

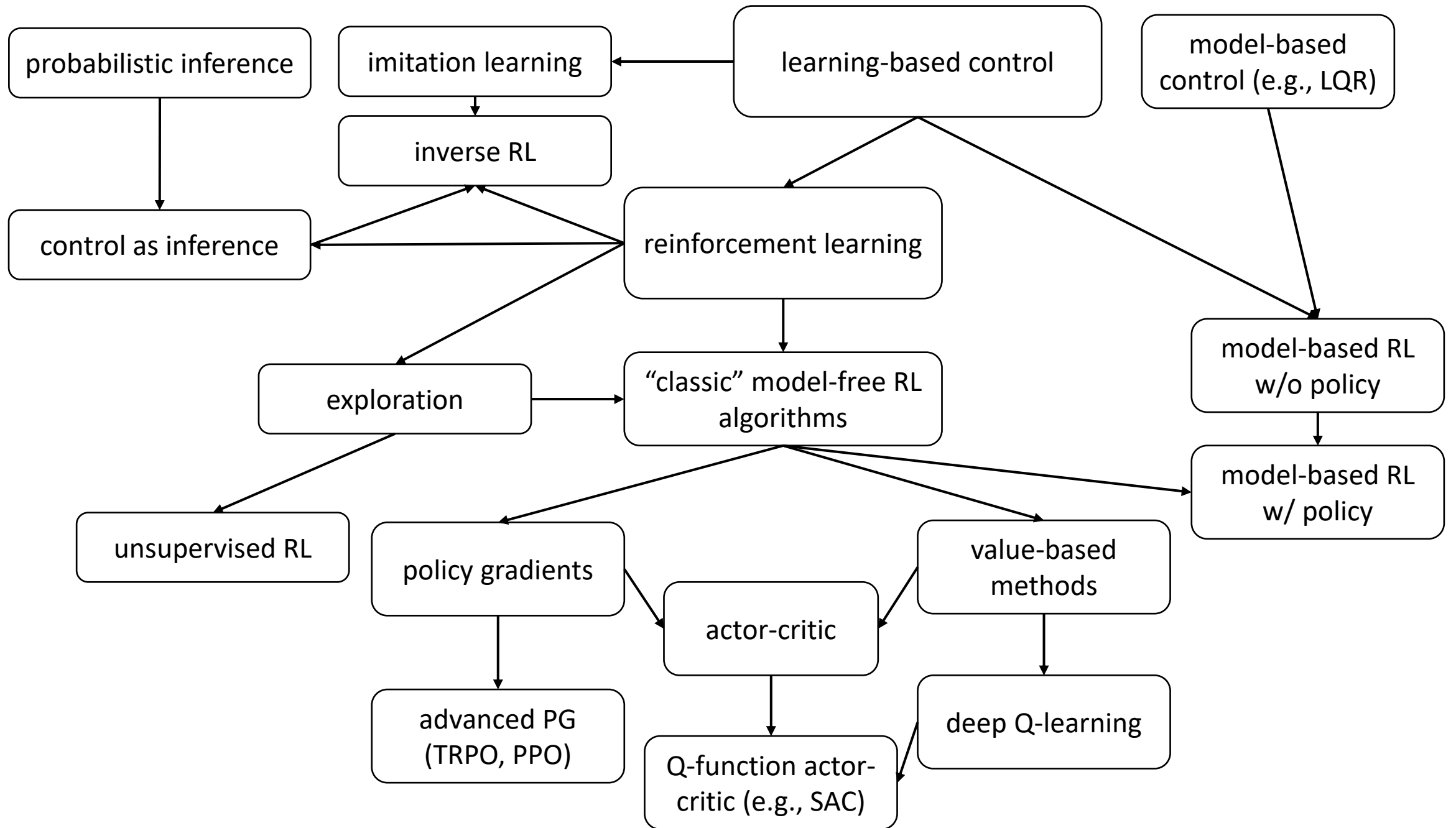
# Challenges and Open Problems

CS 285

Instructor: Sergey Levine  
UC Berkeley



# A Brief Review



# Challenges in Deep Reinforcement Learning

# What's the problem?

## Challenges with **core algorithms**:

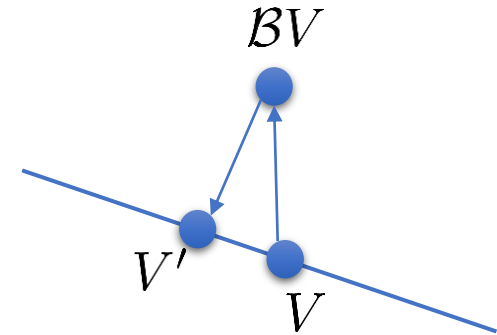
- Stability: does your algorithm converge?
- Efficiency: how long does it take to converge? (how many samples)
- Generalization: after it converges, does it generalize?

## Challenges with **assumptions**:

- Is this even the right problem formulation?
- What is the source of *supervision*?

# Stability and hyperparameter tuning

- Devising stable RL algorithms is very hard
- Q-learning/value function estimation
  - Fitted Q/fitted value methods with deep network function estimators are typically not contractions, hence no guarantee of convergence
  - Lots of parameters for stability: target network delay, replay buffer size, clipping, sensitivity to learning rates, etc.
- Policy gradient/likelihood ratio/REINFORCE
  - Very high variance gradient estimator
  - Lots of samples, complex baselines, etc.
  - Parameters: batch size, learning rate, design of baseline
- Model-based RL algorithms
  - Model class and fitting method
  - Optimizing policy w.r.t. model non-trivial due to backpropagation through time
  - More subtle issue: policy tends to *exploit* the model



gradient-free methods  
(e.g. NES, CMA, etc.)

10x

fully online methods  
(e.g. A3C)

10x

policy gradient methods  
(e.g. TRPO)

10x

replay buffer value estimation methods  
(Q-learning, DDPG, NAF, SAC, etc.)

10x

model-based deep RL  
(e.g. PETS, guided policy search)

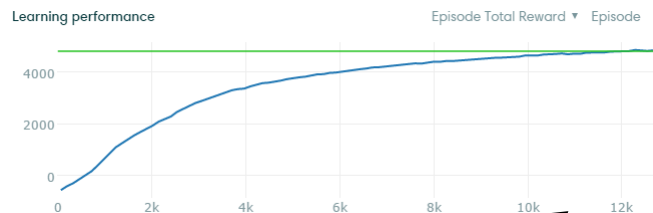
10x

model-based “shallow” RL  
(e.g. PILCO)

## Evolution Strategies as a Scalable Alternative to Reinforcement Learning

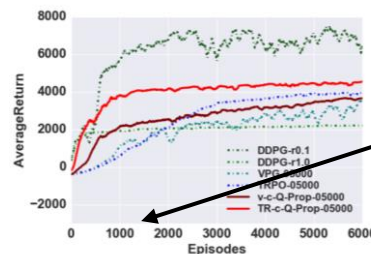
Tim Salimans<sup>1</sup> Jonathan Ho<sup>1</sup> Xi Chen<sup>1</sup> Ilya Sutskever<sup>1</sup>

half-cheetah (slightly different version)

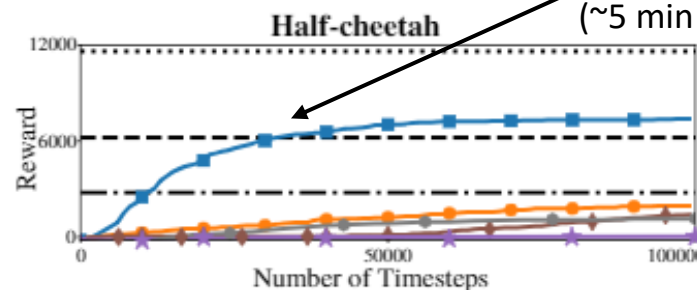


TRPO+GAE (Schulman et al. '16)

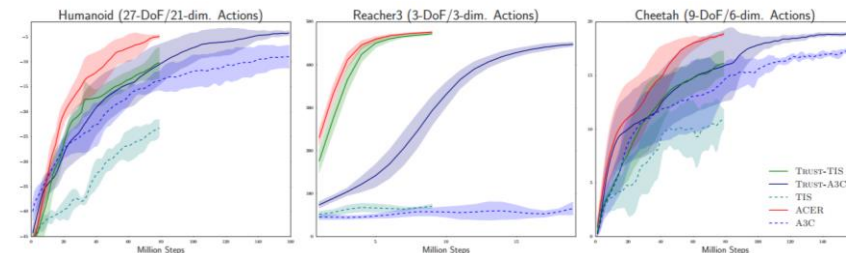
half-cheetah



Gu et al. '16



Chua et al. '18: Deep Reinforcement Learning in a Handful of Trials



Wang et al. '17

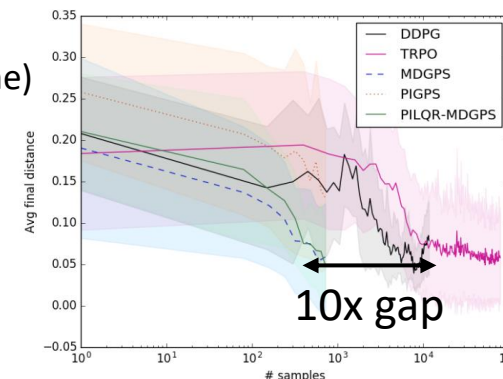
10,000,000 steps  
(10,000 episodes)  
(~ 1.5 days real time)

100,000,000 steps  
(100,000 episodes)  
(~ 15 days real time)

1,000,000 steps  
(1,000 episodes)  
(~3 hours real time)



30,000 steps  
(30 episodes)  
(~5 min real time)

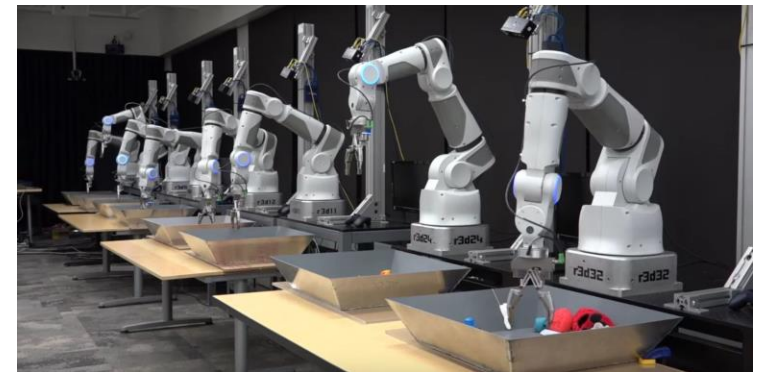


Chebotar et al. '17 (note log scale)

about 20  
minutes of  
experience on a  
real robot

# The challenge with sample complexity

- Need to wait for a long time for your homework to finish running
- Real-world learning becomes difficult or impractical
- Precludes the use of expensive, high-fidelity simulators
- Limits applicability to real-world problems

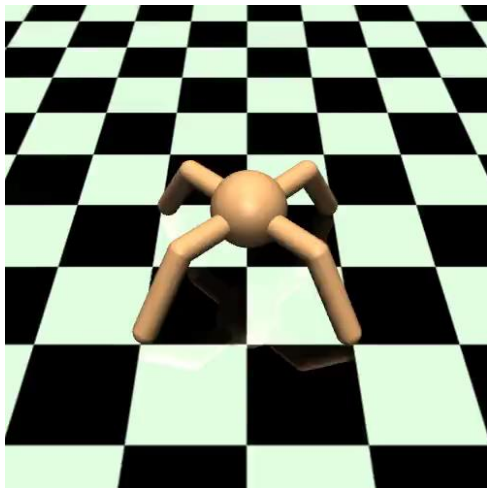




# Scaling up deep RL & generalization



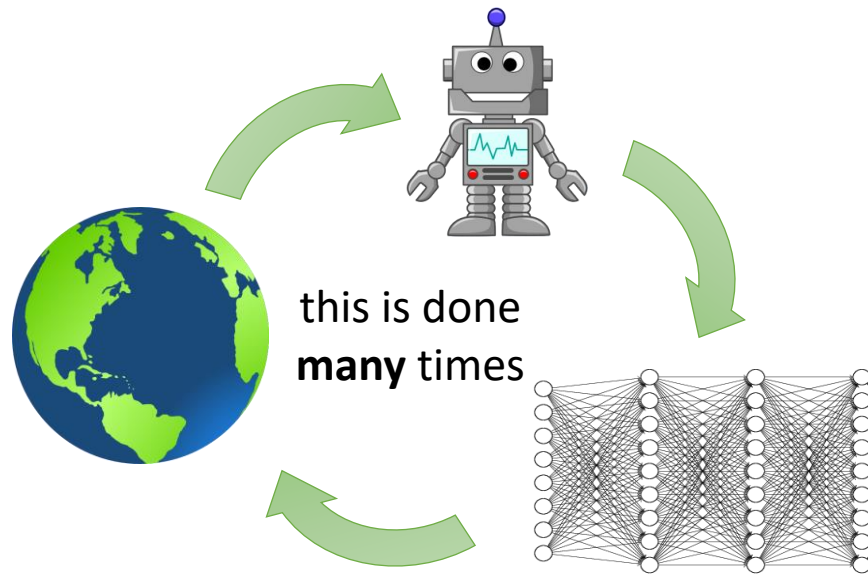
- Large-scale
- Emphasizes diversity
- Evaluated on generalization



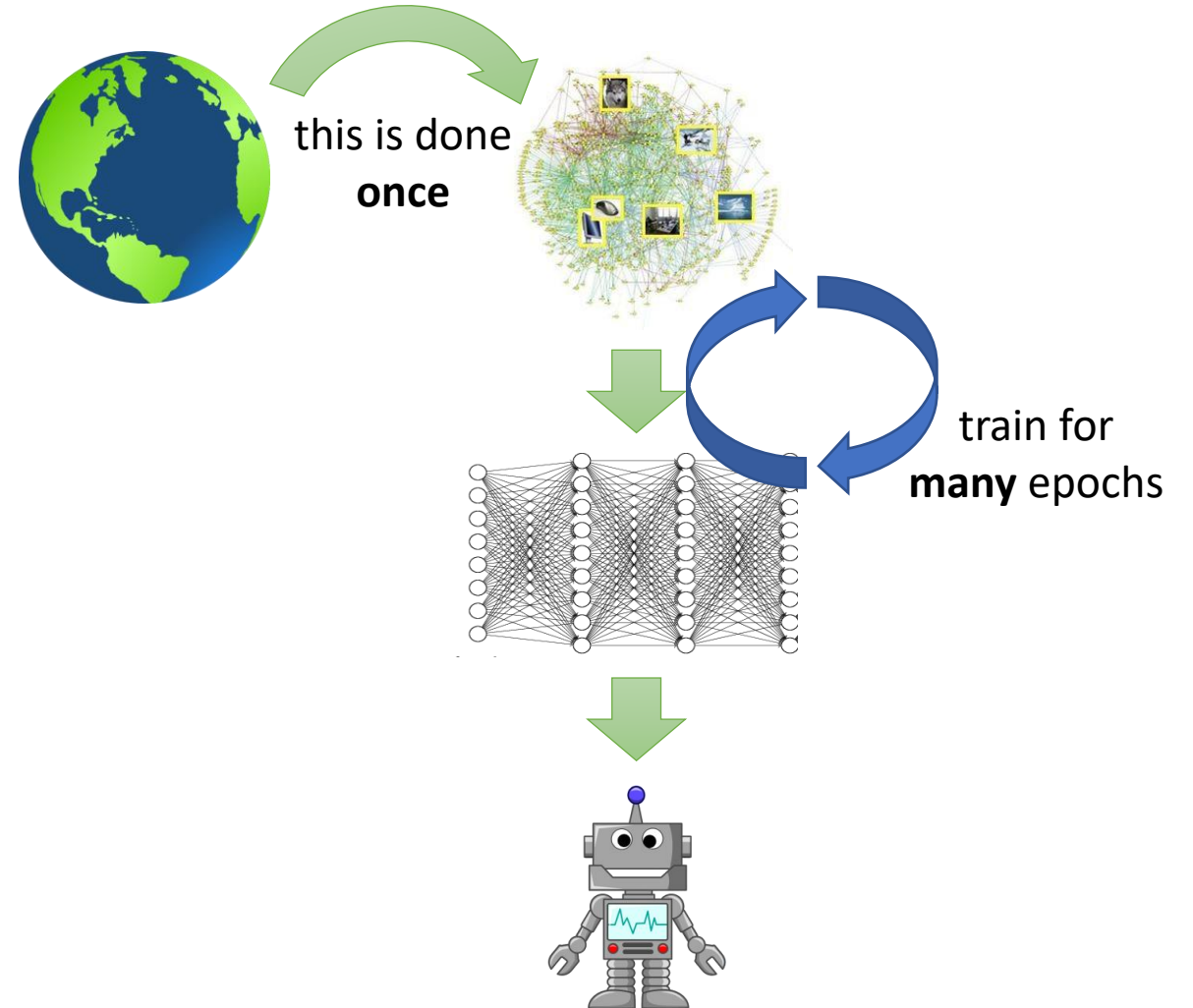
- Small-scale
- Emphasizes mastery
- Evaluated on performance
- Where is the generalization?

# RL has a **big** problem

reinforcement learning

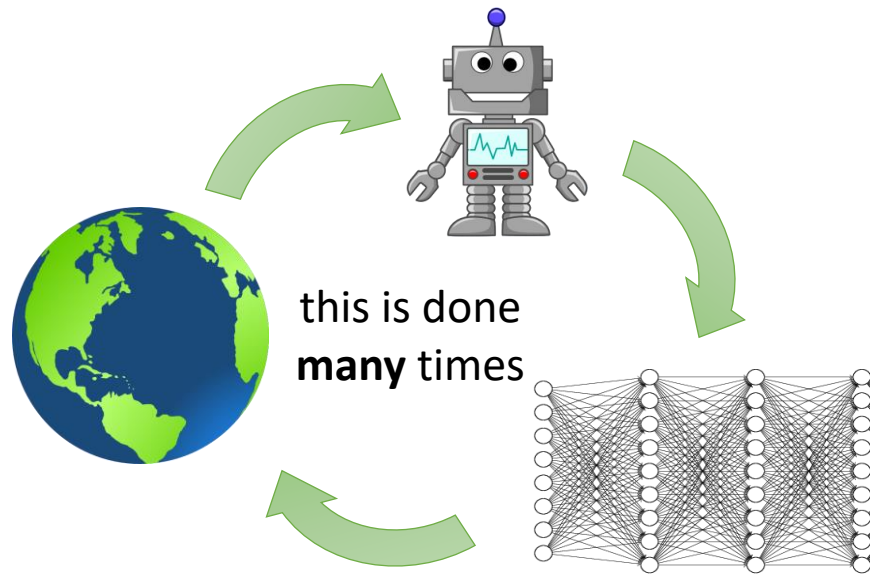


supervised machine learning

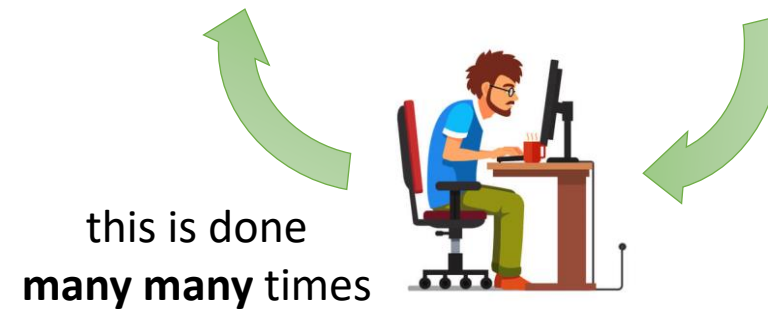
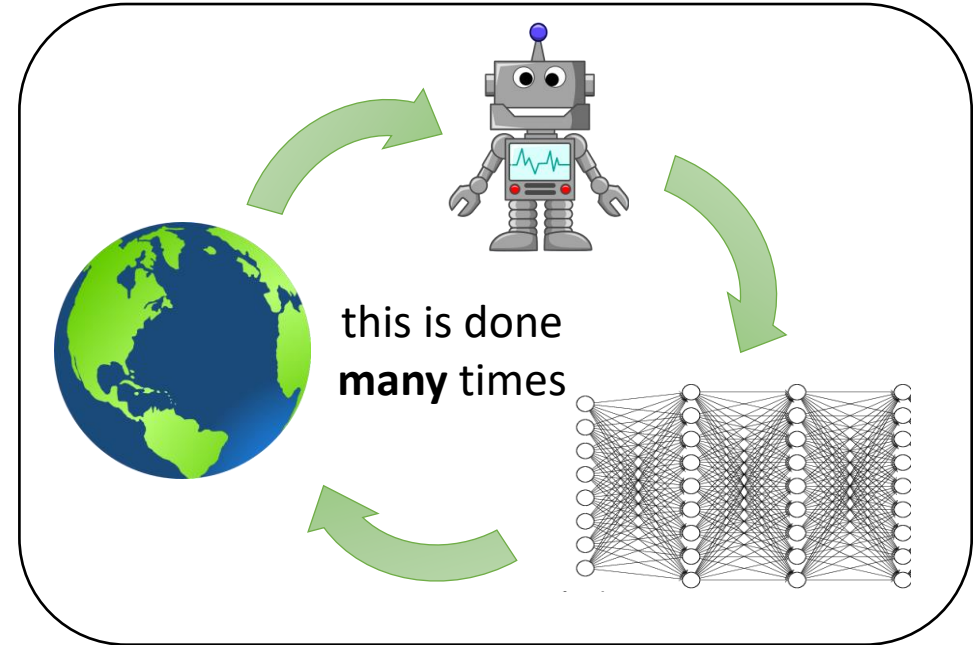


# RL has a **big** problem

reinforcement learning

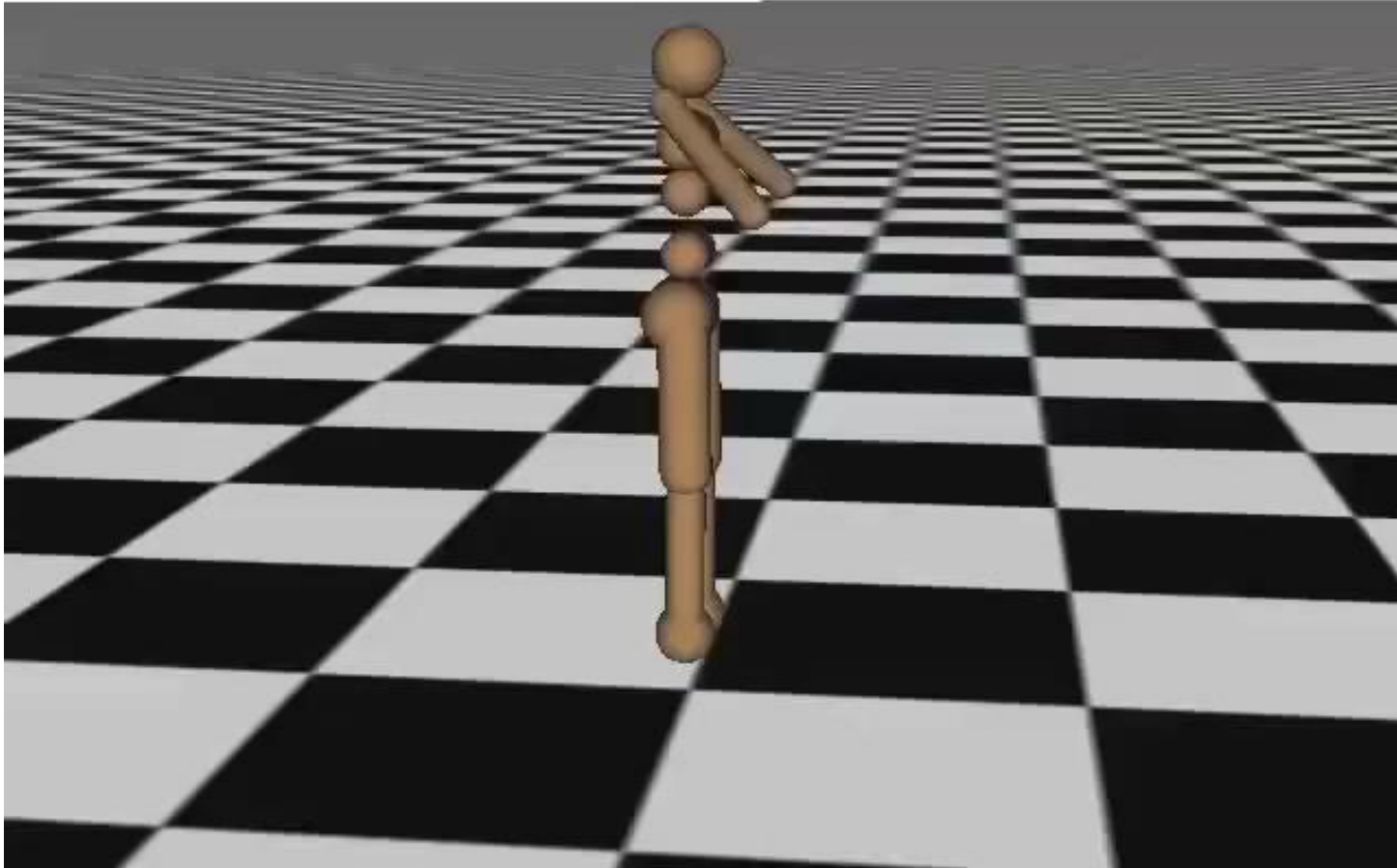


actual reinforcement learning



# How bad is it?

Iteration 0



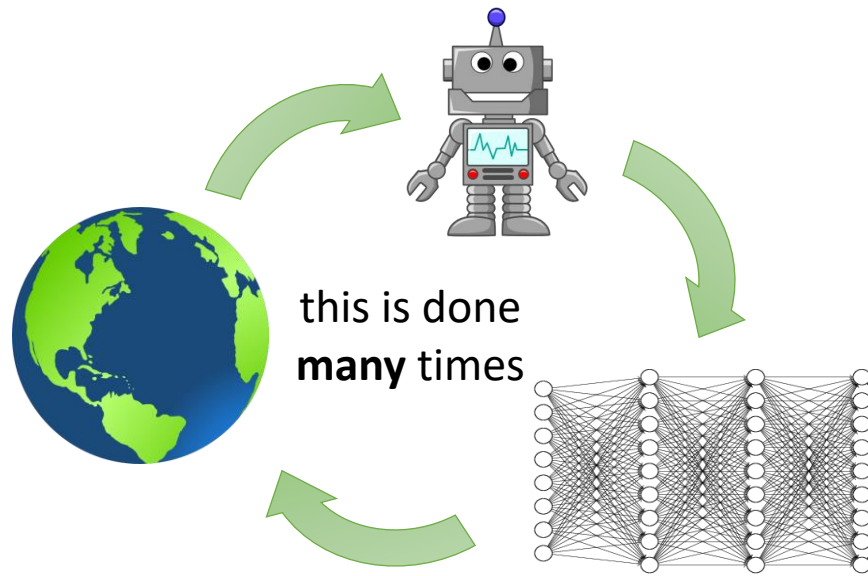
- This is quite cool
- It takes 6 days of real time (if it was real time)
- ...to run on an infinite flat plane



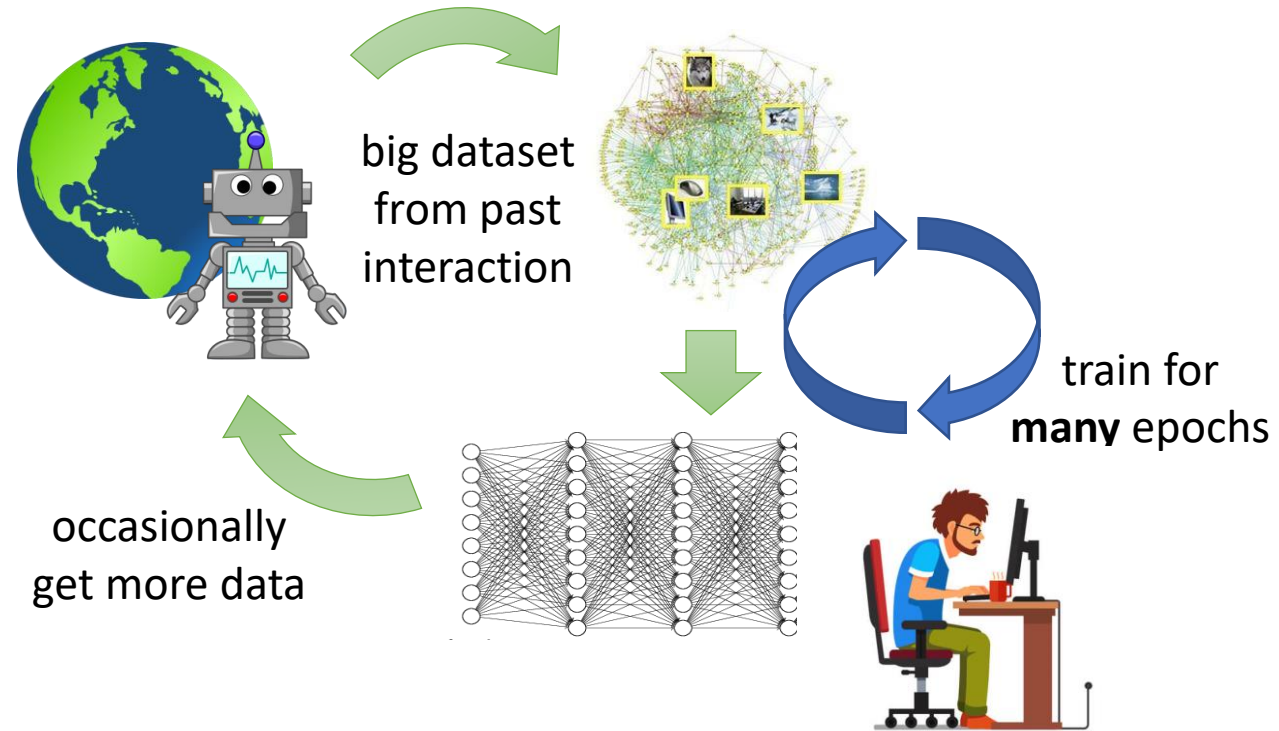
The real world is not so simple!

# Off-policy RL?

## reinforcement learning



## off-policy reinforcement learning





# Single task or multi-task?

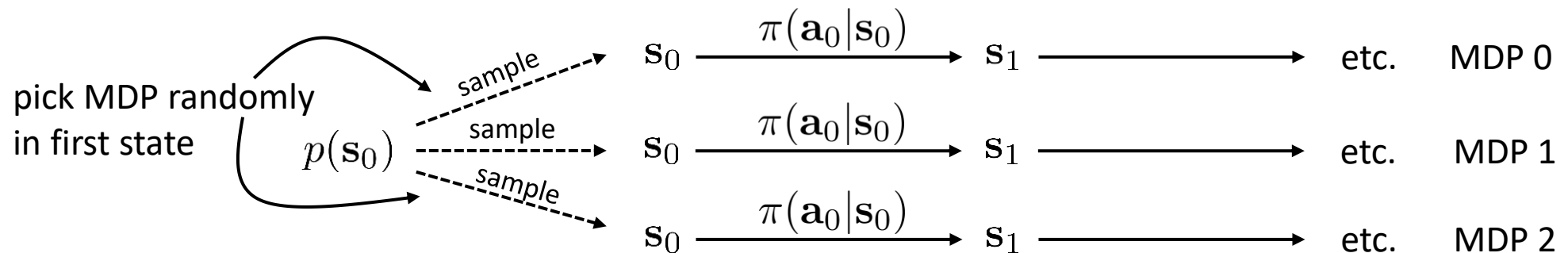


this is where generalization can come from...



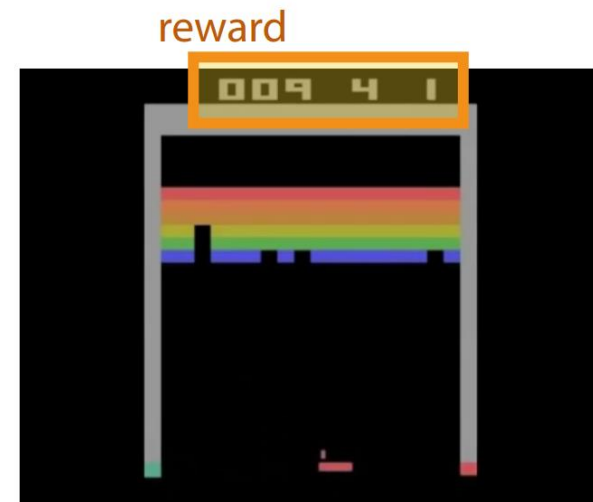
maybe doesn't require any new assumption, but might merit additional treatment

The real world is not so simple!



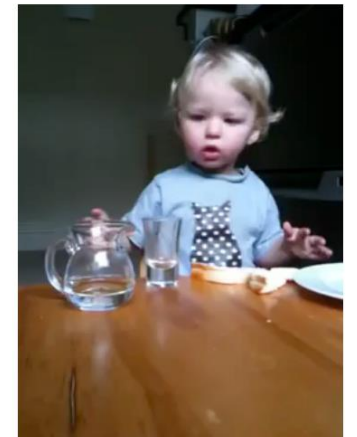
# Where does the **supervision** come from?

- If you want to learn from many different tasks, you need to get those tasks somewhere!
- Learn objectives/rewards from demonstration (inverse reinforcement learning)
- Generate objectives automatically?



Mnih et al. '15

reinforcement learning agent



what is the **reward**?

# Other sources of supervision

- Demonstrations

- Muelling, K et al. (2013). Learning to Select and Generalize Striking Movements in Robot Table Tennis



- Language

- Andreas et al. (2018). Learning with latent language

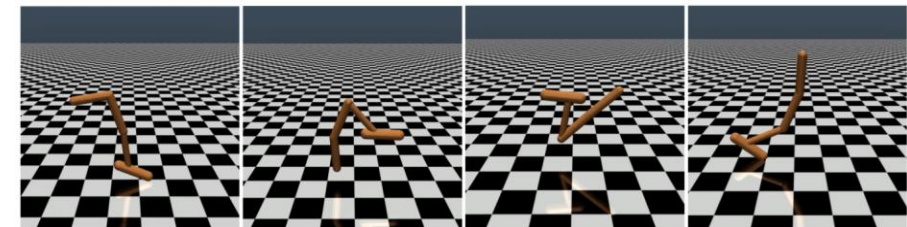
**Human description:**  
move to the star

**Inferred description:**  
reach the star cell



- Human preferences

- Christiano et al. (2017). Deep reinforcement learning from human preferences



Should supervision tell us **what** to do or **how** to do it?



# Rethinking the Problem Formulation

- How should we define a *control* problem?
  - What is the data?
  - What is the goal?
  - What is the supervision?
    - may not be the same as the goal...
- Think about the assumptions that fit your problem setting!
- Don't assume that the basic RL problem is set in stone

Some perspectives...

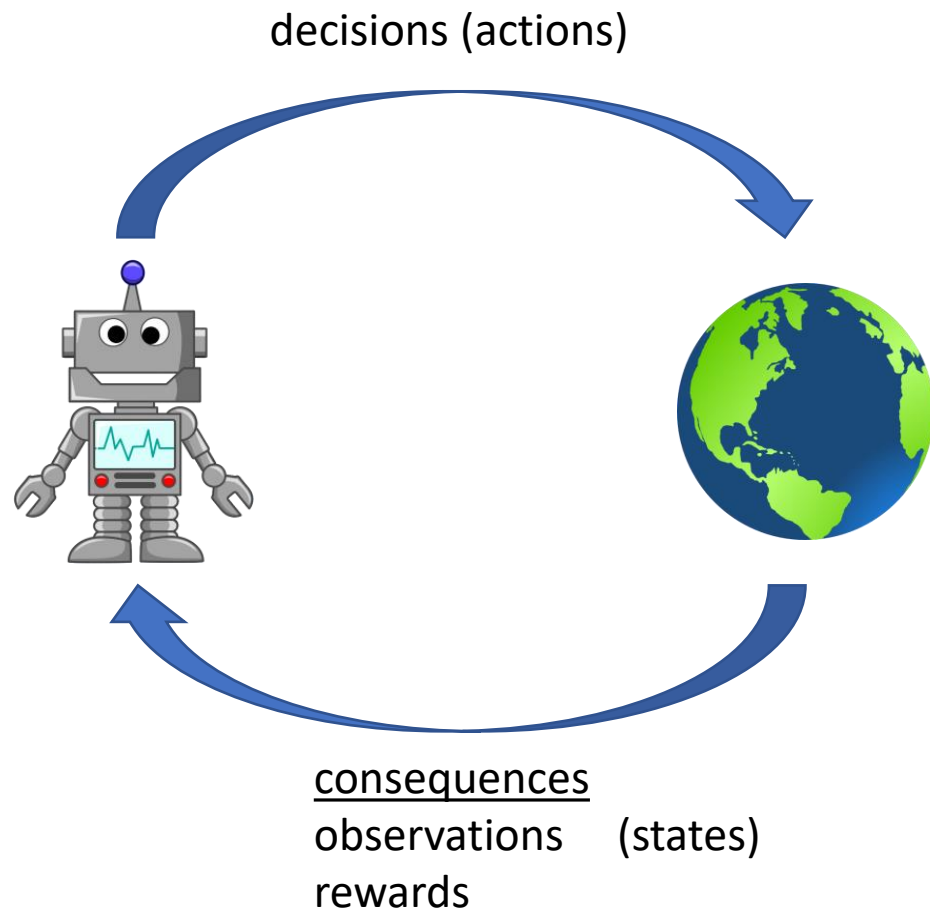
Reinforcement Learning as an Engineering Tool

Reinforcement Learning and the Real World

Reinforcement Learning as “Universal” Learning

# Reinforcement Learning as an Engineering Tool

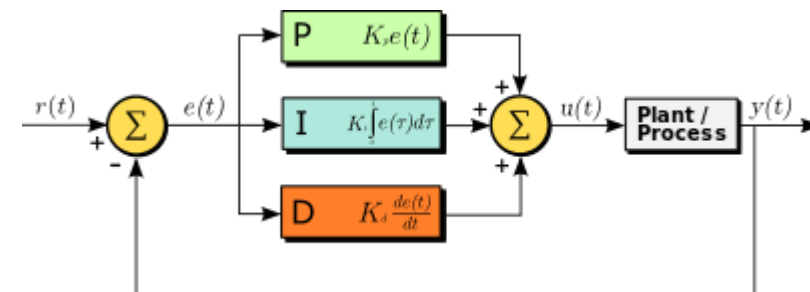
# What we think RL is...



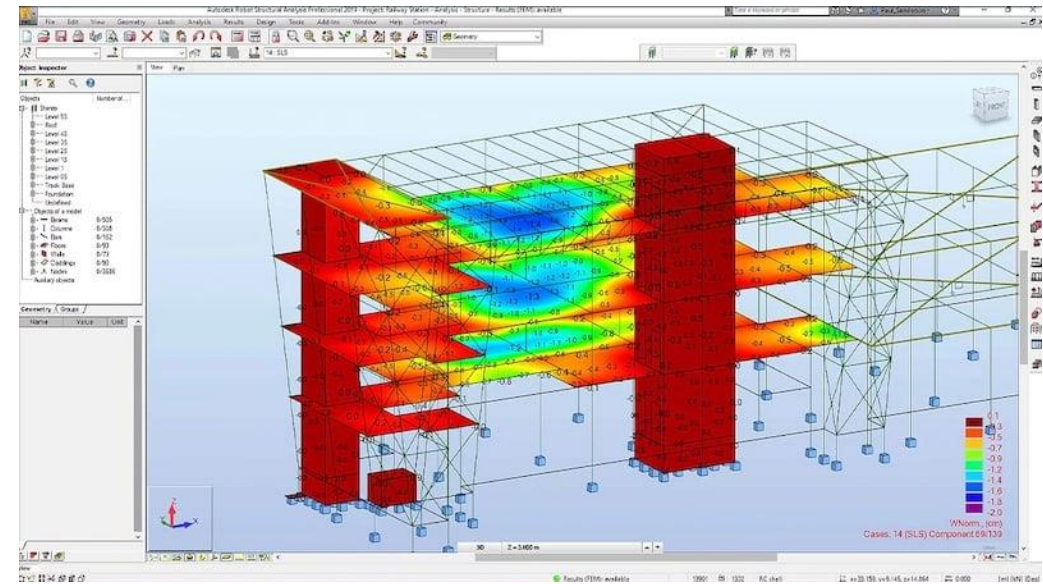
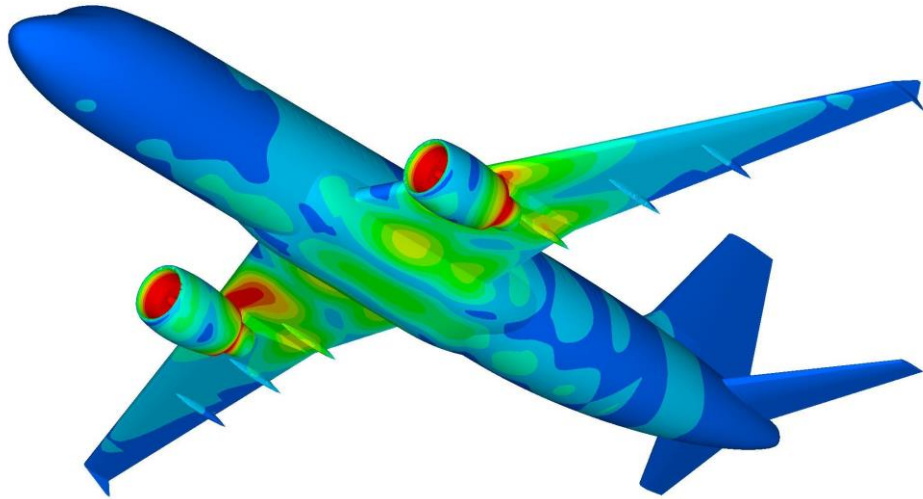
# Engineering a control system



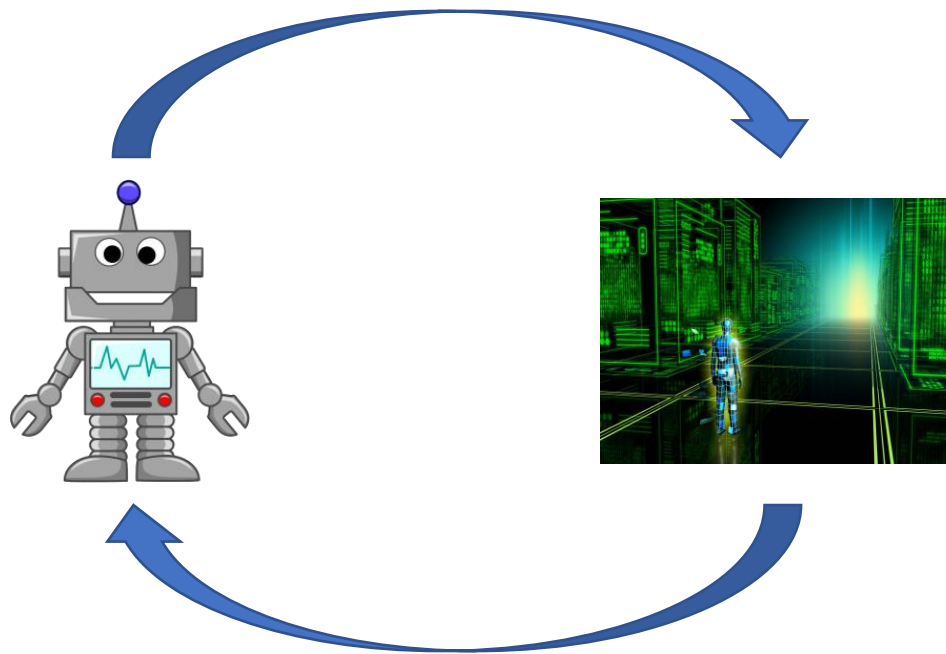
$$\begin{aligned}\mathbf{r} &= \mathbf{r}(t) = r\hat{\mathbf{e}}_r \\ \mathbf{v} &= v\hat{\mathbf{e}}_r + r\frac{d\theta}{dt}\hat{\mathbf{e}}_\theta + r\frac{d\varphi}{dt}\sin\theta\hat{\mathbf{e}}_\varphi \\ \mathbf{a} &= \left( a - r\left(\frac{d\theta}{dt}\right)^2 - r\left(\frac{d\varphi}{dt}\right)^2\sin^2\theta \right)\hat{\mathbf{e}}_r \\ &\quad + \left( r\frac{d^2\theta}{dt^2} + 2v\frac{d\theta}{dt} - r\left(\frac{d\varphi}{dt}\right)^2\sin\theta\cos\theta \right)\hat{\mathbf{e}}_\theta \\ &\quad + \left( r\frac{d^2\varphi}{dt^2}\sin\theta + 2v\frac{d\varphi}{dt}\sin\theta + 2r\frac{d\theta}{dt}\frac{d\varphi}{dt}\cos\theta \right)\hat{\mathbf{e}}_\varphi\end{aligned}$$



# Characterization and simulation...



# RL: anything you can *simulate* you can *control*



- Provides a powerful engineering tool
- Now *that* different from conventional engineering approach!
  - Before: characterize, simulate, control
  - Now: characterize, simulate, run RL
- Main role: powerful *inversion* engine
- Main weakness: still need to simulate!

# Reinforcement Learning and the Real World



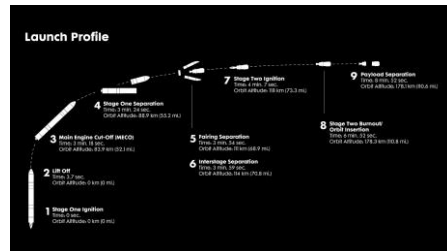


# Moravec's paradox

**Moravec's paradox** seems like a statement about AI

**but it is actually a statement about the physical universe**

“easy” universes



**Why?**

“hard” universes



We are all prodigious olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it.

- Hans Moravec

The main lesson of thirty-five years of AI research is that the hard problems are easy and the easy problems are hard. The mental abilities of a four-year-old that we take for granted – recognizing a face, lifting a pencil, walking across a room, answering a question – in fact solve some of the hardest engineering problems ever conceived.

- Steven Pinker







# What does this have to do with RL?



How do we engineer a system that can deal with the unexpected?

- Minimal external supervision about what to do
  - Unexpected situations that require adaptation
  - Must discover solutions autonomously
  - Must “stay alive” long enough to discover them!
- 
- Humans are extremely good at this
  - Current AI systems are extremely bad at this
  - RL *in principle* can do this, and nothing else can

# So what's the problem?



**RL *should* be really good in the “hard” universes!**

➤ RL *in principle* can do this, and nothing else can

**But we rarely study this kind of setting in RL research!**

## “easy” universes

success = high reward  
 (“optimal control”)

closed world, rules  
 are known

lots of simulation

**Main question:** can RL  
 algorithms **optimize**  
 **really well**

## “hard” universes

success = “survival”  
 (“good enough control”)

open world, everything  
 must come from data

no simulation (because  
 rules are unknown)

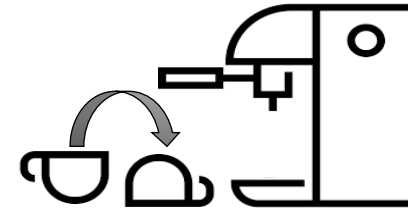
**Main question:** can RL  
 **generalize** and **adapt**

# Some questions that come up in the real world

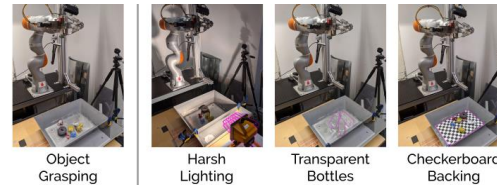
How do we tell RL agents **what we want them to do**?



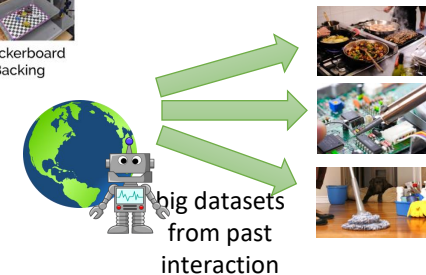
How can we learn **fully autonomously** in continual environments?



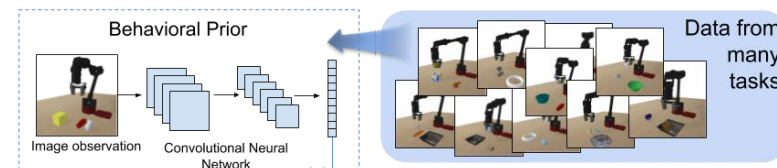
How do remain **robust** as the environment changes around us?



What is the right way to **generalize** using **experience & prior data**?



What is the right way to **bootstrap exploration** with **prior experience**?

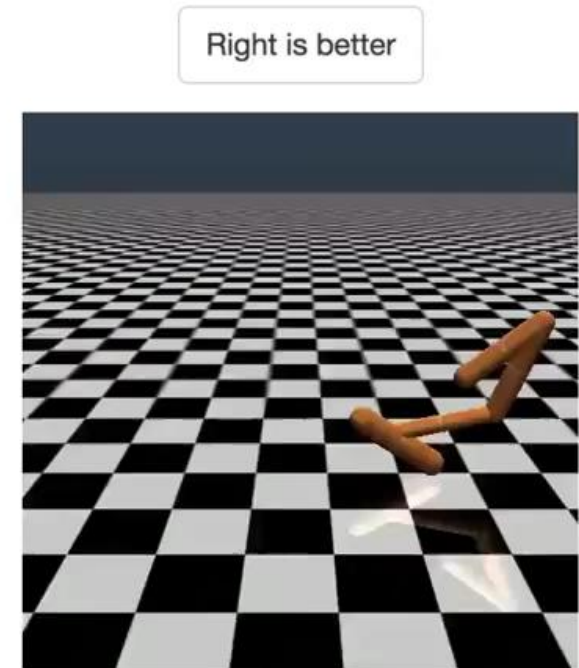
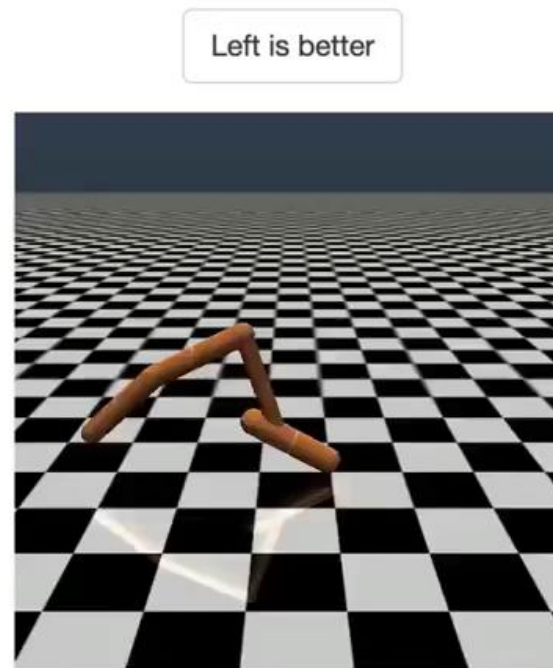
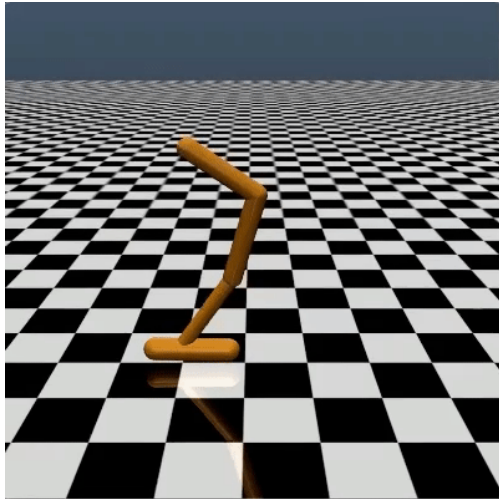
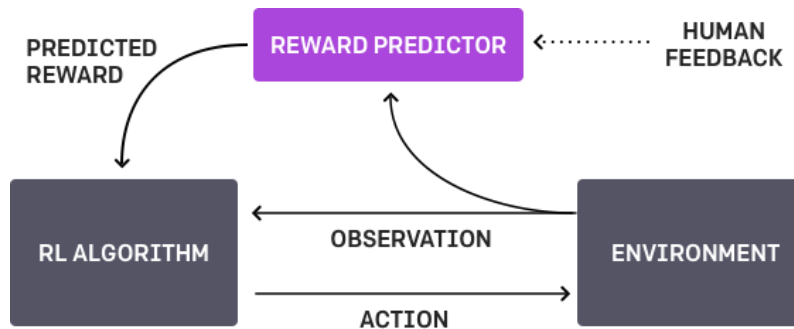


This is not about robots

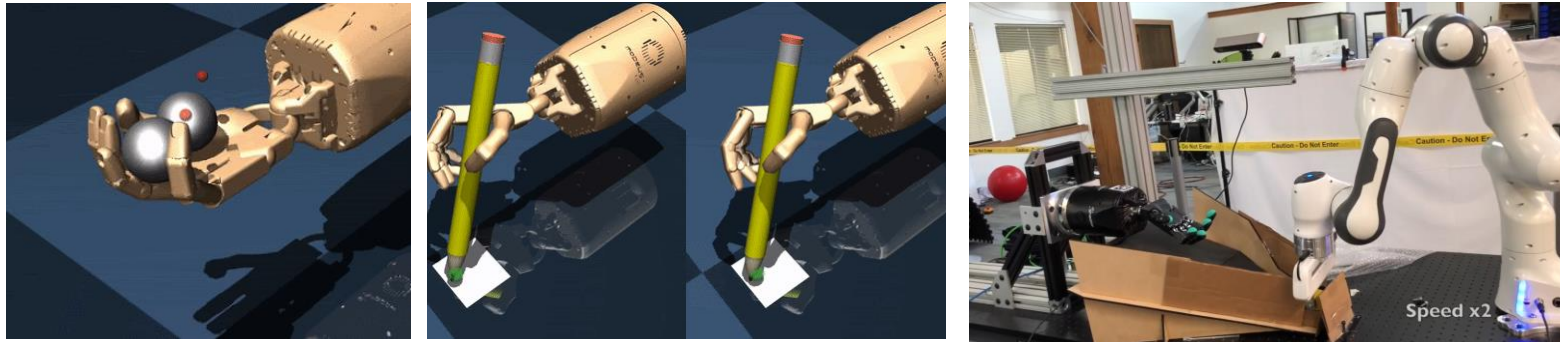
**Robots are the most natural for us to think about, because they are embodied like we are**



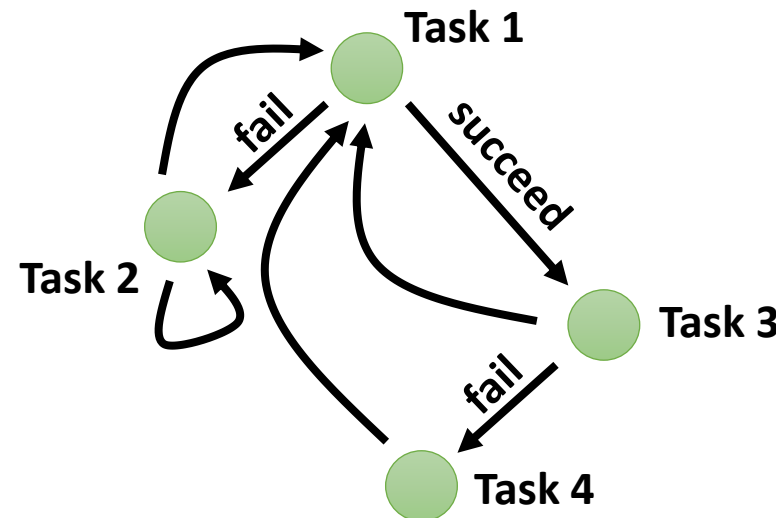
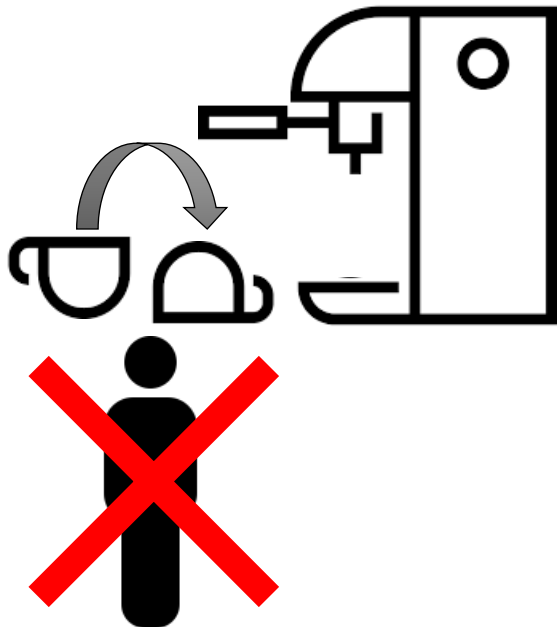
# Other ways to communicate objectives?



# How can we learn fully autonomously?



Nagabandi, Konolige, Levine, Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. CoRL 2019.



**Task 1:** put cup in coffee machine

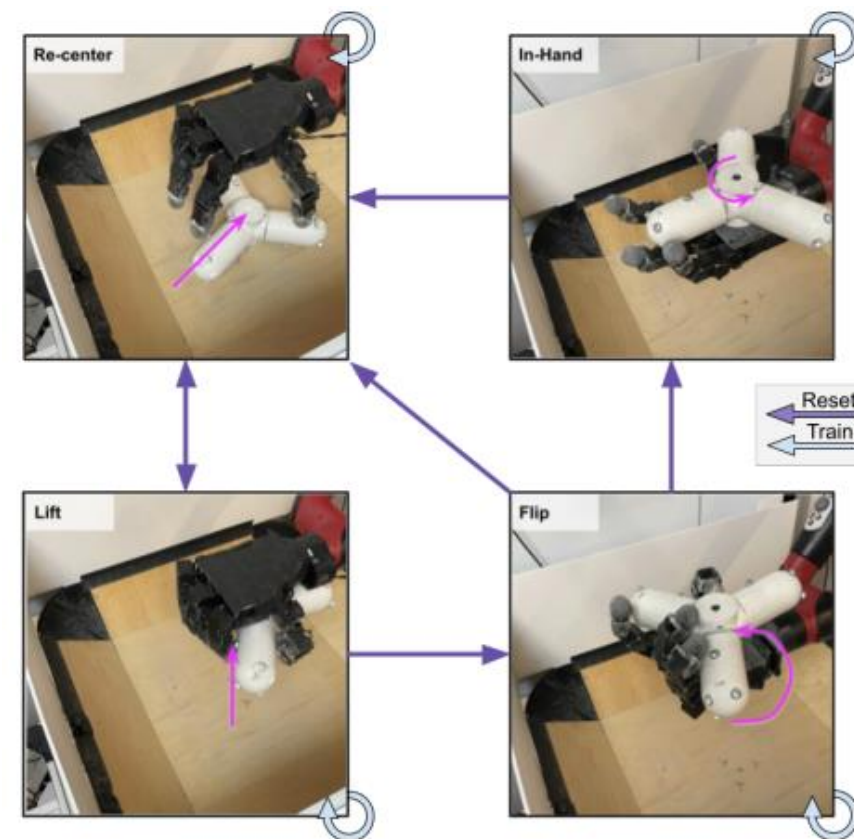
**Task 2:** pick up cup

**Task 3:** replace cup

**Task 4:** clean up spill from cup...

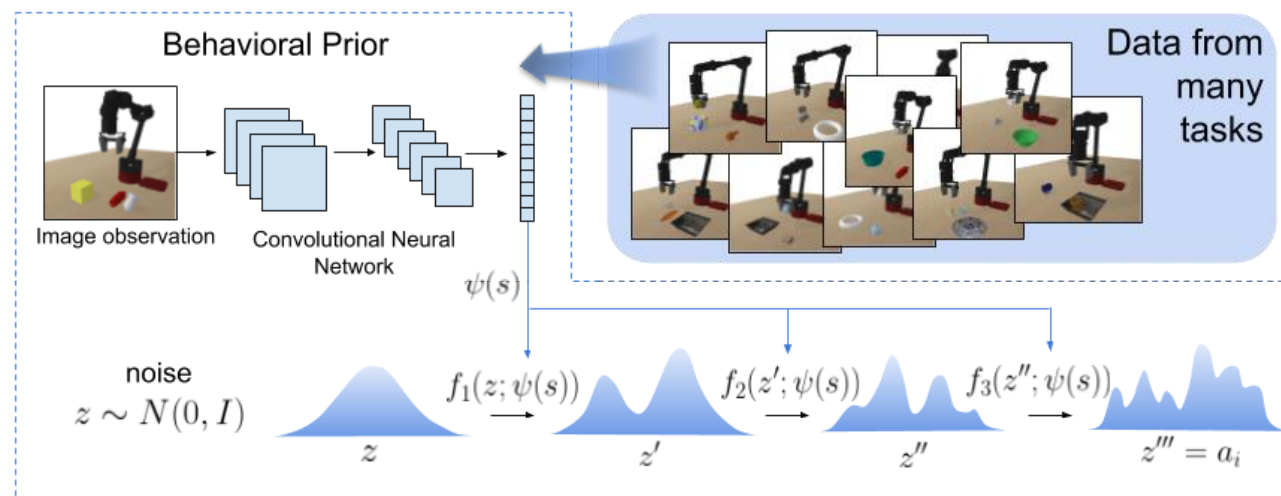
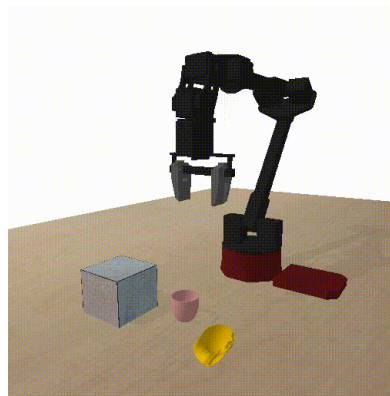


# How can we learn fully autonomously?

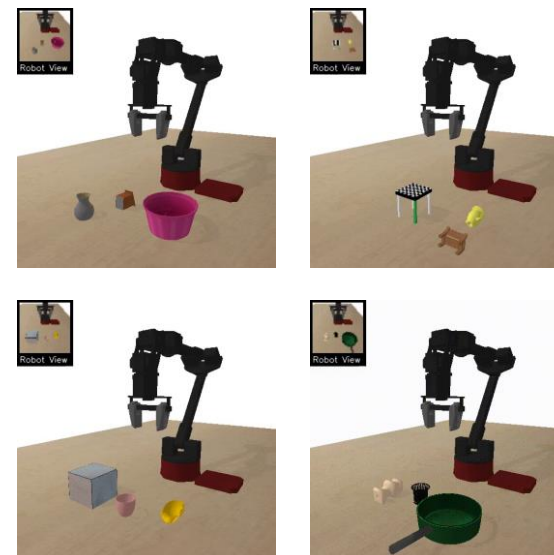
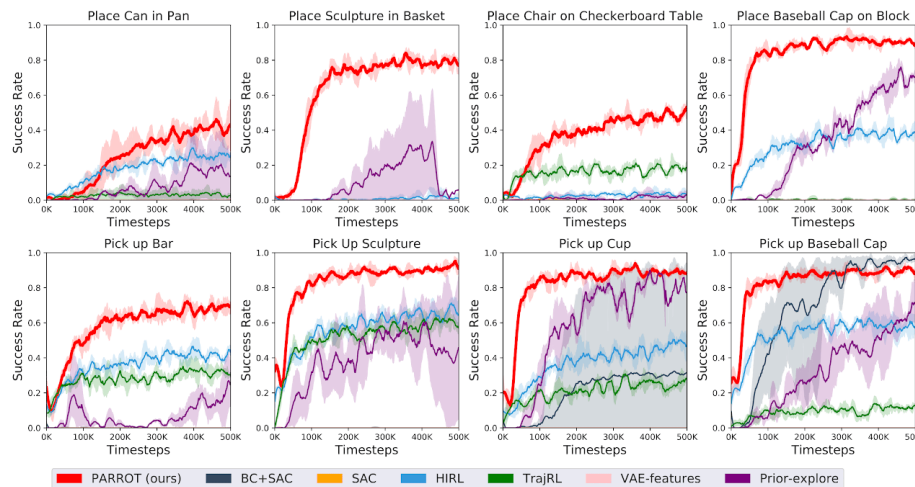
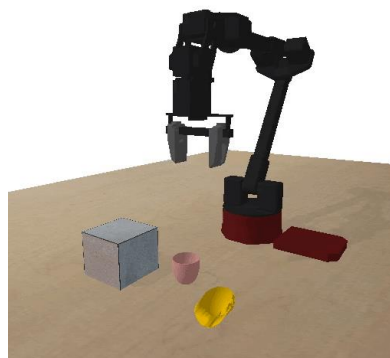


# How to bootstrap exploration from experience?

exploring from scratch



exploring from behavioral prior



# This all seems really hard, what's the point?



Why is this interesting?

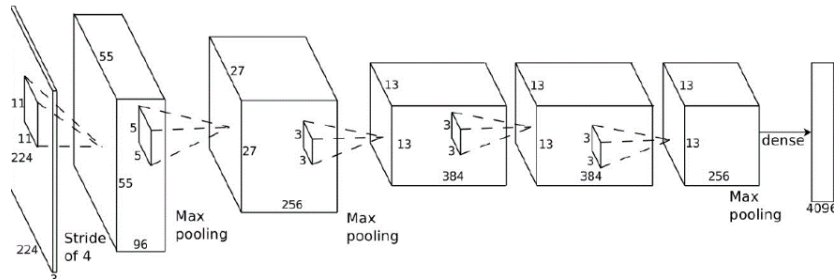
- It's exciting to see what solutions intelligent agents come up with
  - Most exciting if they come up with something we don't expect
  - This requires the world they inhabit to admit novel solutions
  - This means that world must be complicated enough!
- 
- To see interesting emergent behavior, we must train our systems in environments that actually require interesting emergent behavior!
  - RL in the real world may be difficult, but it is also rewarding

# Reinforcement Learning as “Universal” Learning

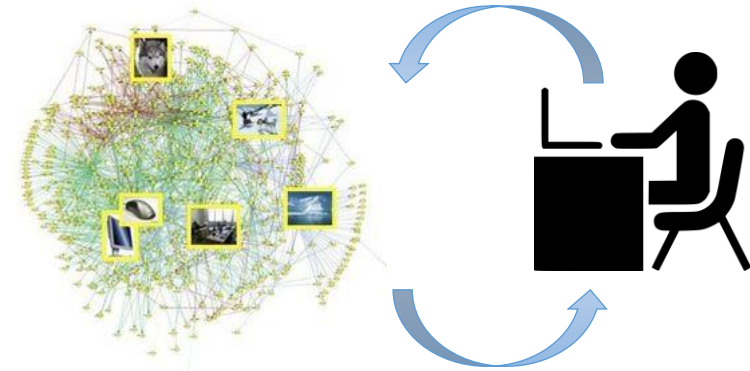
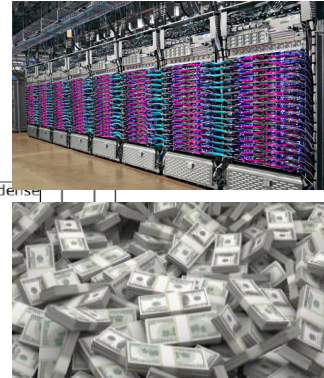


# Large-scale machine learning

Why does deep learning work?

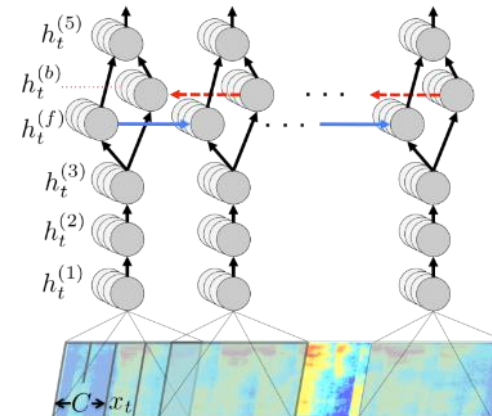
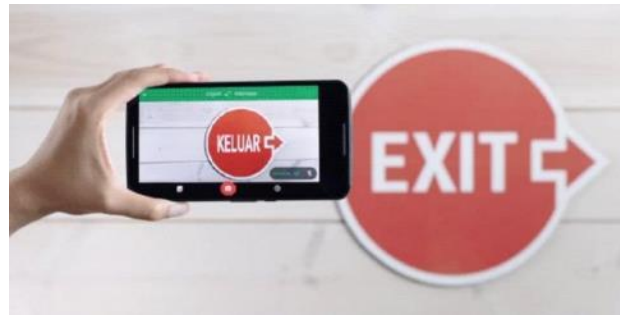
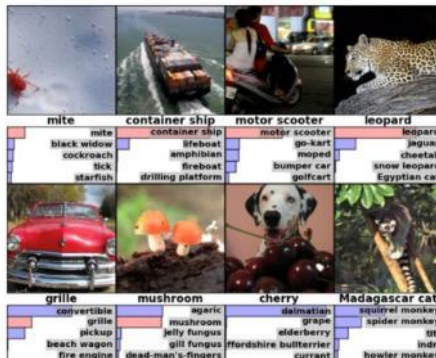


big models



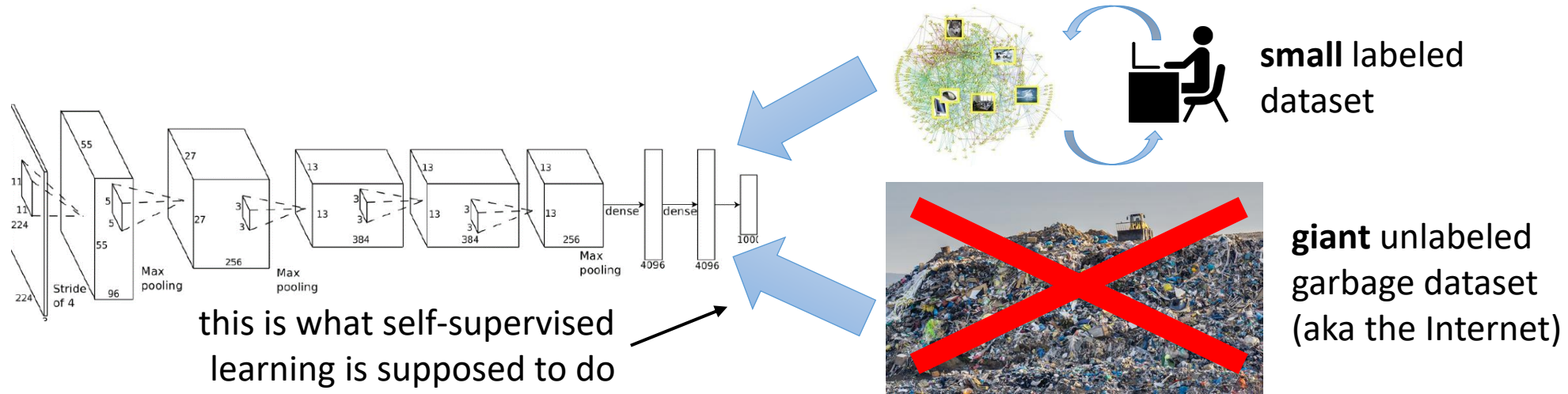
big datasets

labeled





# Reducing the supervision burden



But then where does the knowledge come from?

“Classic” unsupervised learning:  $p_{\theta}(\mathbf{x})$   
(this is what, for example, large language models do)



**Aside:** perhaps this is why “prompting” large language models is such an art!

# Stepping back a bit...

Why do we need **machine learning** anyway?

A postulate:

We need machine learning for one reason and one reason only – that's **to produce adaptable and complex decisions.**



**Decision:** how do I move my joints?



**Decision:** how do I steer the car?



**What is the decision?** The image label?

**Usually not!**

What happens with that label **afterwards**?

Is it used to tag a user's photo?

Detect an endangered animal in a camera trap?

these are **decisions**

they lead to **consequences**

**Aside:** why do we need **brains** anyway?

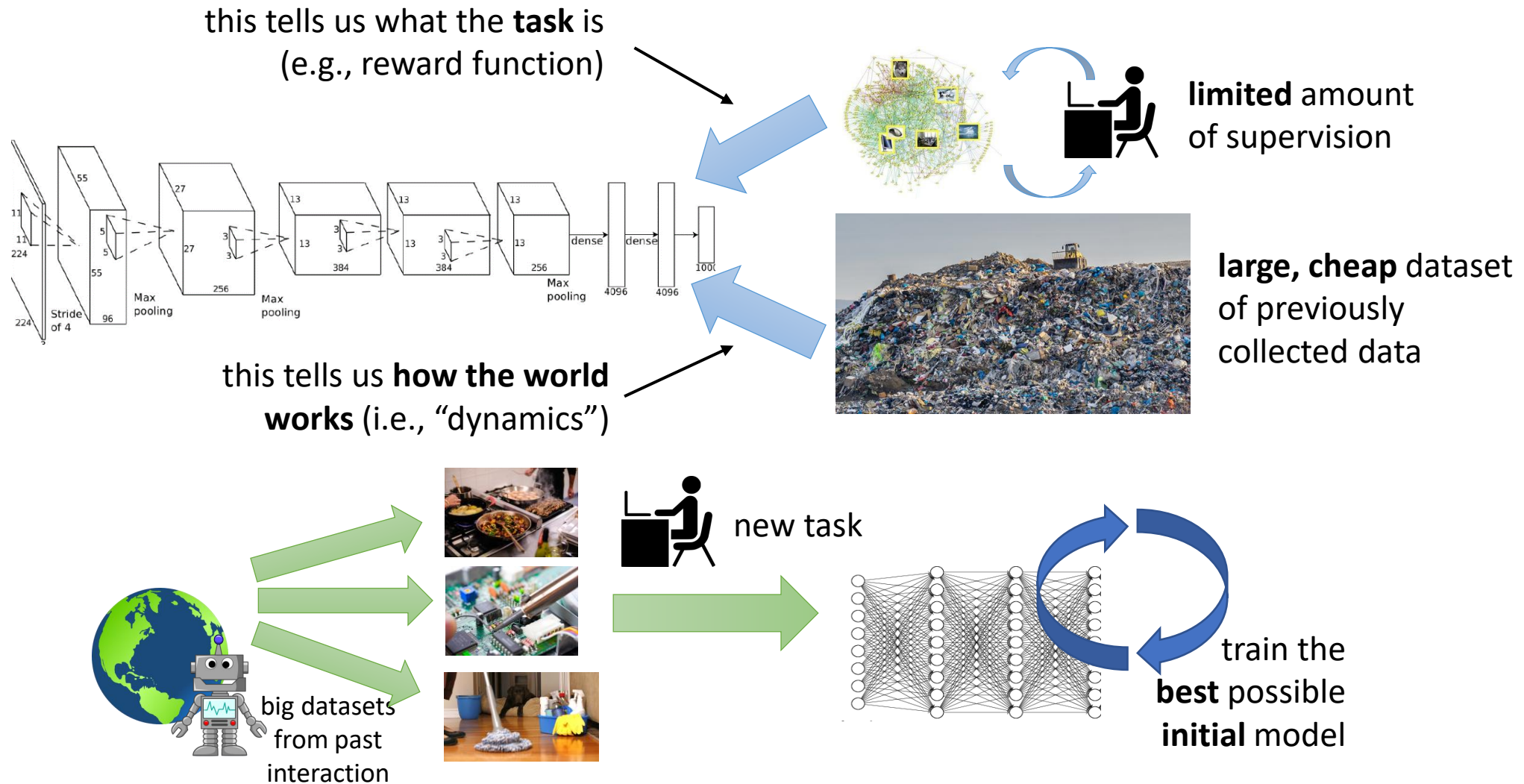


Daniel Wolpert  
(knows quite a lot  
about brains)



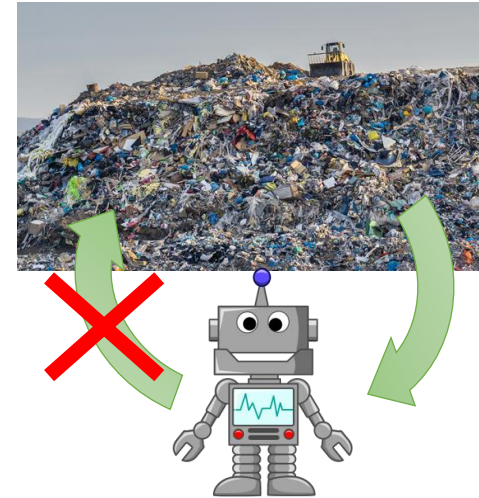
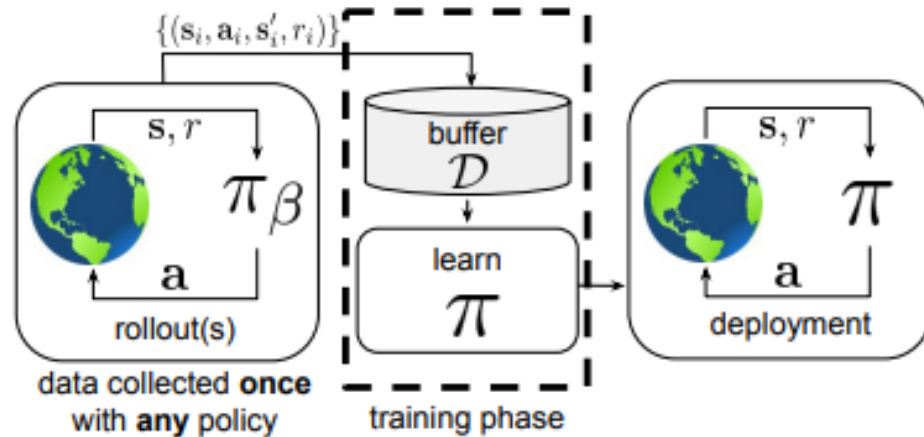
"We have a brain for one reason and one reason only – that's **to produce adaptable and complex movements**. Movement is the only way we have affecting the world around us... I believe that to understand movement is to understand the whole brain."

# Reinforcement learning as a way to use “cheap” (previously collected) data



# The RL + data problem

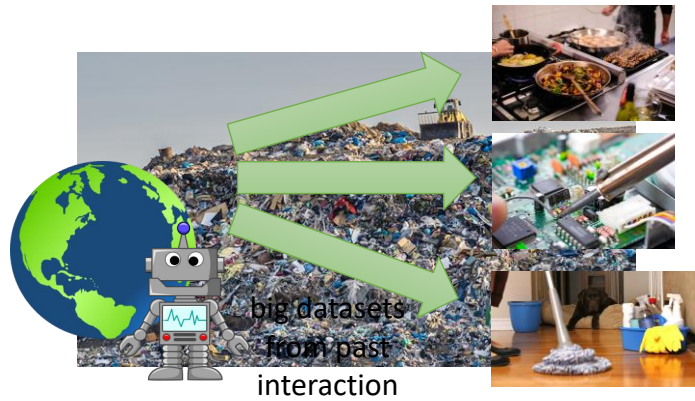
offline reinforcement learning



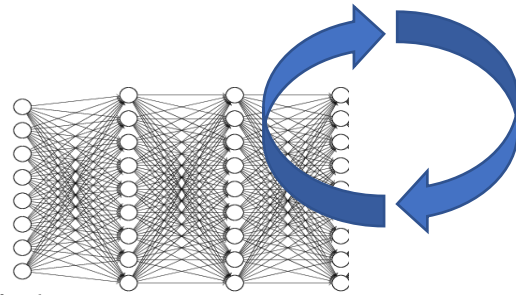
with naïve RL, this is a **costly** interactive process if done in the real world!

but self-supervised learning is about using **cheap** data that we already have lying around (in the garbage heap)!

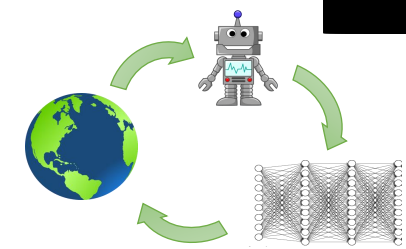
# The recipe



large dataset of diverse  
(but possibly low-quality)  
behavior



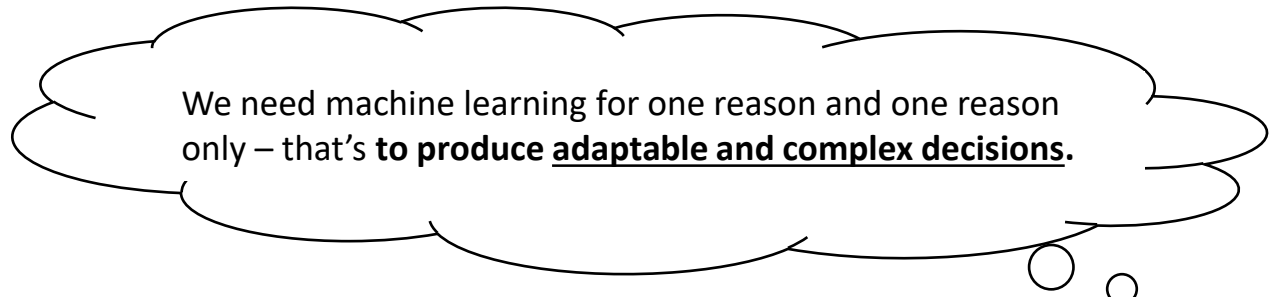
offline  
reinforcement  
learning



learning the downstream task

there are a few different  
choices here:

- human-defined skills
- goal-conditioned RL
- self-supervised skill discovery



We need machine learning for one reason and one reason only – that's to produce adaptable and complex decisions.

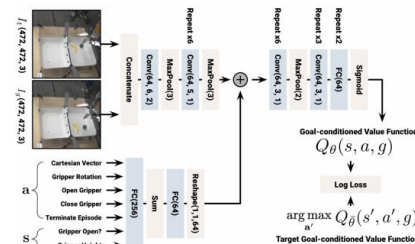


# Can we learn from offline data without **well-defined tasks**?

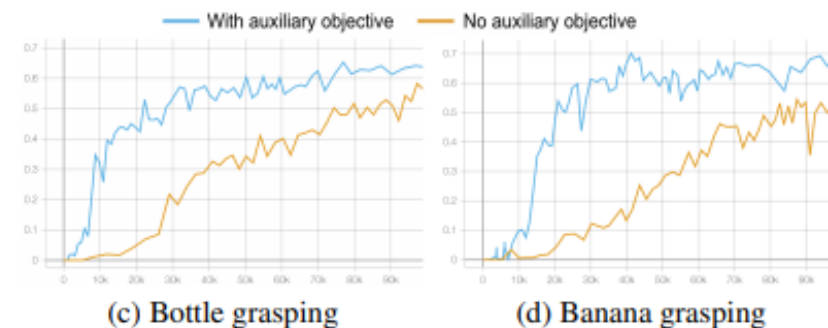
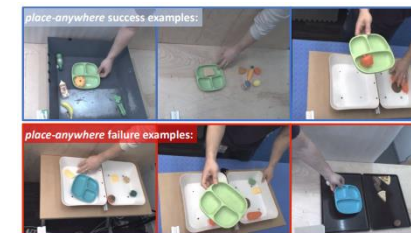


- No reward function at all, task is defined entirely using a **goal image**
- Uses a conservative offline RL method designed for goal-reaching, based on CQL
- Works very well as an **unsupervised pretraining** objective

1. Train **goal-conditioned** Q-function with offline RL



2. Finetune with a **task reward** and limited data



# Can offline RL train large language models?

What we want:



Can you teach me about behavior cloning?



Of course! I'd be happy to explain behavior cloning. To start, could you tell me if you've ever come across the terms "artificial intelligence" or "machine learning" before?



Yes, I've heard those terms, but I'm not exactly sure what they mean.



No problem at all! Let's take it step by step. Have you ever used a computer or a smartphone?

What we get:



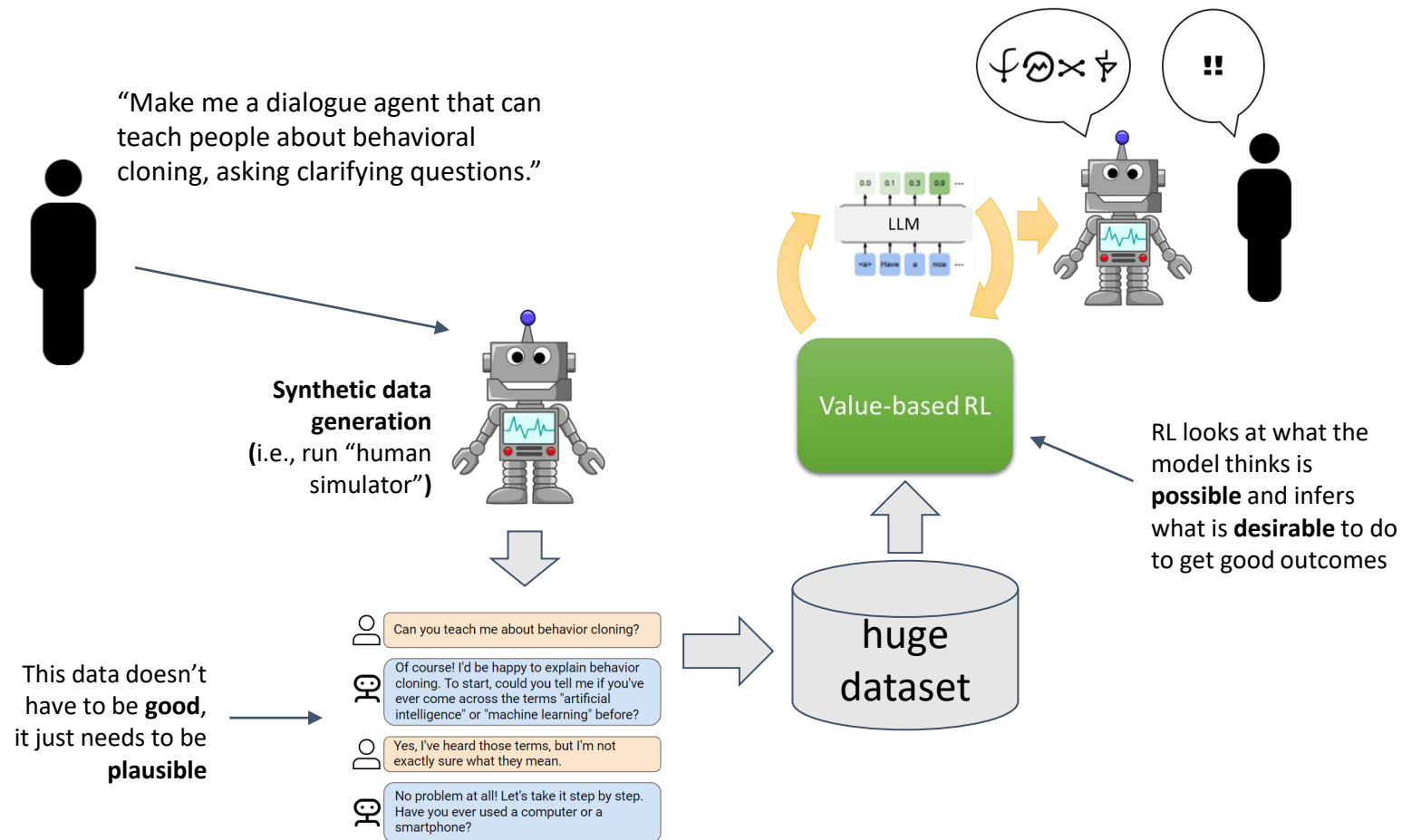
Can you explain the AI concept of behavior cloning to me? But first, I want you to ask me questions to figure out what my background in AI is, then tailor your explanation to match my level.



Certainly! Let's start by understanding your background in AI to provide an explanation that matches your level of knowledge. Please answer the following questions:

1. Have you ever studied or worked with artificial intelligence or machine learning concepts before, even at a basic level?
2. Are you familiar with terms like "neural networks," "training data," or "supervised learning"?
3. Have you ever heard of or worked with specific AI tools or frameworks like TensorFlow, PyTorch, or scikit-learn?
4. Are you interested in a high-level overview or a more detailed explanation of behavior cloning?
5. Do you have any specific examples or applications in mind where you'd like to see behavior

# Model-based offline RL for multi-turn dialogue



RL + LLM is great for tasks that are easy for LLMs to **simulate** but hard for them to **perform optimally**.

With RL, instead of emulating humans, LLMs can learn to achieve desired outcomes using their understanding of how humans behave.



# Example: model-based RL for multi-turn dialogue

## GPT Agent Dialogue:

H: Hi, I've been hearing about something called "behavior cloning" but I don't really understand what it is. Can you explain it to me?

AI: Certainly! Let's start by understanding your background. Please answer the following questions: Have you ever worked with artificial intelligence or machine learning concepts? Are you familiar with terms like "neural networks," "training data," or "machine learning algorithms"? You interested in a high-level overview or a more detailed explanation?

H: I have some basic knowledge of artificial intelligence and machine learning. I'm an expert. I am familiar with neural networks and supervised learning training data.

AI: Great, thank you for that information. Behavior cloning is a type of machine learning algorithm where a computer learns to perform a task by imitating a human's behavior. It does this by closely analyzing and mimicking the actions of a human expert, using a large dataset of examples. This approach is often used in tasks that don't have a clear set of rules or can't be easily solved through traditional programming methods. Does that make sense so far?

H: Yes, that's starting to make sense. What do you mean by expert?

AI: An expert, in this context, refers to a human or an advanced system that knows how to solve a particular task or problem. The AI model learns the expert's behavior to learn how to perform the task itself. Does that clarify things?

H: Yes, thank you.

## IE-RL Agent Dialogue:

H: Hi there! I've heard about this thing called "behavior cloning" in AI, but I don't really understand what it is. Can you explain it to me?

AI: Of course! I'd be happy to explain behavior cloning. To start, could you tell me if you've ever come across the terms "artificial intelligence" or "machine learning" before?

H: Yes, I've heard those terms, but I'm not exactly sure what they mean.

AI: No problem at all! Let's take it step by step. Have you ever used a computer or a smartphone?

H: Yes, I use my computer and phone all the time.

AI: Great! So, imagine that a computer or a smartphone could learn to do something by watching you do it. That's a bit like what behavior cloning is.

H: Oh, interesting. Can you explain more about how that works?

AI: Sure. Behavior cloning is a type of machine learning algorithm where a computer learns to perform a task by imitating a human's behavior. It does this by closely analyzing and mimicking the actions of a human expert, using a large dataset of examples. This approach is often used in tasks that don't have a clear set of rules or can't be easily solved through traditional programming methods. Does that make sense?

H: Yes, I think so. So how does the computer learn to mimic the behavior?

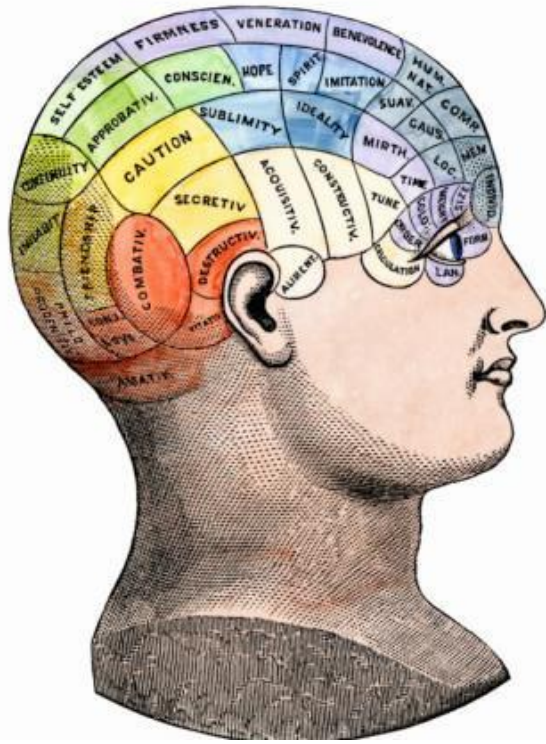
AI: That's a great question. The computer is fed a large amount of data that shows different examples of the task being performed by a human expert. The computer then uses this data to identify patterns and develop a model of the behavior. This model allows the computer to predict the correct actions to take in a given situation. Over time, the computer can continue to refine its model and improve its performance. Does that help clarify things?

H: That does. Thank you!

Back to the Bigger Picture



# Learning as the basis of intelligence




- Reinforcement learning = can reason about decision making
- Deep models = allows RL algorithms to learn and represent complex input-output mappings

Deep models are what allow reinforcement learning algorithms to solve complex problems end to end!

# What is missing?

➤ **How Much Information Does the Machine Need to Predict?** Y LeCun

- **"Pure" Reinforcement Learning (cherry)**
  - ▶ The machine predicts a scalar reward given once in a while.
  - ▶ **A few bits for some samples**
- **Supervised Learning (icing)**
  - ▶ The machine predicts a category or a few numbers for each input
  - ▶ Predicting human-supplied data
  - ▶ **10→10,000 bits per sample**
- **Unsupervised/Predictive Learning (cake)**
  - ▶ The machine predicts any part of its input for any observed part.
  - ▶ Predicts future frames in videos
  - ▶ **Millions of bits per sample**



■ (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

# Where does the *signal* come from?

- Yann LeCun's cake
  - Unsupervised or self-supervised learning
  - Model learning (predict the future)
  - Generative modeling of the world
  - Lots to do even before you accomplish your goal!
- Imitation & understanding other agents
  - We are social animals, and we have culture – for a reason!
- The giant value backup
  - All it takes is one +1
- All of the above

# How should we answer these questions?

- Pick the right problems!
  - Ask: does this have a **chance** of solving an important problem?
  - Optimism in the face of uncertainty is a good exploration strategy!
- Don't be afraid to change the problem statement
  - Many of these challenges won't be met by iterating on existing benchmarks!
- Applications matter
  - Sometimes applying methods to realistic and challenging real-world domains can teach us a lot about the important things that are missing
  - RL has a long history of disregarding this fact
- Think big and start small