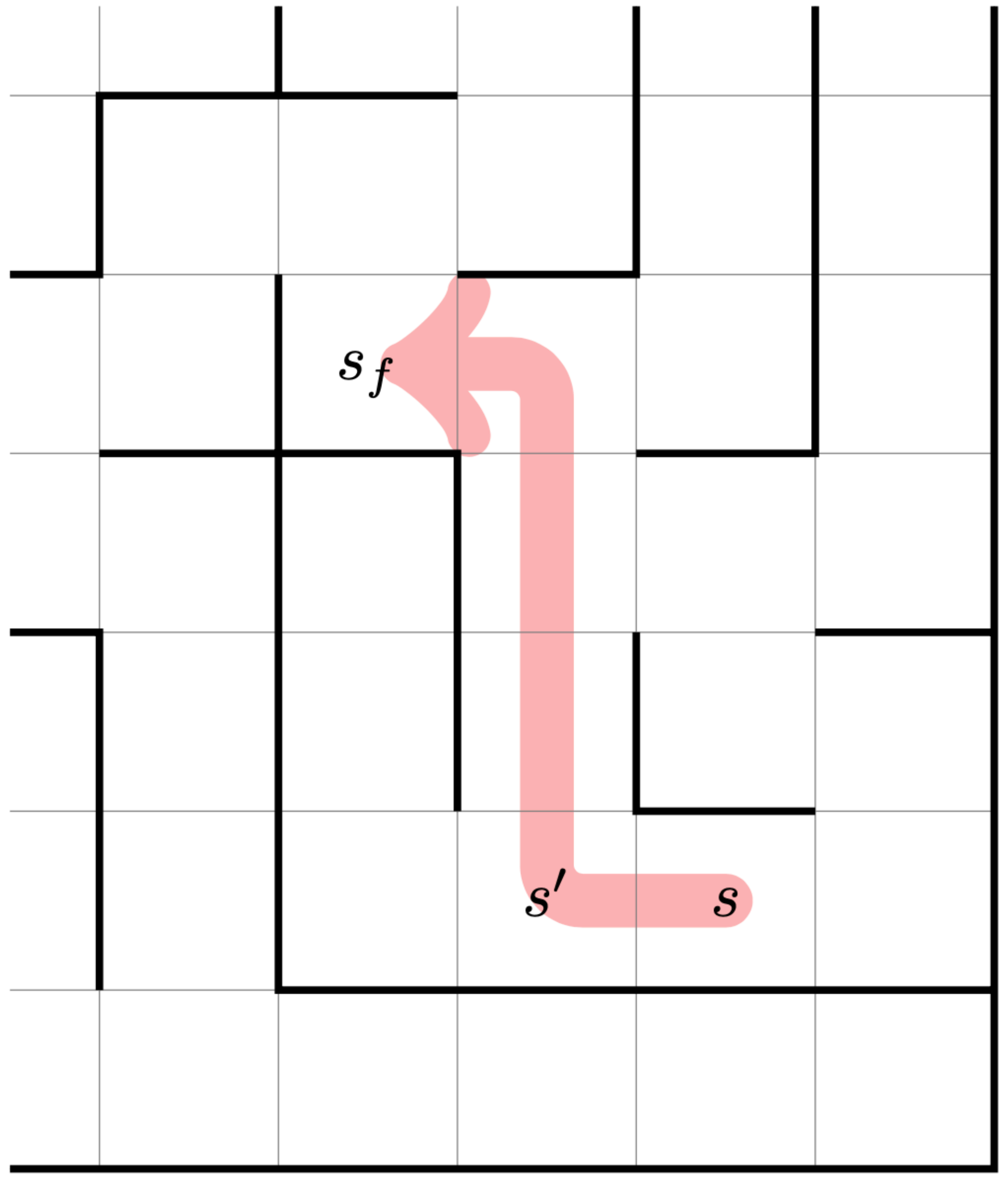
- State = all relevant information from history

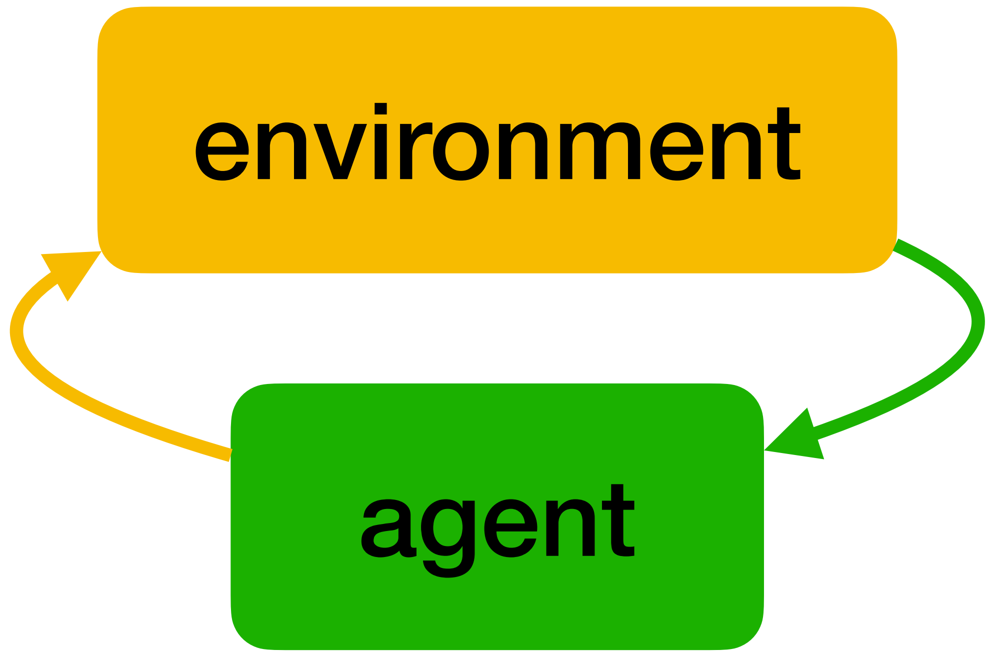
  for future!

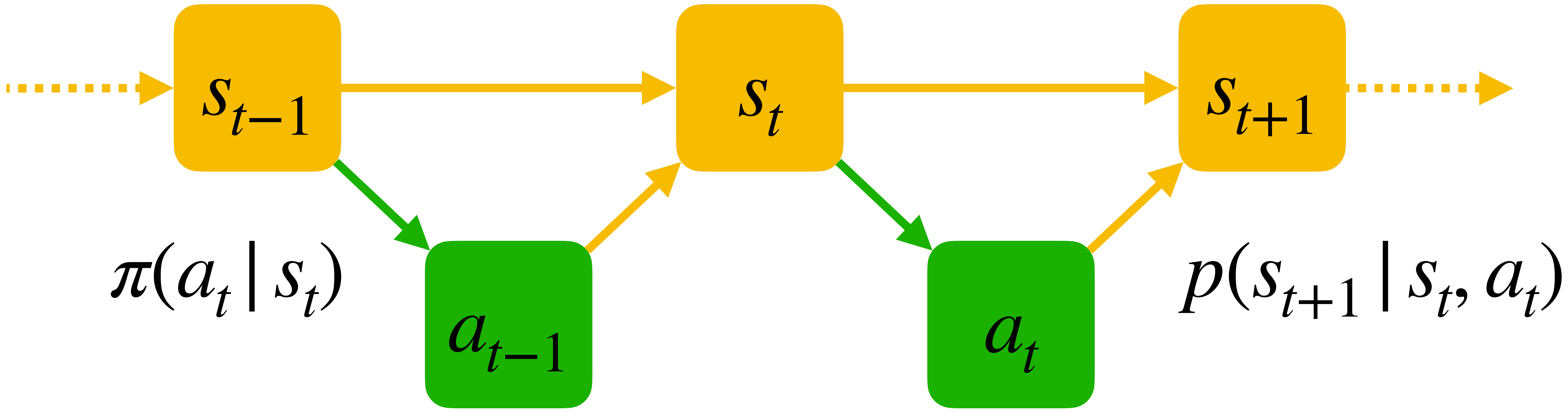
  - Given s_t, the history $h = (s_0, \ldots, s_t)$ and the future $(s_{t+1}, s_{t+2}, \ldots)$ are independent



System = agent + environment

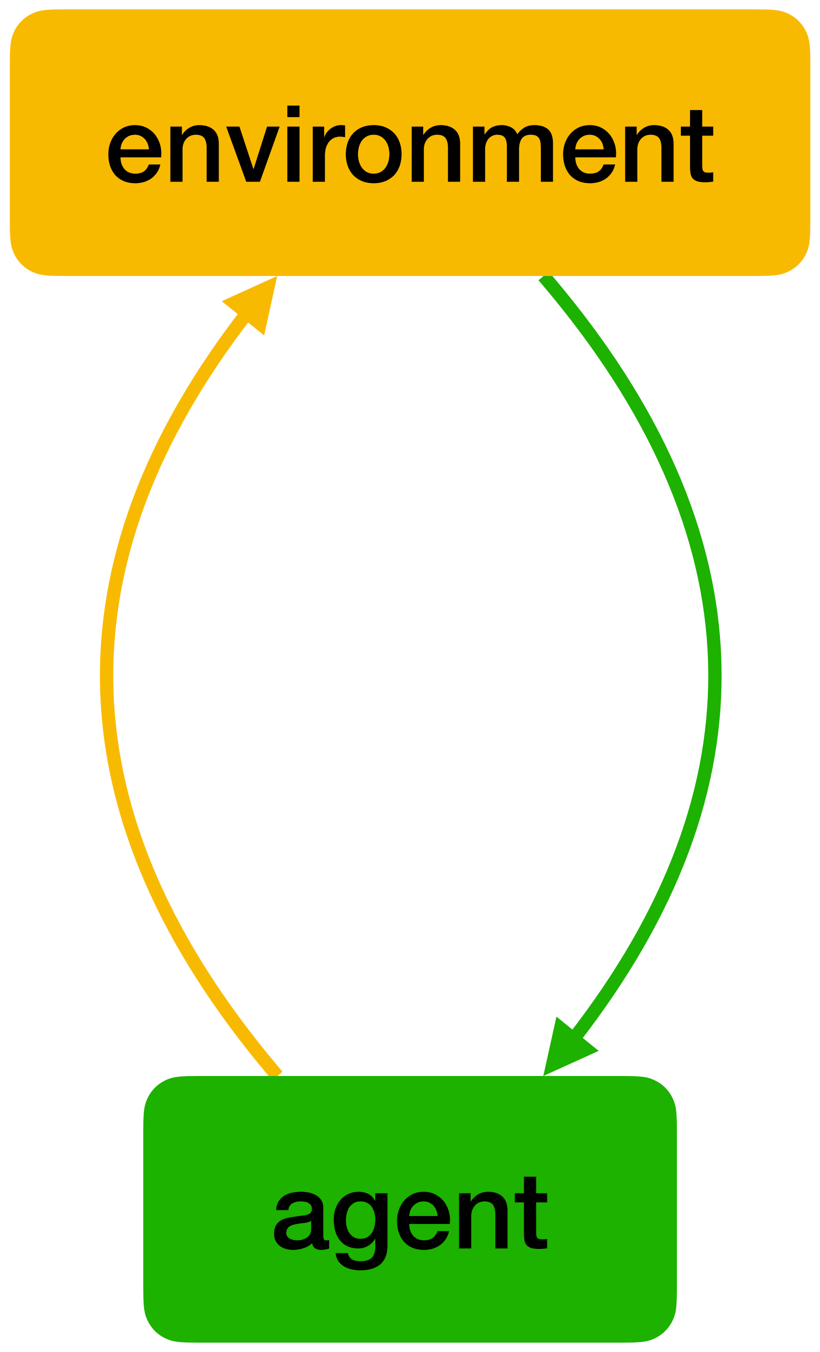






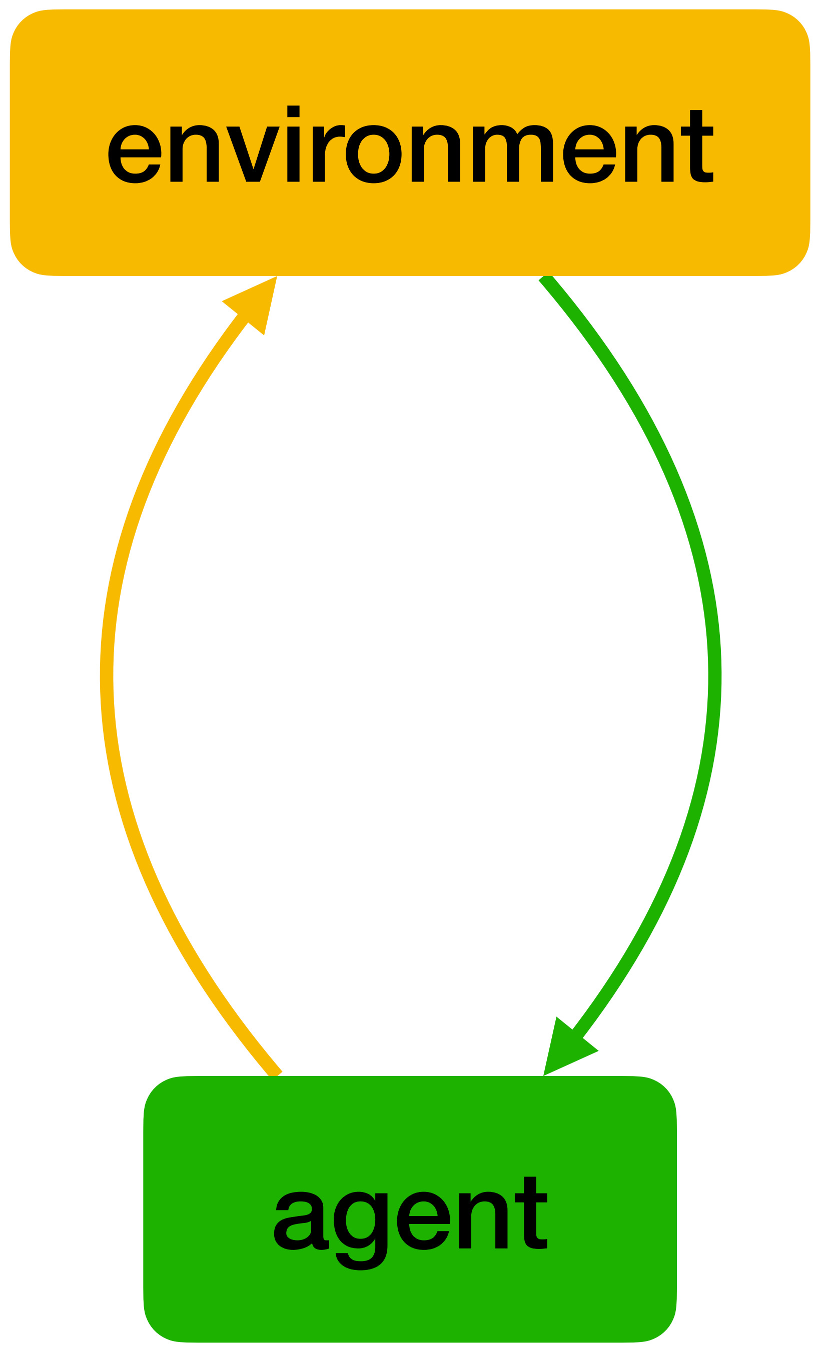
Markov Decision Process (MDP)

- Model of environment
  - $\mathcal{S}$ = set of states
  - $\mathcal{A}$ = set of actions
  - $p(s' \mid s, a)$ = state transition probability
    - Probability that $s_{t+1} = s'$, if $s_t = s$ and $a_t = a$



Agent policy

- "Model" of agent decision-making
  - Policy $\pi(a \mid s)$ = probability of taking action $a_t = a$ in state $s_t = s$
  - In MDP, action a_t only depends on current state s_t:
    - Markov property = s_t is all that matters in history
    - Causality = cannot depend on the future
  - Should the policy depend on time? $\pi_t : s_t \mapsto a_t$
    - Sometimes; can add t as feature: $s_t \to (t, s_t)$



Trajectories

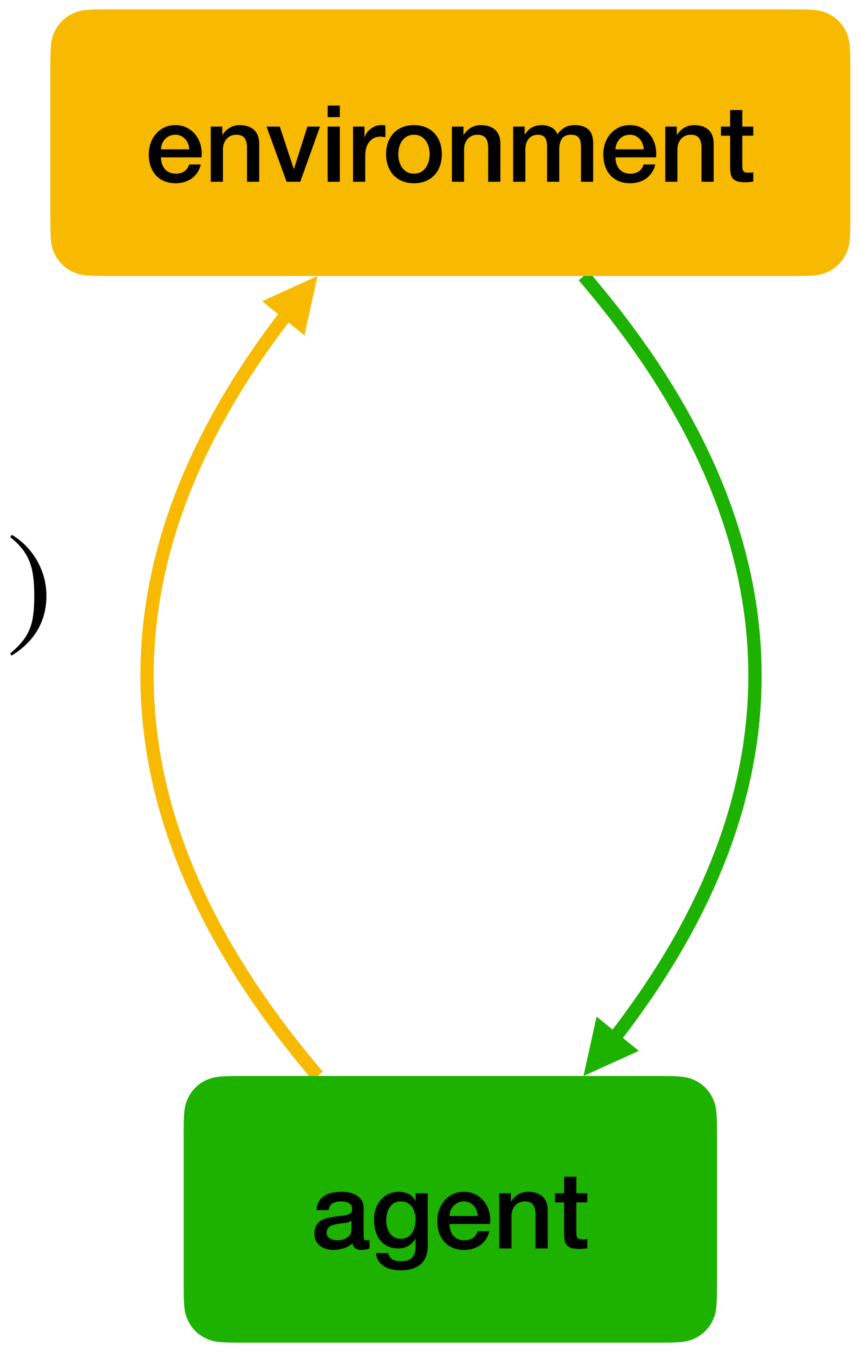
- The agent's behavior iteratively uses (rolls out) the policy

- Trajectory: $\xi = (s_0, a_0, s_2, a_2, \ldots, s_T)$

- MDP + policy induce distribution over trajectories

$$p_\pi(\xi) = p(s_0)\pi(a_0 \mid s_0)p(s_1 \mid s_0, a_0)\cdots\pi(a_{T-1} \mid s_{T-1})p(s_T \mid s_{T-1}, a_{T-1})$$

$$= p(s_0)\prod_{t=0}^{T-1}\pi(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t)$$



- Imitation learning: learn from dataset of expert demonstrations
  - Supervised learning of $\pi(a \mid s)$ from “labeled” states (s_t, a_t)

Learning from rewards

- Providing demonstrations is hard
  - Particularly for learner-generated trajectories

as in online learning

- Can the teacher just score learner actions?
  - Reward: $r(s, a) \in \mathbb{R}$
- High reward is positive reinforcement for this behavior (in this state)
  - Much closer to how natural agents learn
  - Designing and programming r often easier than programming / demonstrating π

Actions have long-term consequences

- Tradeoff: short-term rewards vs. long-term returns (accumulated rewards)
  - Fly drone: slow down to avoid crash?
  - Games: slowly build strength? block opponent? all out attack?
  - Stock trading: sell now or wait for growth?
  - Infrastructure control: reduce power output to prevent blackout?
  - Life: invest in college, obey laws, get started early on course project
- Forward thinking and planning are hallmarks of intelligence

Returns

- Return = total reward = $R = \sum_{t \geq 0} \gamma^t r(s_t, a_t)$
  - Summarize reward sequence $r_t = r(s_t, a_t)$ as single number to be maximized
- Discount factor $\gamma \in [0,1]$
  - Higher weight to short-term rewards (and costs) than long-term
  - Good mathematical properties:
    - Assures convergence, simplifies algorithms, reduces variance
  - Vaguely economically motivated (inflation)

Horizon classes

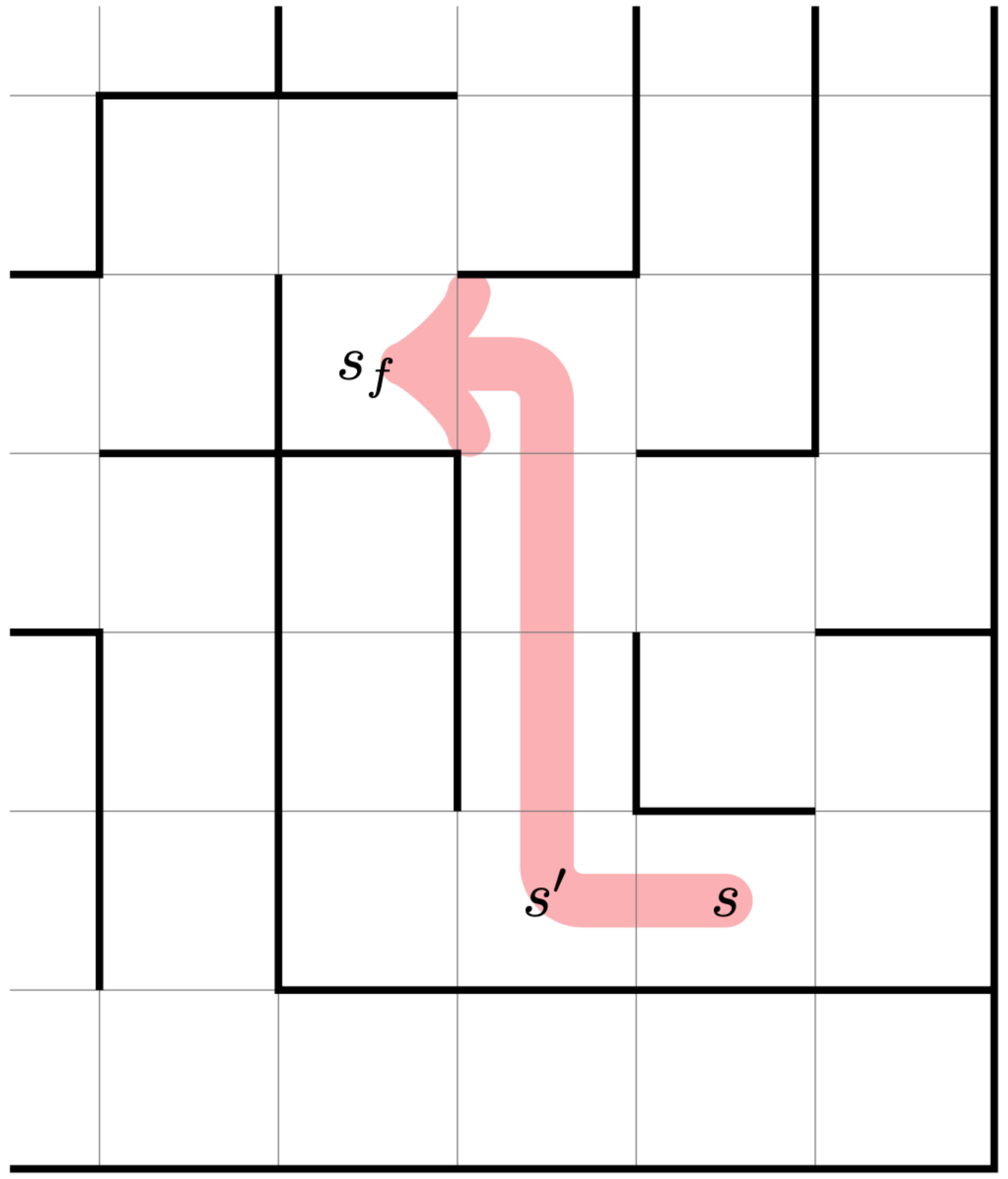
- Finite: $R(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$
- Infinite: $R(\xi) = \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} r(s_t, a_t)$
- Discounted: $R(\xi) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \qquad 0 \leq \gamma < 1$
- Episodic: $R(\xi) = \sum_{t=0}^{T-1} r(s_t, a_t)$ s.t. $s_T = s_f$

Basic RL concepts

- State: $s \in \mathcal{S}$; action: $a \in \mathcal{A}$; reward: $r(s, a) \in \mathbb{R}$
- Dynamics: $p(s_{t+1} \mid s_t, a_t)$ for stochastic; $s_{t+1} = f(s_t, a_t)$ for deterministic
- Policy: $\pi(a_t \mid s_t)$ for stochastic; $a_t = \pi(s_t)$ for deterministic
- Trajectory: $p_\pi(\xi = s_0, a_0, s_1, a_1, \ldots) = p(s_0) \prod_t \pi(a_t \mid s_t) p(s_{t+1} \mid s_t, a_t)$
- Return: $R(\xi) = \sum_t \gamma^t r(s_t, a_t) \qquad 0 \leq \gamma < 1$
- Value: $V(s) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s]$

  $Q(s, a) = \mathbb{E}_{\xi \sim p_\pi}[R \mid s_0 = s, a_0 = a]$

Special case: shortest path



- Deterministic dynamics: in state s, take action a to get to state $s' = f(s, a)$
  - Example above: $s' = f(s, a_{\text{left}})$
- Reward: (-1) in each step (until the goal s_f is reached)

Today's lecture

Course overview

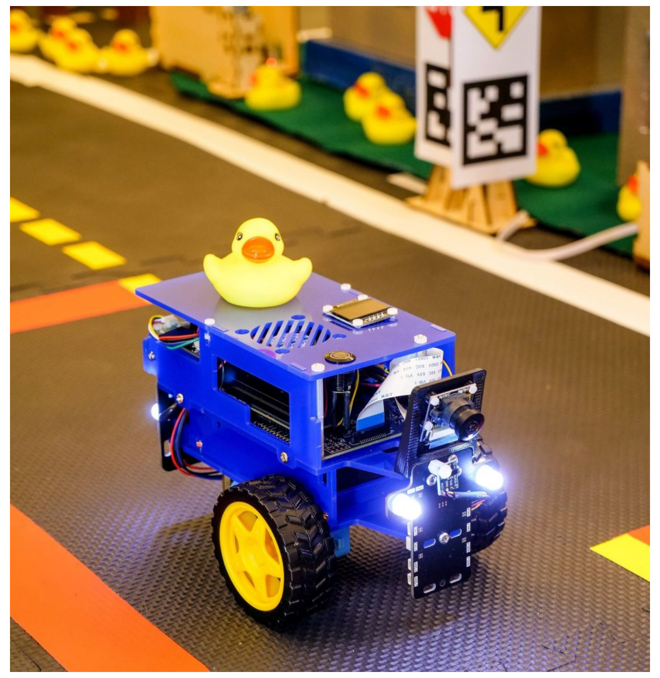
What is a project

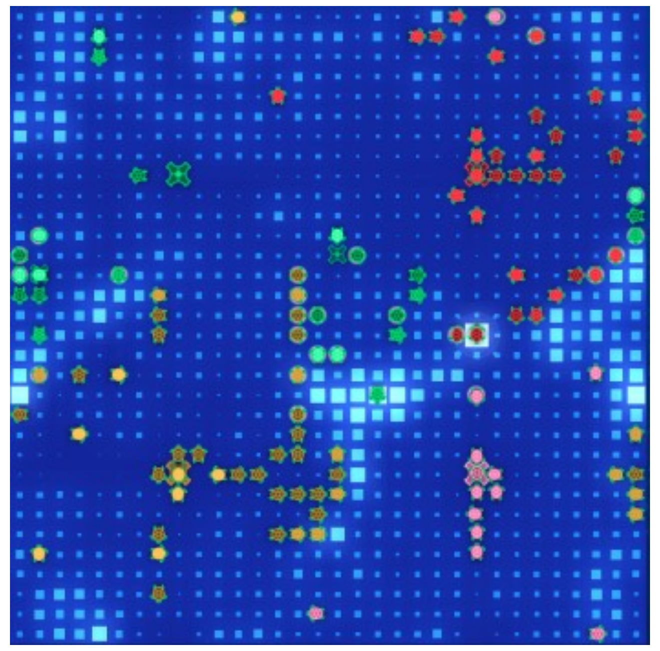
What is reinforcement learning

Project ideas

Some project ideas

- Applications:
  - MineCraft
  - DuckieTown
  - Obstacle Tower
  - Hanabi
  - Halite
  - Diplomacy

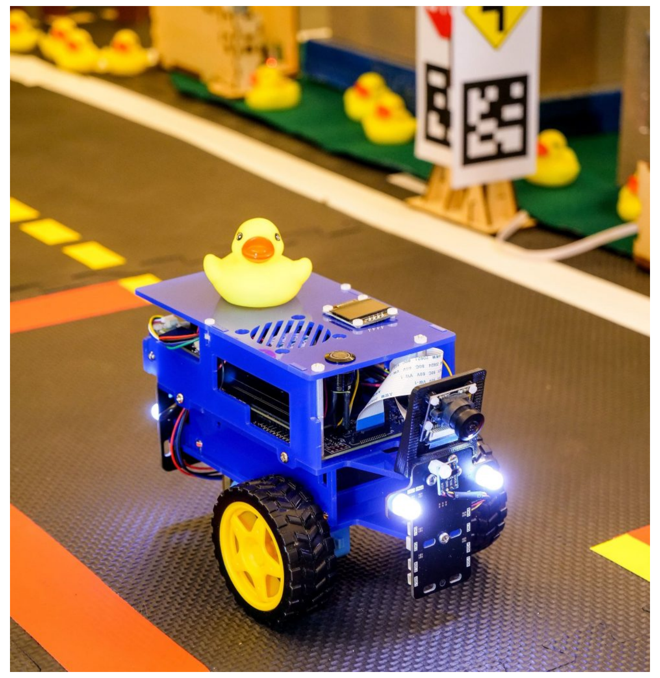




MineCraft

- Open world: can define many scenarios and tasks
- Done many many times before, so you'd have to get very creative
- One interesting option: MindCraft lets language agents play MineCraft
  - https://github.com/kolbytn/mindcraft

DuckieTown



- Drive a small vehicle on a foam track
- Common tasks: lane following, multi-agent collision avoidance
- You'd mostly work in a simulator
  - Successful projects can be deployed to real DuckieBots!

Obstacle Tower

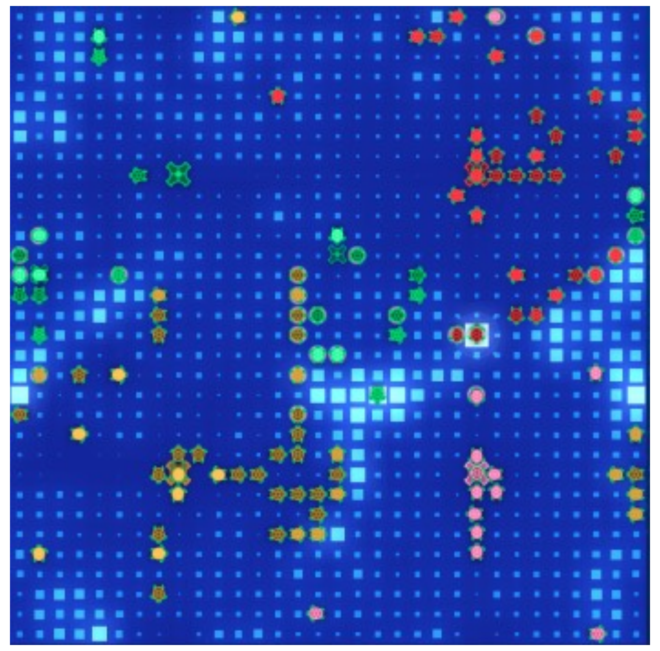
- Algorithmically generated locomotion puzzles
- Visual control + planning
- Progressively more challenging
  - Need generalization, continual learning, maybe symbolic planning

Hanabi

- Collaborative game, simple with many challenging expansions
- Distributed observability, solution can be centralized or not
- How to induce zero-shot cooperation?
  - Will the policy collaborate with humans / other training seeds?

Halite

- Competitive resource management (and combat) game
- Fully observable (Markov game) in a large but structured space
- Evaluation may be non-transitive: $\pi_1 > \pi_2 > \pi_3 > \pi_1$
  - Carefully evaluate against populations



Diplomacy

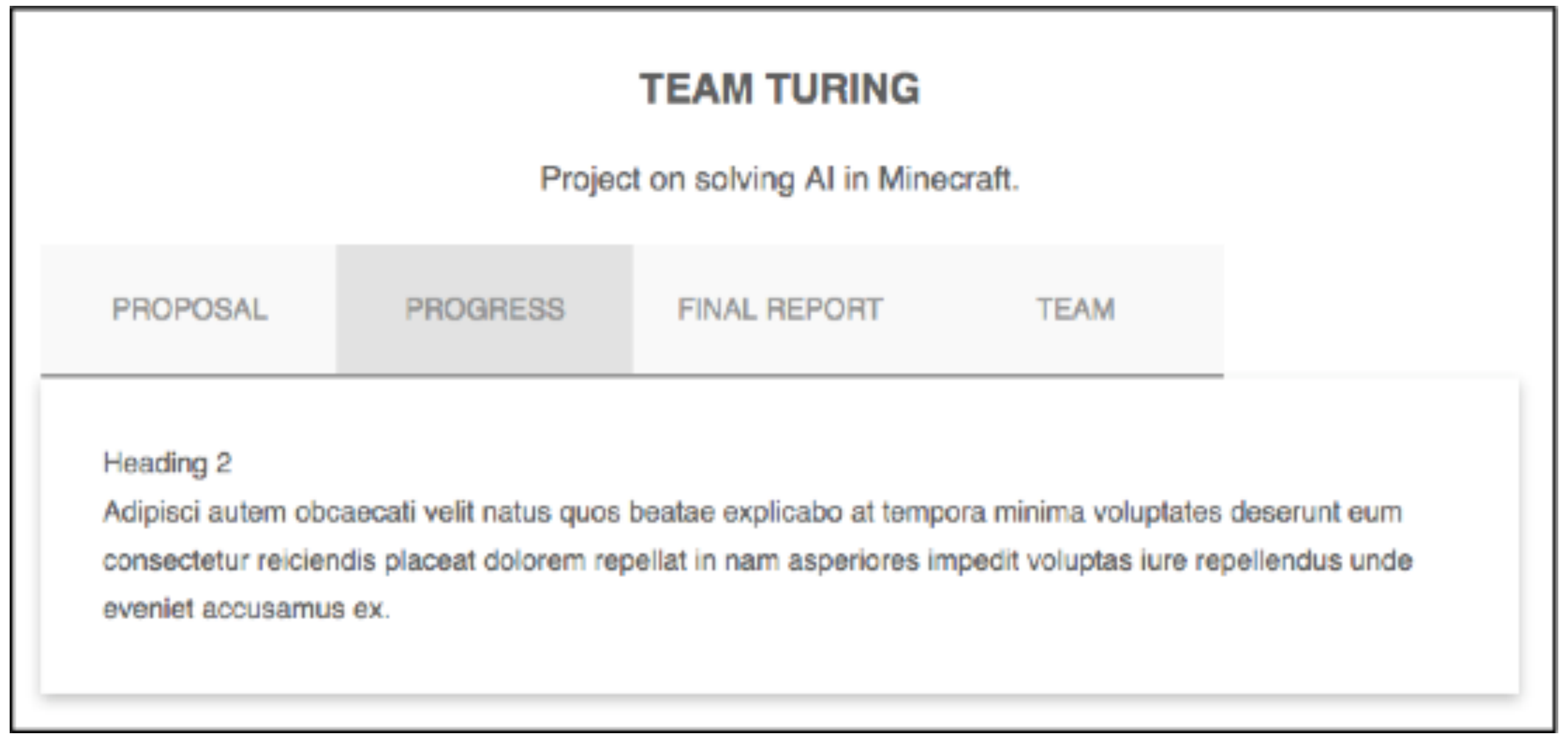
- Multi-player alliance and betrayal game
- What do we even optimize? Worst-case performance is always bad
- Humans play with text communication
  - Why? Can AI learn to ally with / betray each other / humans?

More project ideas

- Applications:
  - MineCraft
  - DuckieTown
  - Obstacle Tower
  - Hanabi
  - Halite
  - Diplomacy
  - More “serious”: robots, infrastructure
- Method:
  - RL from non-reward feedback
  - Off-policy to on-policy RL
  - MaxEnt RL learning dynamics
  - RL for language generation
  - Model-based multi-agent RL
  - RL with sparse rewards
  - Large comparative study

Resources and tools

- GitHub — sync your work with teammates and course staff
- GitHub Pages — maintain project website



- Program in Python
  - Use libraries (numpy, scikit-learn, pytorch, jax)
  - Many domains and algorithms have existing implementations
    - May be a reason to prefer one over another
- Compute resources: campus-wide HPC3 cluster https://rcic.uci.edu/hpc3/