University of California, Irvine

CS 295: Optimal Control and Reinforcement Learning

Winter 2020

Lecture 18: Multi-Task Learning

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

Today's lecture

- Transfer learning
  - World model
  - Perceptual features
  - Policy
- Domain randomization + adaptation
- Curriculum learning
- Shared learning

Learning from very little data

- As the number of learnable tasks grows
  - sample complexity per task must drop to be practical
- Our goal: learn a new task with
  - 0-shot: no new training interactions (exploration / demonstration)
  - 1-shot: single training episode
  - few-shot: very few training episodes

Prior knowledge

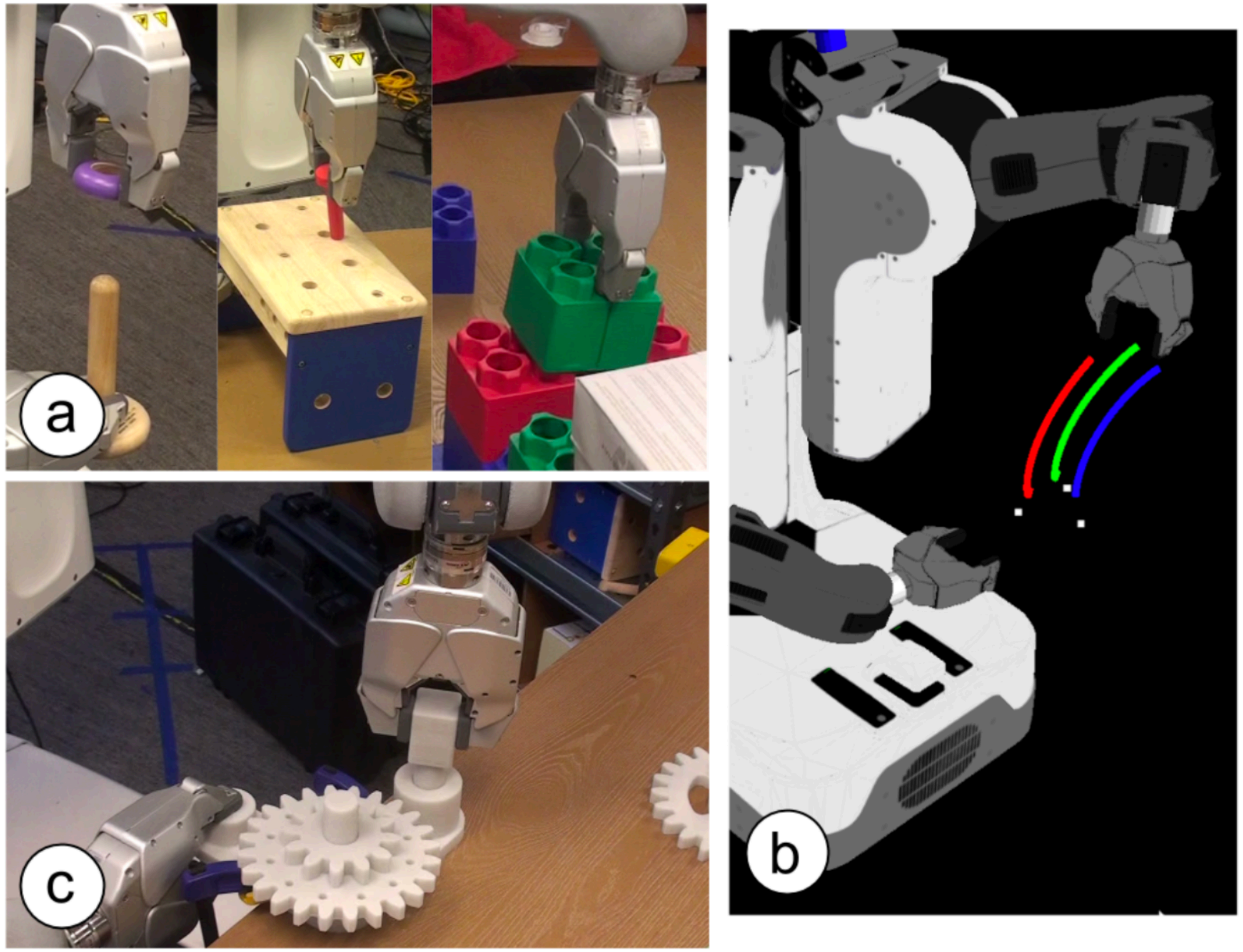
- To only need little information from data, the rest must be a-priori
- Programmed prior knowledge:
  - Programmed policy / skills
  - Choice of observation and action representations
    - Feature extraction
  - World model (dynamics / reward)
  - Learner model class / neural network architecture

Learning prior knowledge from other tasks

- Transfer learning: first learn other task(s), then solve new task
  - with (>0-shot) or without (0-shot) more learning in the new task
- Practical question: what knowledge is transferred / shared
  - World model
  - Perceptual features
  - Value function / policy
  - More later...

World model transfer

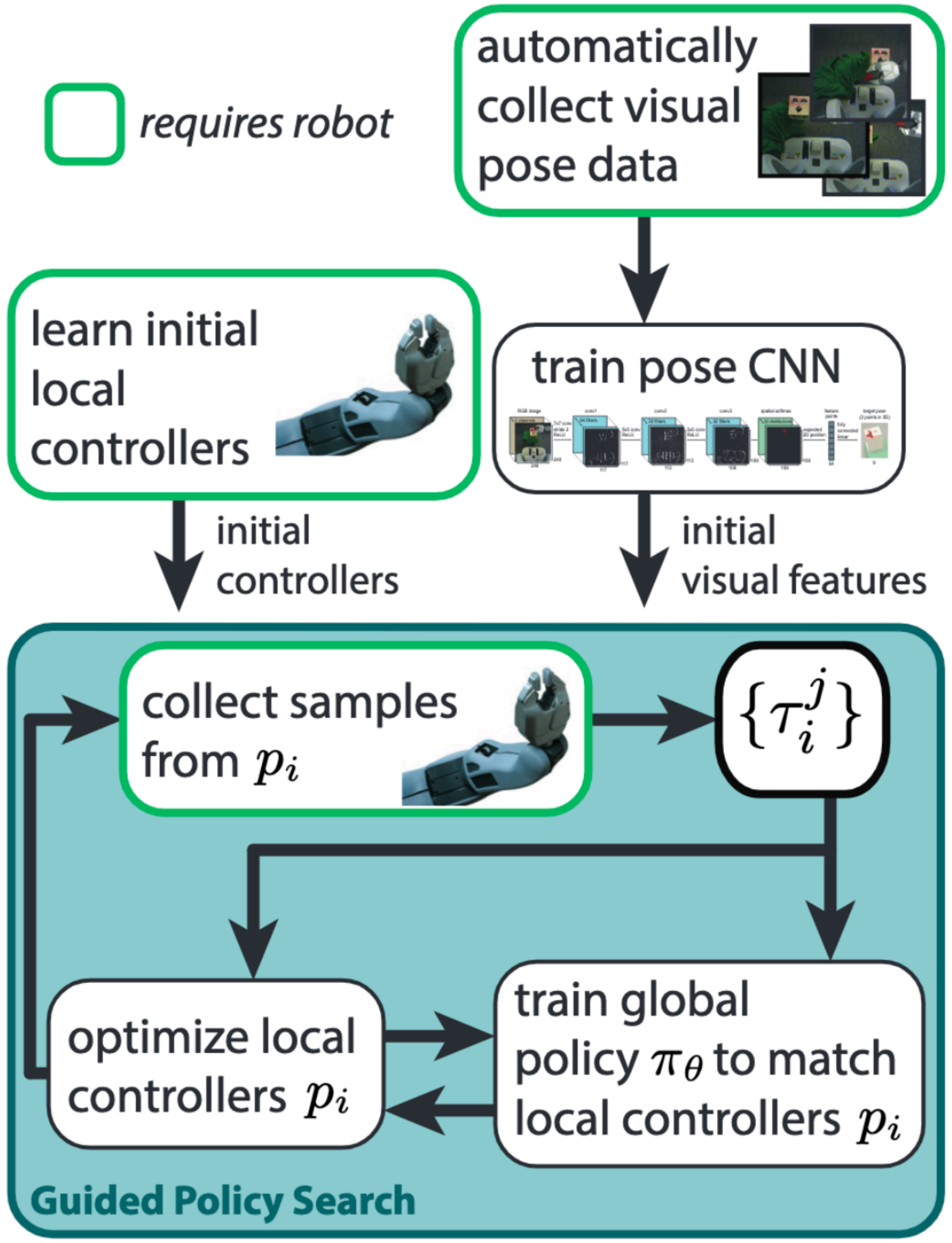
- Interact on many related task(s)
- Fit a world model to the dynamics
- Model-based RL of a new task



- Problem: prior model is inaccurate
- Solution: take the pre-trained model
    - and fine-tune it using new task data

Perceptual features transfer

- Interact to collect (robot pose, image) data
- Train perceptual features that recover image → pose
- RL with perceptual features as observations
- May again benefit from fine-tuning



Policy transfer

- Find similar task(s) where data is abundant
  - Many demonstrations / exploration episodes can be obtained
  - E.g. simulator of the world → real world (sim2real)
- Train policy with RL / IL in the abundant domain
- Execute policy in the scarce domain
  - Or fine-tune with further few-shot RL / IL, as needed

Soft-optimal policies for fine-tuning

- Problem: policy can "overfit" to pre-training task
  - Policy may become deterministic
    - unfit for exploration
    - optimizer may struggle to again introduce uncertainty
  - Perceptual features may deteriorate to only what's needed for actions
- Solution: keep policy soft-optimal
  - Max entropy subject to sufficiently high value

SQL pre-training helps fine-tuning

Domain randomization

- Choosing a source domain to match the target domain may be hard
- Can we do better with multiple source domains?
  - Define distribution over tasks that supports the target = interpolation
  - Generalize even outside the support = extrapolation

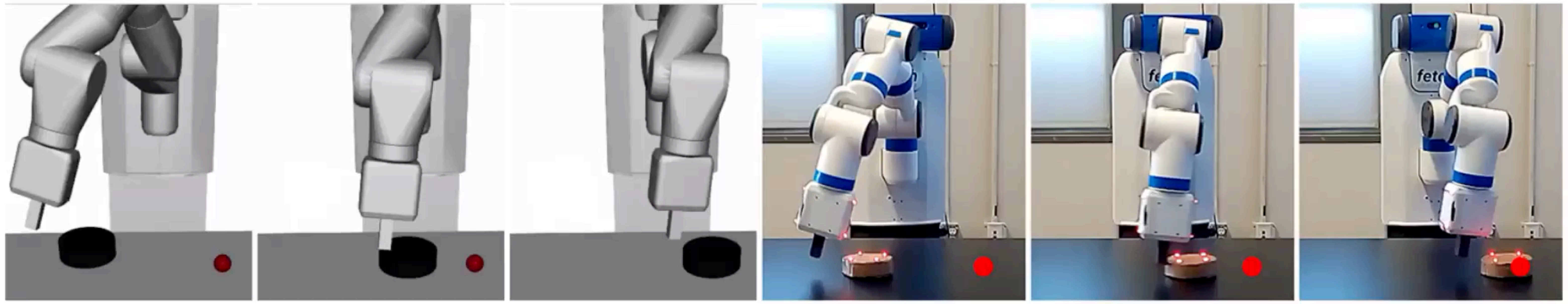
Sim2real with domain randomization

Sim-to-Real Transfer of Robotic Control with Dynamics Randomization

Xue Bin Peng[1,2], Marcin Andrychowicz[2], Wojciech Zaremba[2], Pieter Abbeel[1,2]

[1]Electrical Engineering and Computer Sciences, UC Berkeley, USA

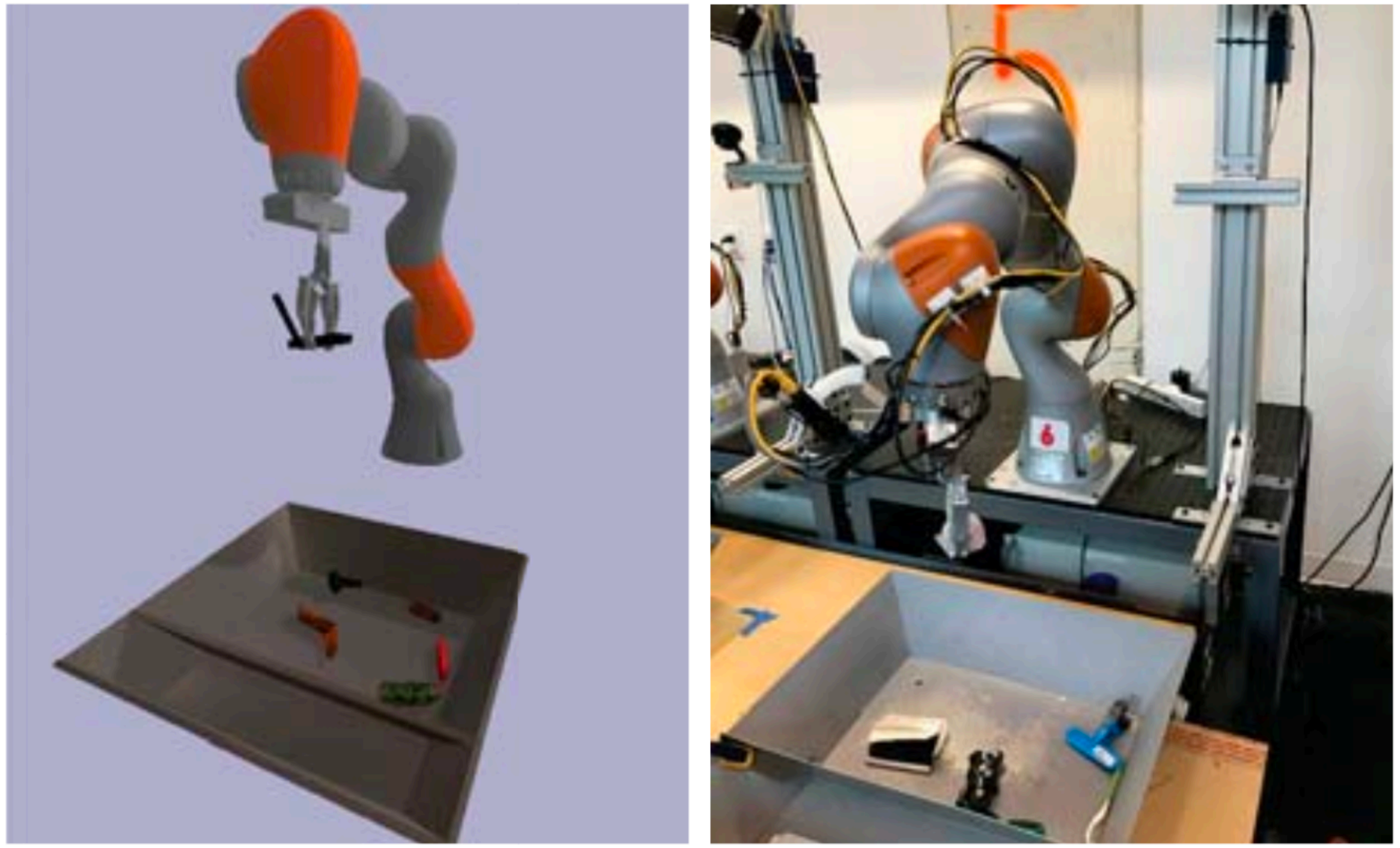
[2]OpenAI, USA



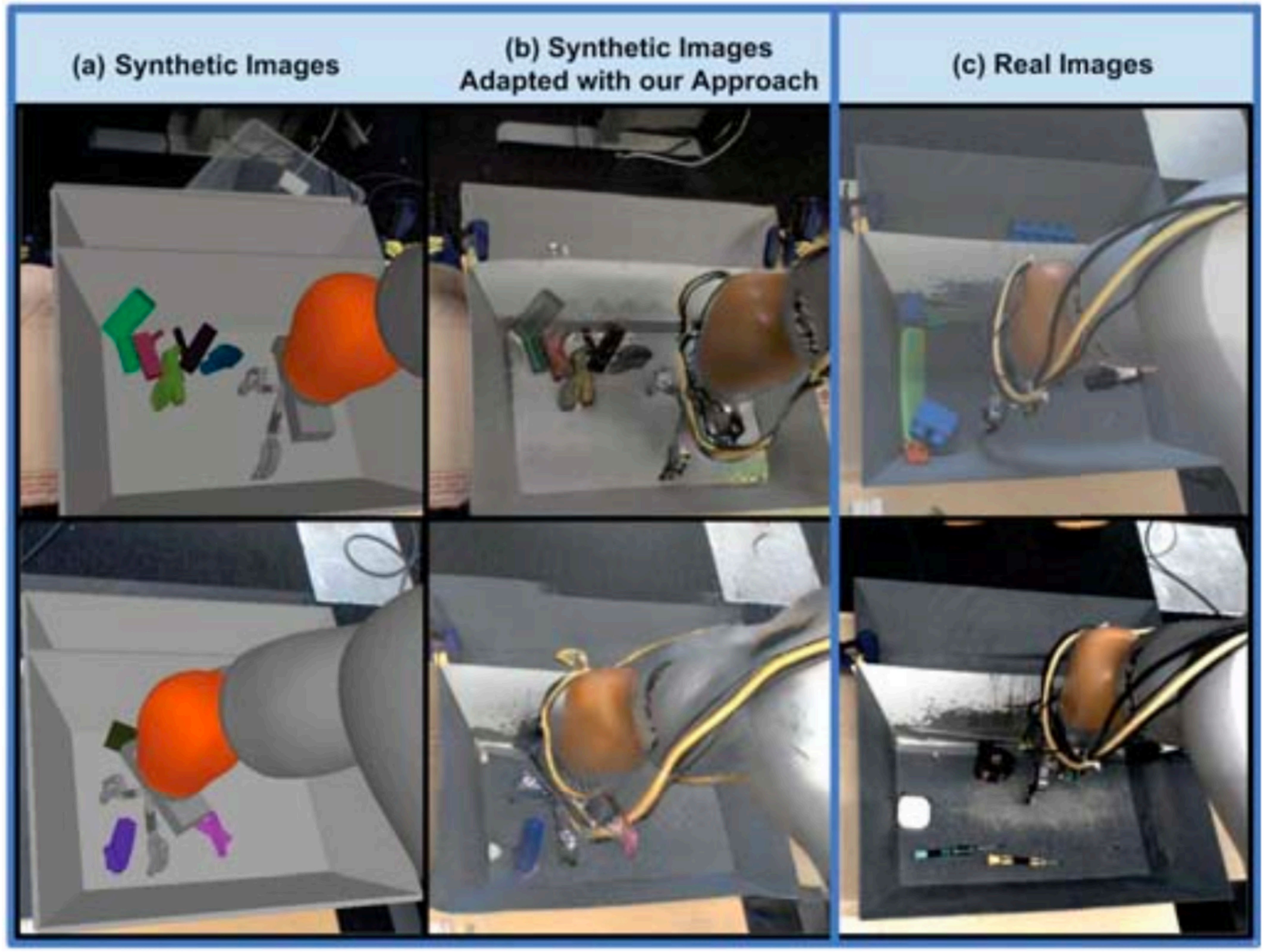
Domain adaptation

- Domain randomization alleviates the need for target domain knowledge
  - But much is still needed: a simulator in the ballpark, the randomization ranges
- The more we know about target domain, the better we can adapt source
- Can we automate this adaptation process?
  - Using target-domain data

Sim2real with domain adaptation



(a) Simulated World (b) Real World

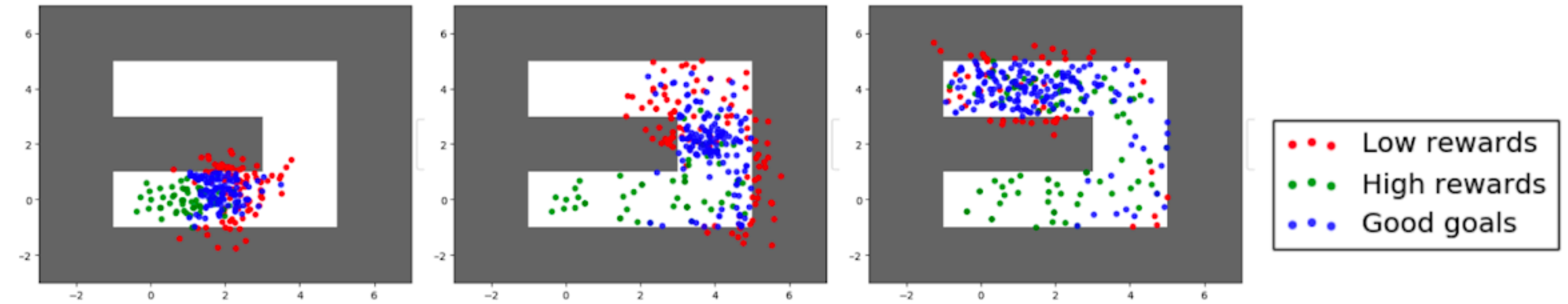


Curriculum learning

- Another purpose for policy pre-training
  - (So far: to have more data)
  - Start by learning easy versions of the task, make it gradually harder
- If a task is hard to solve by itself, "training wheels" can help
  - Exploration never finds rewards? Shorten task
  - Rewards don't encourage exploring / reaching subgoals? Leave "breadcrumbs"
  - Poor SGD convergence properties? Coarsen states / actions / time
  - Challenging state inference under partial observability? Add observability

Goal GAN

- Sample goals, roll out policy
    - Reject goals with too low / high rewards



(a) Iteration 5 (b) Iteration 90 (c) Iterartion 350

- Train GAN to generate goals with this distribution
- Train agent on generated goals

- Generated goals are intermediate-level:
    - just hard enough for the agent to learn something new
    - not so hard that it struggles to do it

Multi-task learning settings

- <u>Transfer learning</u>
  - Earlier domains / tasks are only stepping stones towards the ultimate task
- <u>Shared learning</u>
  - Learn multiple tasks jointly, have them inform each other
- <u>Lifelong learning</u>
  - Learn tasks as they occur, but also keep past abilities
    - No <u>catastrophic forgetting</u>, where fine-tuning a model degrades its quality for old task

Shared learning

- Sharing a world model / perceptual features
  - Similar to above
- Sharing a policy
  - Multi-task policy distillation
  - Task-aware policy
- Sharing modules in a structured policy
  - Multi-task hierarchical imitation learning (HIL-MT)

Policy distillation

- Policy distillation = behavior cloning of existing policy with cross-entropy loss
- Wait, but... why?! if we already have the policy
  - Network compression
  - Track "average" policy (stabilize training, fictitious play in game theory, etc.)
  - Combine multiple policies into one

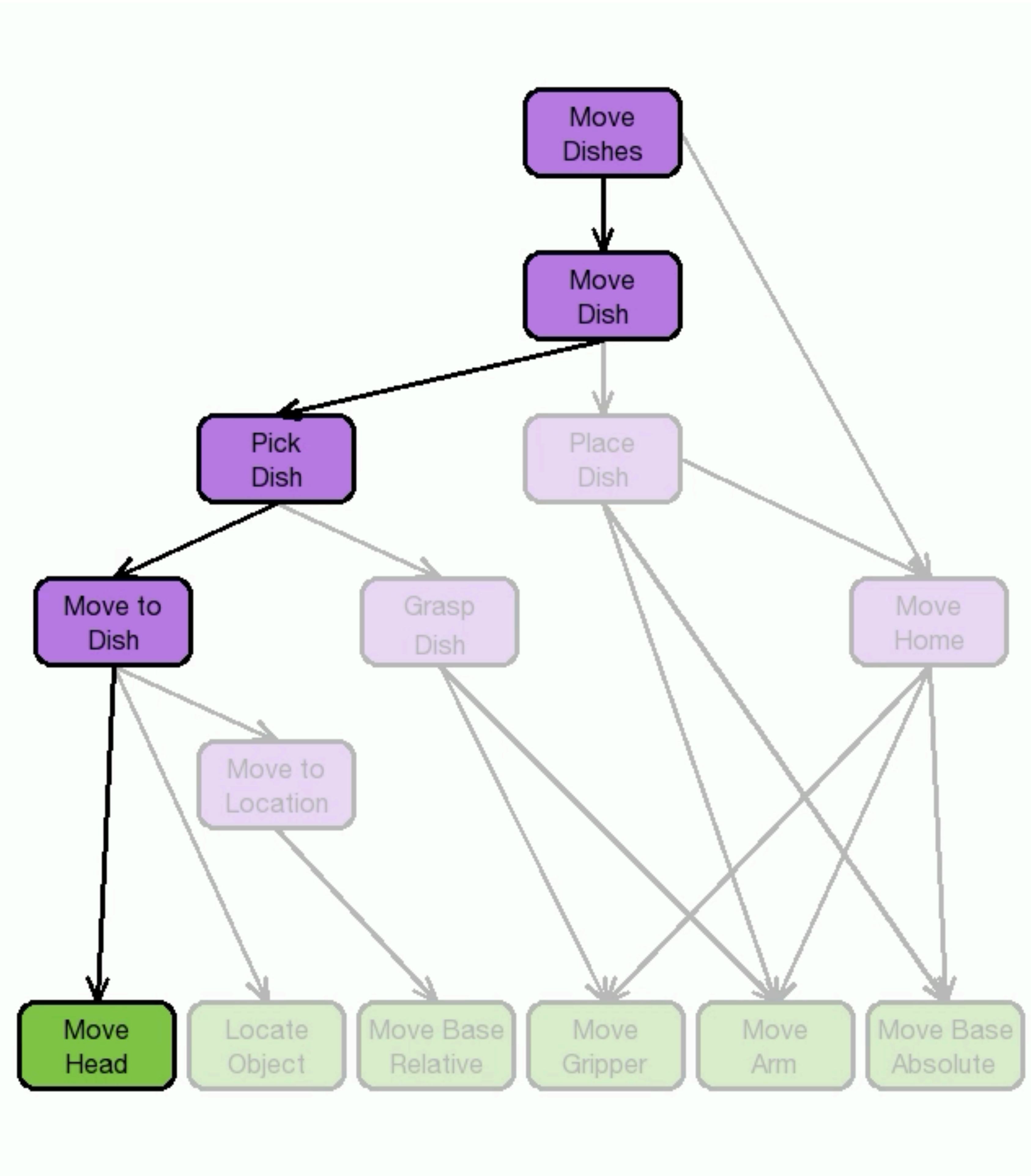
Multi-task policy distillation

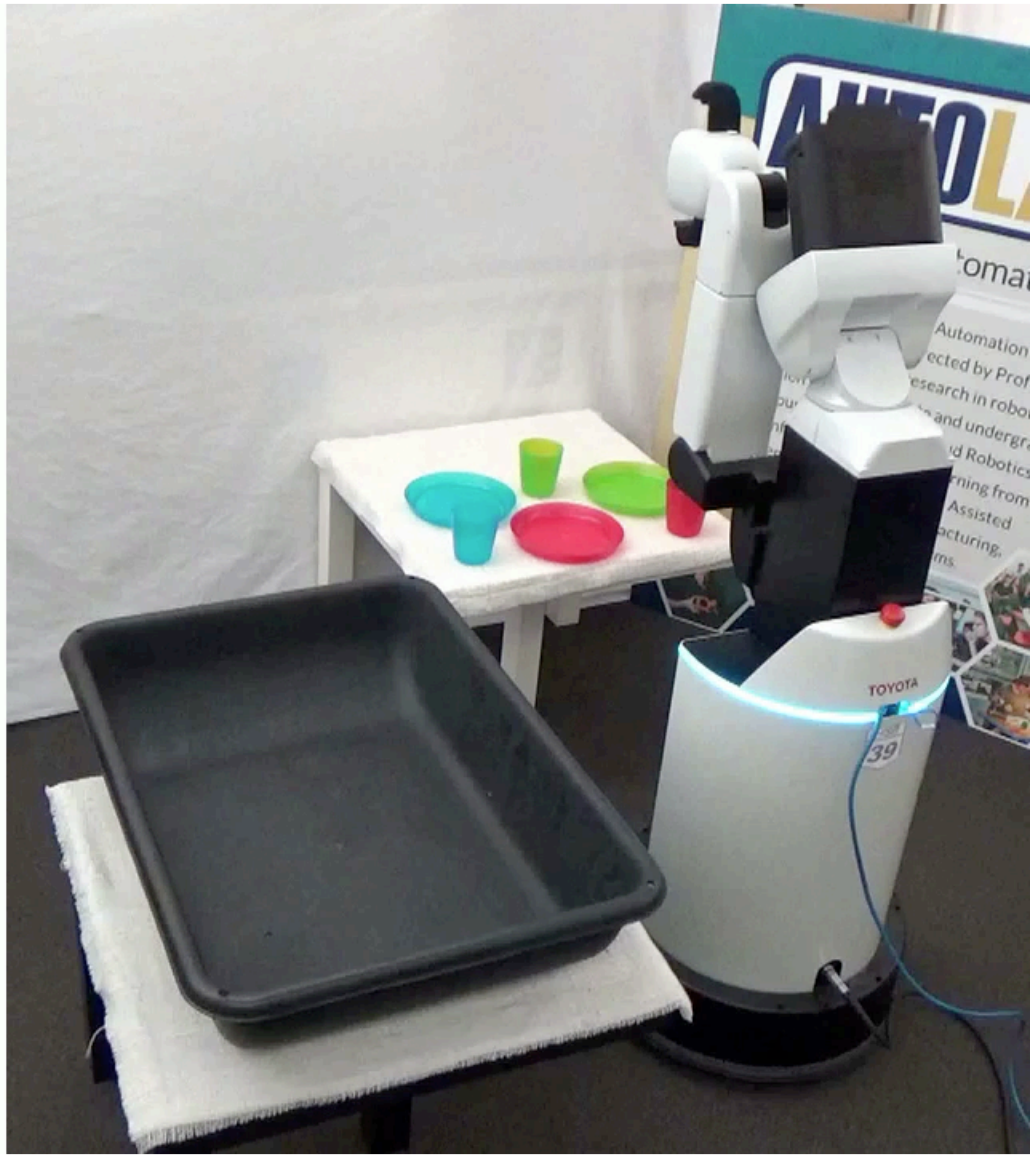
- Train a policy for each task
- Distill them into one policy
- Pros:
  - This is like combining the empirical evidence from all datasets = more data
  - If tasks are related, distilled policy can be more stable
- Cons:
  - If tasks aren't related, they compete for network capacity
  - One very wrong distilled policy can ruin it for everyone

Task-aware policy

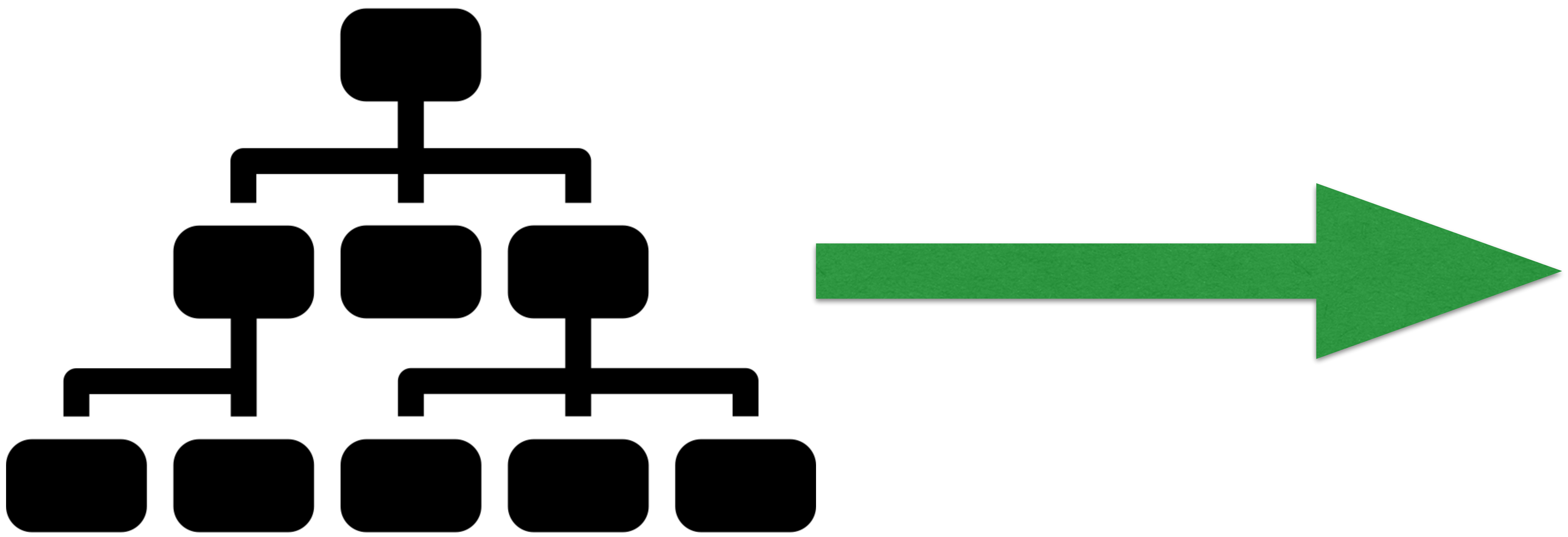
- Similar to goal-conditioned behavior cloning (Lecture 2)
  - Except, a goal is a state, a task = (dynamics + reward) is more general
- Separate policy $\pi_\tau(a|s)$ for each task is like one task-aware $\pi(a|s,\tau)$
  - With the task given by its index
- Sometimes, a task has a natural embedding
  - Direction + speed for Walker / Swimmer / Hopper
  - Index of block + position to place it
- Otherwise, can learn to embed task specification (demo / text / target image)

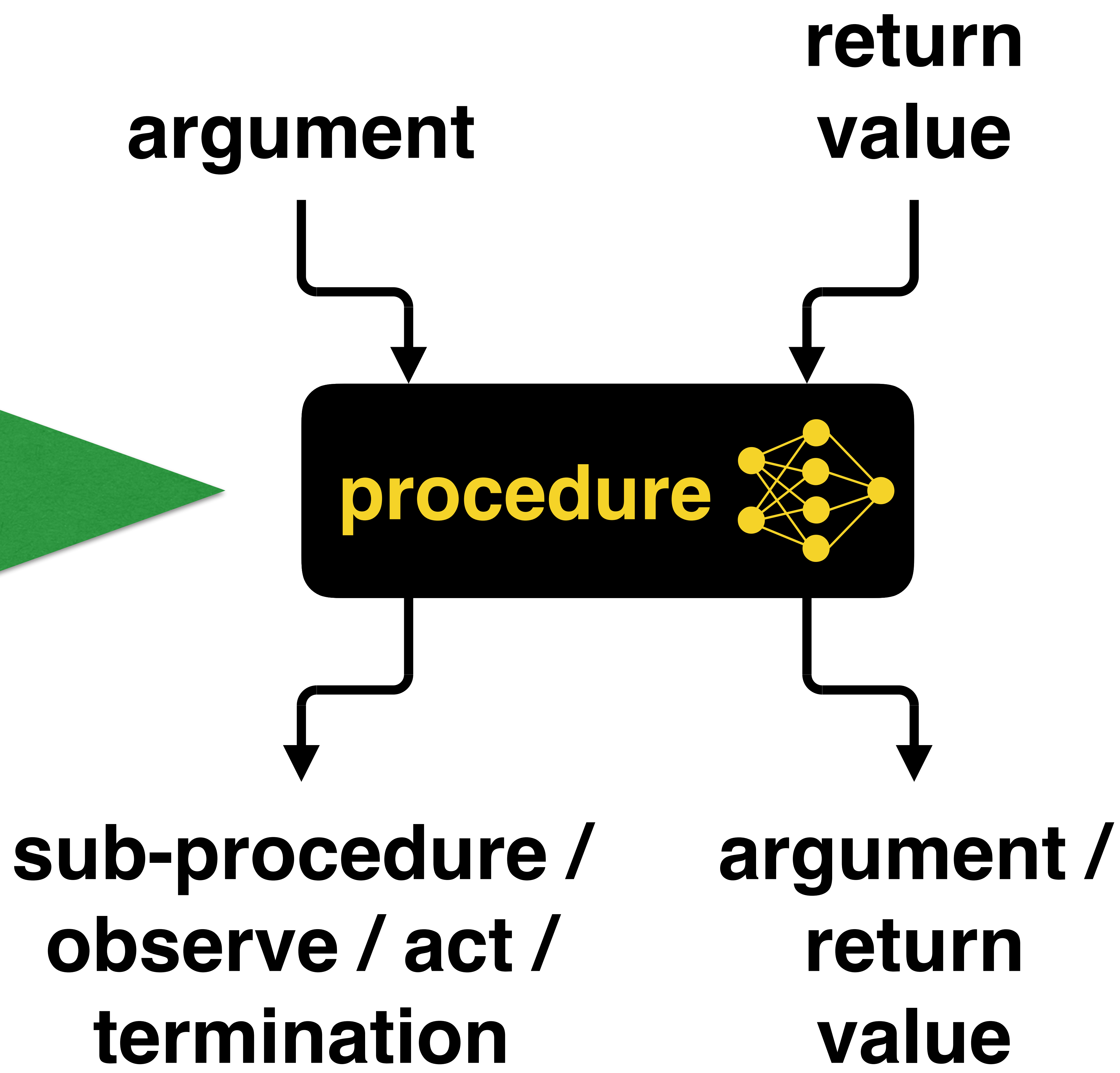
Learning from annotated demonstrations





Supervised learning each skill

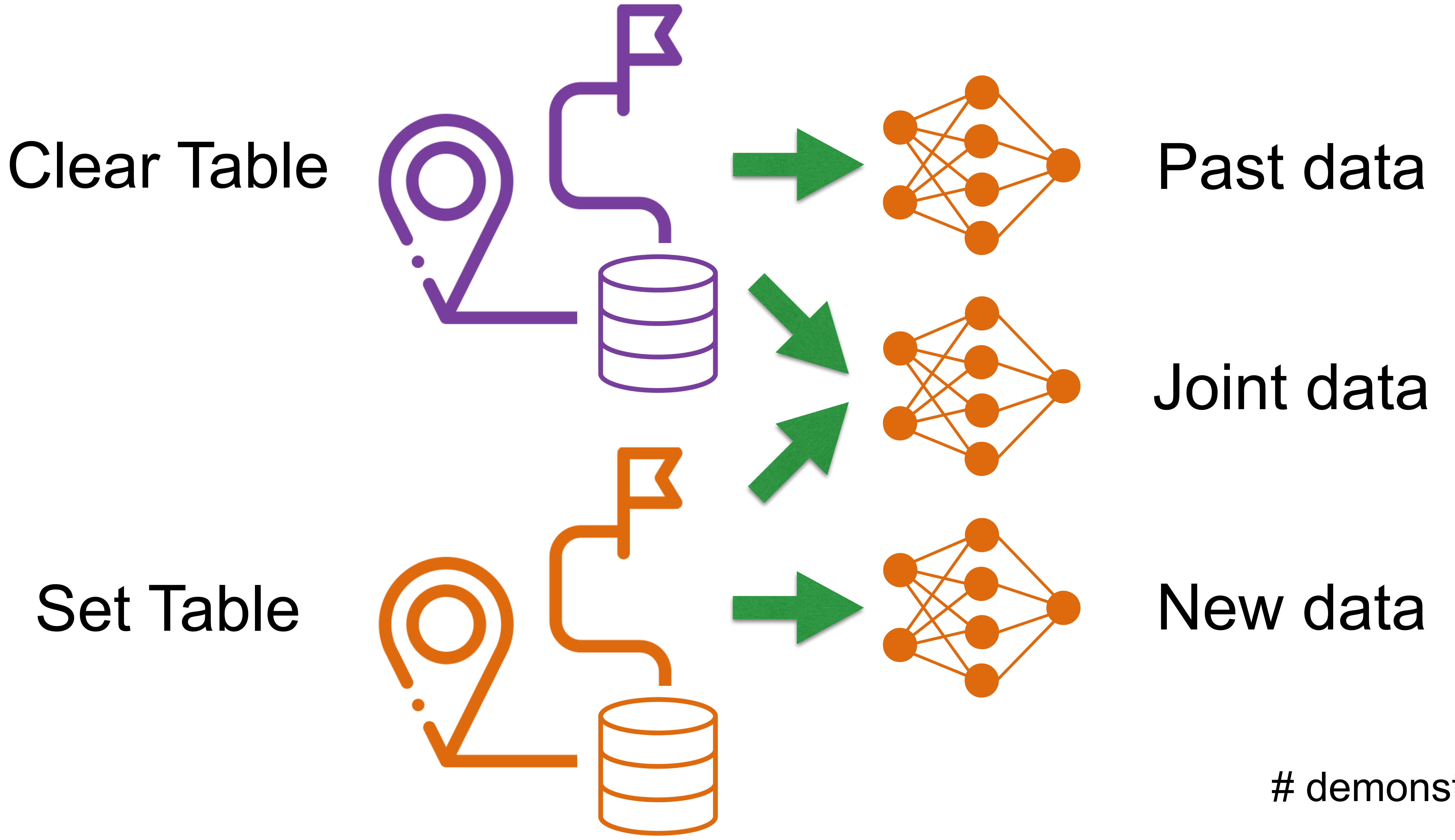




$$\log p_\theta(\text{procedure steps}) = \sum_i \log p_\theta(\text{procedure step } i)$$

Multi-task hierarchical imitation learning (HIL-MT)

- Hierarchical control allows per-procedure selection of multi-task mode



demonstrations to learn Clear Table → Set Table

$\mathcal{D}_{clear}$	$\mathcal{D}_{set}$	$\mathcal{D}_{clear} \cup \mathcal{D}_{set}$	Per-skill selection
Failed	19 ± 0.3	Failed	**11.6**± 0.25

Recap

- Reuse data between related tasks
  - May hurt if tasks are unrelated
- To improve the task overlap: soft-optimality, randomization, adaptation
- Shared learning may benefit both ways
- Modularity allows mix-and-match of best approach
- Did not talk about: meta-learning, lifelong learning