[Deep]
v
Transfer in Reinforcement Learning

April 3, 2017

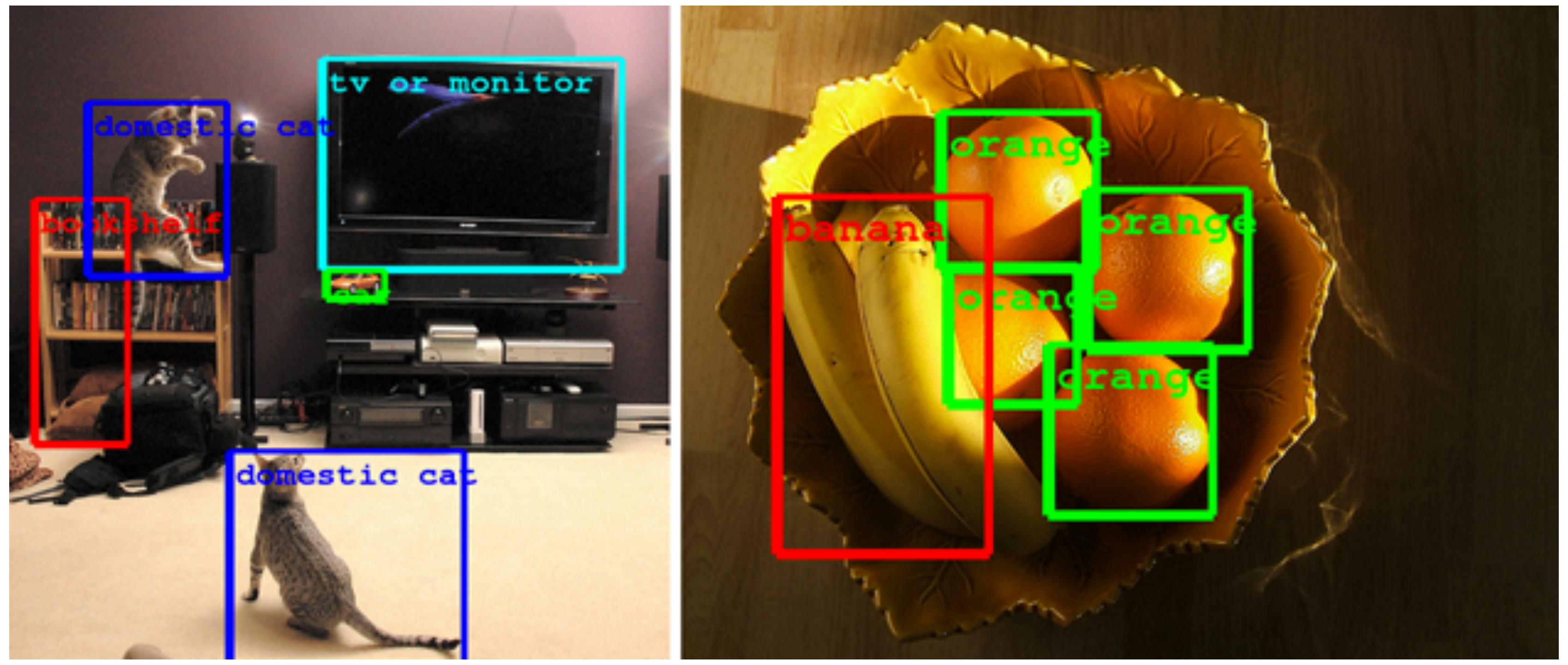
Chelsea Finn

Course Reminders:
~~March 22nd: Project group & title due~~
April 17th: Milestone report due & milestone presentations
April 26th: Beginning of project presentations

Starting Wednesday: guest lectures

Deep Learning Success Stories


speech recognition



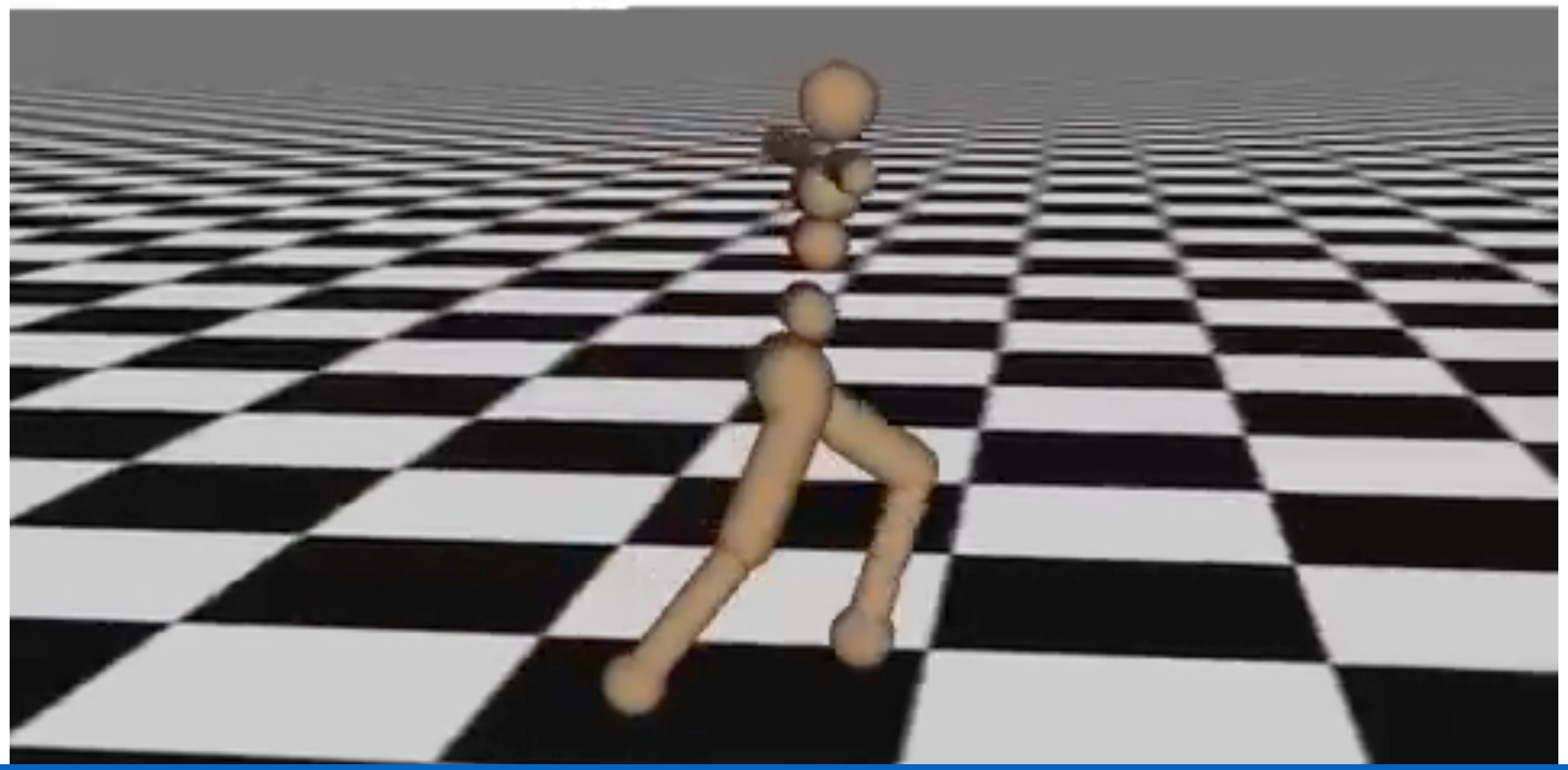
object detection

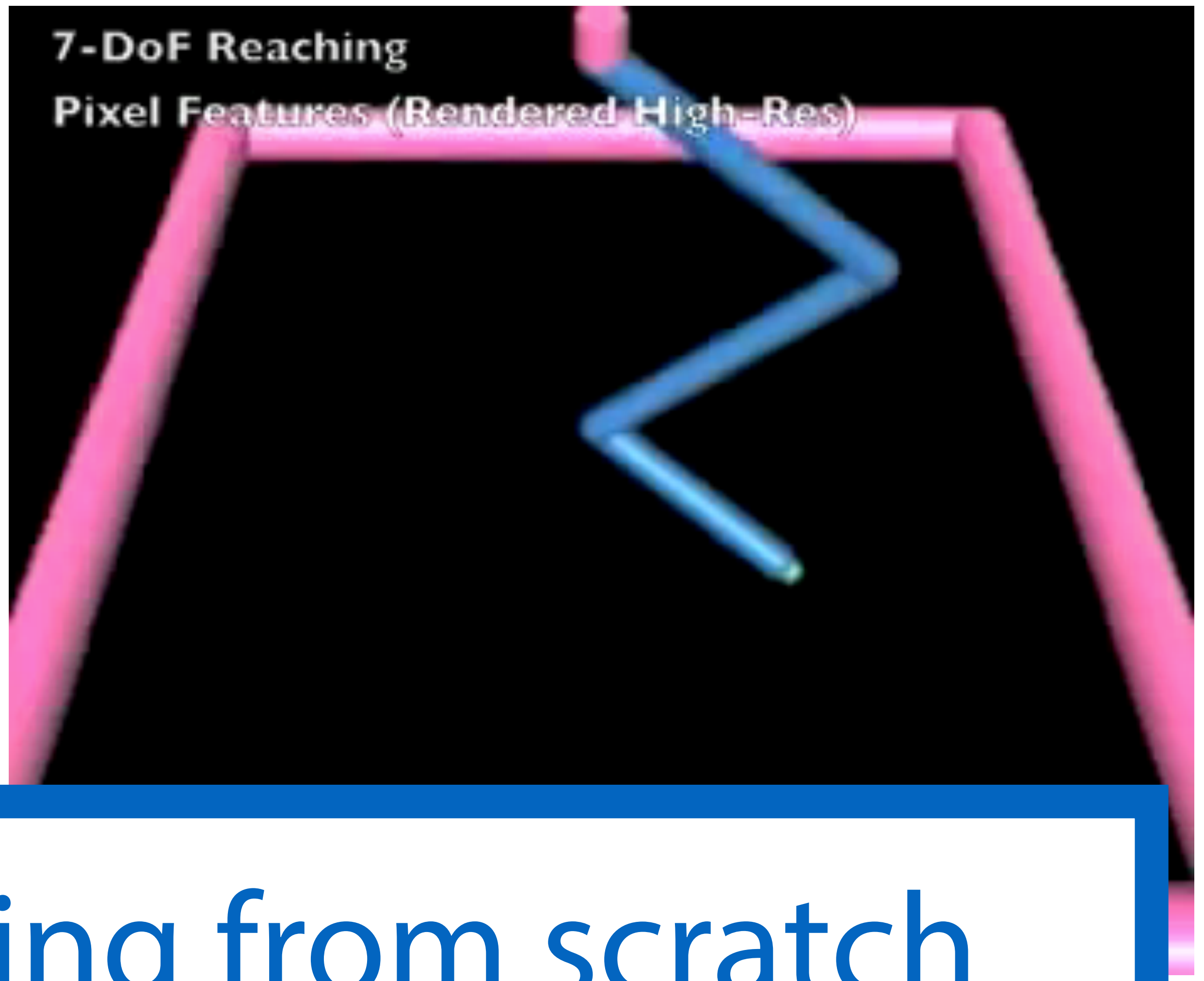
machine translation

+ can handle raw sensory observations ✓
+ scales to diversity of the real-world ✗

This course: deep learning for behavior

Deep Reinforcement Learning





train/test on 1 task in 1 environment, starting from scratch

Levine*, Finn*, et al. '15

Chebotar et al. '17

Yahya et al. '17

Why Transfer?

Don’t start from scratch every time.

Use cheap experience in simulation for learning real-world skills.

Enable agent to effectively act in an environment it hasn’t seen before.

What is Transfer?

many definitions

this lecture:

transfer learning: using experience from one set of tasks for faster learning and/or better performance on a new task

A broad notion of "task":
- varying objectives (reward)
- varying robots (can affect state, action, and dynamics)
- varying environments (can affect observation space, dynamics, reward)

Often make assumptions about what will change across tasks.

Note: can treat whole world as a single MDP (and not worry about transfer)
but usually more efficient to model how the world changes

What is Transfer?

transfer learning: using experience from **one set of tasks** for faster learning and/or better performance on a **new task**

Some terminology…

source domain(s) -> **target domain**

0-shot generalization: don't use any data from target domain

few-shot generalization: use small amount of data from target domain

faster learning: use less data than training from scratch

Evaluating Transfer

How should we evaluate how well learning transfers?

<u>For real robots</u>: largely ad-hoc

Popular simulated benchmarks with no generalization evaluation

Bellemare et al. '13

Brockman et al. '16

(Though, because these are well-known, people have used the environments to evaluate methods for transfer.)

Evaluating Transfer

How should we evaluate how well learning transfers?

Some recently proposed environments with diversity:

MINECRAFT
PROJECT MALMO PUBLIC
navigation, collaboration

DeepMind Lab
maze navigation

UNIVERSE
wide range of video games

VIZDOOM
DOOM video game

CommAI-env
communication-based tasks

not yet released: Starcraft II (DeepMind)

generally, no consensus on how to evaluate

Approach 0: Try it, and hope for the best

Sometimes, a policy trained in one domain will generalize in others
but no reason for it to succeed with enough variation

Approach 0.5: Fine-tuning

Initialize with a policy trained on source task(s) and fine-tune on target task
works well with ImageNet pre-training, doesn't seem to work well for RL*

Some potential reasons as for why:

- We don't have ImageNet for behavior
- RL networks tend to be much smaller than vision networks
- RL algorithms are unstable at beginning, when there is no data

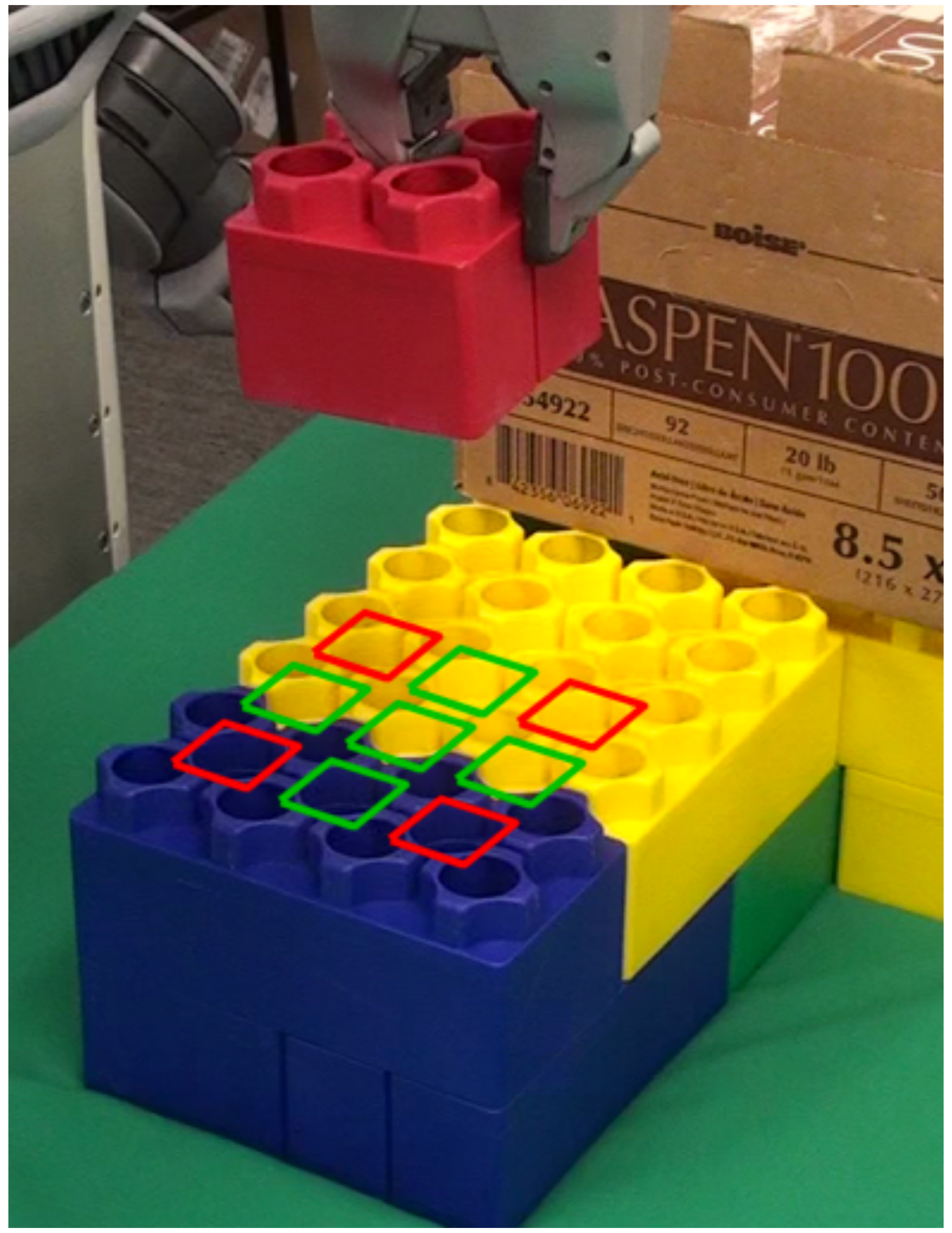
*Note: a couple approaches discussed today will involve fine-tuning

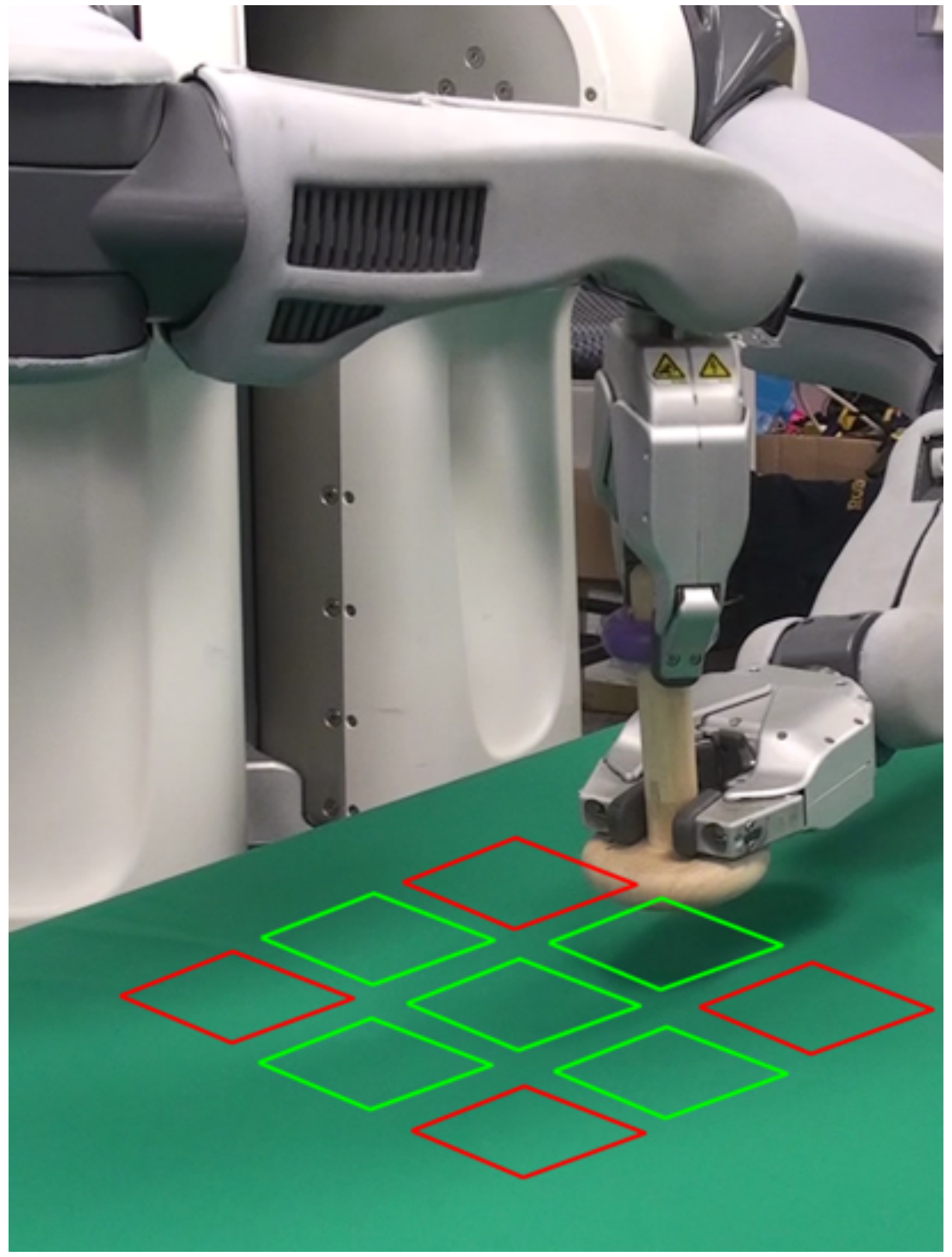
Outline: Achieving Transfer in RL

1. **Handling changes in reward**
   a. task represented in the observation
2. **Handling changes in environment**
   a. diversity for sim-to-real transfer
3. **Reusing representations**
   a. progressive networks
   b. PathNet
   c. modular networks
4. **Meta-learning**
   a. Learning to learn quickly
   b. Few-shot adaptation

Approach 1: Pass in Task Representation

Goal represented in observation, train policy across goals





Pros:
- simple
- 0-shot generalization to new goals

Cons:
- need to densely sample goals for good generalization/transfer
- task may be hard to represent

approach also applicable to differing objects/dynamics

Case Study: Multi-task Learning on Atari

Goal: learn a single policy that can play all Atari games

POLICY DISTILLATION

Andrei A. Rusu, Sergio Gómez Colmenarejo, Çağlar Gülçehre,* Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu & Raia Hadsel
Google DeepMind

ACTOR-MIMIC DEEP MULTITASK AND TRANSFER REINFORCEMENT LEARNING

Emilio Parisotto, Jimmy Ba, Ruslan Salakhutdinov
Department of Computer Science
University of Toronto

Note: no need to explicitly pass in task representation

Background: Ensembles & Distillation

Easy way to extract knowledge from training data:

train many different models in parallel, then take the average prediction

"ensemble"

how almost all ML competitions are won

but: expensive at test time…

Idea: "distill" knowledge from ensemble of networks into a single smaller network

train on soft targets:

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}$$

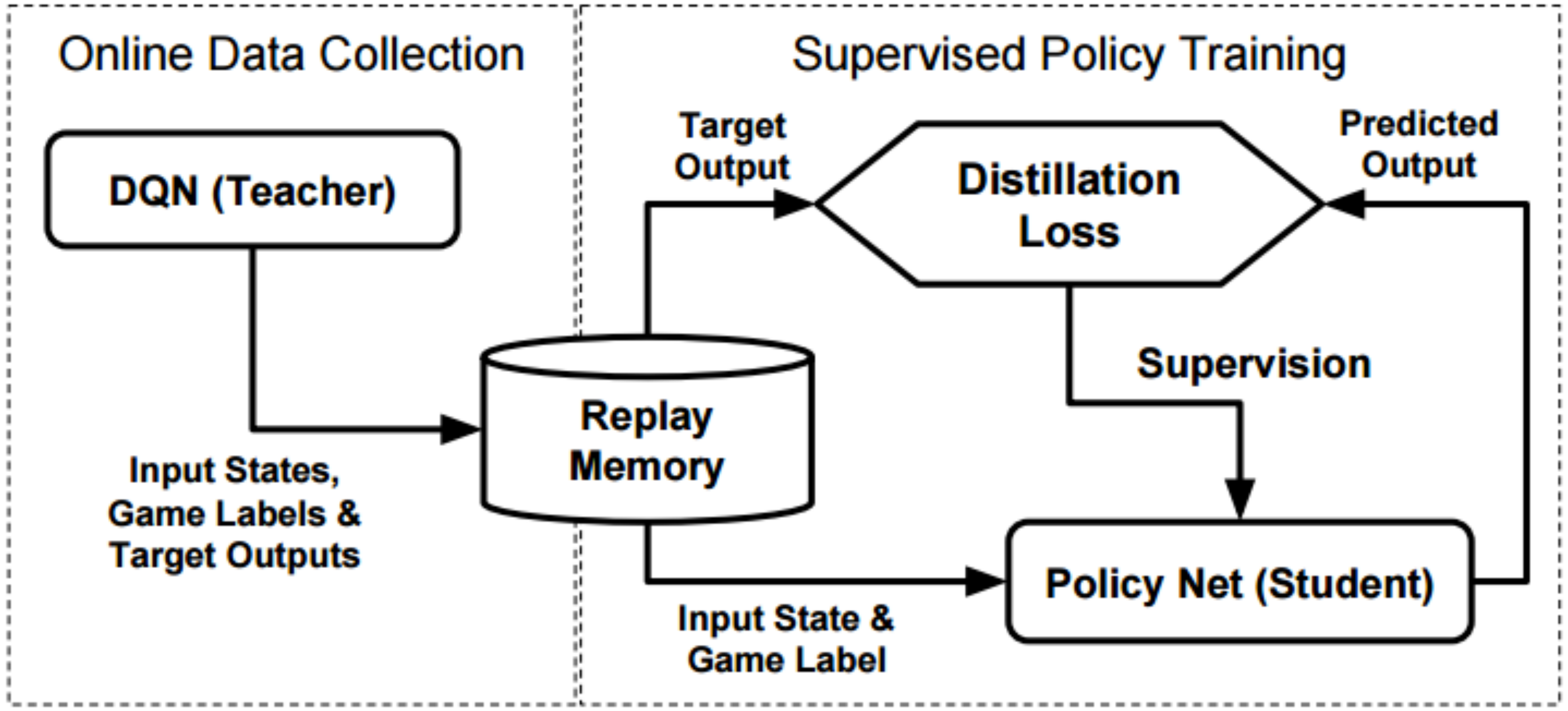
average probabilities over models to get z_i, further soften with temperature T

slide adapted from Geoff Hinton

Case Study: Multi-task Learning on Atari

1. train N DQN agents on N tasks, simultaneously
2. train single student network to mimic Boltzmann distribution of DQN agent [distillation*]

$$\pi_{E_i}(a|s) = \frac{e^{\tau^{-1} Q_{E_i}(s,a)}}{\sum_{a' \in \mathcal{A}_{E_i}} e^{\tau^{-1} Q_{E_i}(s,a')}} \quad \mathcal{L}^i_{policy}(\theta) = \sum_{a \in \mathcal{A}_{E_i}} \pi_{E_i}(a|s) \log \pi_{\mathrm{AMN}}(a|s;\theta)$$



*Hinton et al. '14

Case Study: Multi-task Learning on Atari

blue: single actor-mimic policy

red: separate DQN policies

Parisotto et al. '16

Case Study: Multi-task Learning

Pros:

- Learn single policy for multiple tasks

Cons:

- still need to train on each game for same amount of time
- performance drops slightly from multi-task training (no transfer)

caveat: Atari games are likely not good for transfer

Memory for better Generalization in POMDPs

Control of Memory, Active Perception, and Action in Minecraft

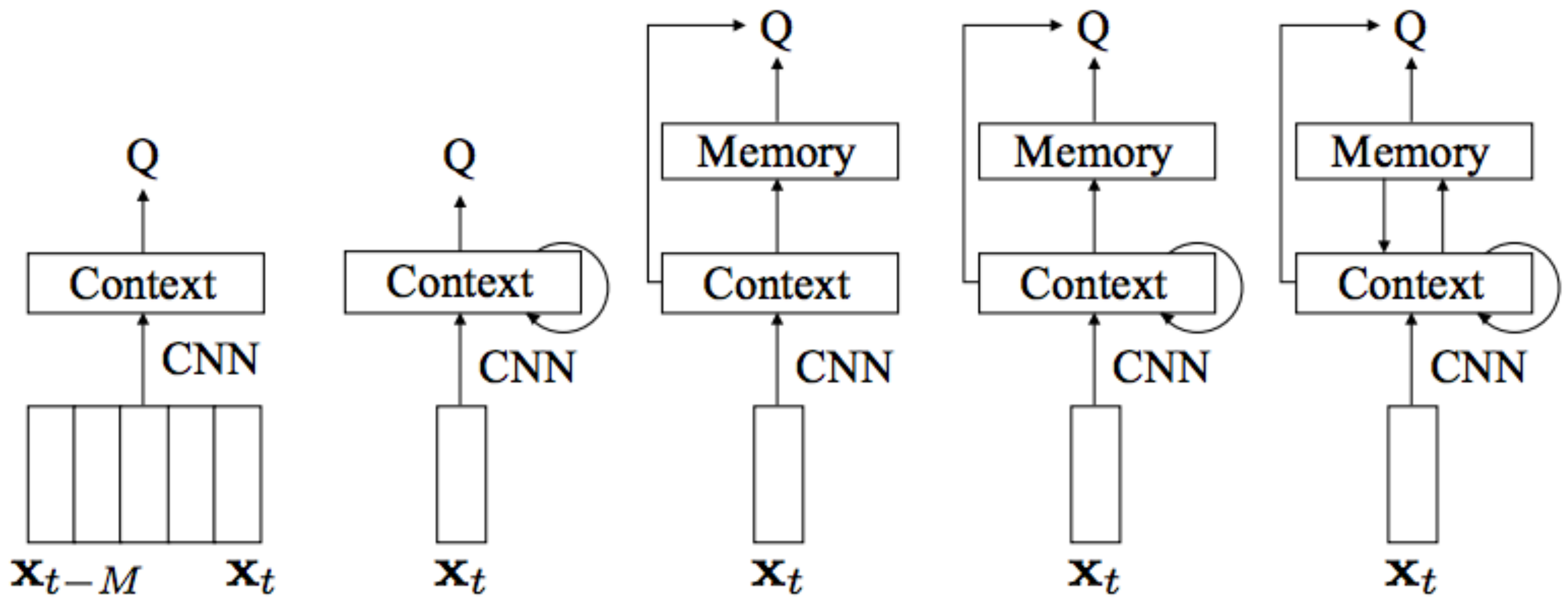
Junhyuk Oh JUNHYUK@UMICH.EDU
Valliappa Chockalingam VALLI@UMICH.EDU
Satinder Singh BAVEJA@UMICH.EDU
Honglak Lee HONGLAK@UMICH.EDU
Computer Science & Engineering, University of Michigan

feedforward networks may be forced to memorize training environments

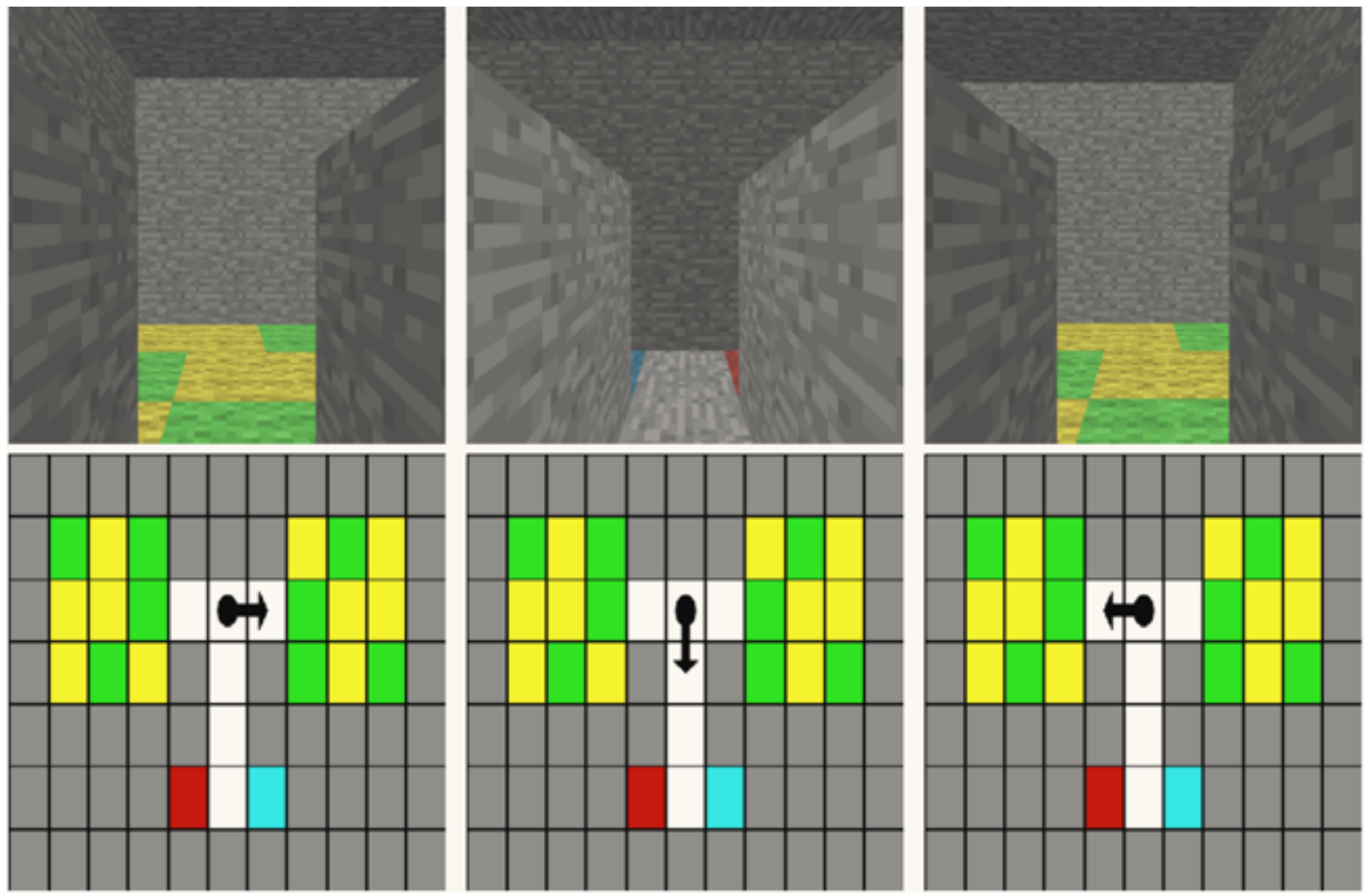
the wrong memory mechanism may also lead to memorization

Memory for better Generalization in POMDPs

Can improve generalization with appropriate memory mechanism



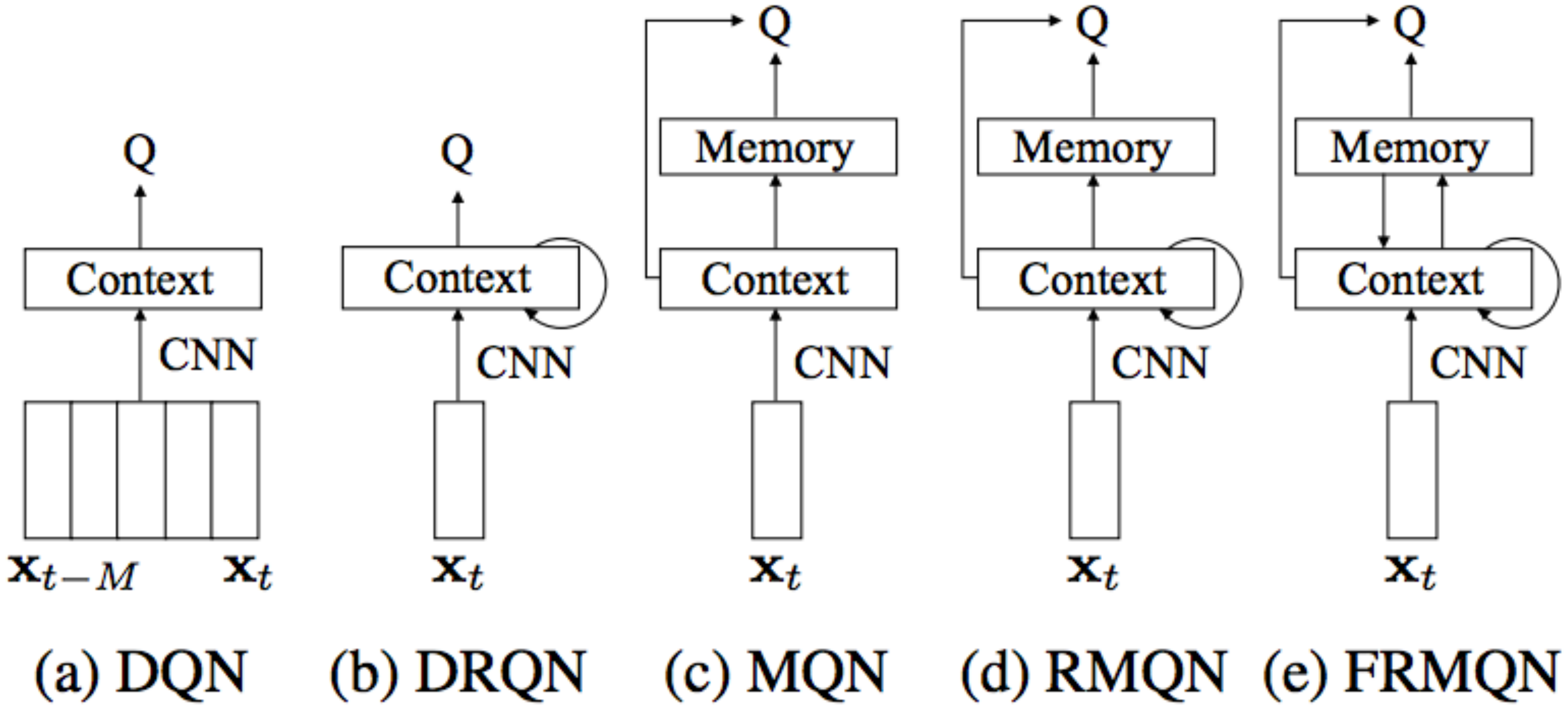
(a) DQN (b) DRQN (c) MQN (d) RMQN (e) FRMQN



	TRAIN	UNSEEN
DQN	62.9% (±3.4%)	60.1% (±2.8%)
DRQN	49.7% (±0.2%)	49.2% (±0.2%)
MQN	99.0% (±0.2%)	69.3% (±1.5%)
RMQN	82.5% (±2.5%)	62.3% (±1.5%)
FRMQN	**100.0%** (±0.0%)	**91.8%** (±1.0%)

Memory for better Generalization in POMDPs

Can improve generalization with appropriate memory mechanism



Pros:

- easy to combine with other approaches

Cons:

- doesn't completely solve the problem

Outline: Achieving Transfer in RL

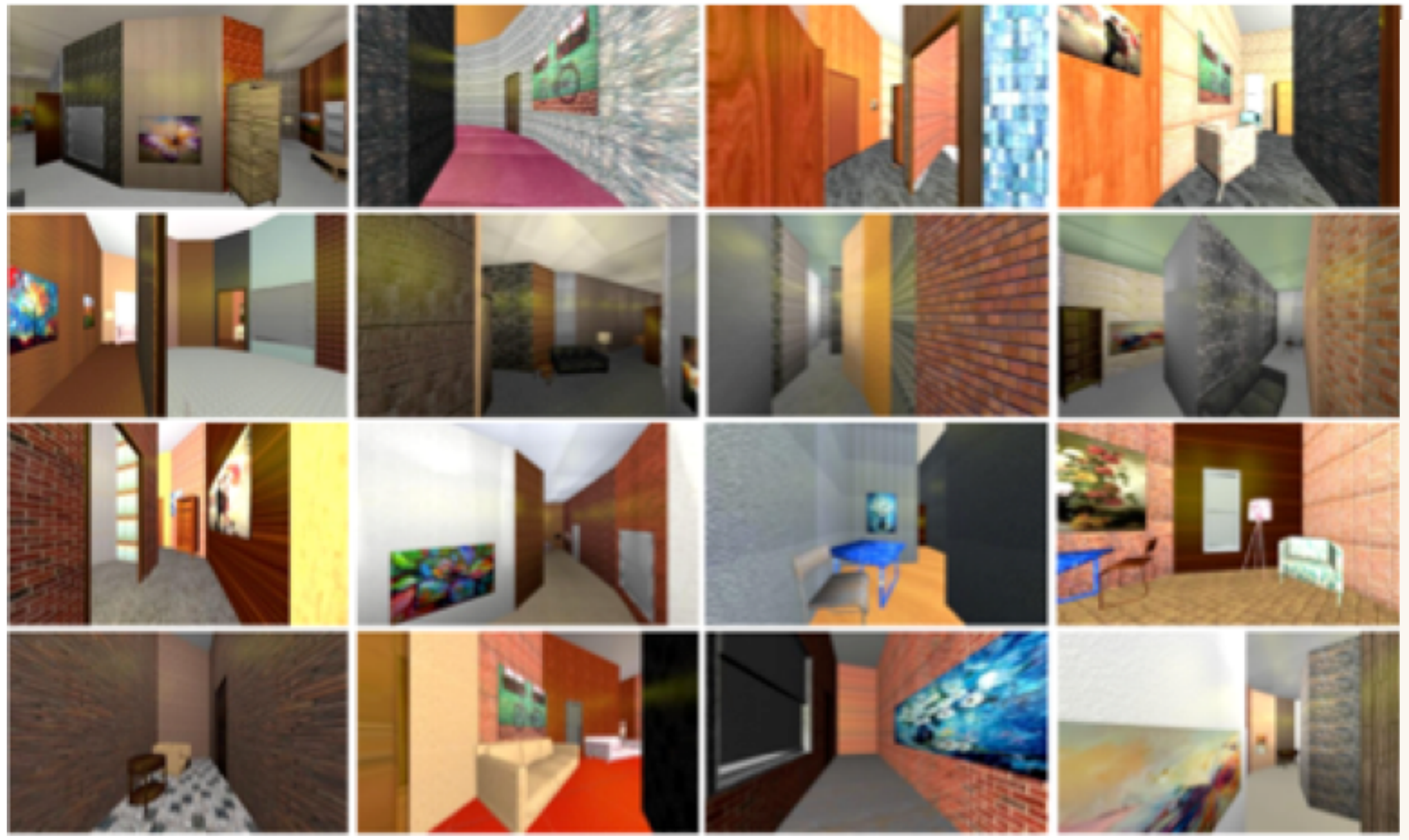
1. **Handling changes in reward**
   a. task represented in the observation
2. **Handling changes in environment**
   a. diversity for sim-to-real transfer
3. **Reusing representations**
   a. progressive networks
   b. PathNet
   c. modular networks
4. **Meta-learning**
   a. Learning to learn quickly
   b. Few-shot adaptation

Transfer across Environments: Diversity

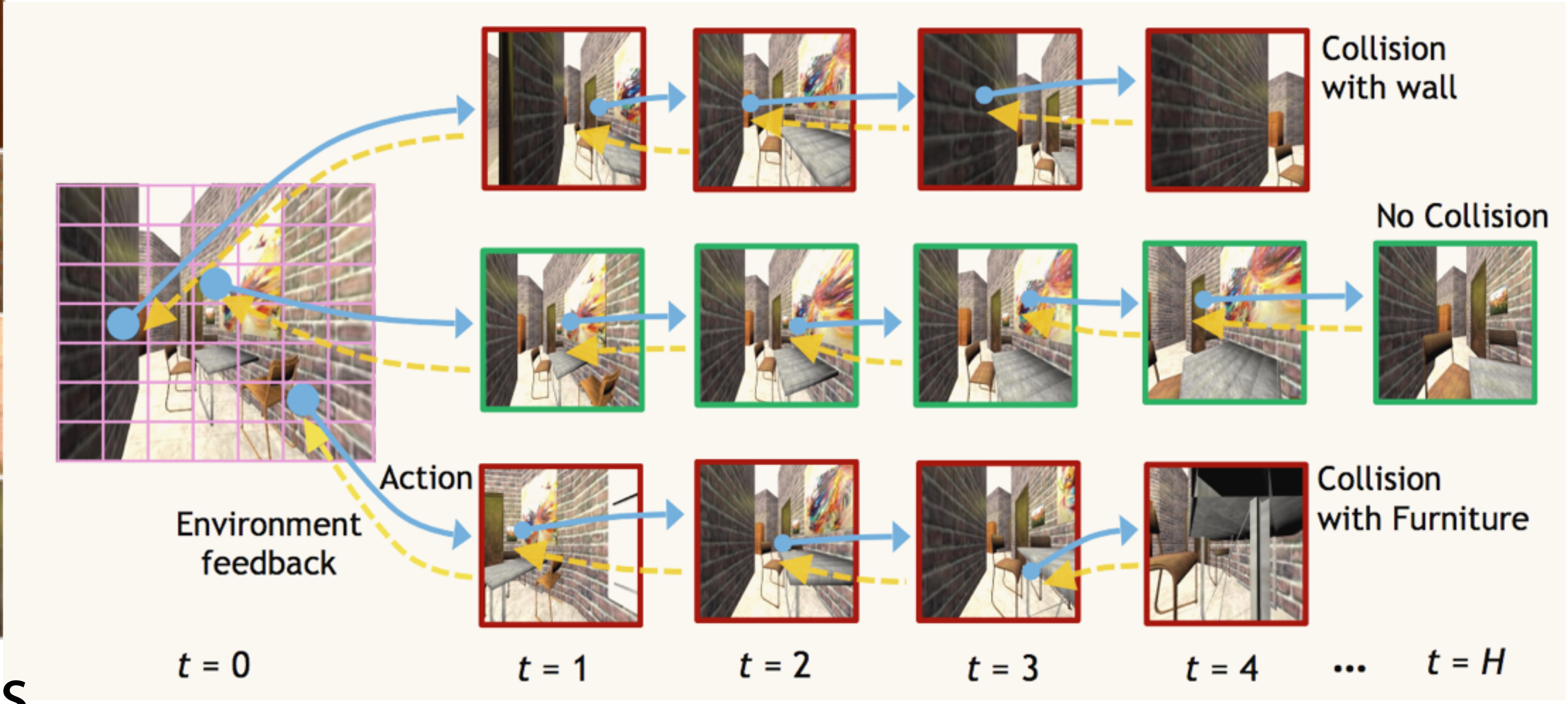
Case-Study: Simulation-to-real world transfer

(CAD)^{2}RL: Real Single-Image Flight without a Single Real Image

Fereshteh Sadeghi[1] and Sergey Levine[2]

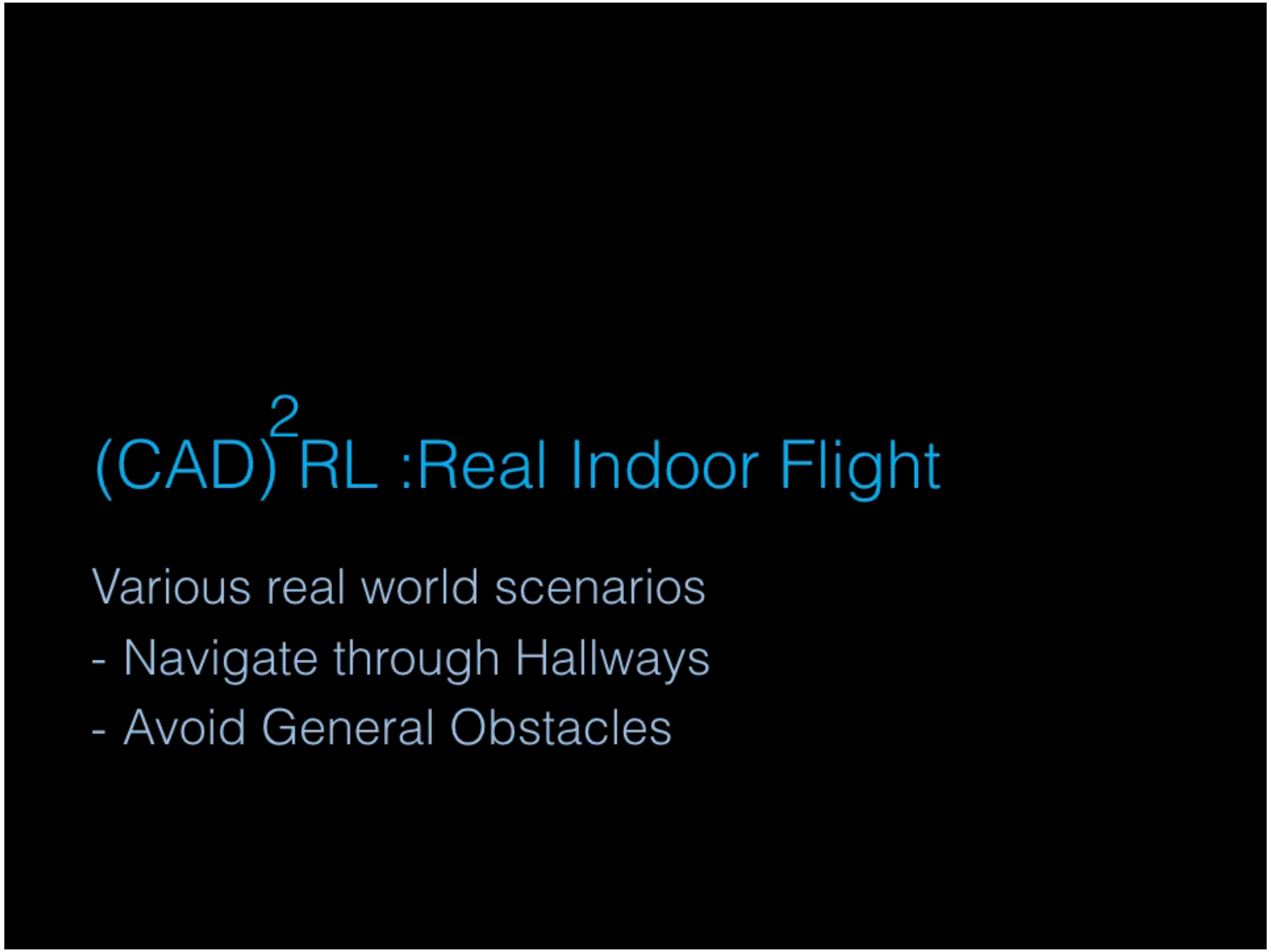


vary textures & hallway geometries



Case-Study: Simulation-to-real world transfer

After training entirely in simulation:

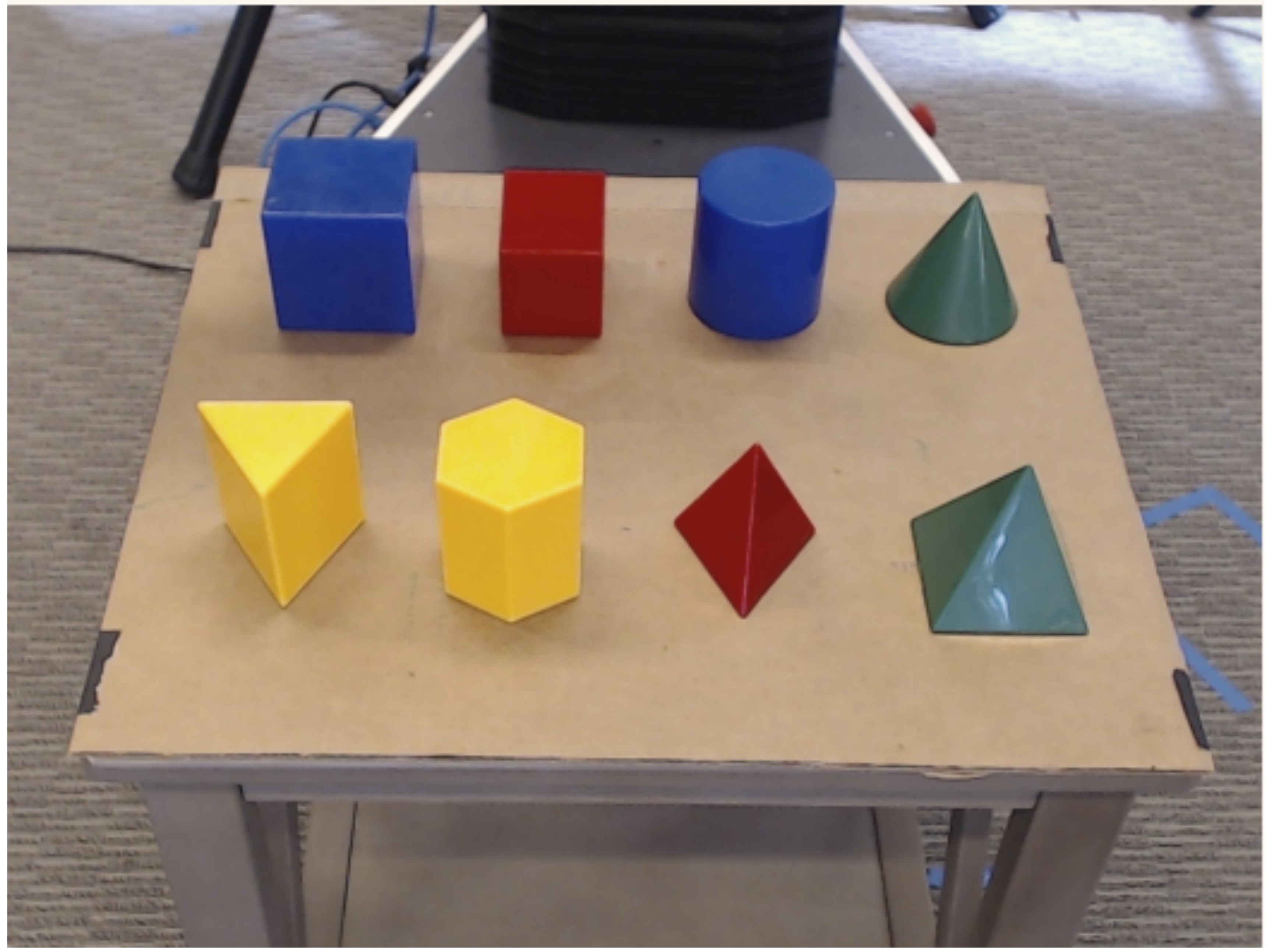


Case-Study: Simulation-to-real world transfer

Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World

Josh Tobin[1], Rachel Fong[2], Alex Ray[2], Jonas Schneider[2], Wojciech Zaremba[2], Pieter Abbeel[3]

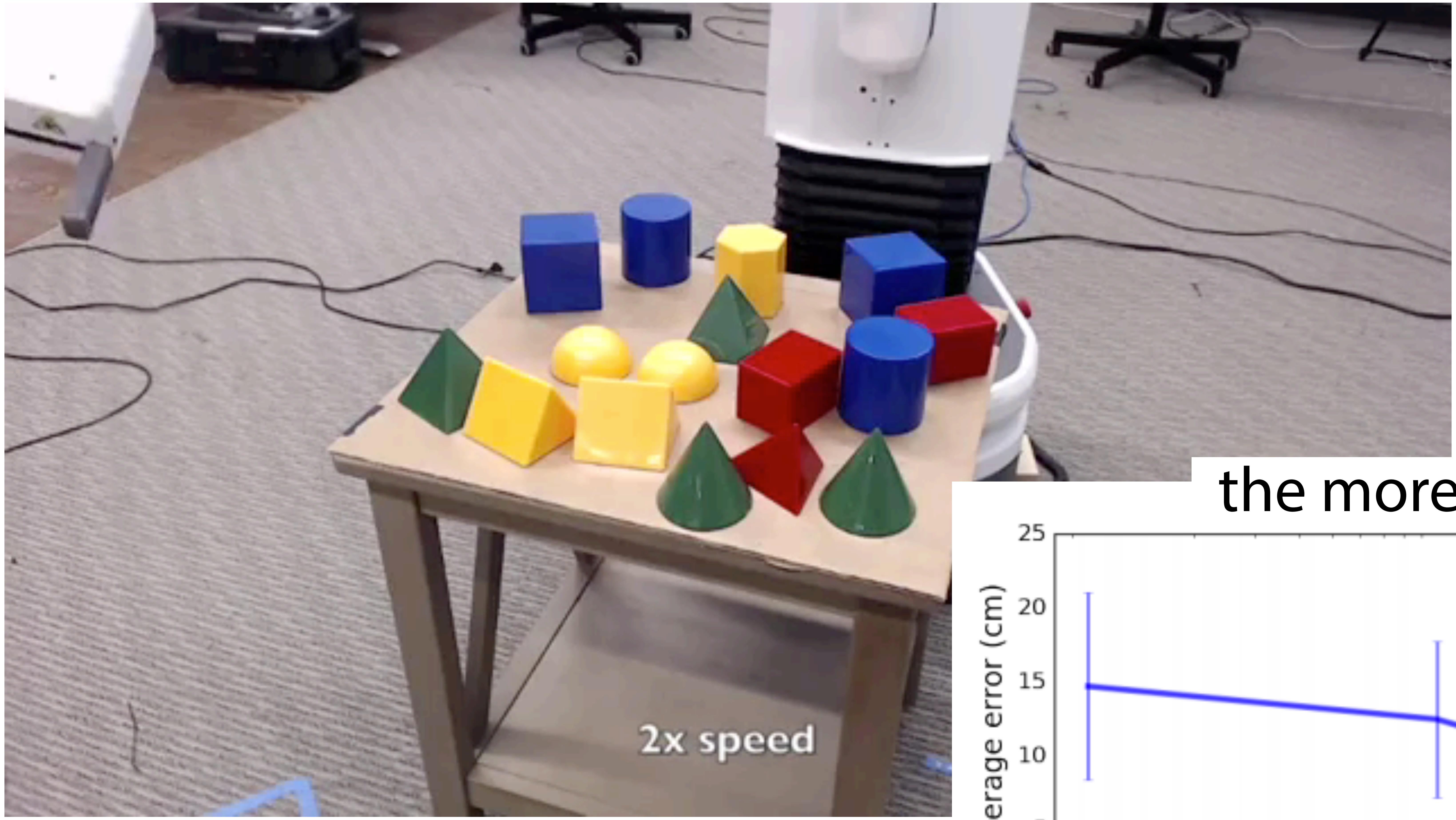
object localization



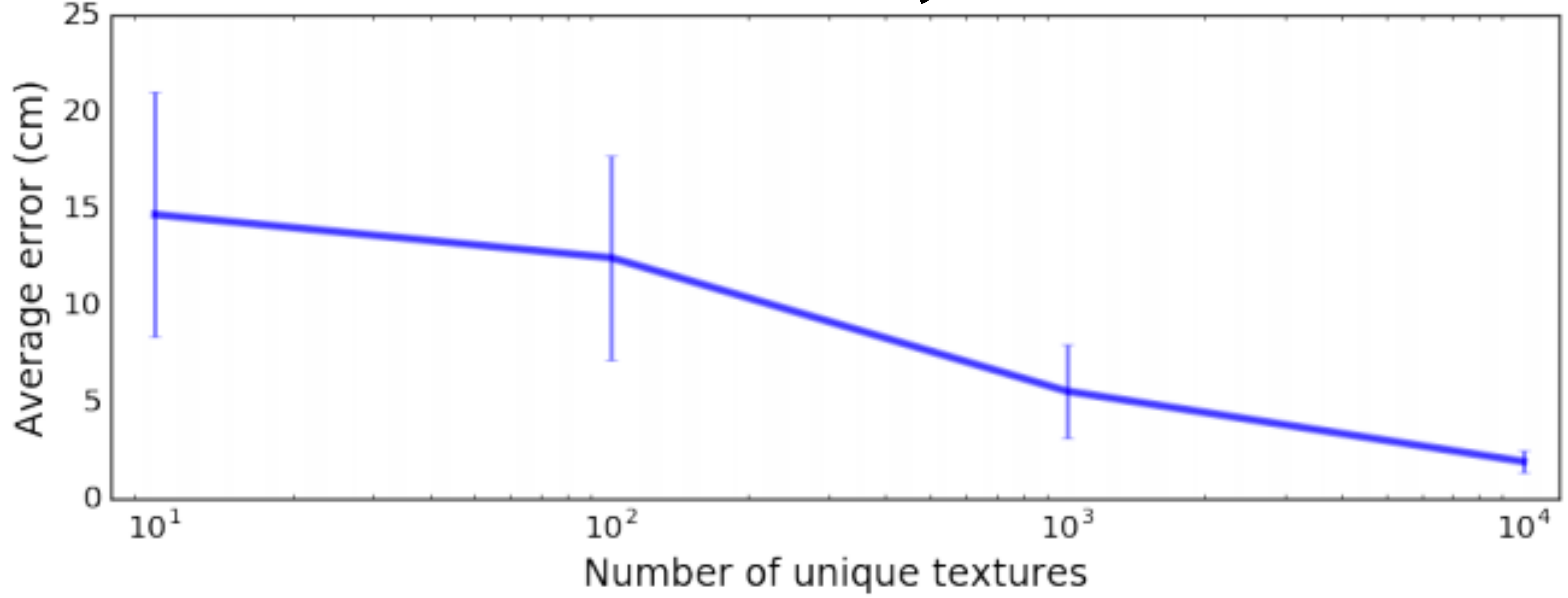
vary object colors, lighting, camera angle

Case-Study: Simulation-to-real world transfer

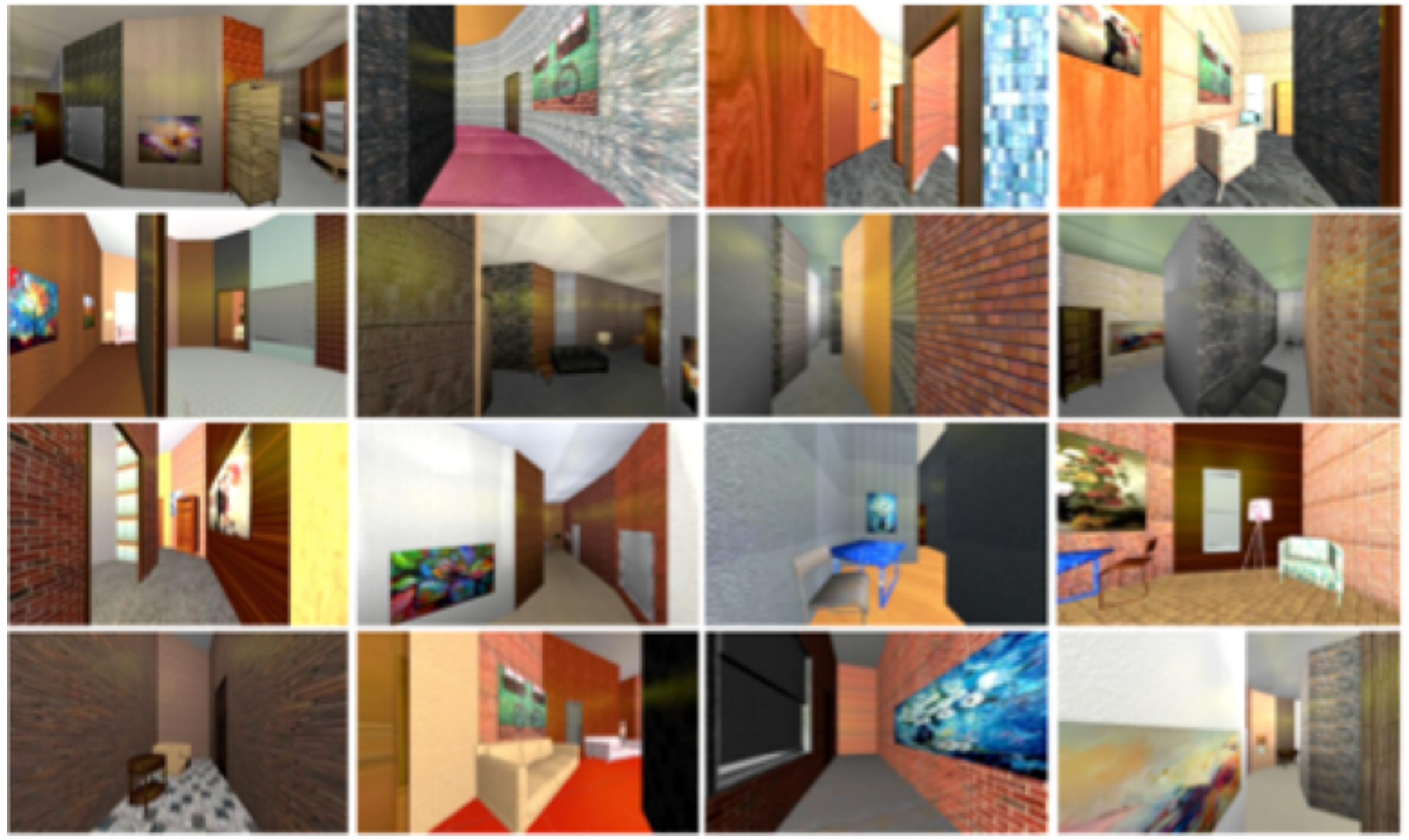
object localization

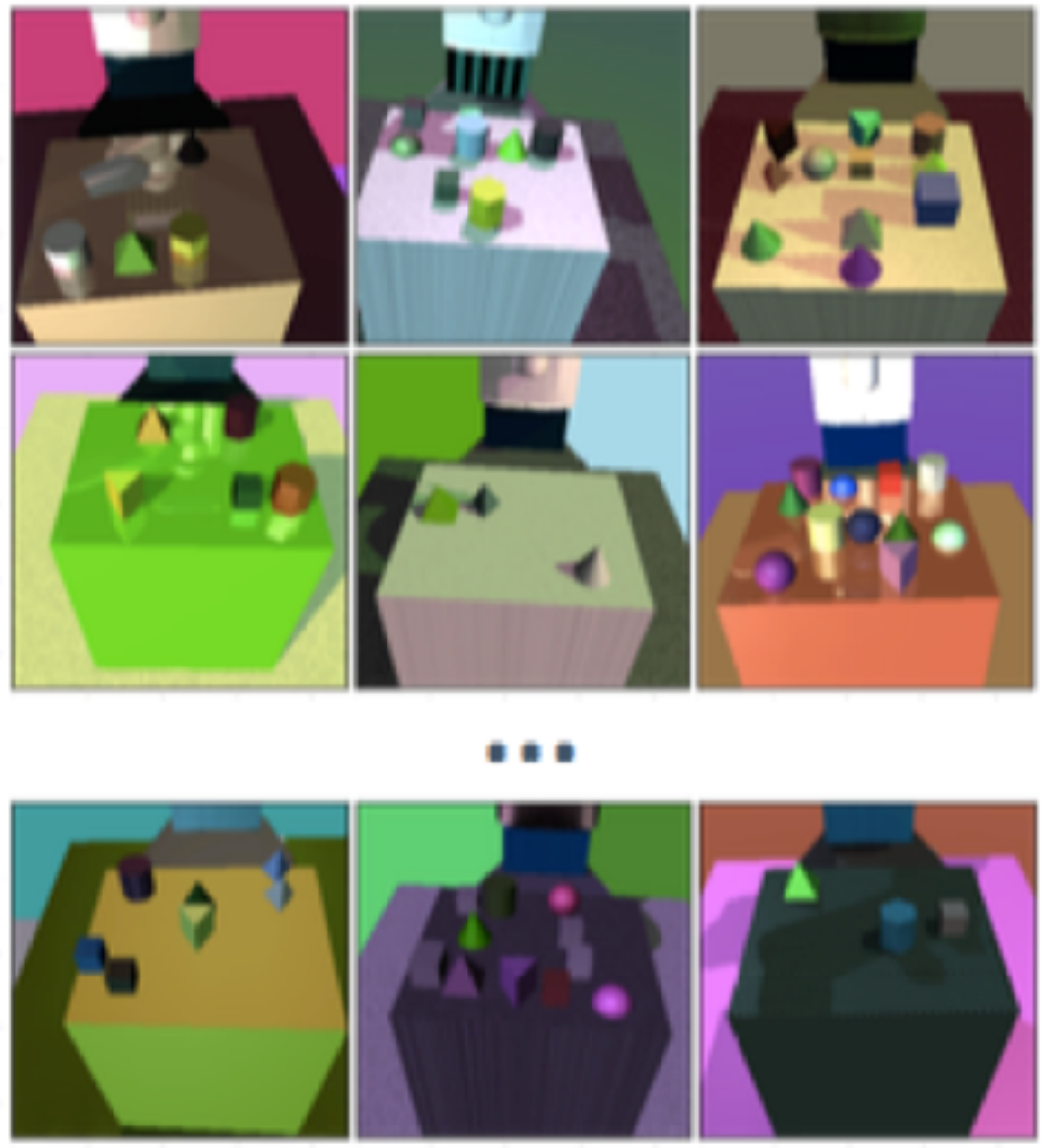


the more diversity, the better!



Transfer across Environments: Diversity





Pros:

- works surprisingly well, 0-shot generalization to real world

Cons:

- Content creation requires large engineering effort
- No use of target domain data (not even unsupervised data)
- Only demonstrated for shift in observation space (not yet for dynamics/reward)

Outline: Achieving Transfer in RL

1. **Handling changes in reward**
    a. task represented in the observation
2. **Handling changes in environment**
    a. diversity for sim-to-real transfer
3. **Reusing representations**
    a. progressive networks
    b. PathNet
    c. modular networks
4. **Meta-learning**
    a. Learning to learn quickly
    b. Few-shot adaptation

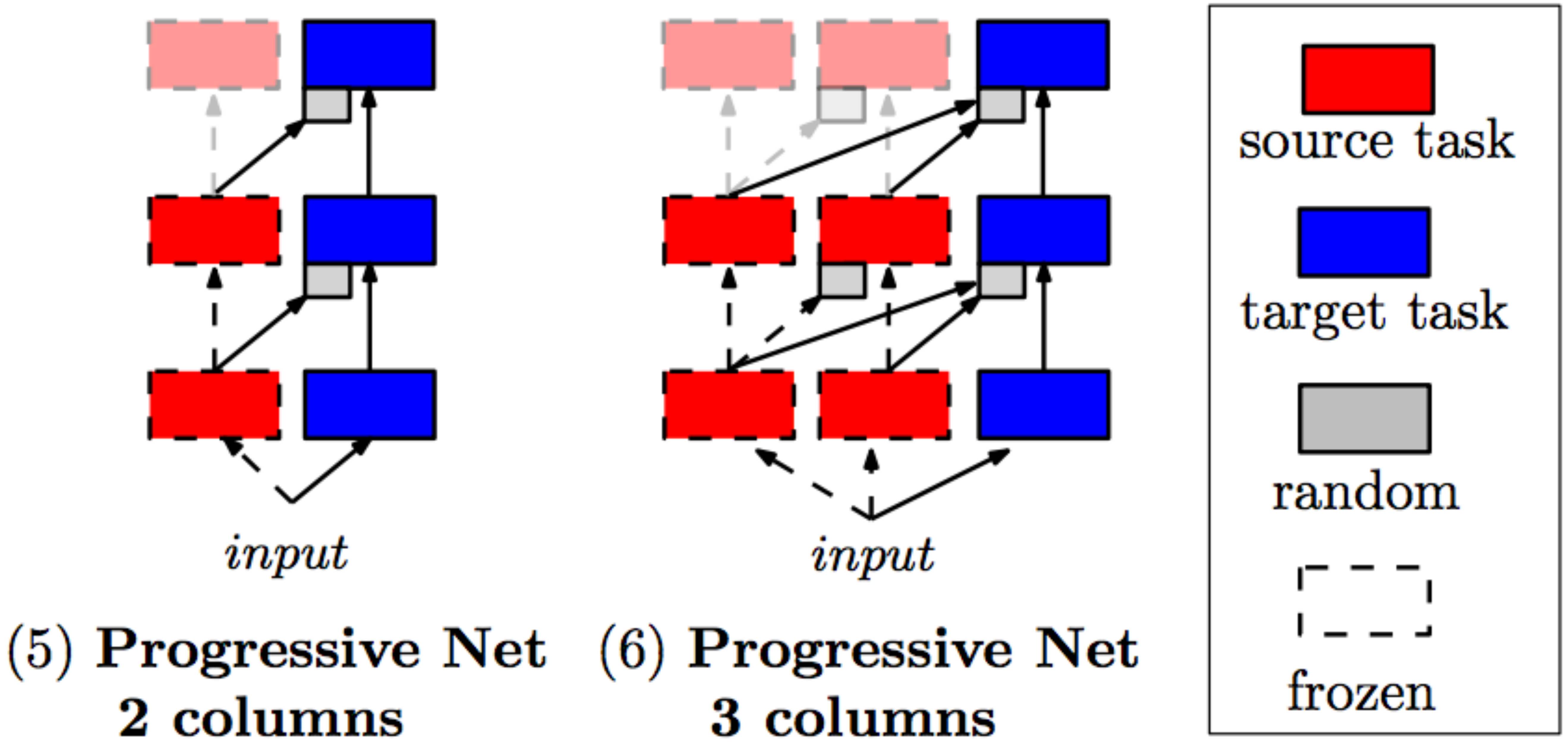
Break

Outline: Achieving Transfer in RL

1. **Handling changes in reward**
   a. task represented in the observation
2. **Handling changes in environment**
   a. diversity for sim-to-real transfer
3. **Reusing representations**
   a. progressive networks
   b. PathNet
   c. modular networks
4. **Meta-learning**
   a. Learning to learn quickly
   b. Few-shot adaptation

Reusing Representations: Progressive Networks

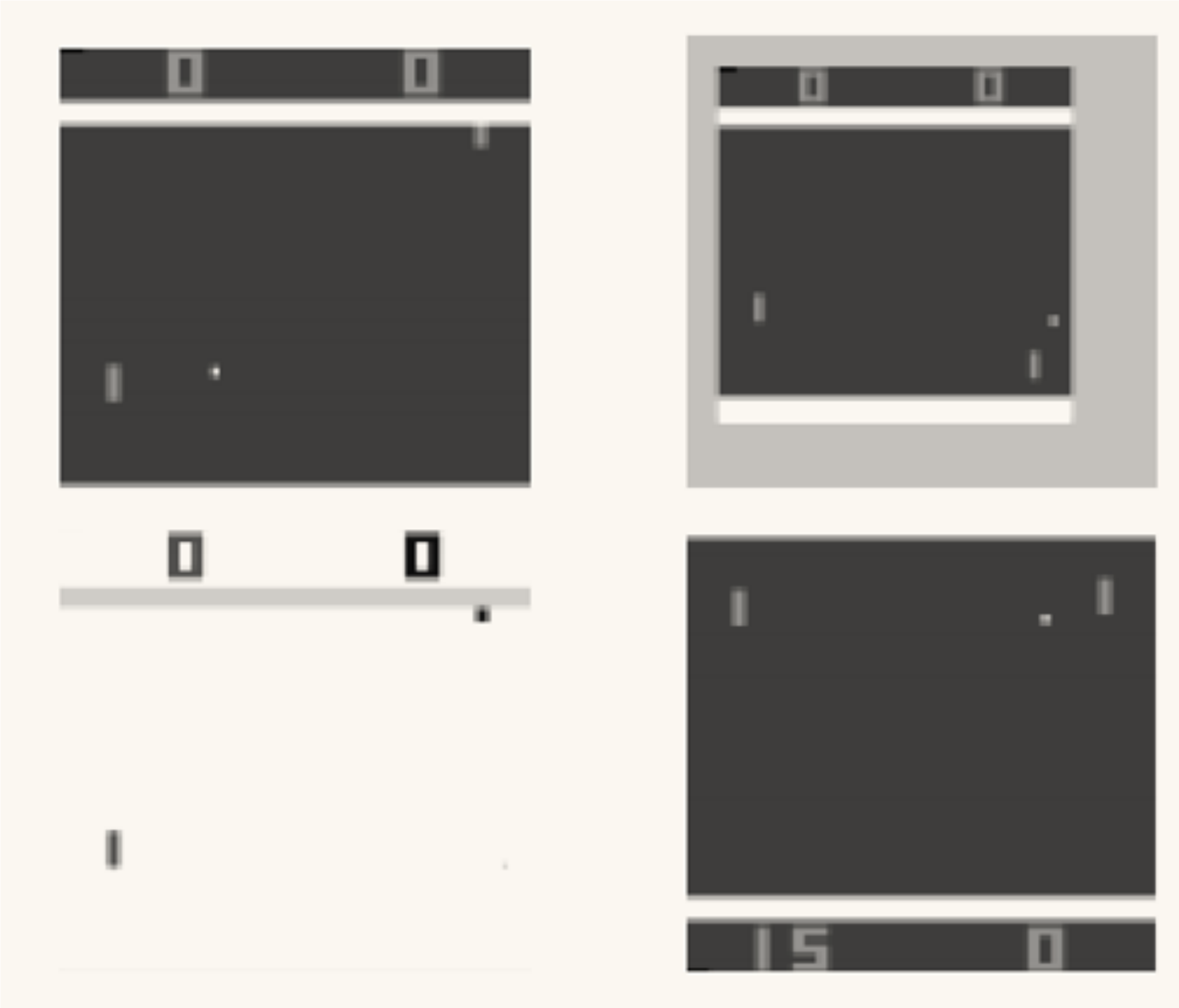
1. Train on new domain
2. Freeze weights on that domain
3. Reuse frozen representation on that domain when training on new domain
4. Repeat

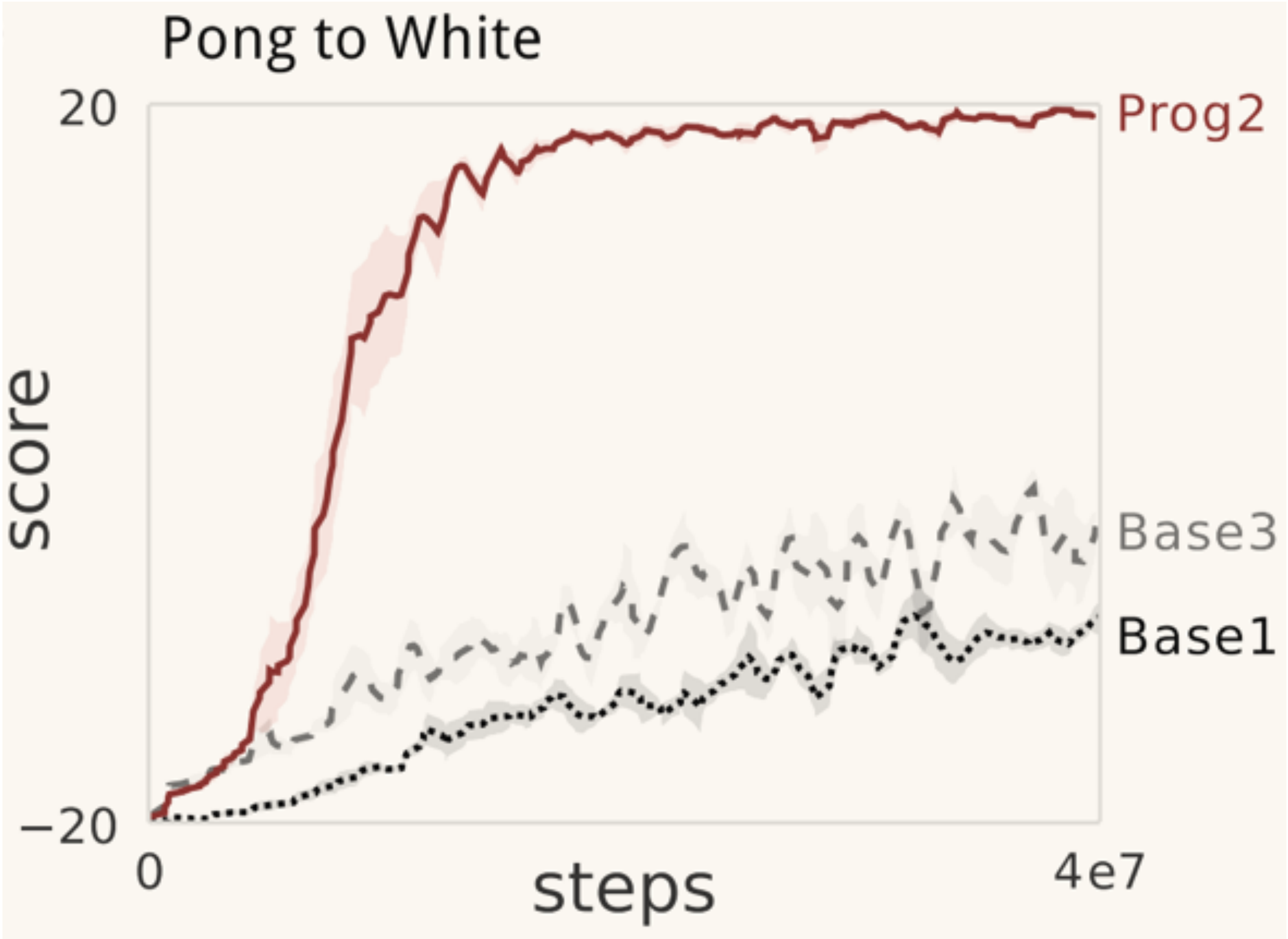


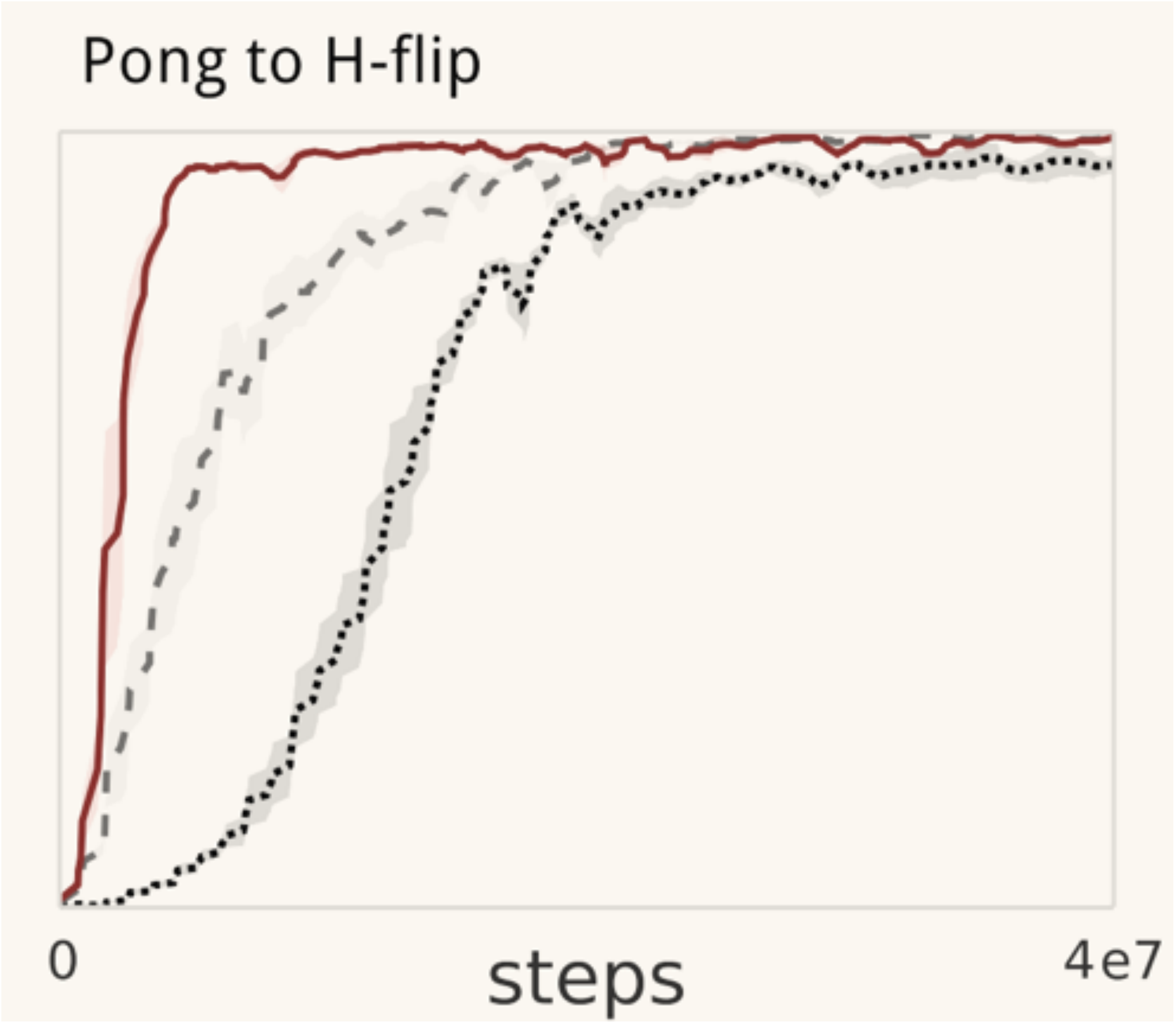
Rusu et al. '16

Reusing Representations: Progressive Networks

Variations on Pong

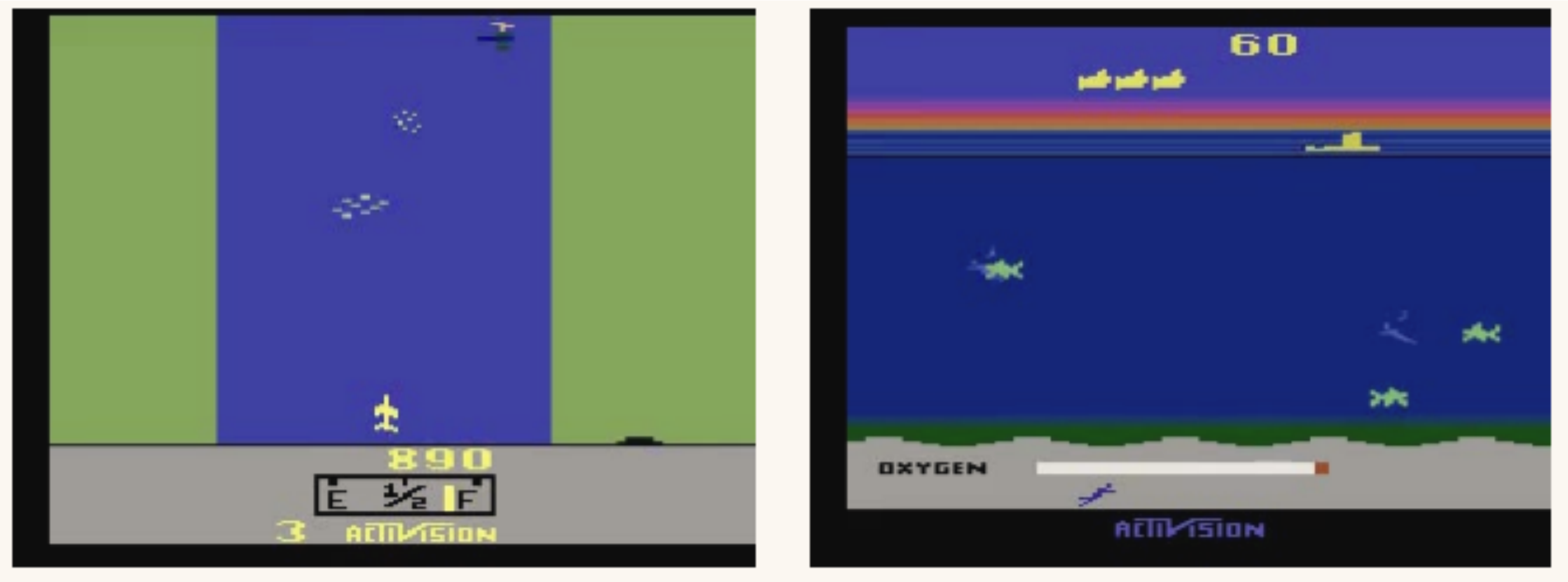


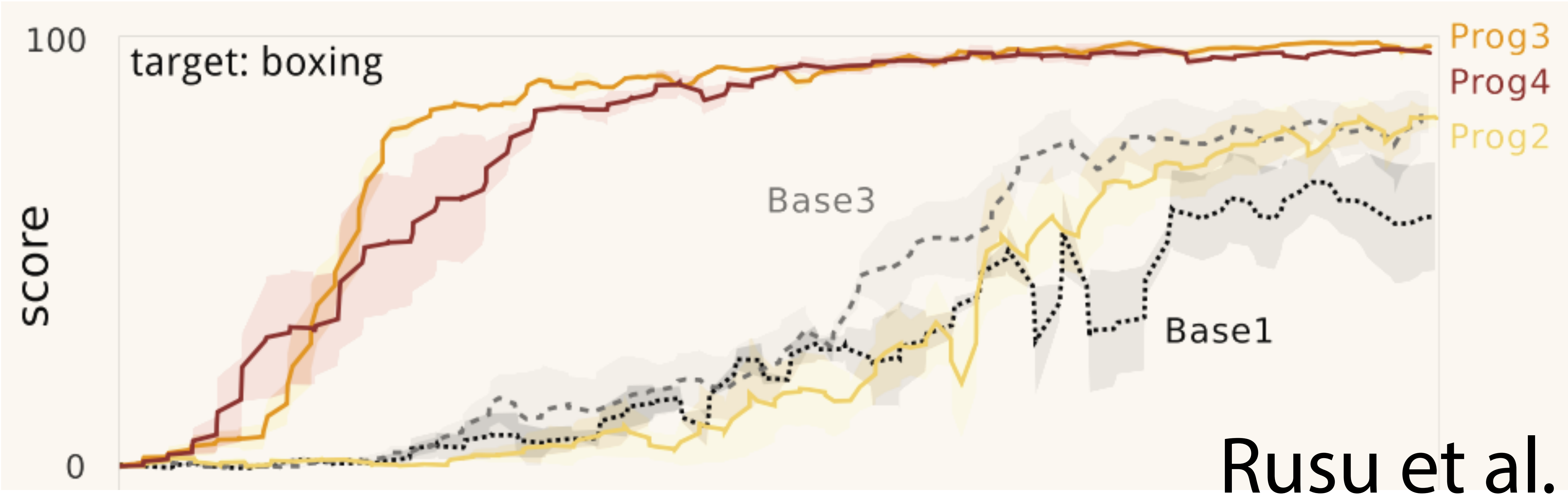




compared to scratch & fine-tuning

transfer from
pong, river raid, seaquest



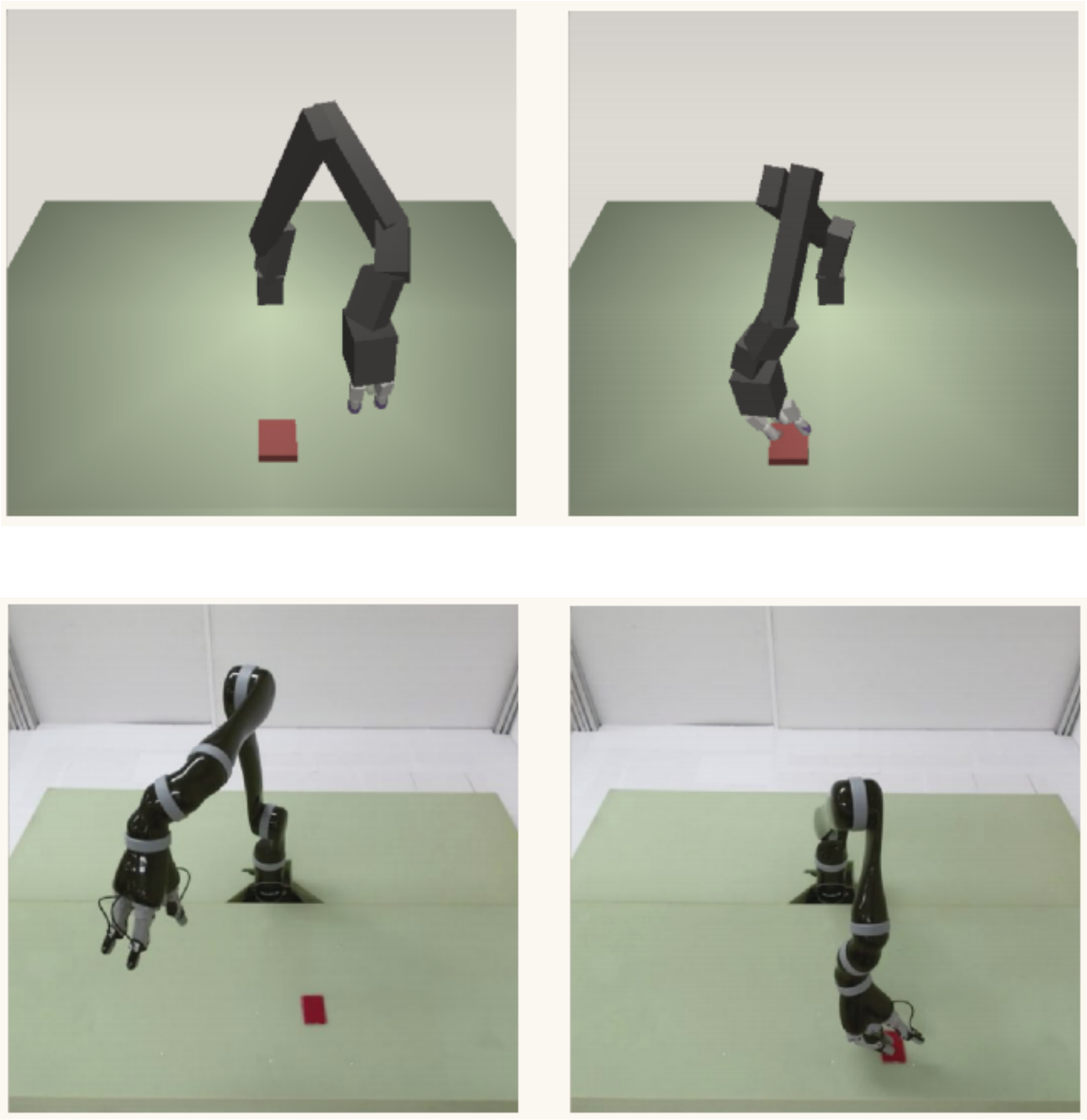


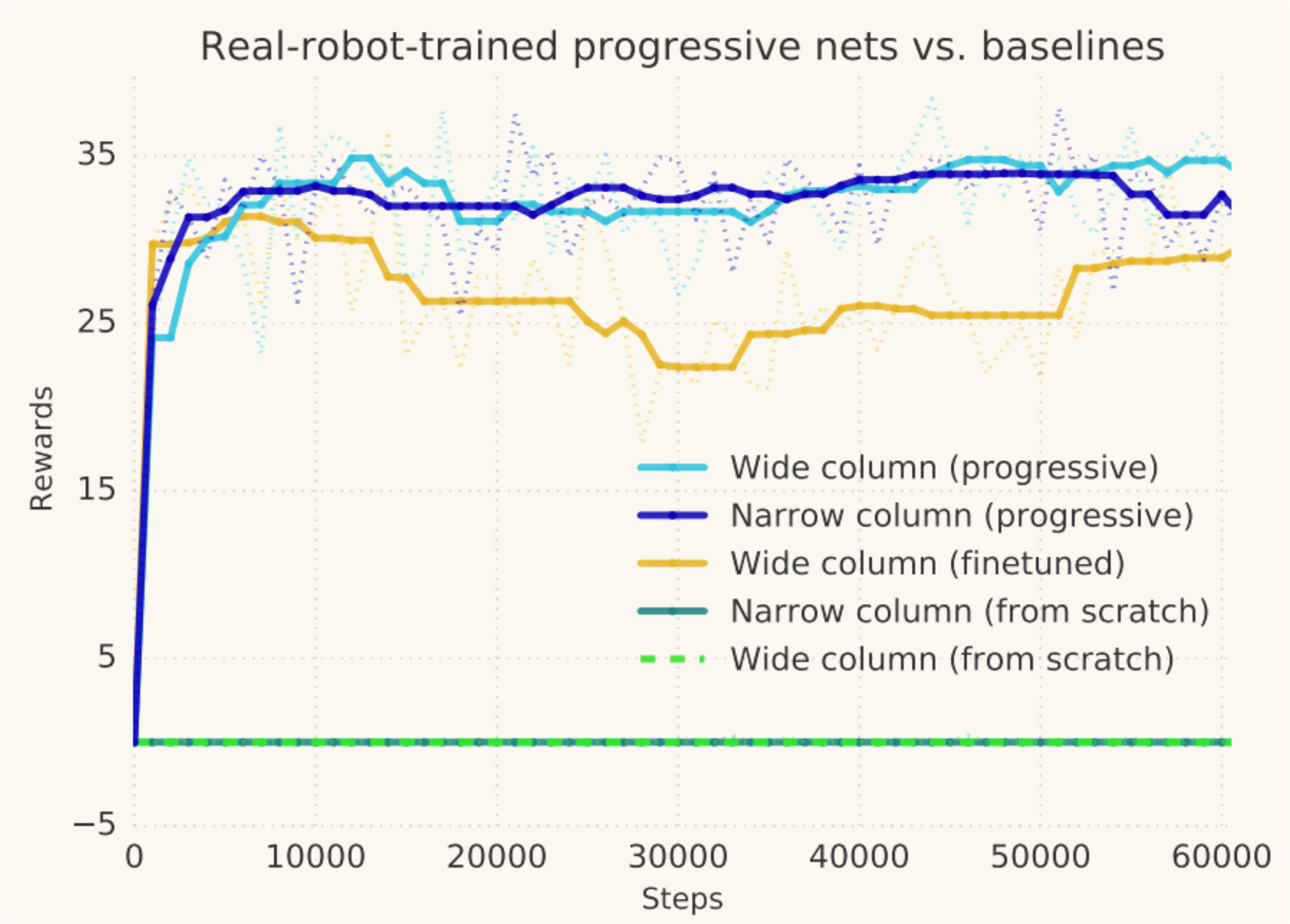
Rusu et al. '16

Reusing Representations: Progressive Networks

Simulation (A3C) to Real World (A2C)

target reaching task





Rusu et al. '16

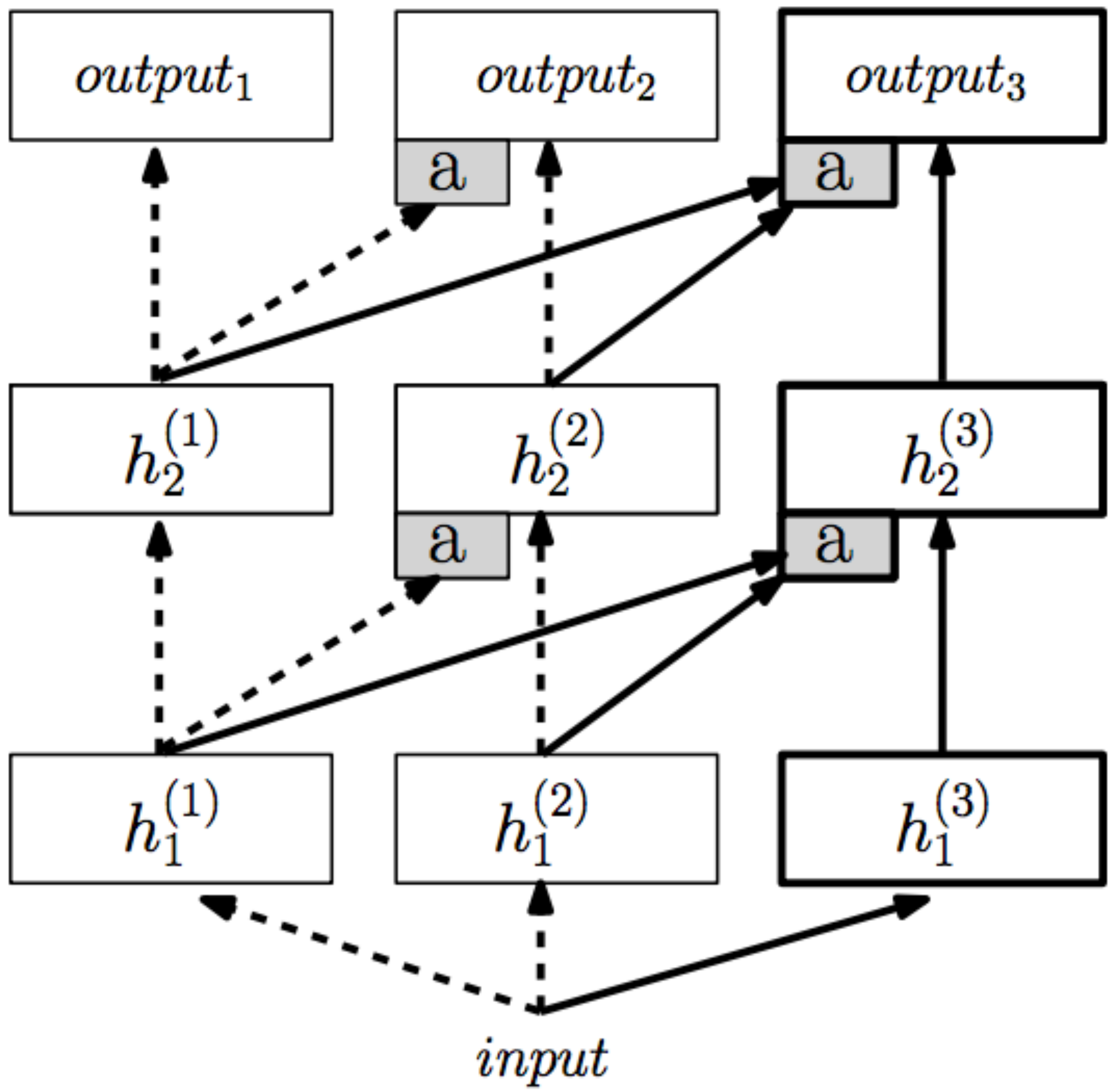
Reusing Representations: Progressive Networks

Pros:

- effective
- simple

Cons:

- dissatisfying: network grows larger and larger with each new domain (fixed topology)
- experience in new domains doesn't help old domains (unless you train on them again)
- learning still fairly slow



Reusing Representations: PathNet

Contrast to progressive nets: "evolve" relationships between columns

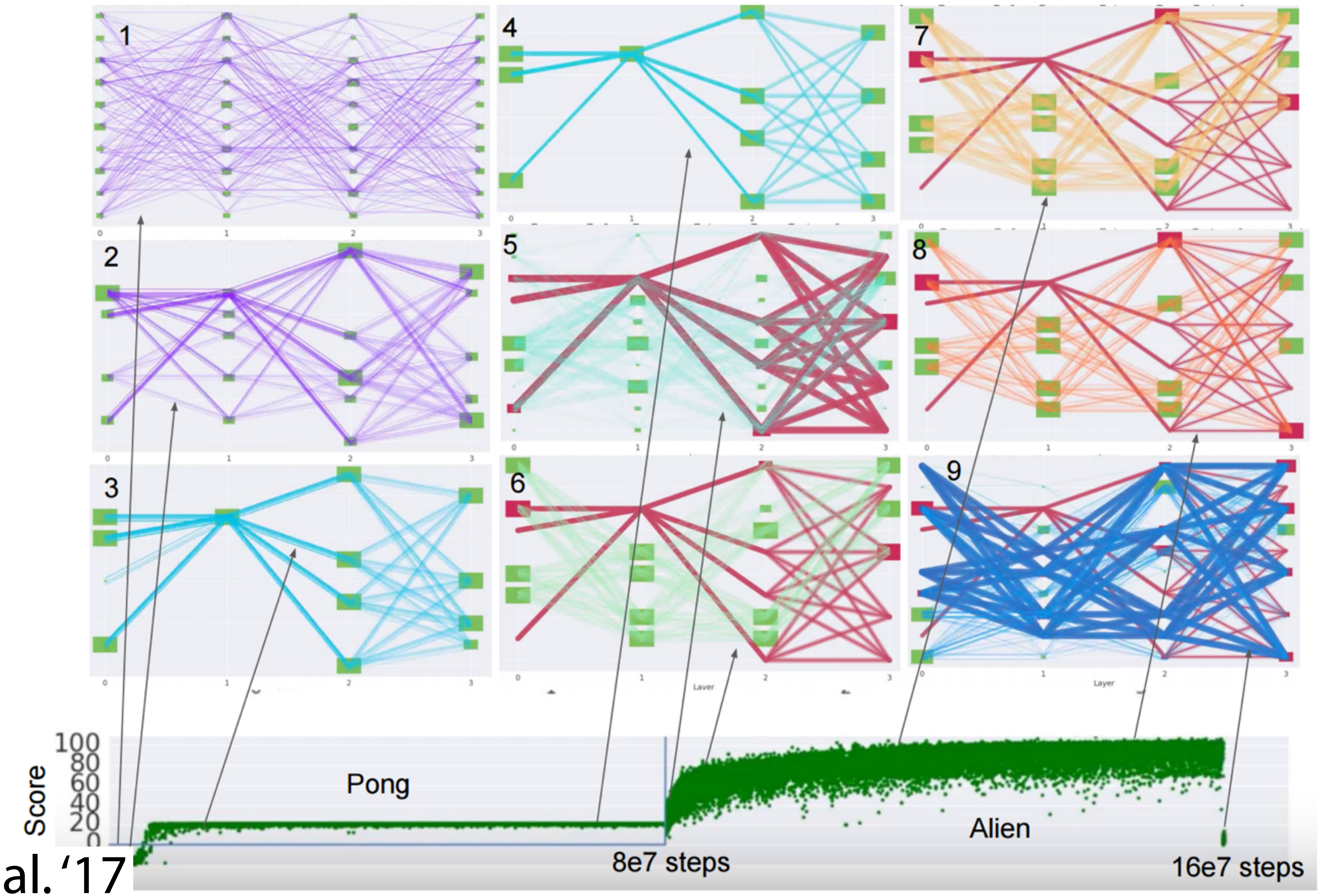
1. Pick new task
2. Evolve path through network and learn weights along path
3. Freeze weights along evolved path
4. Repeat, only changing unfrozen weights

genetic algorithm select paths (which parameters to train)

gradient-based RL (A3C) to learn parameter values

Fernando et al. '17

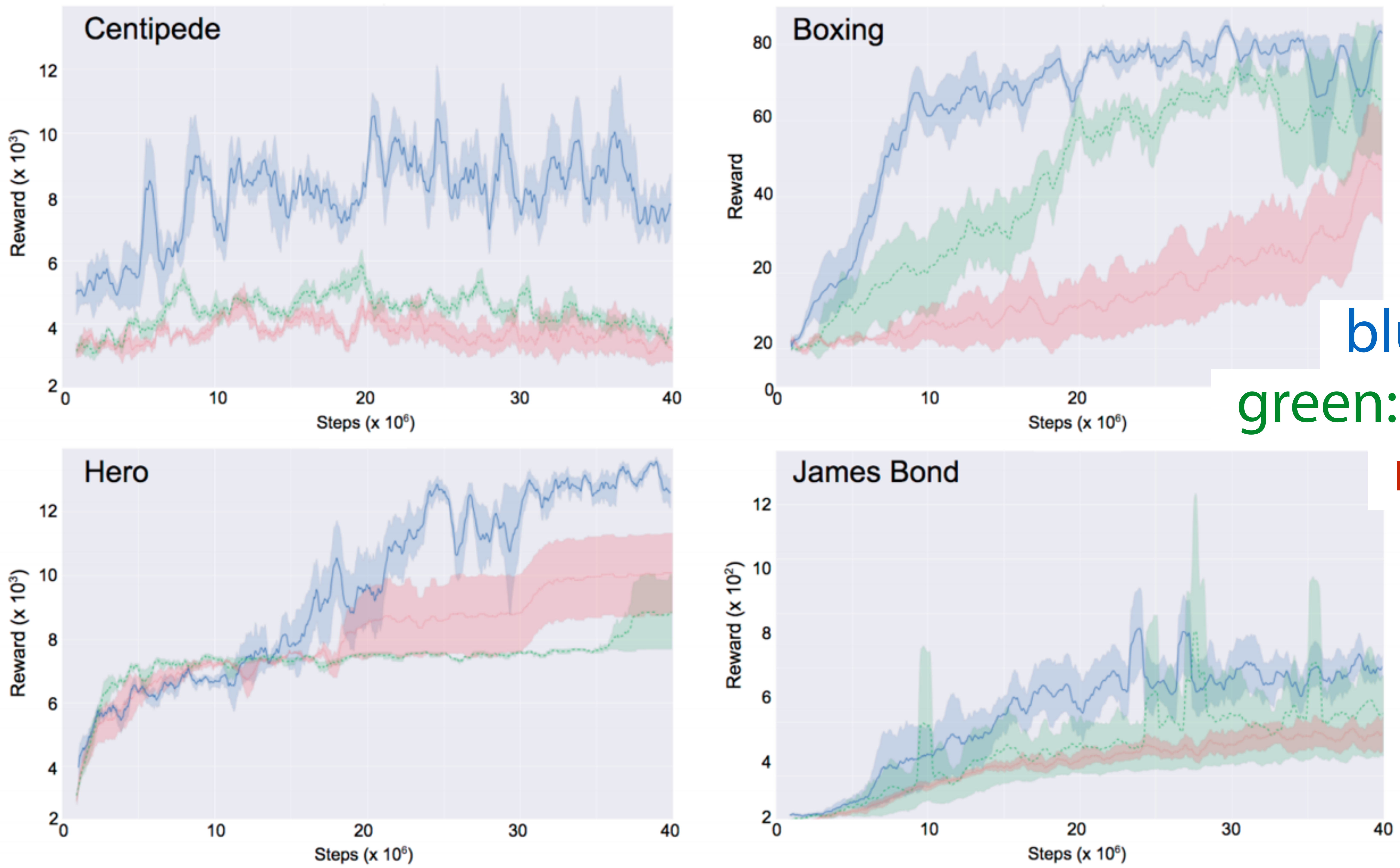
PathNet



Fernando et al. ‘17

Reusing Representations: PathNet

Transfer from River Raid to other games:



blue: PathNet
green: fine-tuning
red: scratch

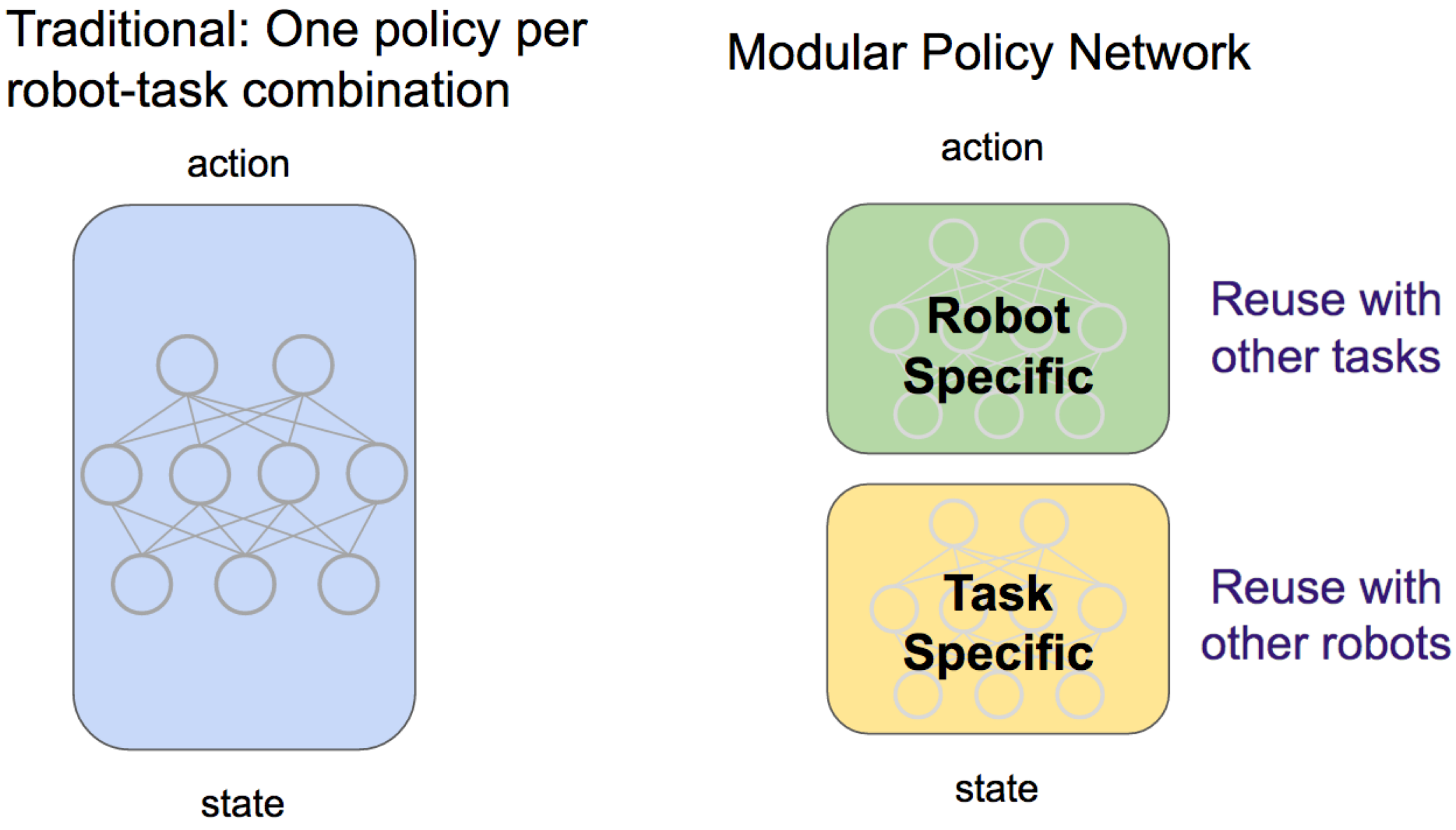
Reusing Representations: PathNet

Pros:
- network size is fixed (and straight-forward to grow if needed)
- effective combination of evolutionary & gradient-based learning

Cons:
- learning a new task is still fairly slow
- experience in new domains doesn't help old domains (unless you train on them again)

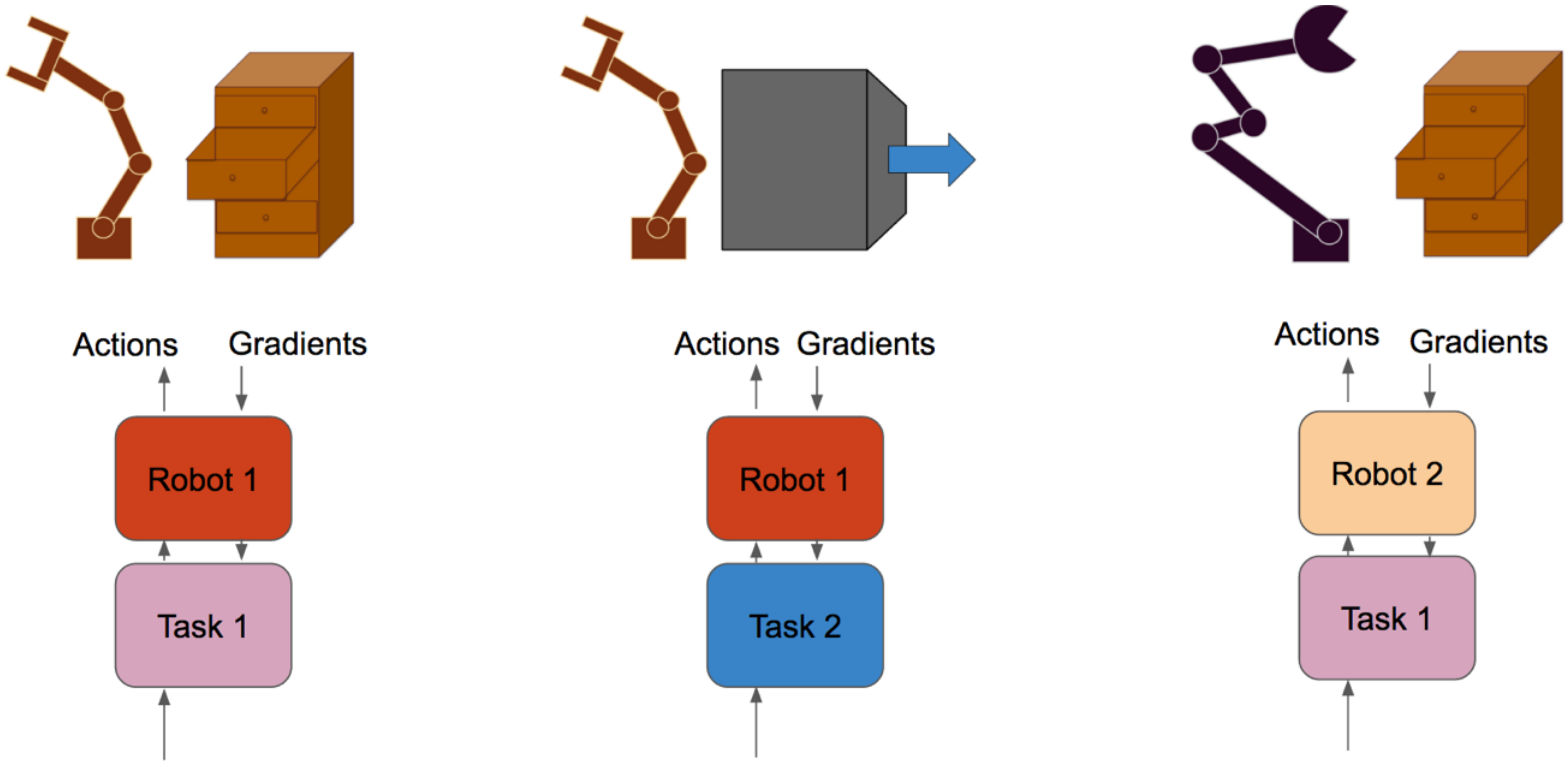
Reusing Representations: Modular Networks

Main Idea: transfer across robots/tasks by training modules for each



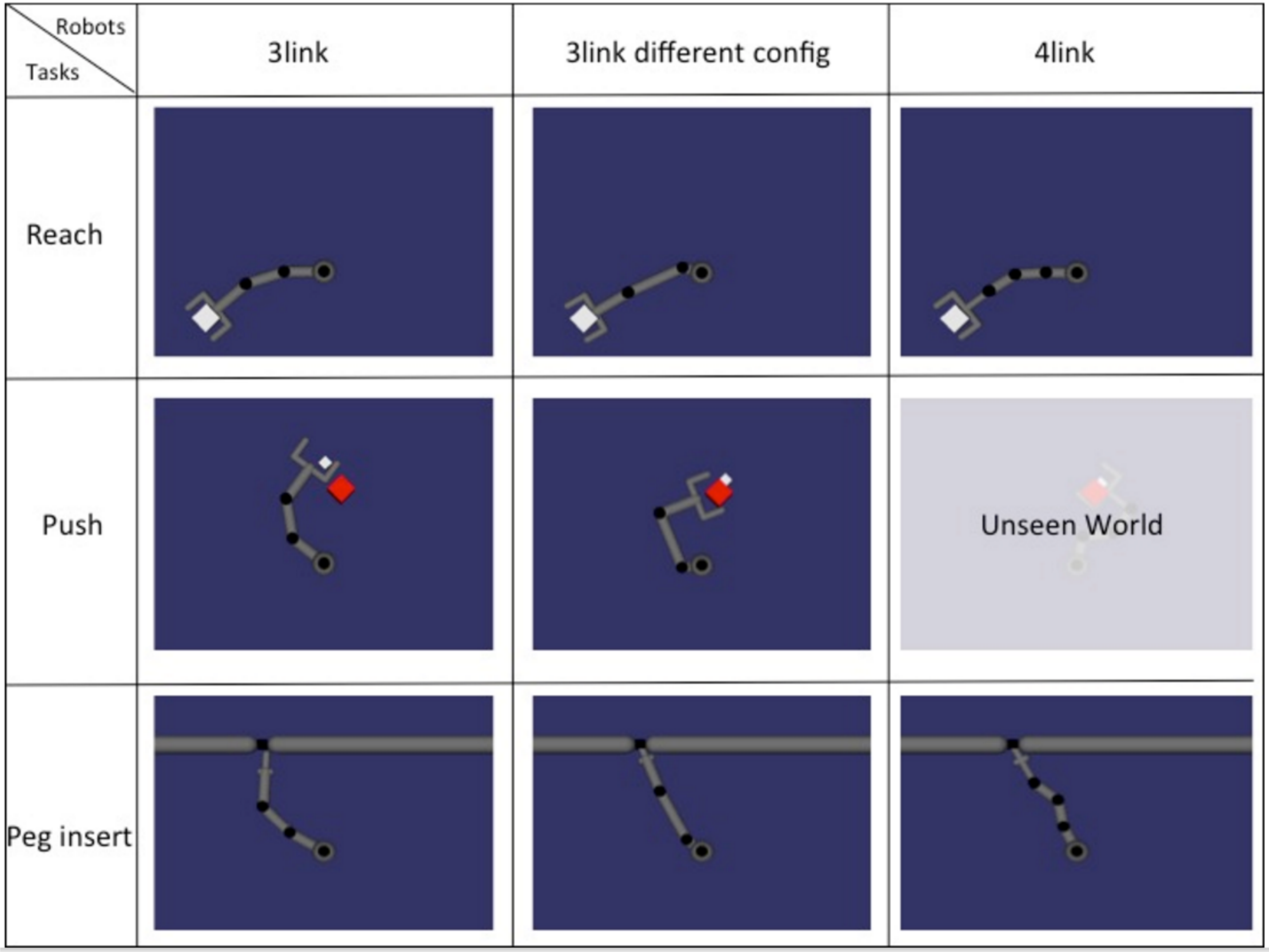
Reusing Representations: Modular Networks

Main Idea: transfer across robots/tasks by training modules for each

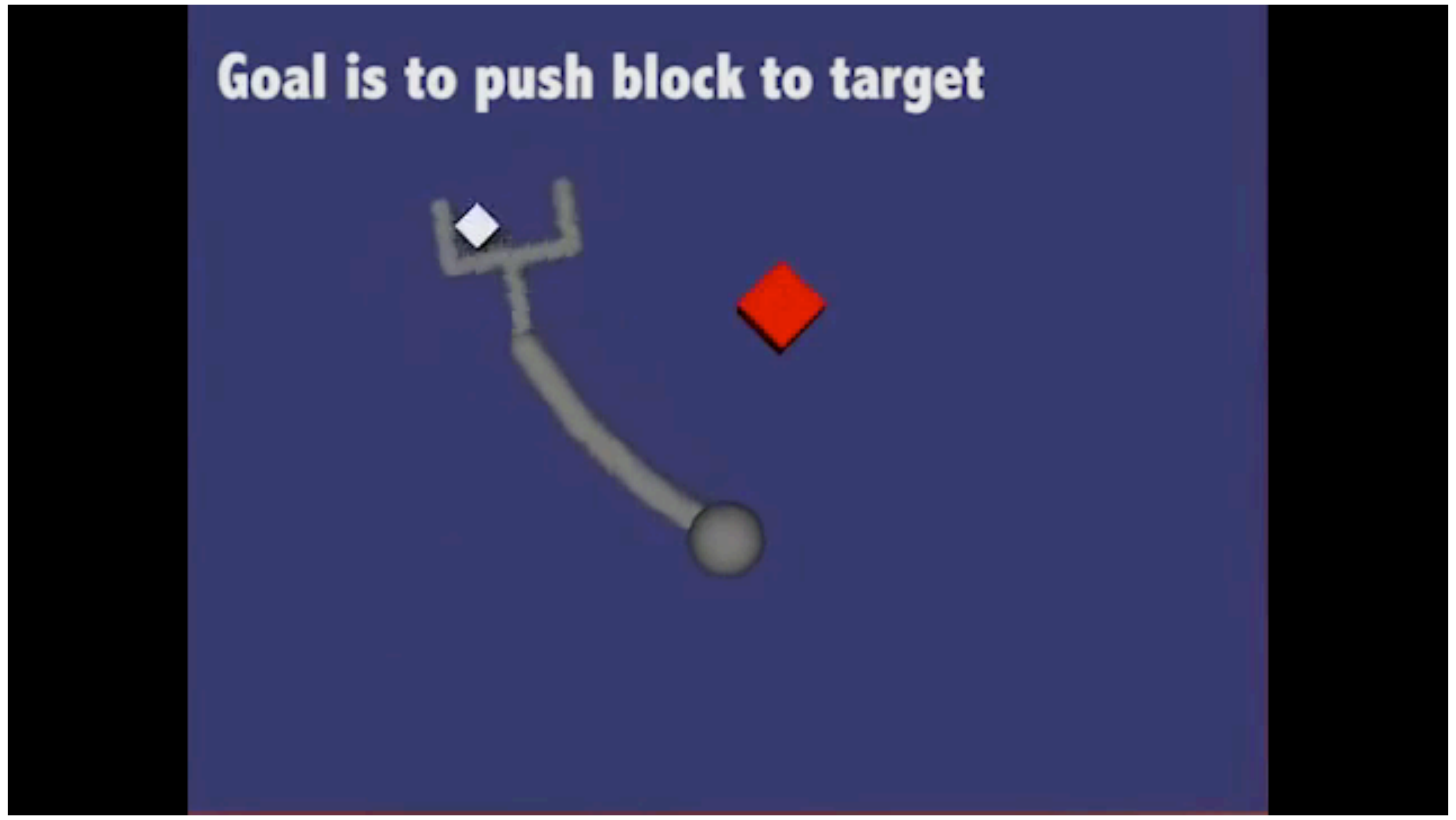


To prevent modules from overfitting to each other: **dropout, bottleneck**

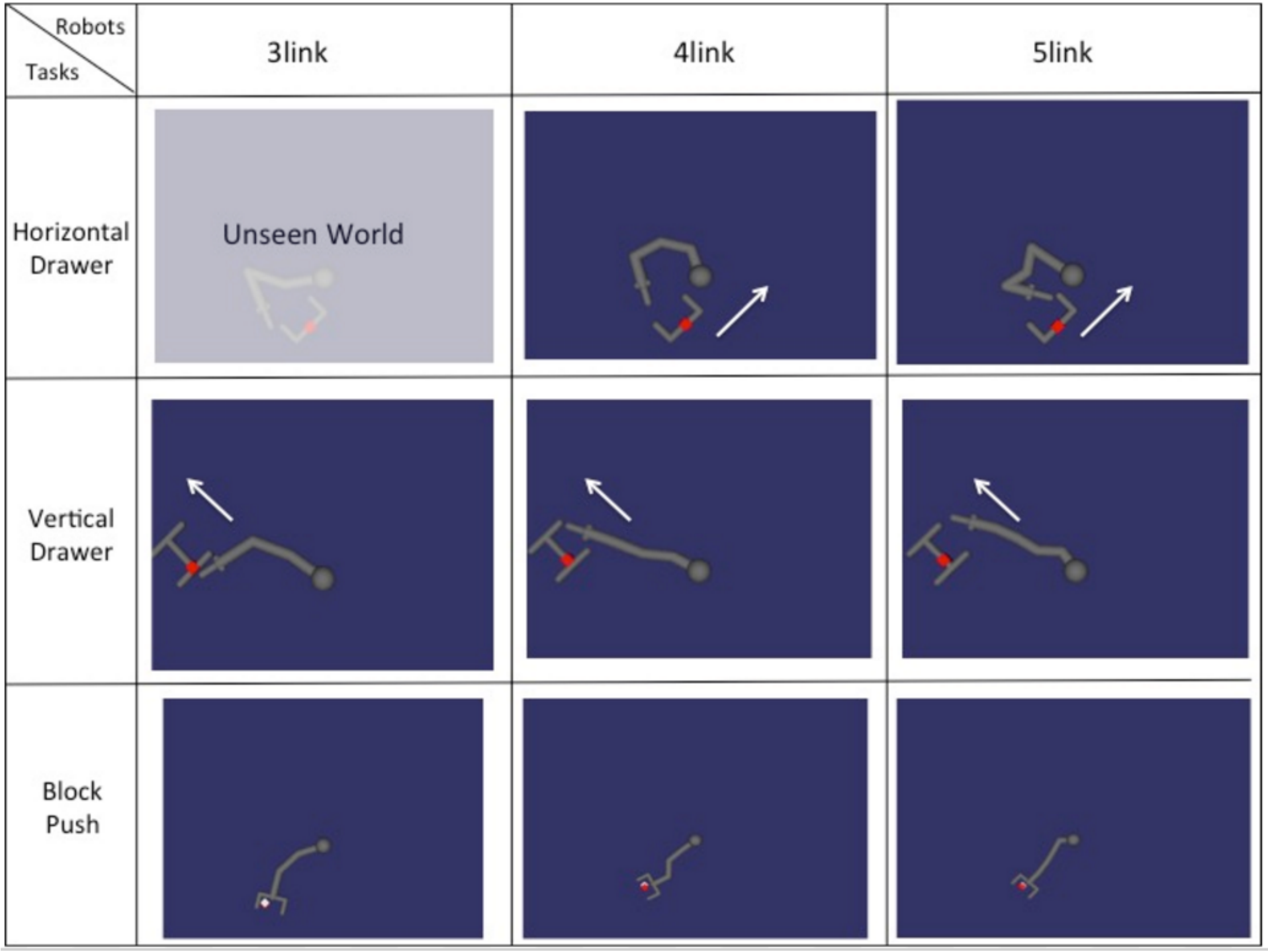
Reusing Representations: Modular Networks



0-shot generalization to new combination

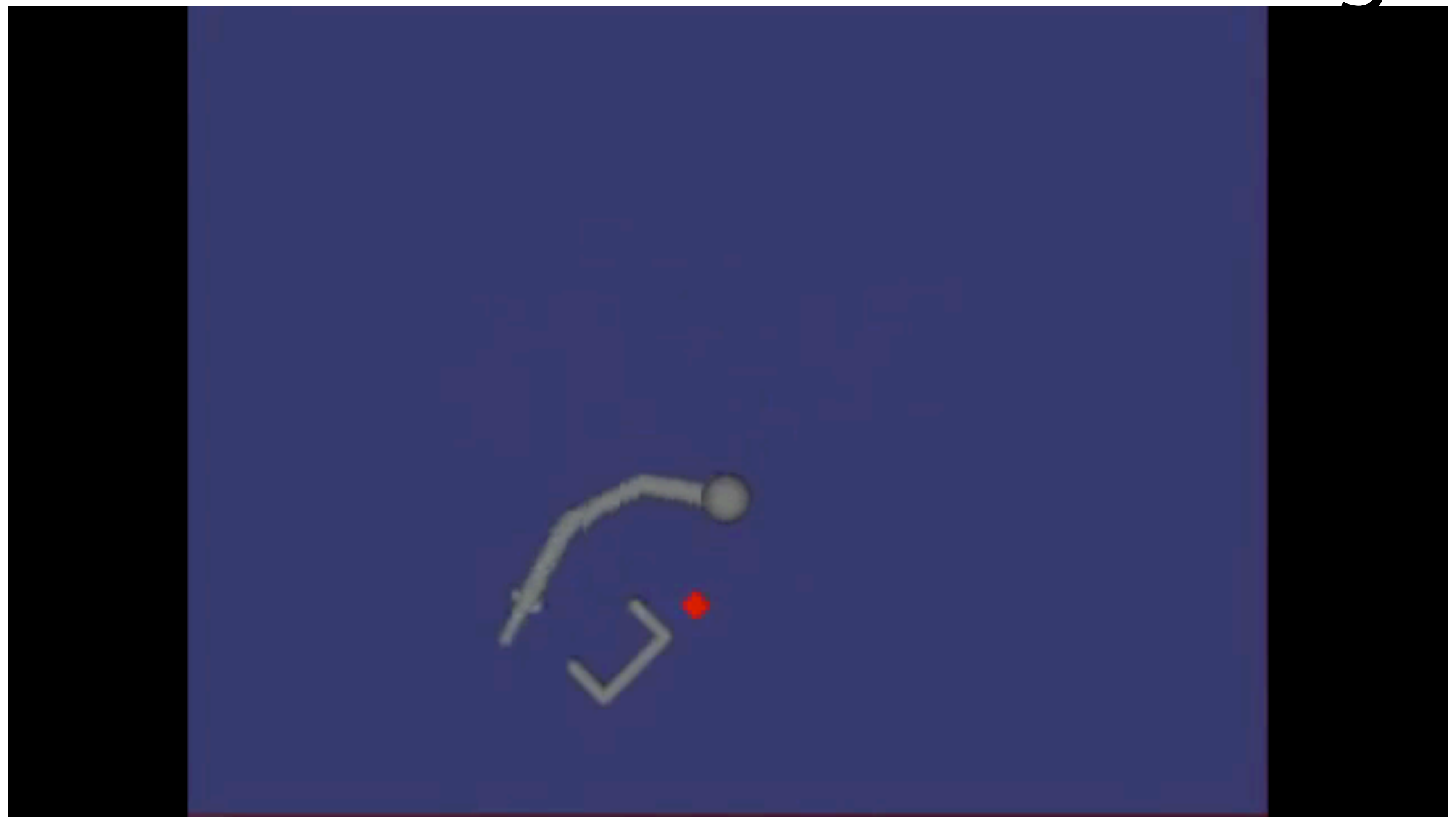


Reusing Representations: Modular Networks



0-shot

with 10 iterations of training



Reusing Representations: Modular Networks

Pros:
- interpretable representation
- can achieve 0-shot generalization in some cases, and if not, can fine-tune

Cons:
- need to limit information-passing & regularize for modular effect

Can we learn to adapt quickly?

Outline: Achieving Transfer in RL

1. **Handling changes in reward**
   a. task represented in the observation
2. **Handling changes in environment**
   a. diversity for sim-to-real transfer
3. **Reusing representations**
   a. progressive networks
   b. PathNet
   c. modular networks
4. **Meta-learning**
   a. Learning to learn quickly
   b. Few-shot adaptation

Meta-Learning

Learn structure across tasks, such that learning on a new task is faster

Learn an update rule / generating weights

Schmidhuber '87, '92
Bengio et al. '90, '92
Ravi & Larochelle '17
Ha et al. '17
Li & Malik '17
Andrychowicz et al. '17

Bayesian Modeling

Tenenbaum '99, '00
Fei-Fei et al. '05, '07
Lake et al. '11
Edwards & Storkey '17

Memory-Augmentation

Santoro et al., '16
Vinyals et al. '16

[*and more in the last month*]

Meta-Learning: Learning Fast RL

RL^2: FAST REINFORCEMENT LEARNING VIA SLOW REINFORCEMENT LEARNING

Yan Duan[†‡], John Schulman[†‡], Xi Chen[†‡], Peter L. Bartlett[†], Ilya Sutskever[‡], Pieter Abbeel[†‡]
[†] UC Berkeley, Department of Electrical Engineering and Computer Science
[‡] OpenAI

LEARNING TO REINFORCEMENT LEARN

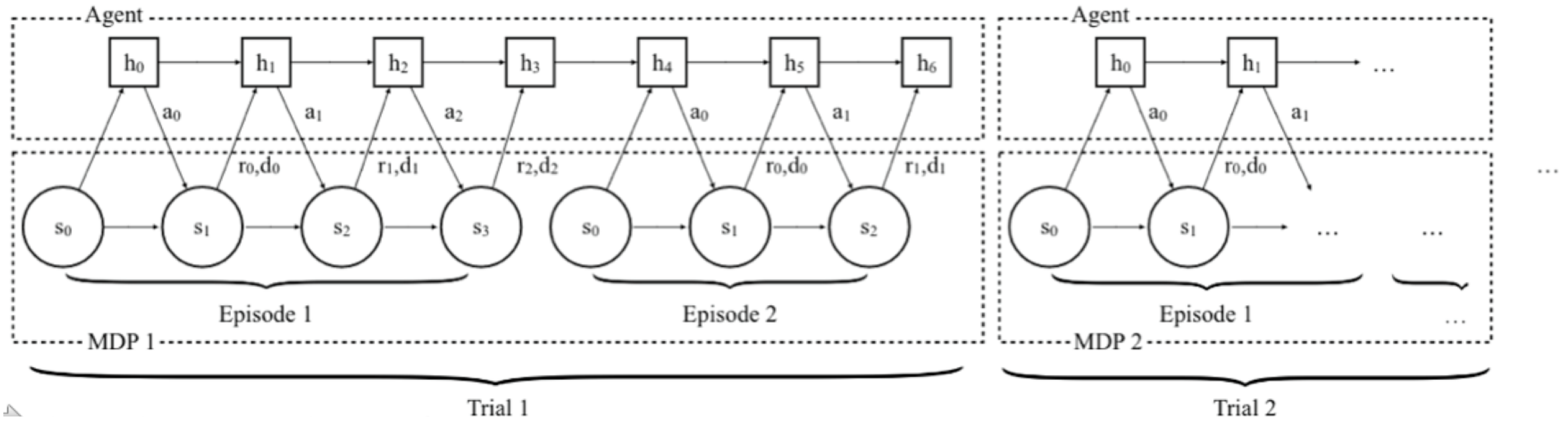
JX Wang[1], Z Kurth-Nelson[1], D Tirumala[1], H Soyer[1], JZ Leibo[1],
R Munos[1], C Blundell[1], D Kumaran[1,3], M Botvinick[1,2]
[1]DeepMind, London, UK
[2]Gatsby Computational Neuroscience Unit, UCL, London, UK
[3]Institute of Cognitive Neuroscience, UCL, London, UK

Key idea: Train an RNN to learn in new MDPs

Meta-Learning: Learning Fast RL

Representation of the fast RL algorithm:

- RNN = generic computation architecture
- different weights in the RNN means different RL algorithm
- different activations in the RNN means different current policy

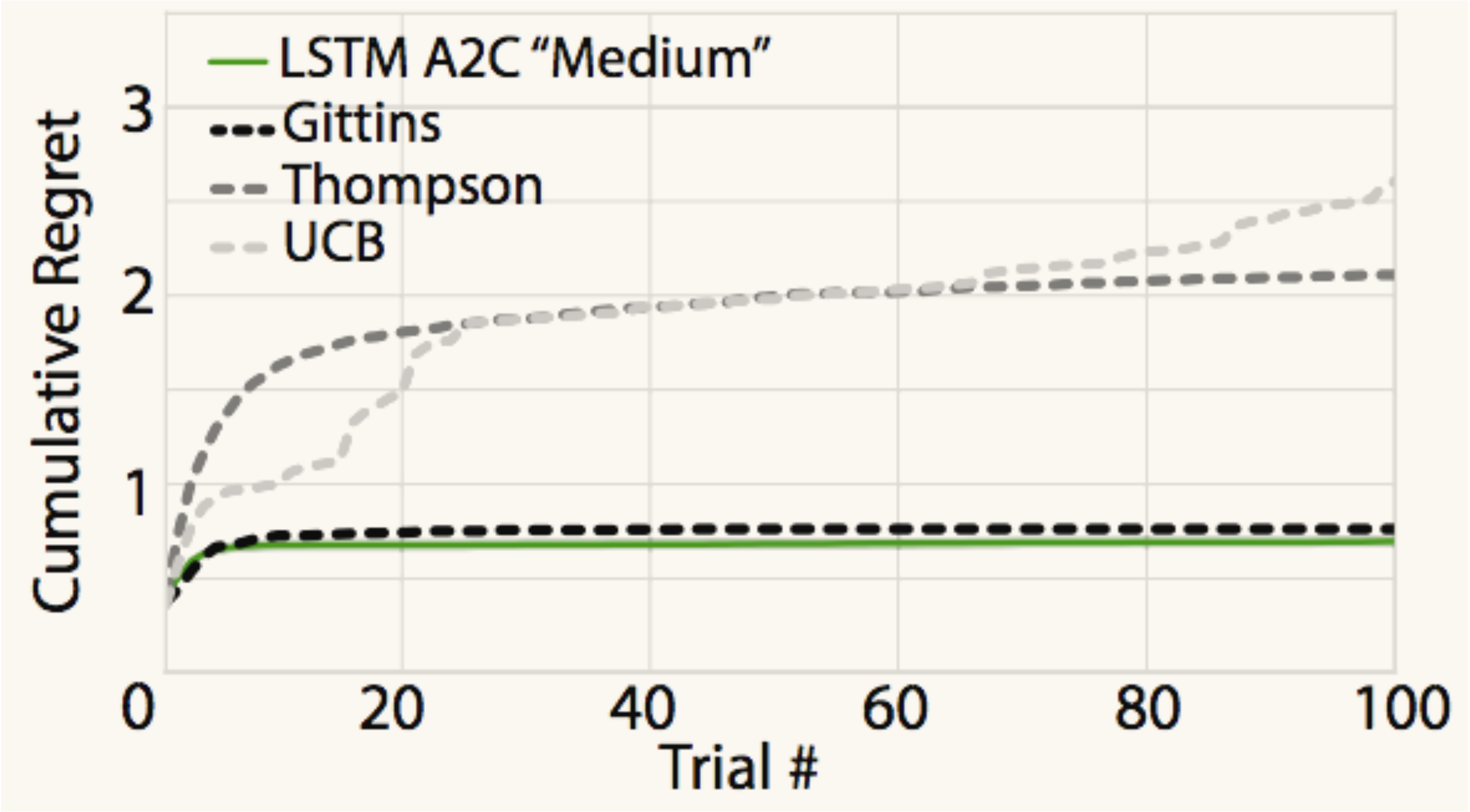


slide adapted from Pieter Abbeel

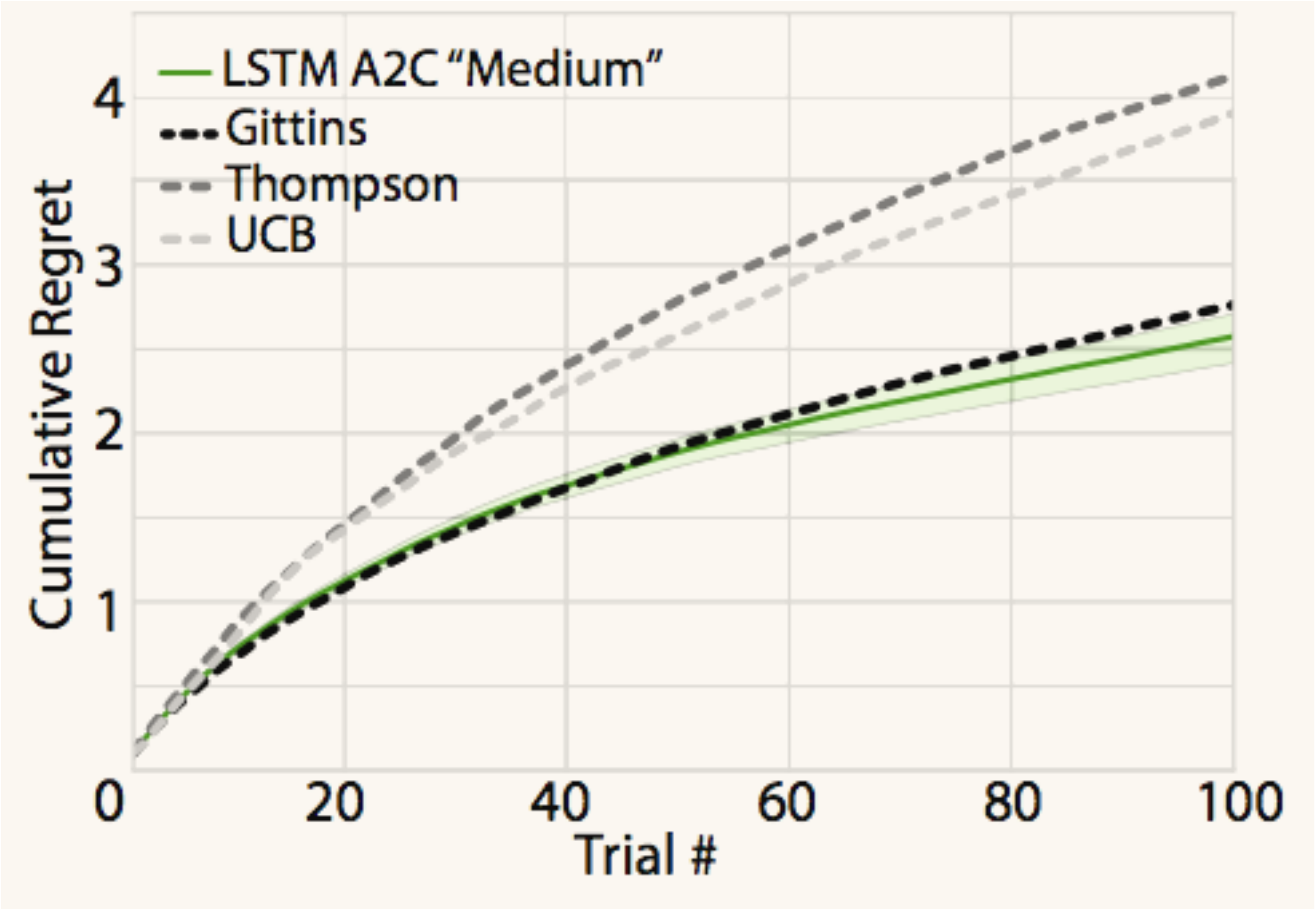
Meta-Learning: Learning Fast RL

comparison to asymptotically optimal algorithms on correlated-arm bandit problems

trained on medium difficulty



test on medium difficulty

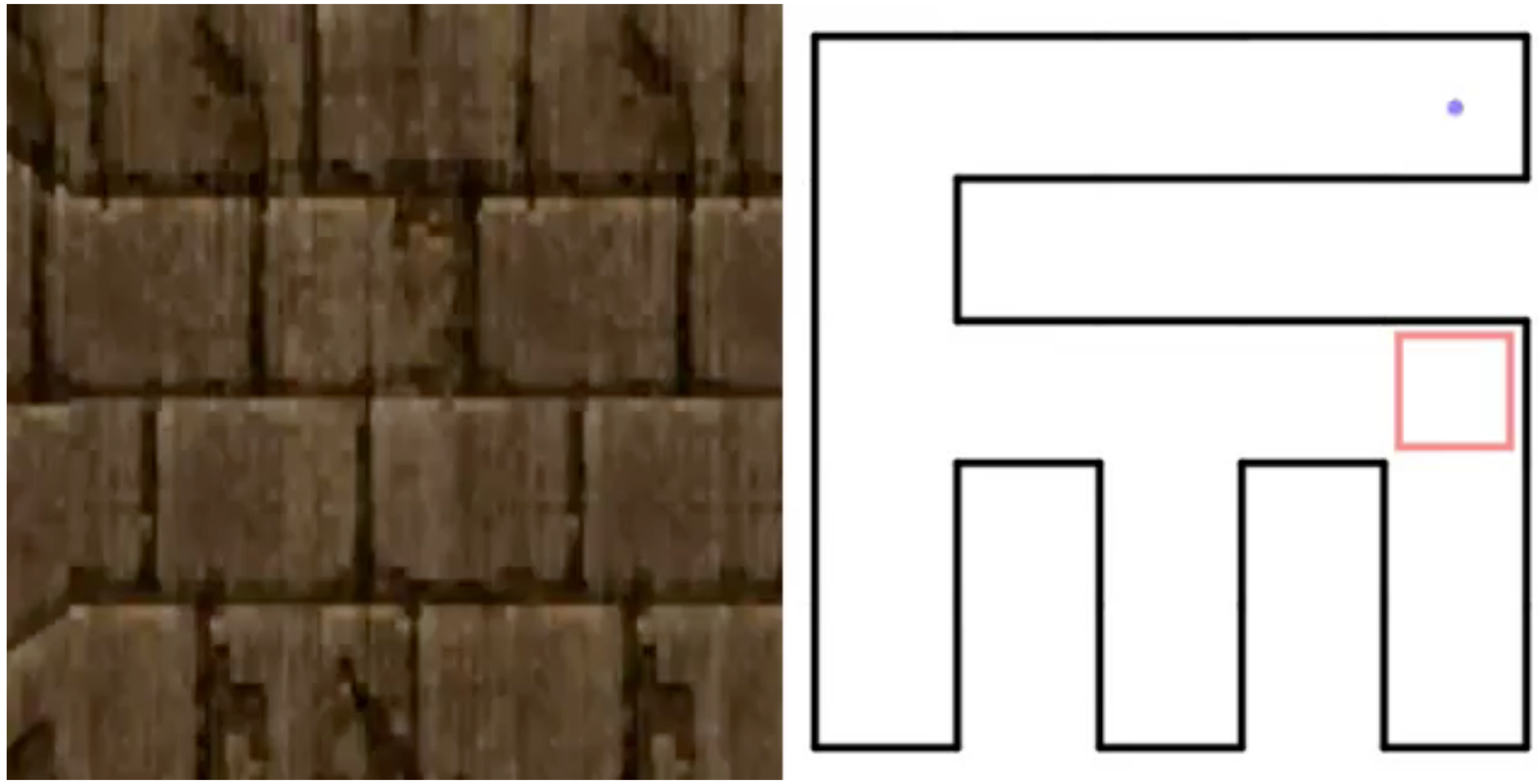


test on hard difficulty (arms less correlated)

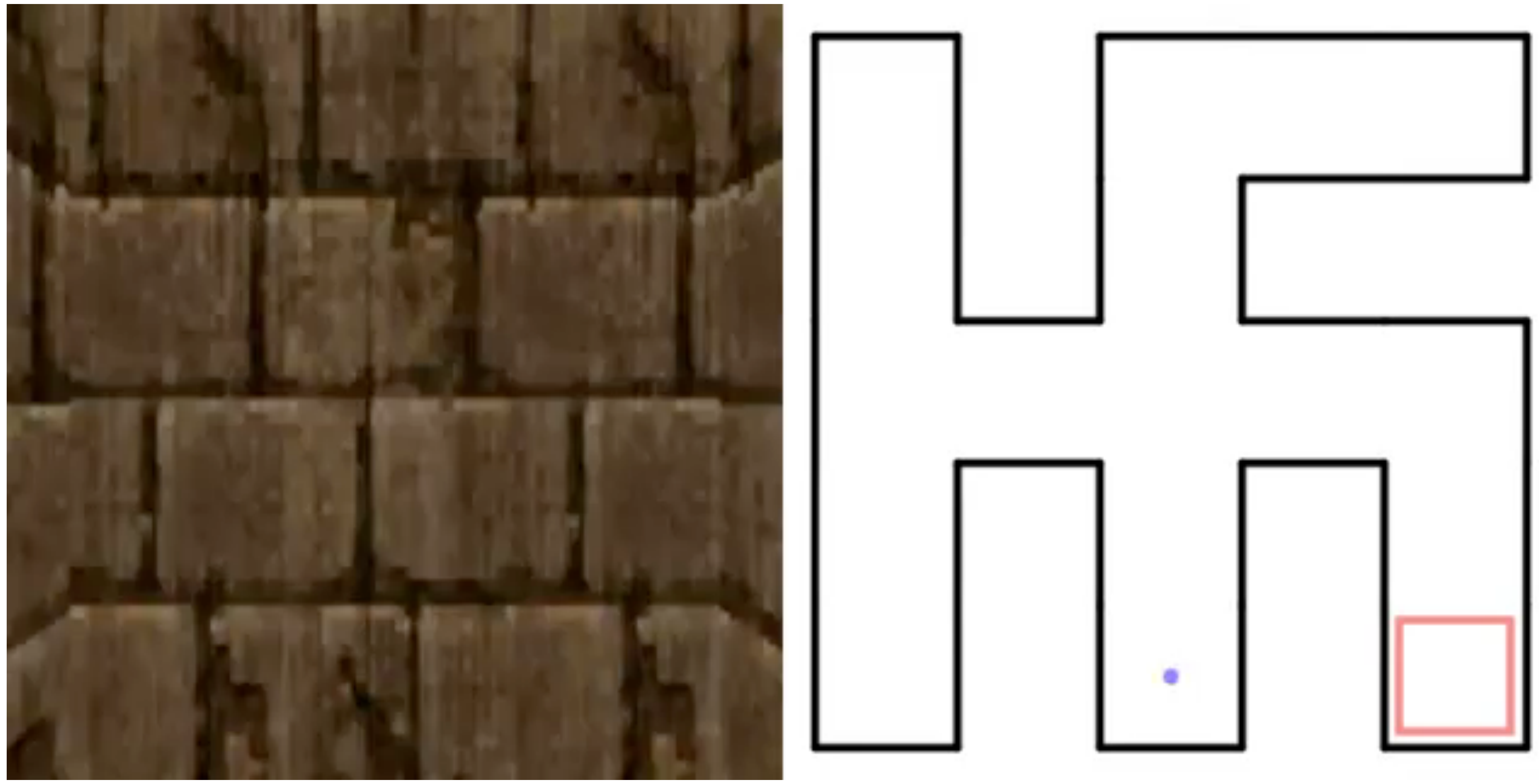
Wang et al. '17

Meta-Learning: Learning Fast RL

before meta-learning



after meta-learning



Duan et al. '17

Meta-Learning: Learning Fast RL

Pros:
- simple approach
- can learn new tasks from the distribution seen during training

Cons:
- asking a lot of the LSTM
- might not handle tasks from outside the distribution well (doesn't reduce to running RL at test-time)

Learning Few-Shot Adaptation

Few-shot/Transfer learning: incorporate prior knowledge from other tasks for fast learning

pretrained parameters

Fine-tuning: $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_\mathcal{T}(\theta)$

[test-time]

task

Our method: $\min_\theta \sum_{\text{task } \mathcal{T}} \mathcal{L}_\mathcal{T}(\theta - \alpha \nabla_\theta \mathcal{L}_\mathcal{T}(\theta))$

Key idea: Train over many tasks, to learn parameter vector θ that transfers

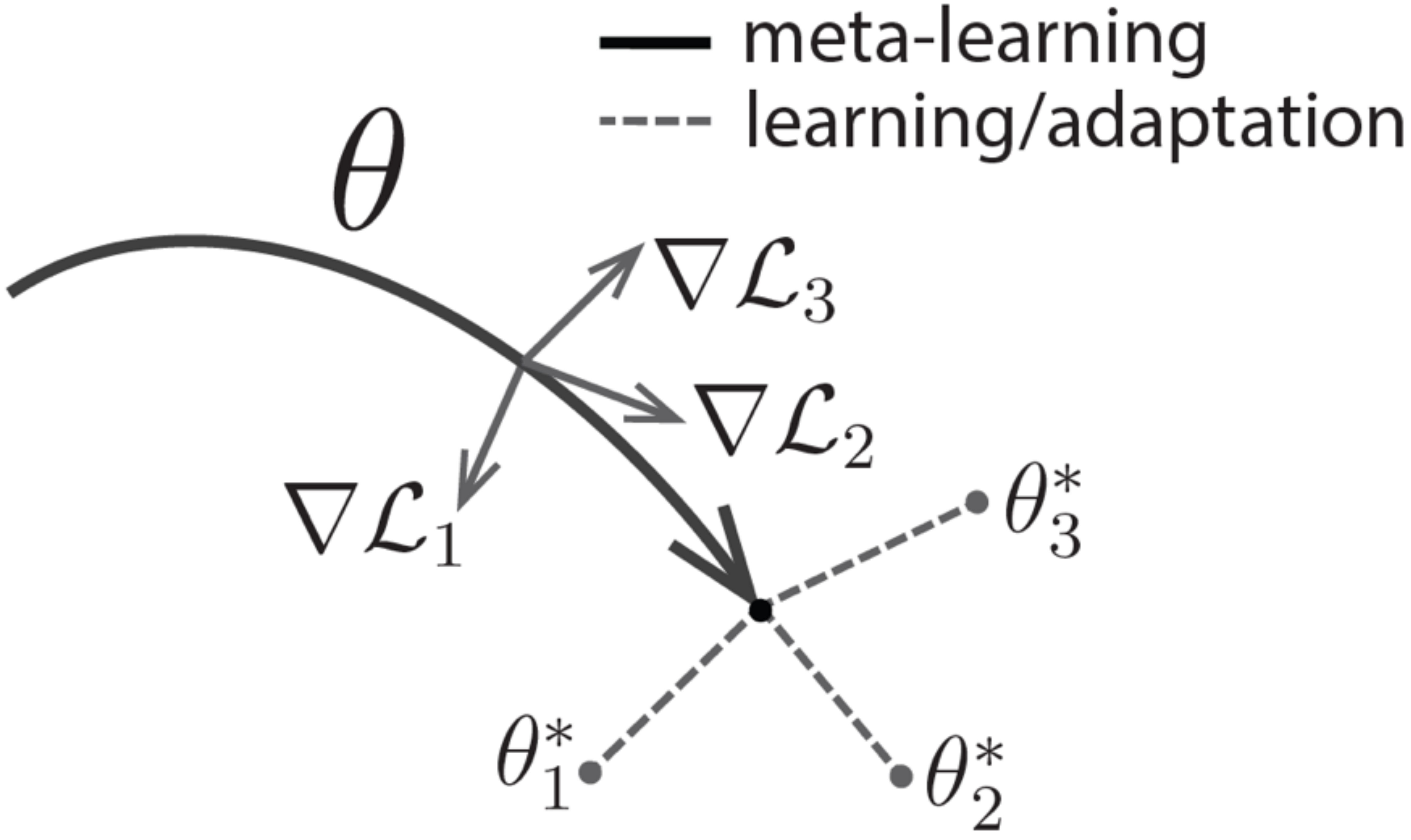
Finn et al. '17

Learning Few-Shot Adaptation

$$\min_{\theta} \sum_{\text{task } \mathcal{T}} \mathcal{L}_{\mathcal{T}}(\theta - \alpha \nabla_{\theta} \mathcal{L}_{\mathcal{T}}(\theta))$$

θ parameter vector being meta-learned

θ_i^* optimal parameter vector for task i

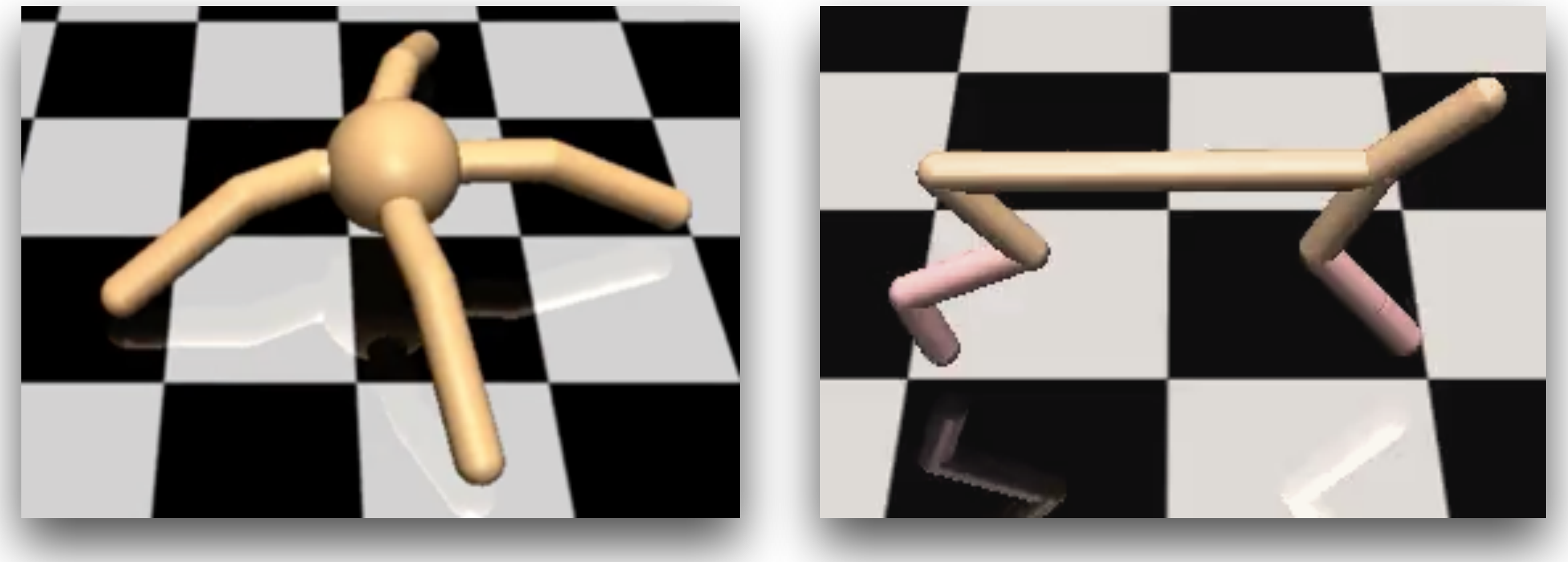


Model-**A**gnostic **M**eta-**L**earning

Finn et al. '17

Fast Adaptation in Reinforcement Learning

Locomotion problems:



1. run at goal velocity
   (continuous range of tasks)
2. run forward or backward
   (2 tasks)

Methods:

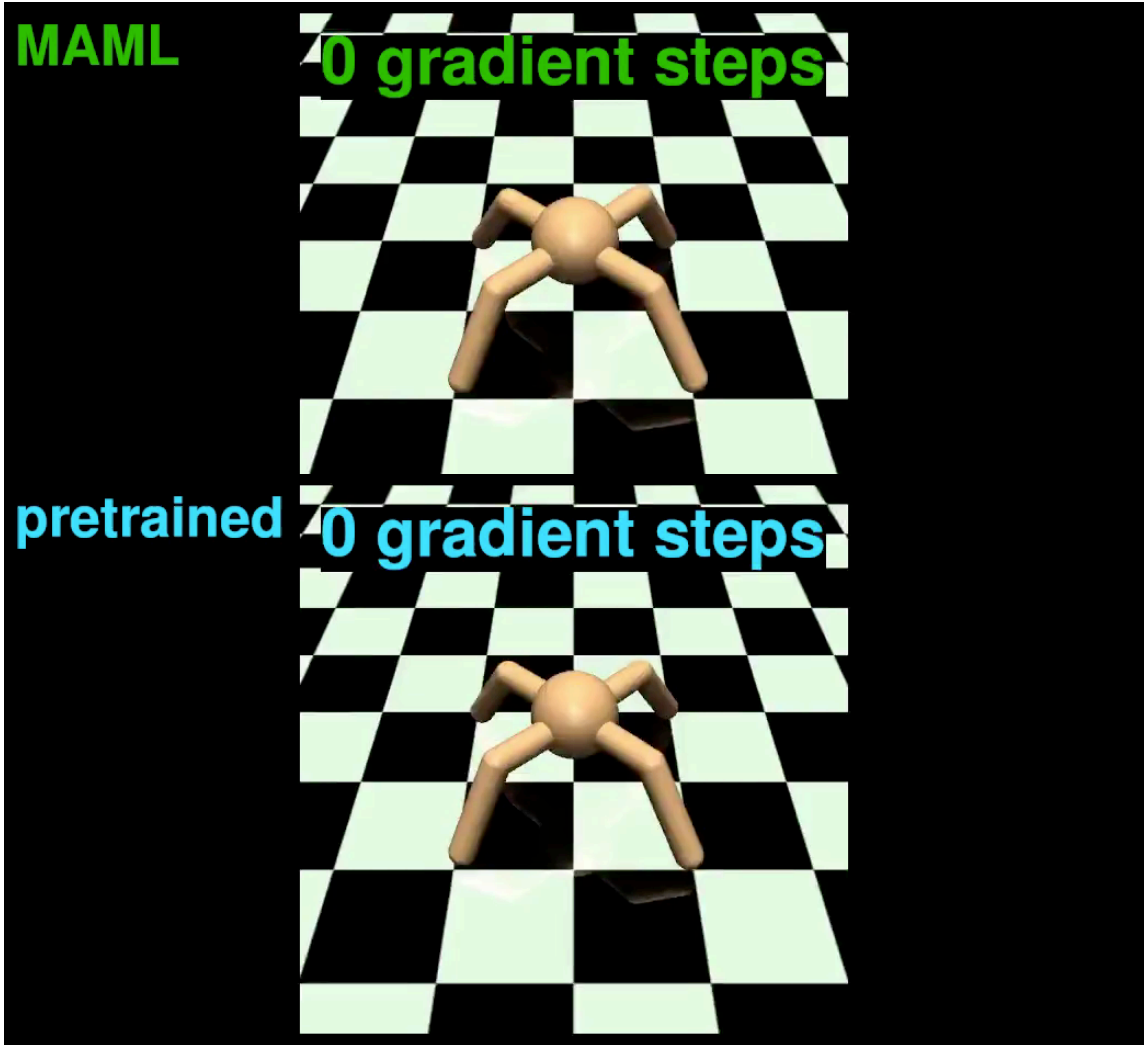
MAML
meta-learner: TRPO
learner: vanilla policy gradient (REINFORCE)

Baselines: oracle (gets task as input) pretrain on all tasks random init

all: 20 roll-outs for learner update

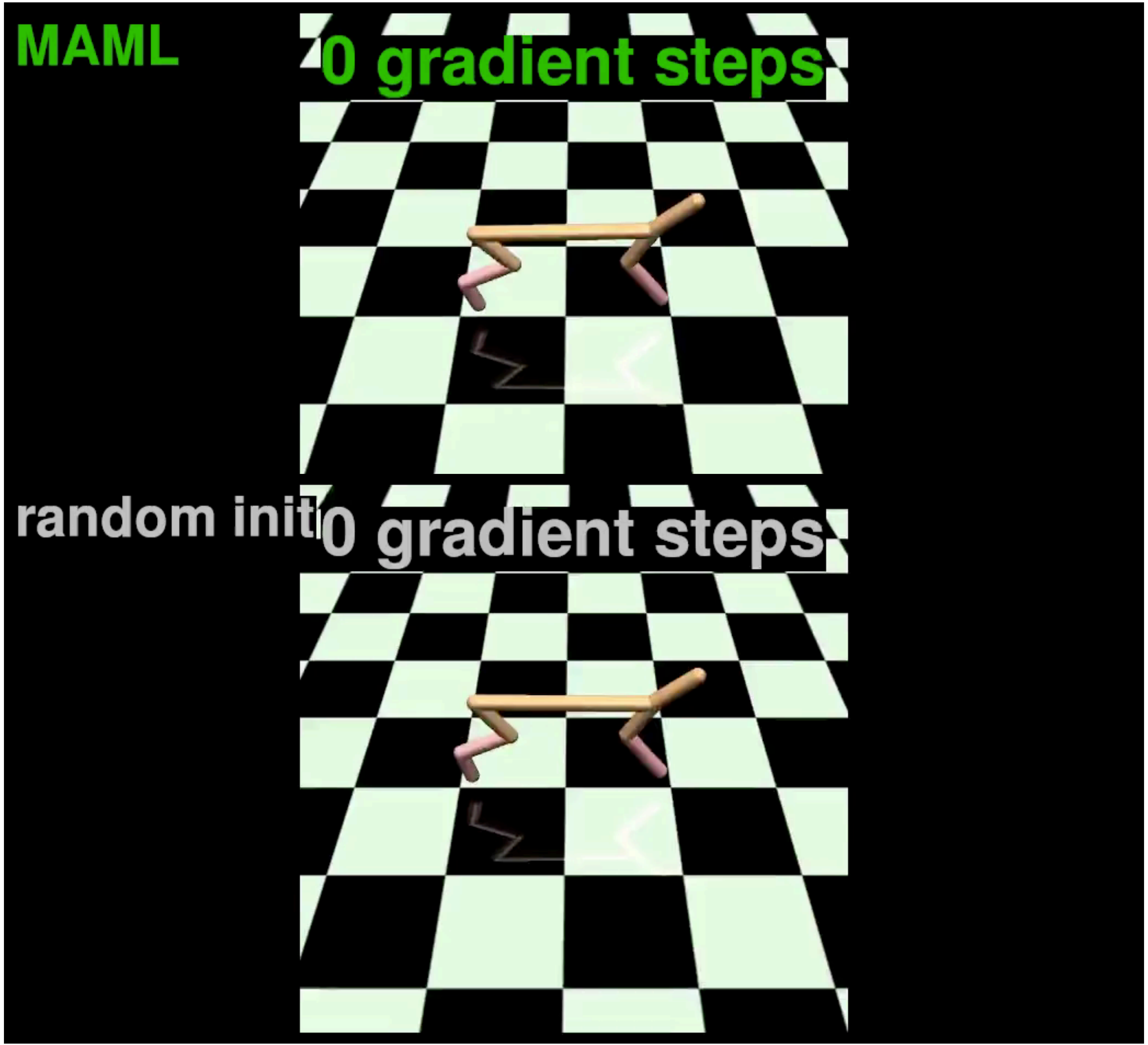
Finn et al. '17

walk/run at goal velocity



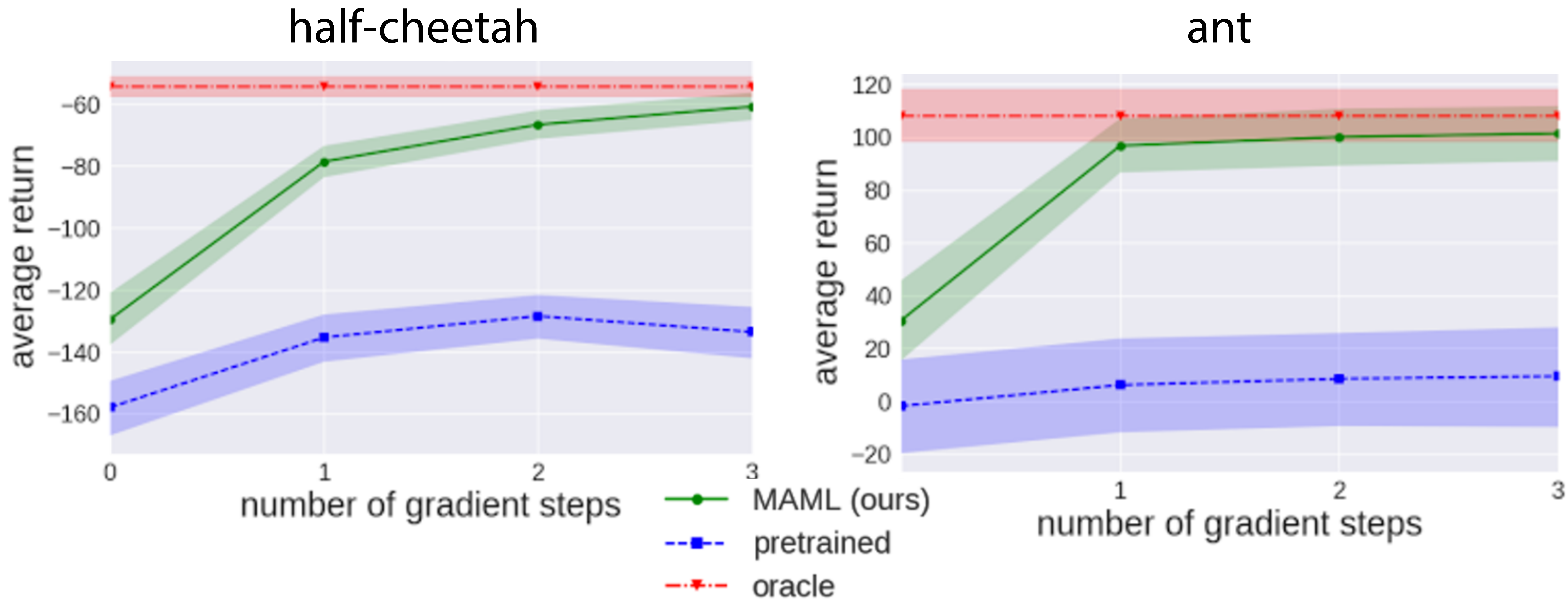
Finn et al. ‘17

run backward or forward



Finn et al. '17

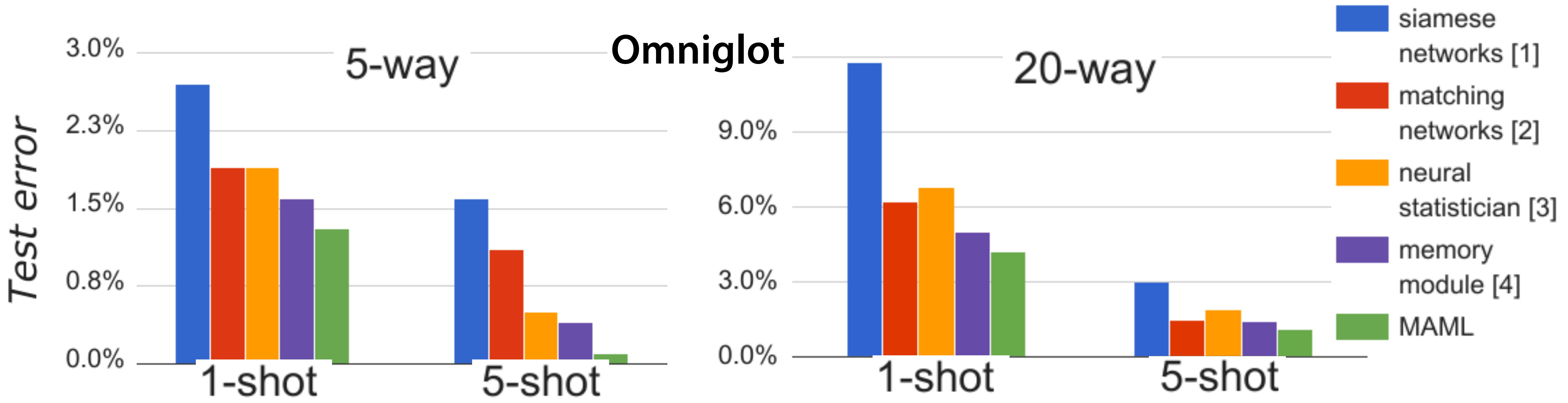
Quantitative Results



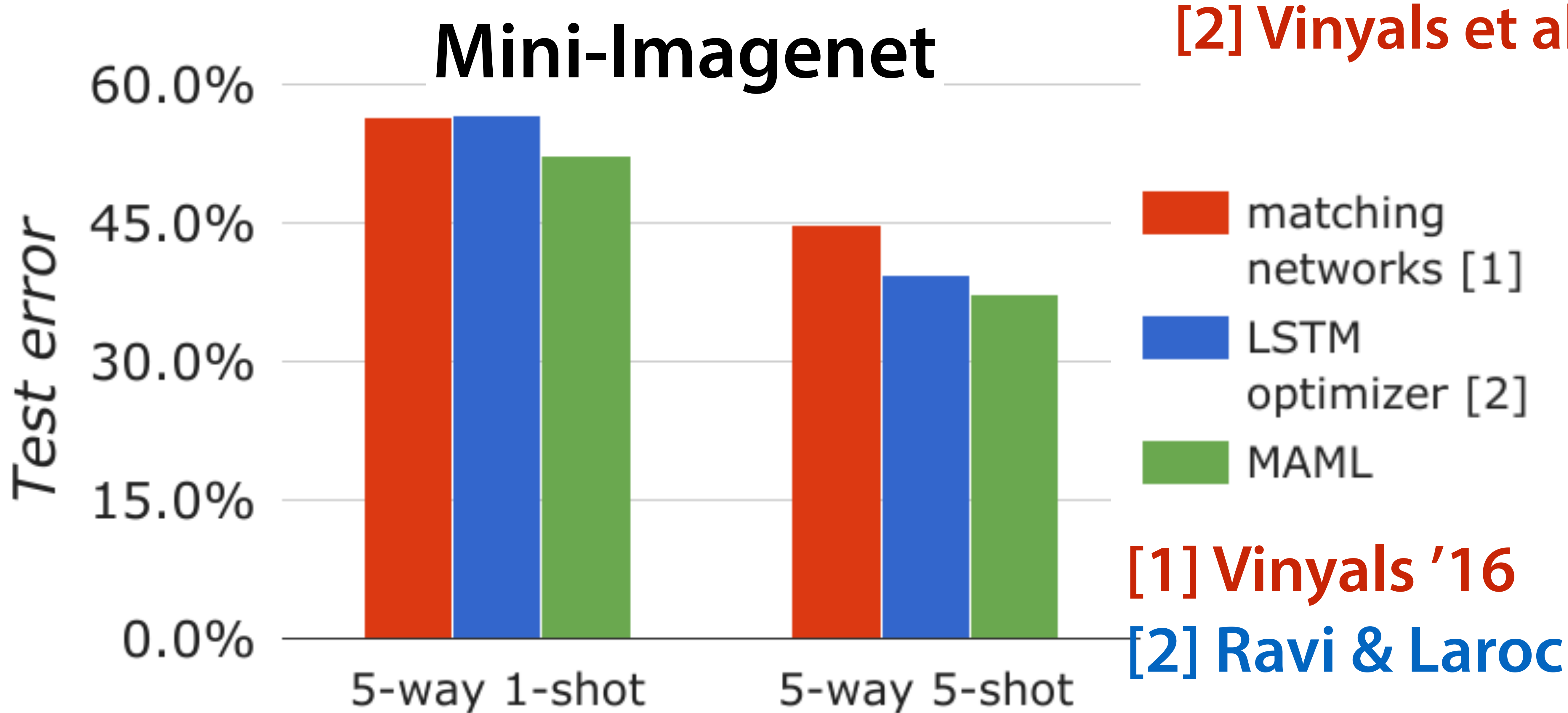
key finding: continues to improve with more updates

Finn et al. '17

Few-Shot Image Classification



[1] Koch '15 [3] Edwards & Storkey '17
[2] Vinyals et al. '16 [4] Kaiser et al. '17



[1] Vinyals '16
[2] Ravi & Larochelle '17

Finn et al. '17

Meta-Learning: MAML

Pros:
- same learning rule at test time — just run RL
- few-shot adaptation

Cons:
- need to enumerate tasks at meta-training time

Takeaways: Achieving Transfer in RL

Most popular RL benchmarks evaluate mastery, not generalization.

Approaches for transfer in RL

1. **Task represented in the observation**
2. **Diversity for sim-to-real transfer**
3. **Reusing representations**
4. **Meta-learning**

Not covered: catastrophic forgetting, options framework (Sutton, Precup, & Singh, '99)

Next time: Quoc Le & Barret Zoph (Google Brain)