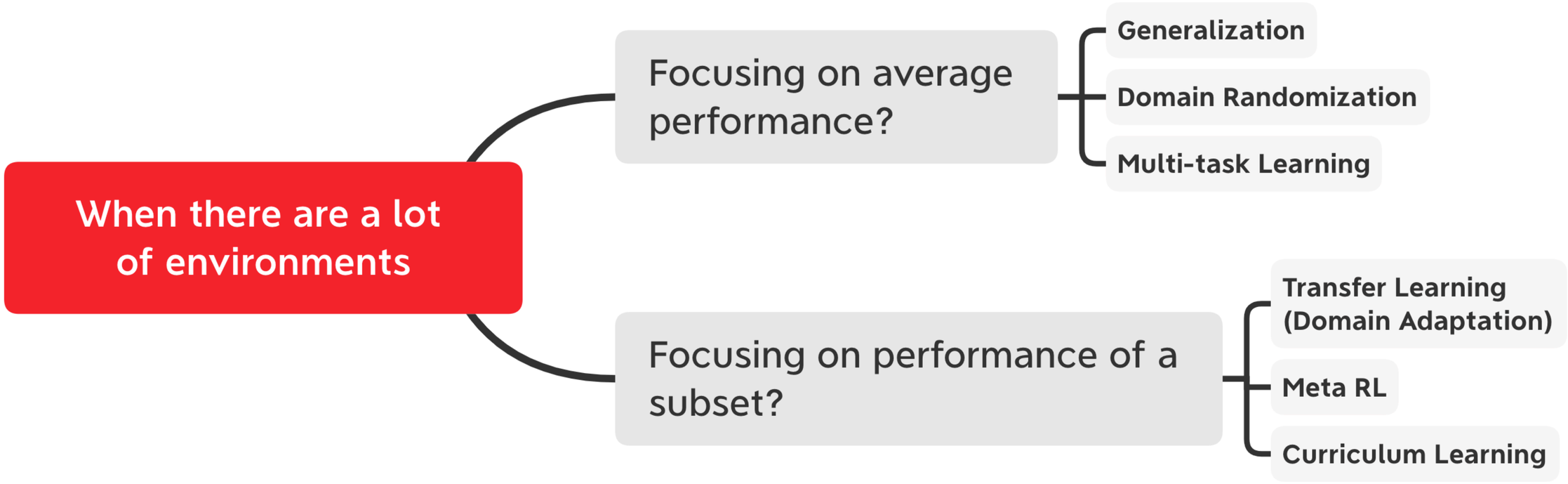
Generalization Over Environment Distribution

Peng Zhenghao
2020-11-24

Overview

What if we have a set (or a distribution) of environments?



Generalization

- A critical ability of RL agent: The ability to generalize in unseen environments.
- $\mathop{\mathbb{E}}_{\tau} R(\tau)$ is changed to $\mathop{\mathbb{E}}_{env} \mathop{\mathbb{E}}_{\tau} R(\tau)$
- But this feature is hardly tested, because a key problem in current RI community: **test on training set**!

Generalization

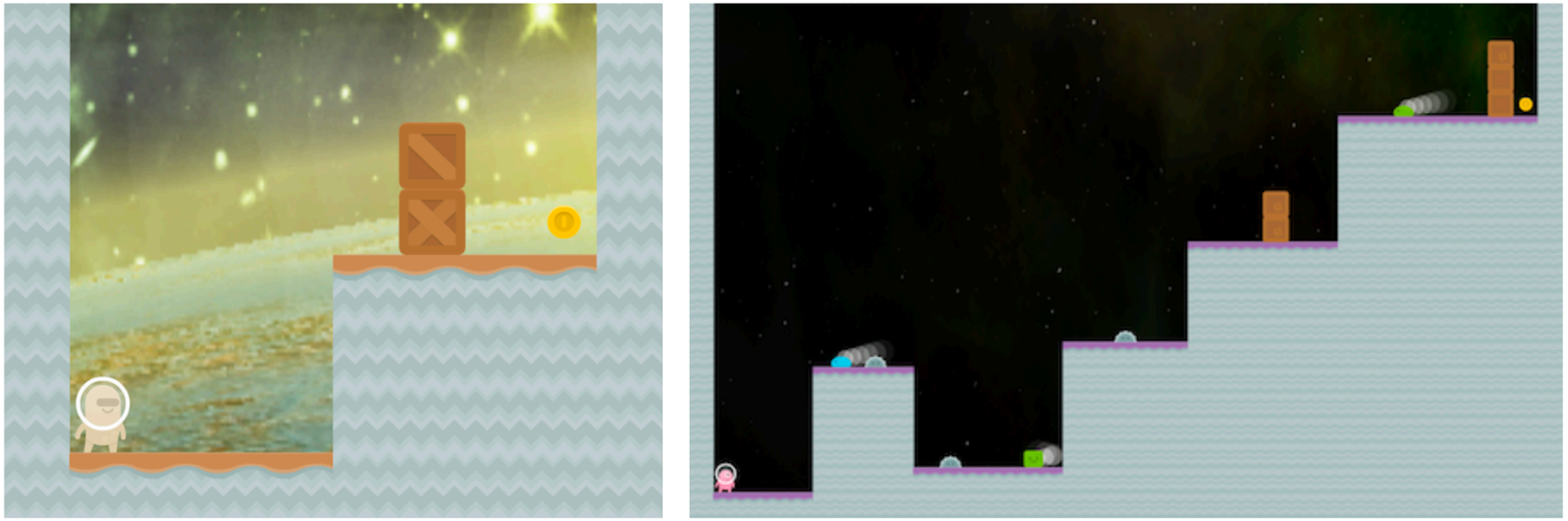
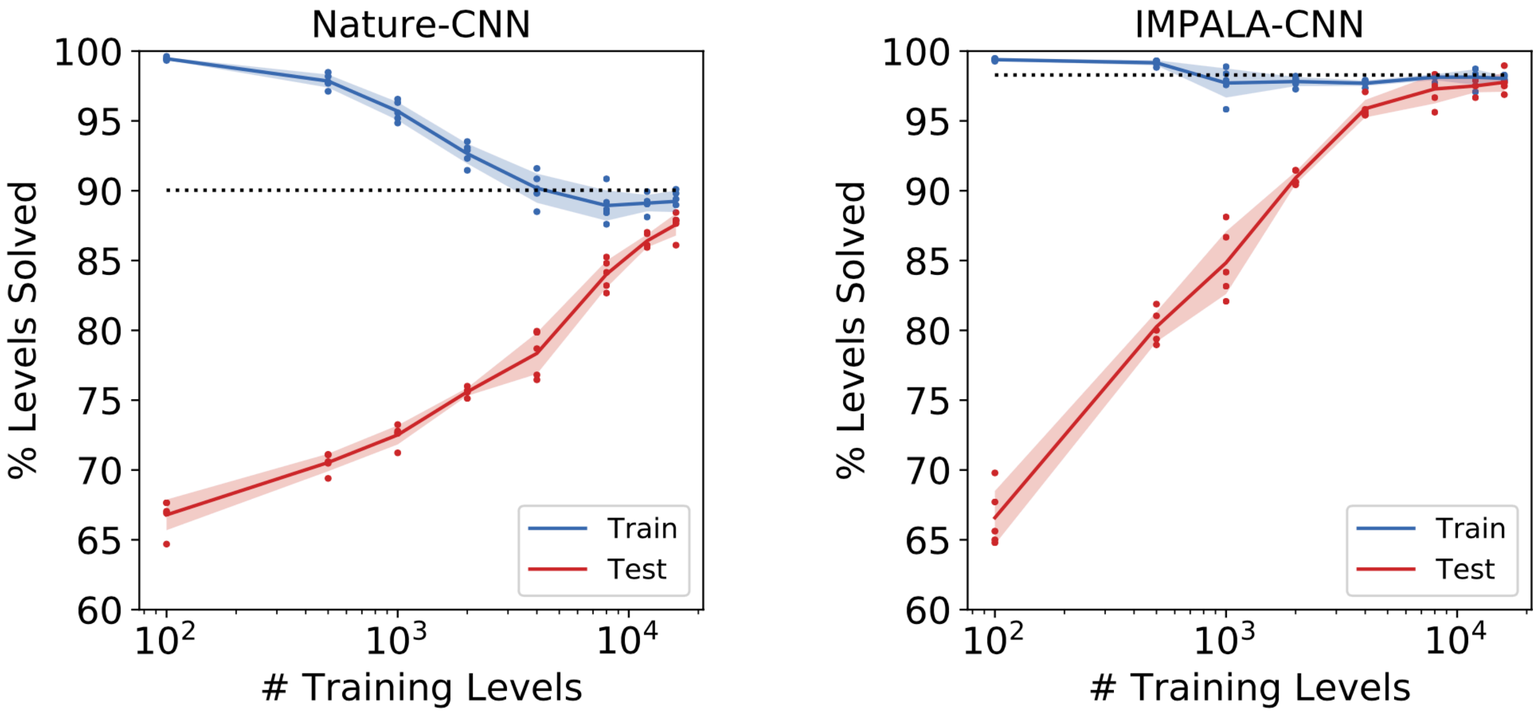


Figure 1. Two levels in CoinRun. The level on the left is much easier than the level on the right.

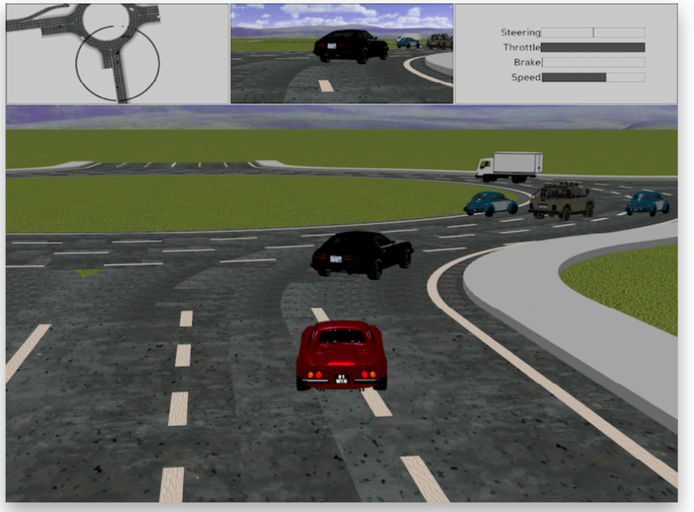
- Procedurally generated infinite number of levels.
- Train and test agents in different environments.

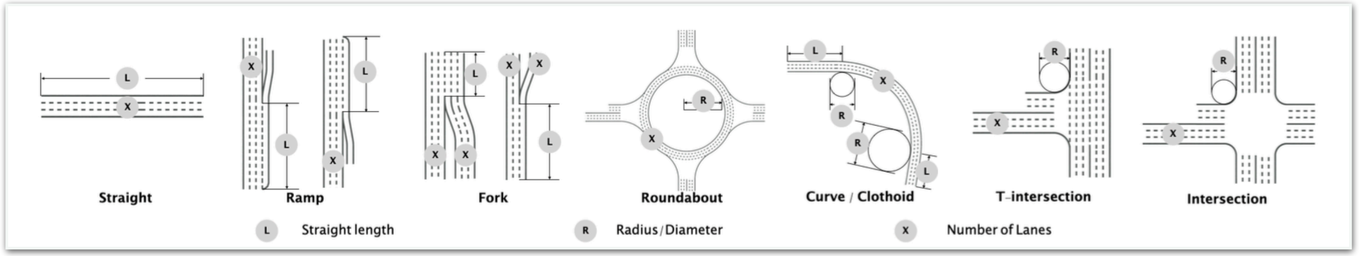
Generalization

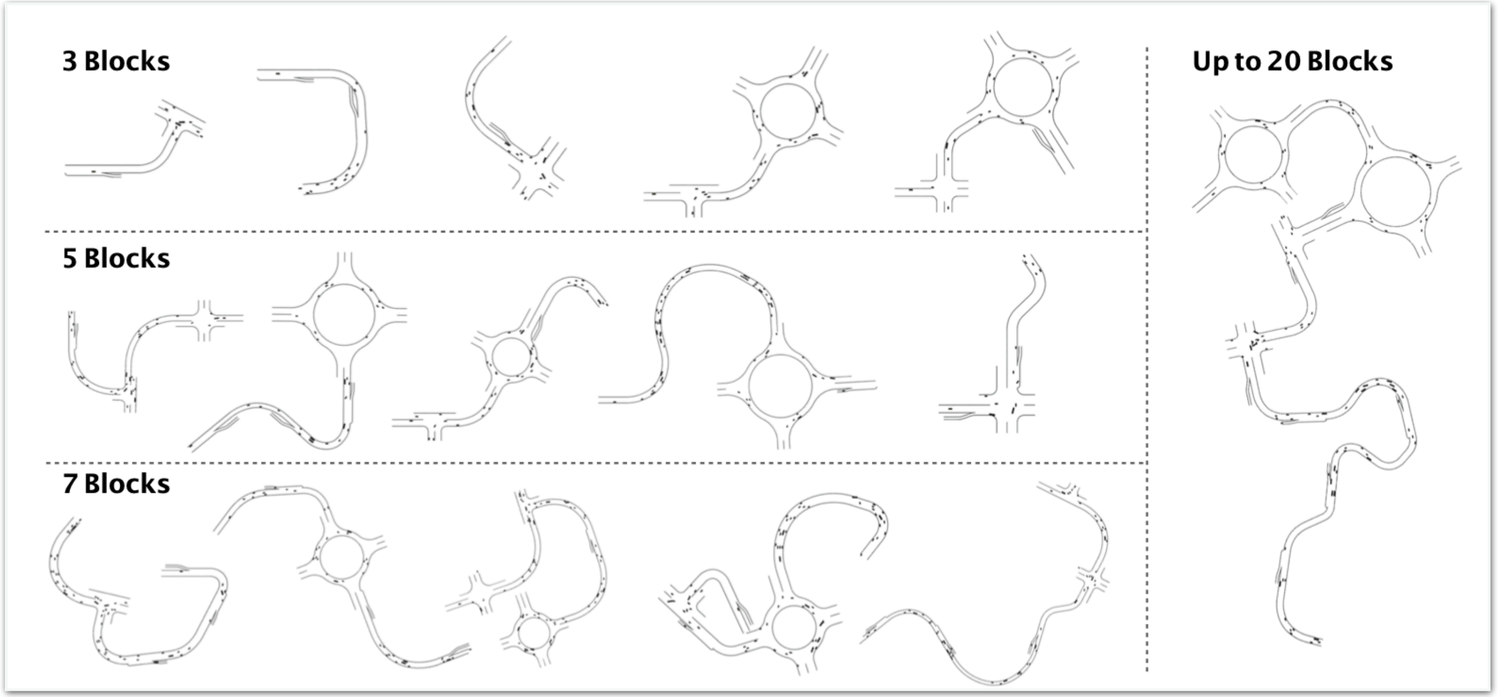


- How to evaluate generalization? The gap between training and test performance.

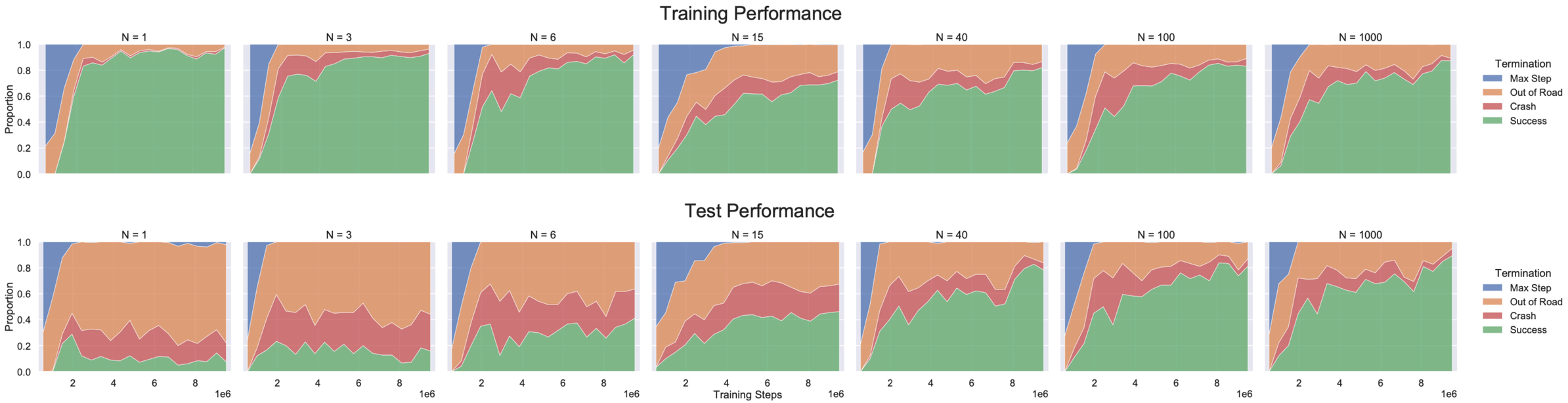
Generalization





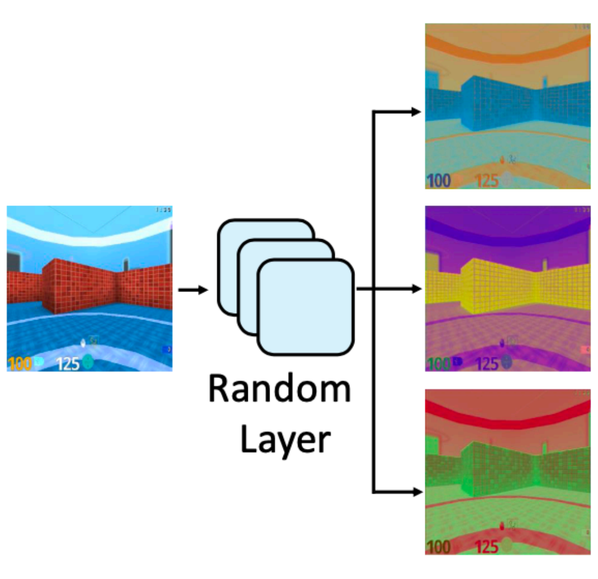


Generalization

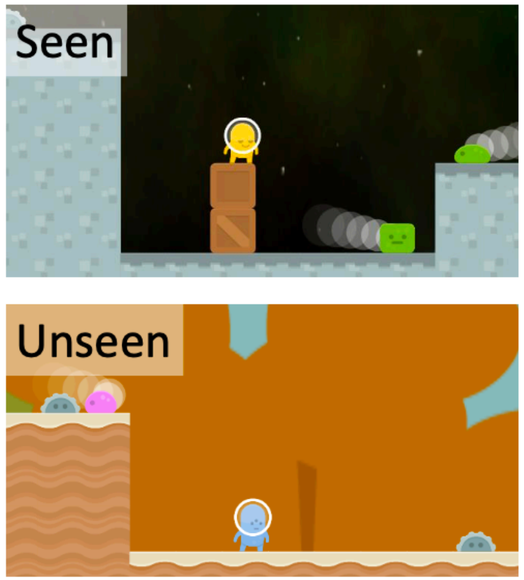


- Training success rate drop as the number training environments increase, while test success rate is increasing.

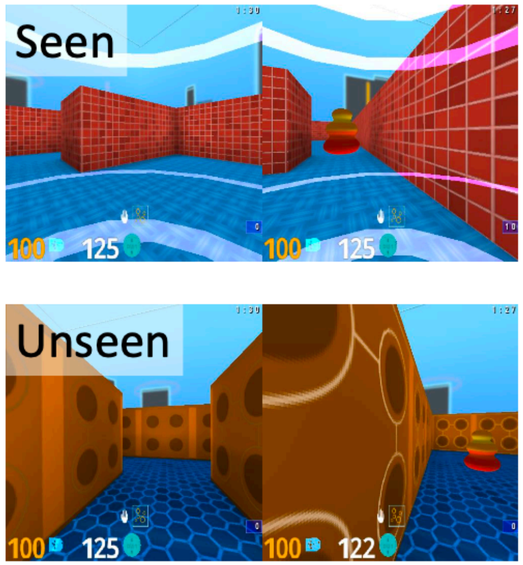
Generalization



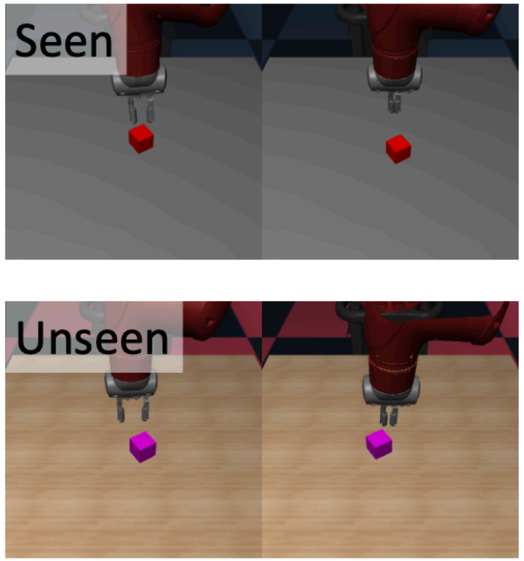
(a) Outputs of random layer



(b) 2D CoinRun



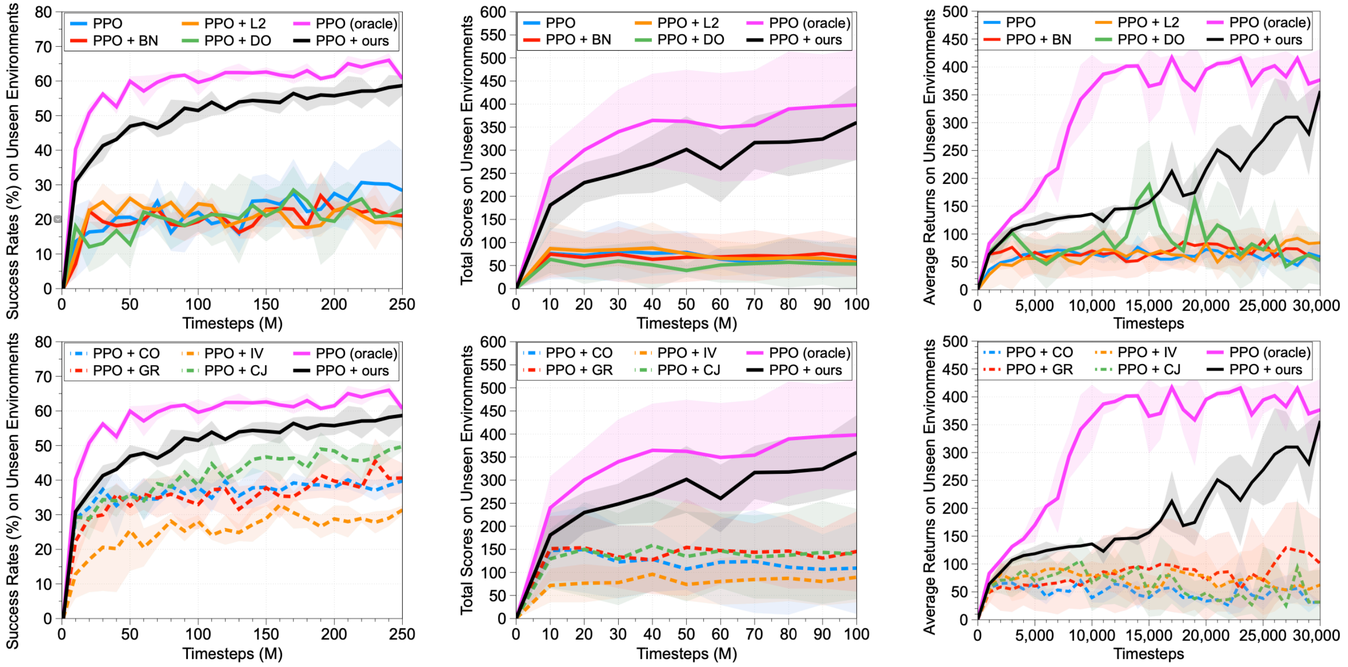
(c) 3D DeepMind Lab



(d) 3D Surreal robotics

- How to improve generalization?
- Increase the diversity of input data! Just like data augmentation in supervised / imitation learning.

Generalization

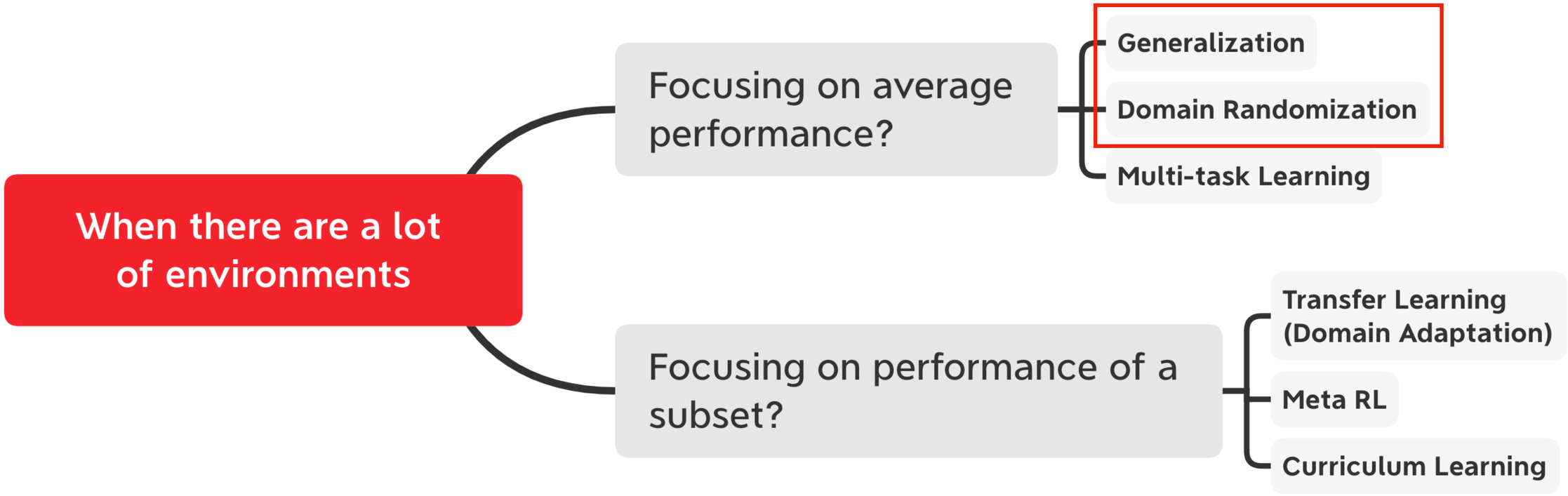


(a) Large-scale CoinRun (b) DeepMind Lab (c) Surreal robotics

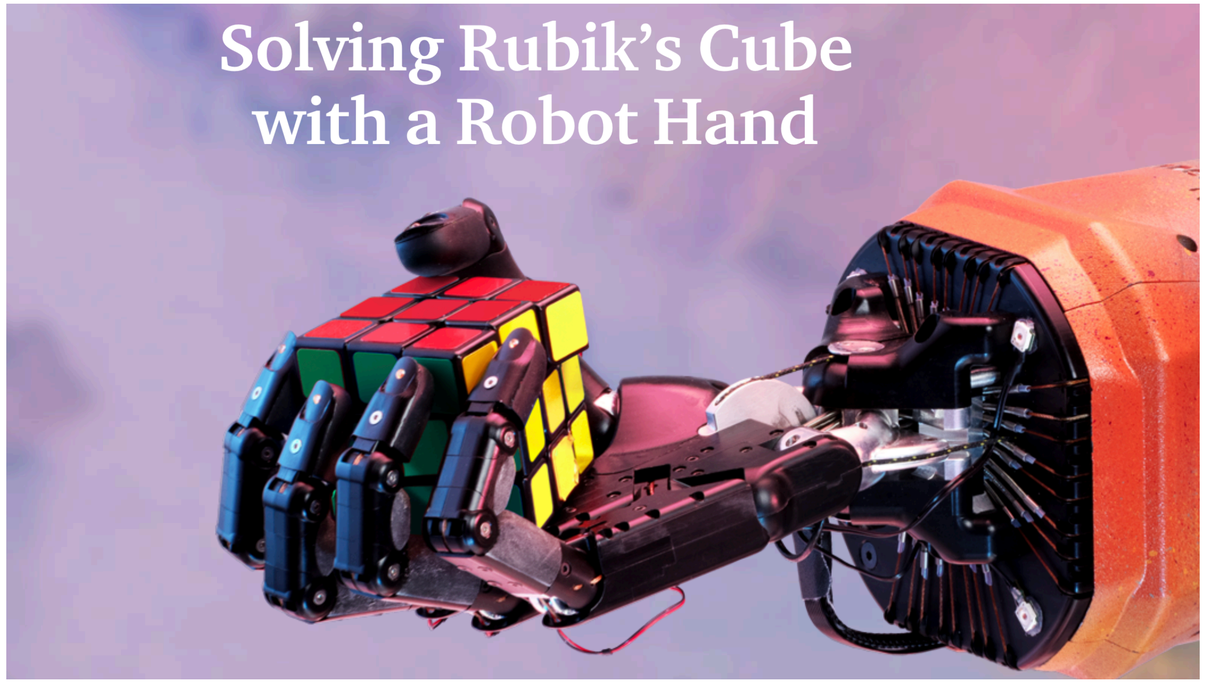
Figure 5: The performances of trained agents in unseen environments under (a) large-scale CoinRun, (b) DeepMind Lab and (c) Surreal robotics control. The solid/dashed lines and shaded regions represent the mean and standard deviation, respectively.

Overview

What if we have a set (or a distribution) of environments?



Domain Randomization



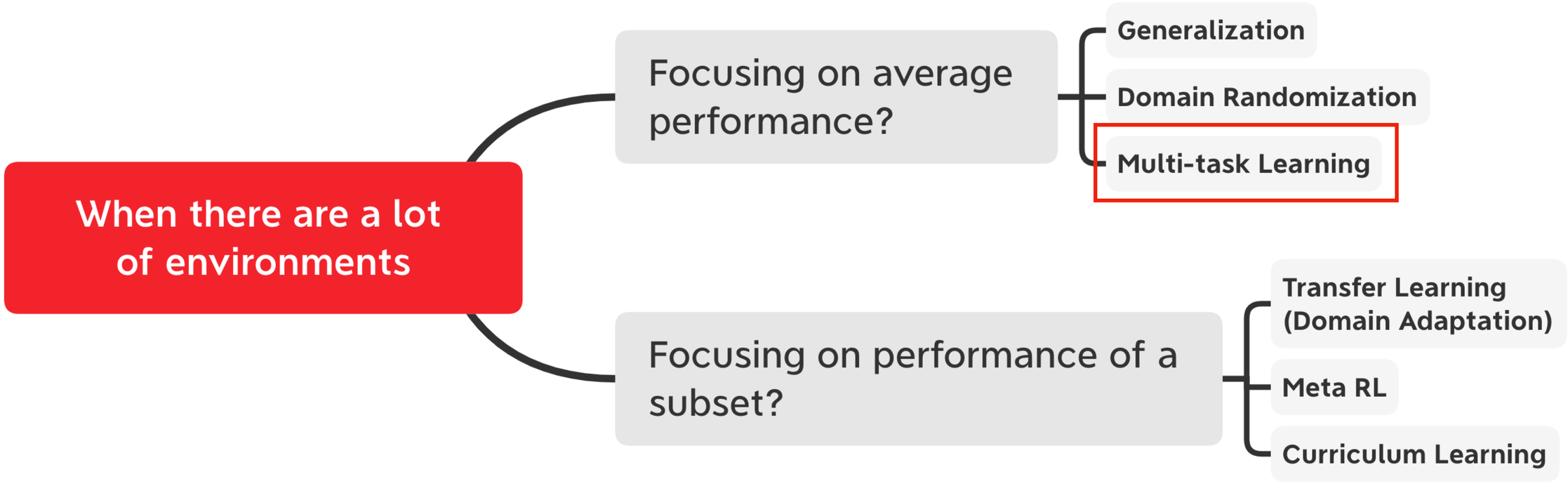
- Improve generalization via adequate randomness.
- When agent is robust in a wide range of environments, it would robust too in real world.
- Use widely in sim-to-real task

Domain Randomization

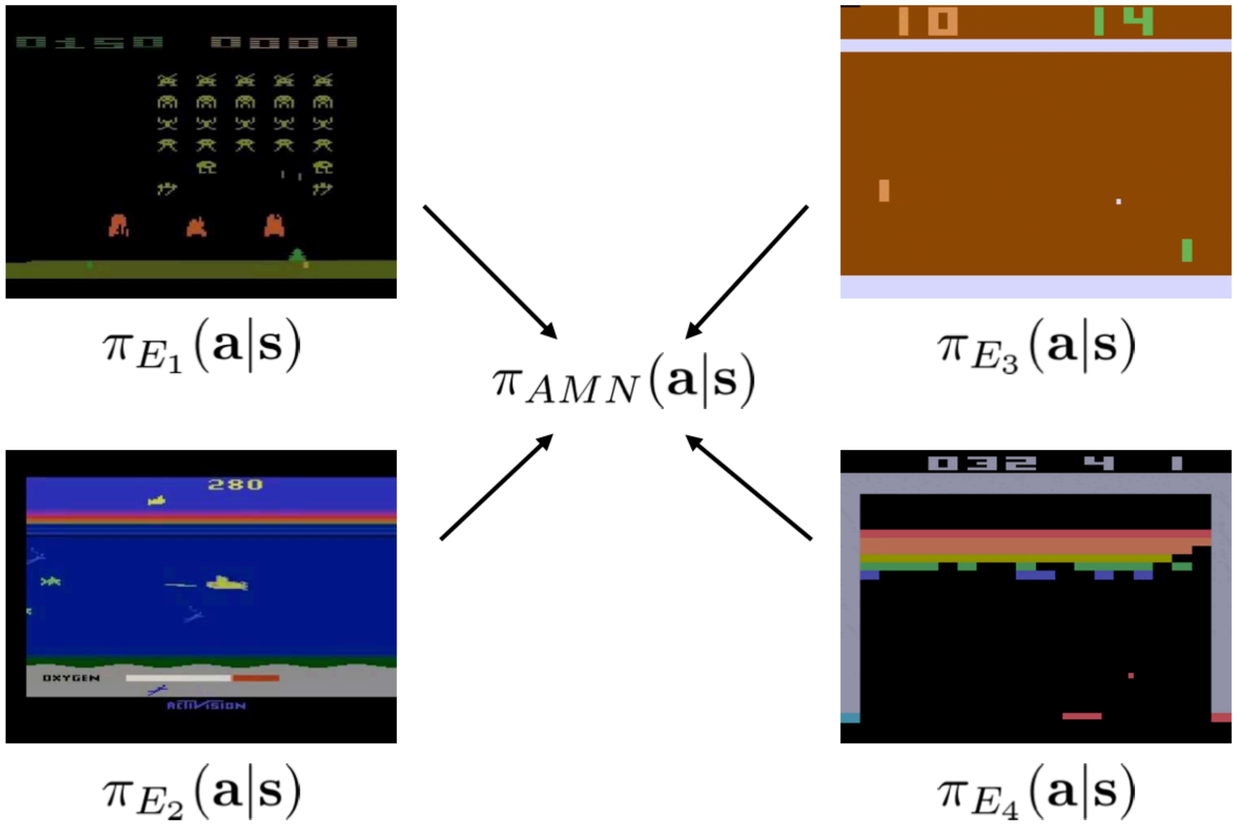
- Improve generalization via adequate randomness.
- When agent is robust in a wide range of environments, it would robust too in real world.
- Use widely in sim-to-real task

Overview

What if we have a set (or a distribution) of environments?



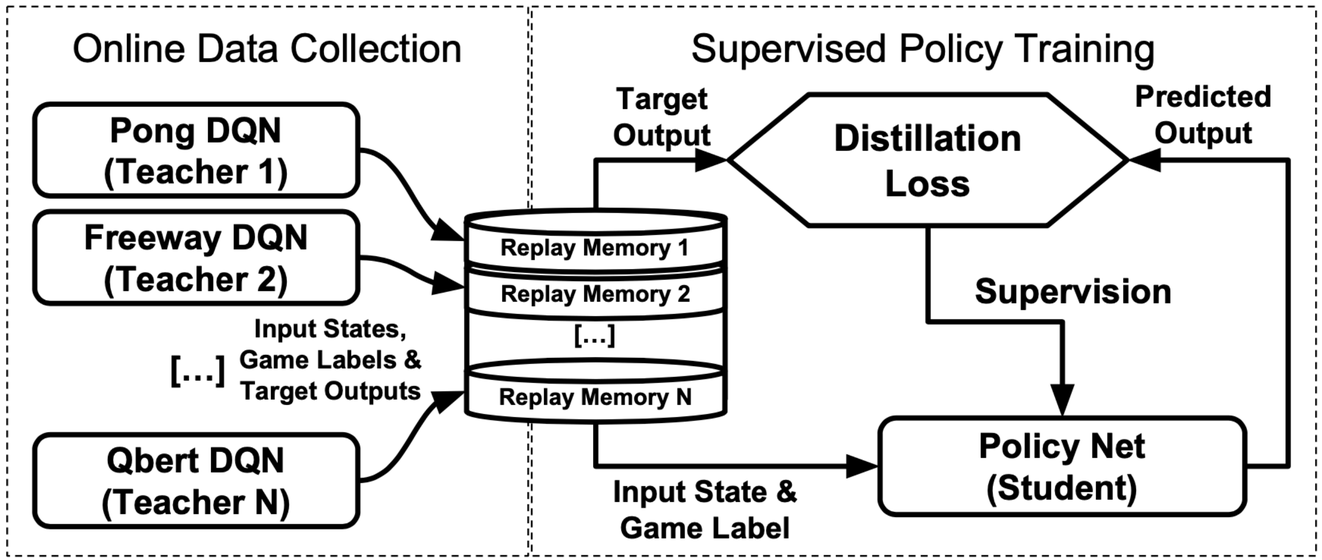
Multi-task Learning



$$\mathcal{L} = \sum_{\mathbf{a}} \pi_{E_i}(\mathbf{a}|\mathbf{s}) \log \pi_{AMN}(\mathbf{a}|\mathbf{s})$$

- Wish to train a generalist in multiple tasks
- Train experts in separated task
- Collect transitions in each task and supervised learn an agent

Multi-task Learning



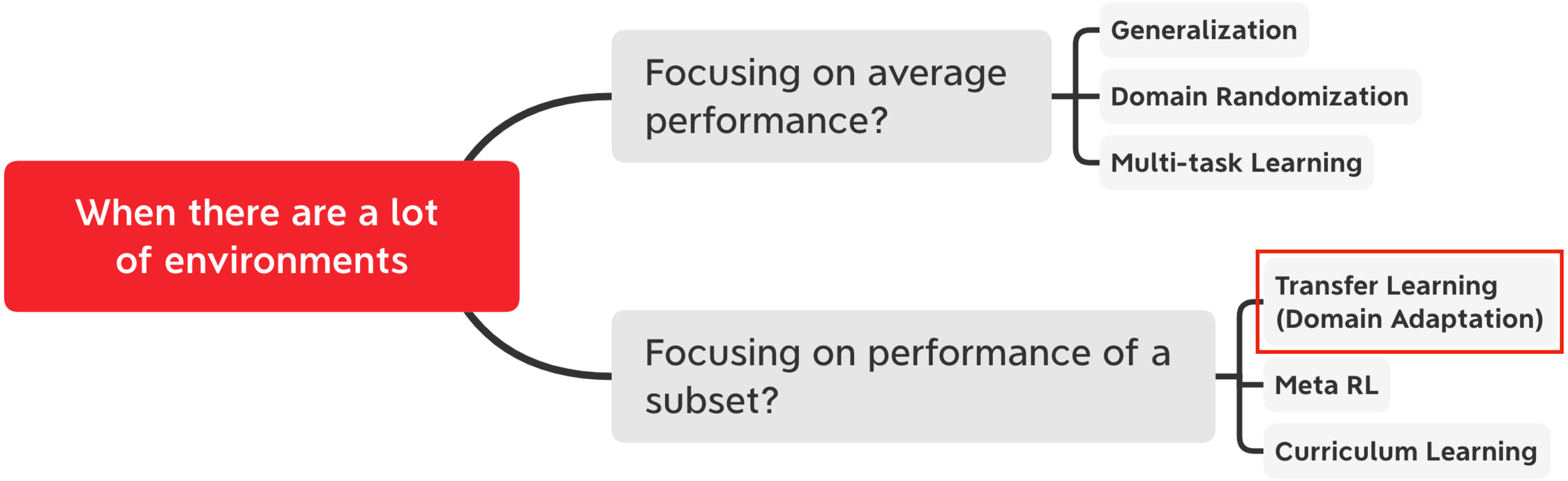
$$L_{KL}(\mathcal{D}^T, \theta_S) = \sum_{i=1}^{|D|} \text{softmax}(\frac{\mathbf{q}_i^T}{\tau}) \ln \frac{\text{softmax}(\frac{\mathbf{q}_i^T}{\tau})}{\text{softmax}(\mathbf{q}_i^S)}$$

- Wish to train a generalist in multiple tasks
- Train experts in separated task
- Collect transitions in each task and supervised learn an agent

Both are at ICLR 2016!

Overview

What if we have a set (or a distribution) of environments?



Transfer Learning (Domain Adaptation)

- Multi-task / Domain Randomization:
  - Learn a batch of tasks, solve arbitrary tasks.
  - Maximize average performance.

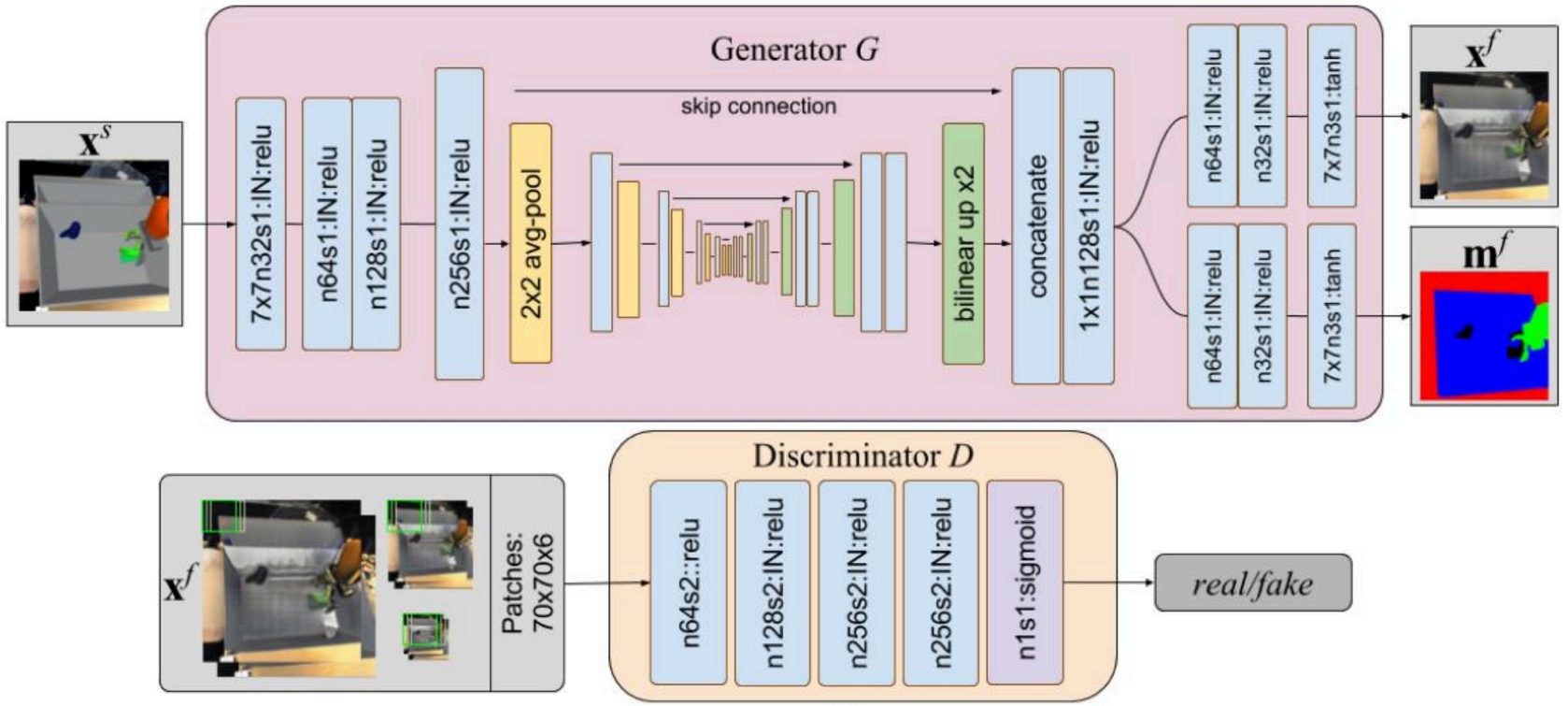
- Transfer Learning / Domain Adaptation:
  - Learn a batch of training tasks, then learn test tasks.
  - Maximize test tasks performance.

Transfer Learning (Domain Adaptation)

- Fine-tuning is OK.
- Sometime the agent becomes deterministic (pure exploitation), so we need to encourage exploration in new tasks.
- Use maximum-entropy policies:
  - $\arg\max_{\pi} \sum_{t} \{\mathbb{E}_{\pi}[r(s_t, a_t)] + \mathbb{E}_{\pi}[H(\pi(a_t \mid s_t))]\}$

Transfer Learning (Domain Adaptation)

Assumption: Good feature can generate realistic image, as well as guiding RL training.



- Use a generator to generate "domain adapted image"
- Use a classifier (discriminator) to distinguish real image and generated image
- Use the latent generator feature to train policy

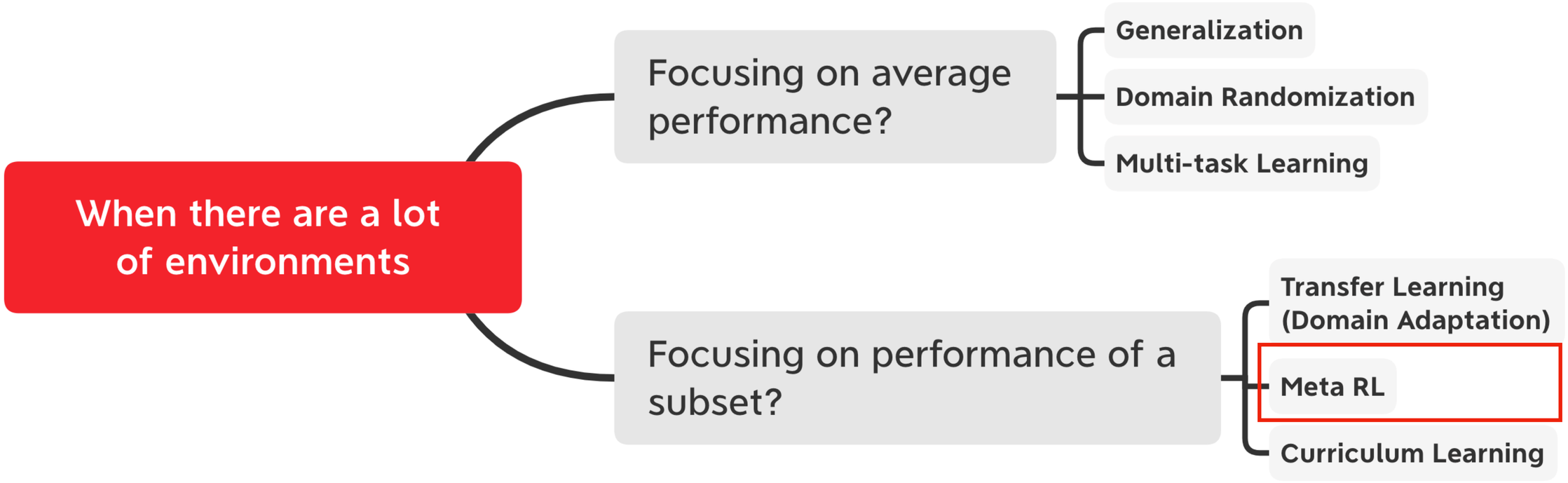
Transfer Learning (Domain Adaptation)

- Multi-task / Domain Randomization:
  - Let agent consumes diverse data!

- Transfer Learning / Domain Adaptation:
  - Let agent consumes data that fits the target domain!

Overview

What if we have a set (or a distribution) of environments?

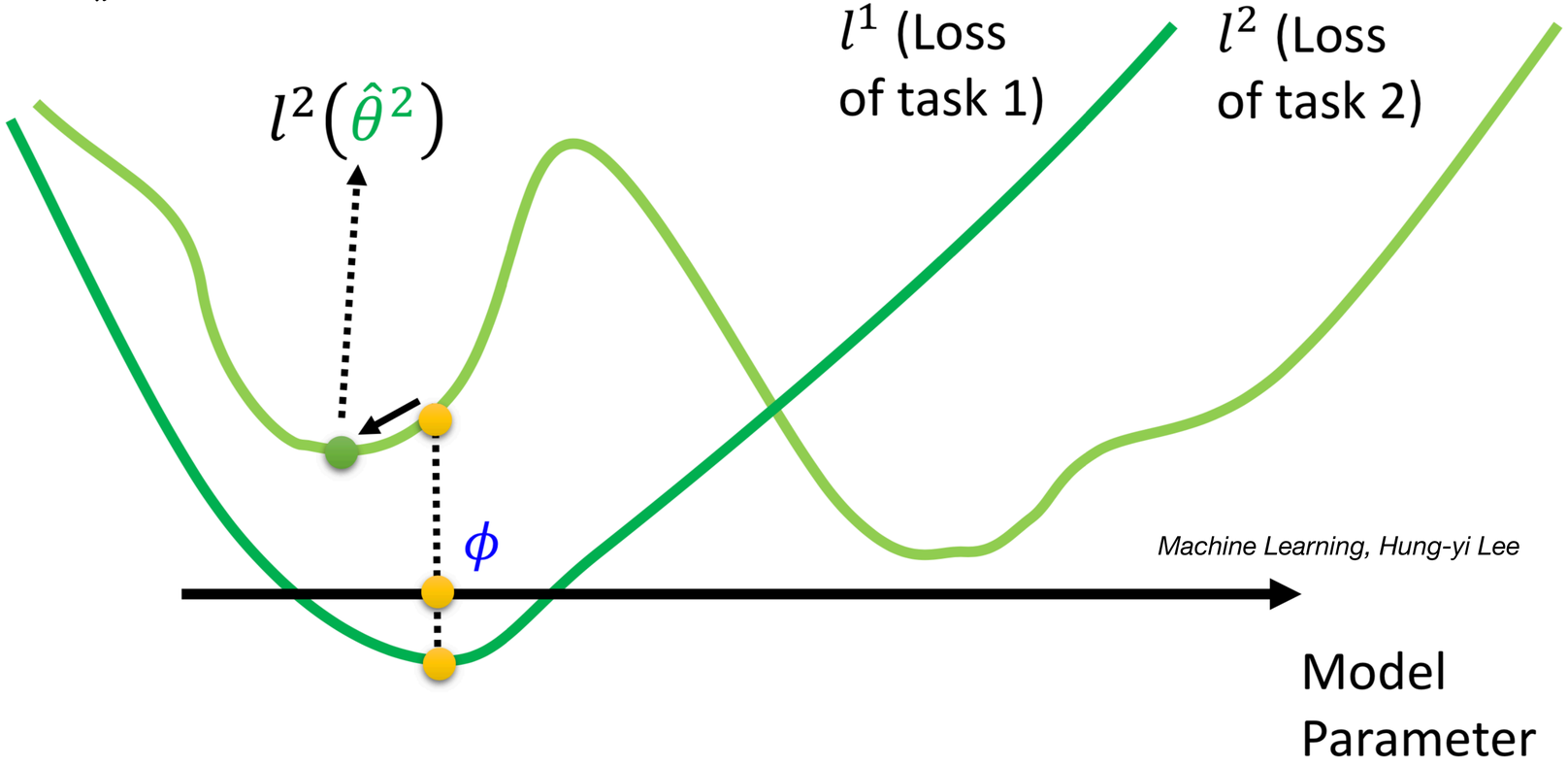


Meta RL

- Multi-task Learning maximizes average performance
  - $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_\pi} R(\tau_\pi)$
- Transfer Learning maximizes test tasks performance
  - $\arg\max_{\pi} \mathbb{E}_{test\ env} \mathbb{E}_{\tau_\pi} R(\tau_\pi)$
- Meta Learning finds the agent with highest **potential** performance
  - $\arg\max_{\pi} \mathbb{E}_{test\ env} \mathbb{E}_{\tau_{\pi^*}} R(\tau_{\pi^*})$, wherein π^* is the trained π

Difference between meta learning and multi-task learning

- Multi-task Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_\pi} R(\tau_\pi)$
- Meta Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_{\pi^*}} R(\tau_{\pi^*})$

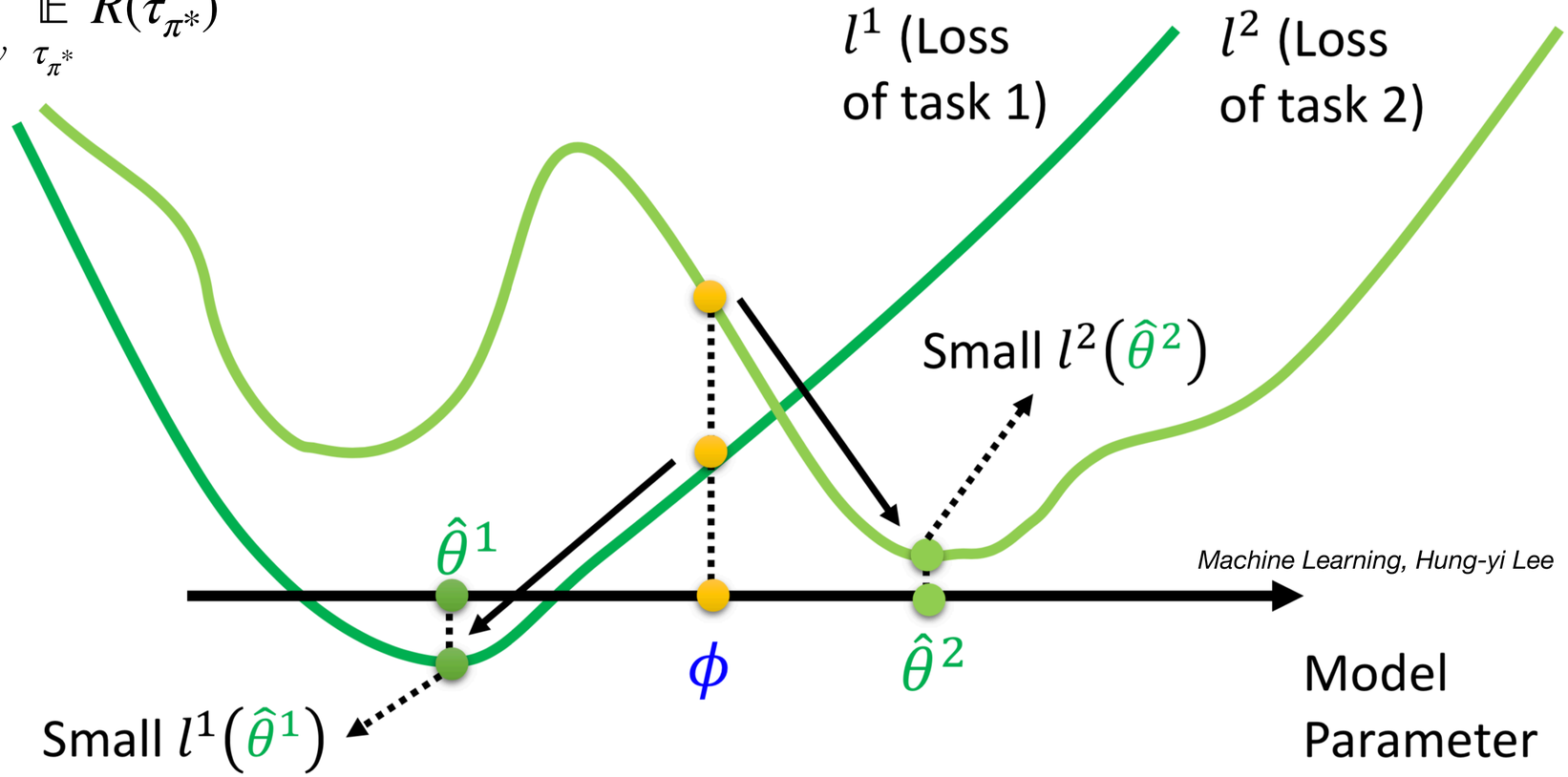


Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Difference between meta learning and multi-task learning

- Multi-task Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_\pi} R(\tau_\pi)$
- Meta Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_{\pi^*}} R(\tau_{\pi^*})$

Quickly Adaptation!

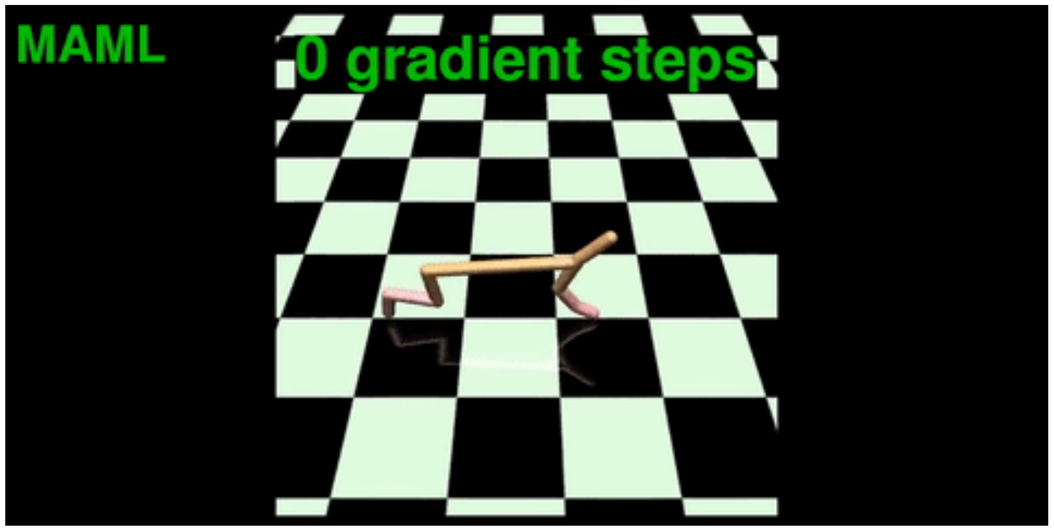


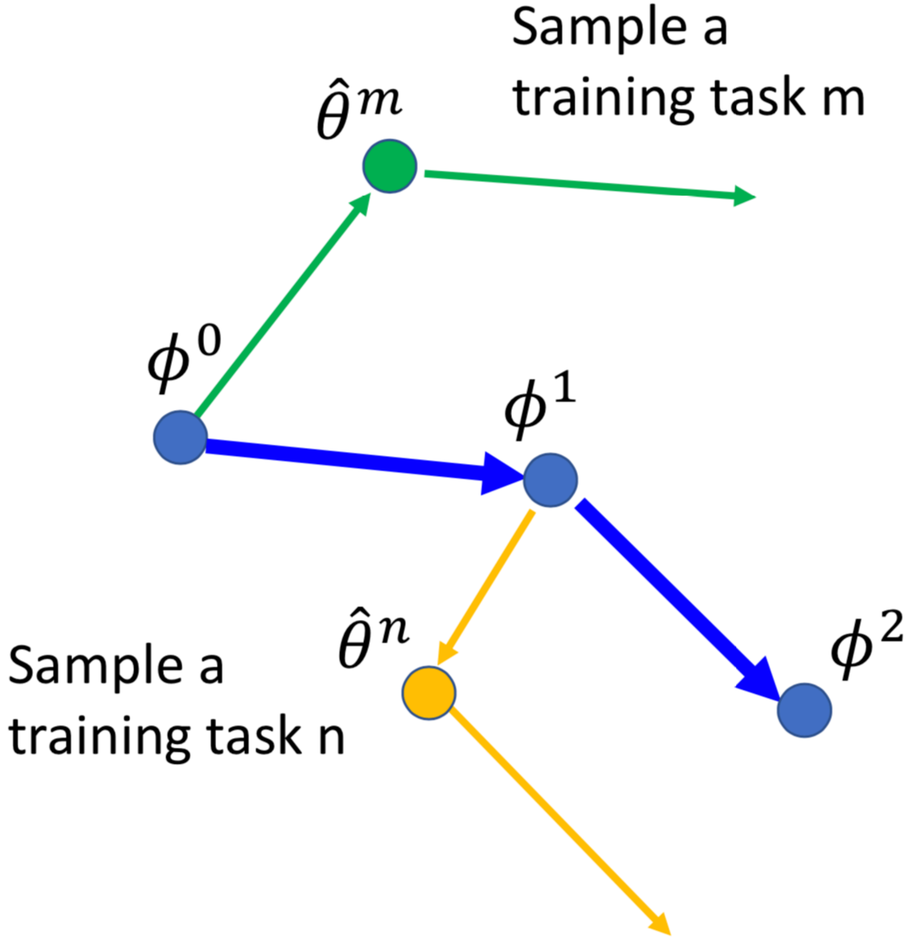
Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks

Meta RL

- Multi-task Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_\pi} R(\tau_\pi)$
- Meta Learning: $\arg\max_{\pi} \mathbb{E}_{env} \mathbb{E}_{\tau_{\pi^*}} R(\tau_{\pi^*})$

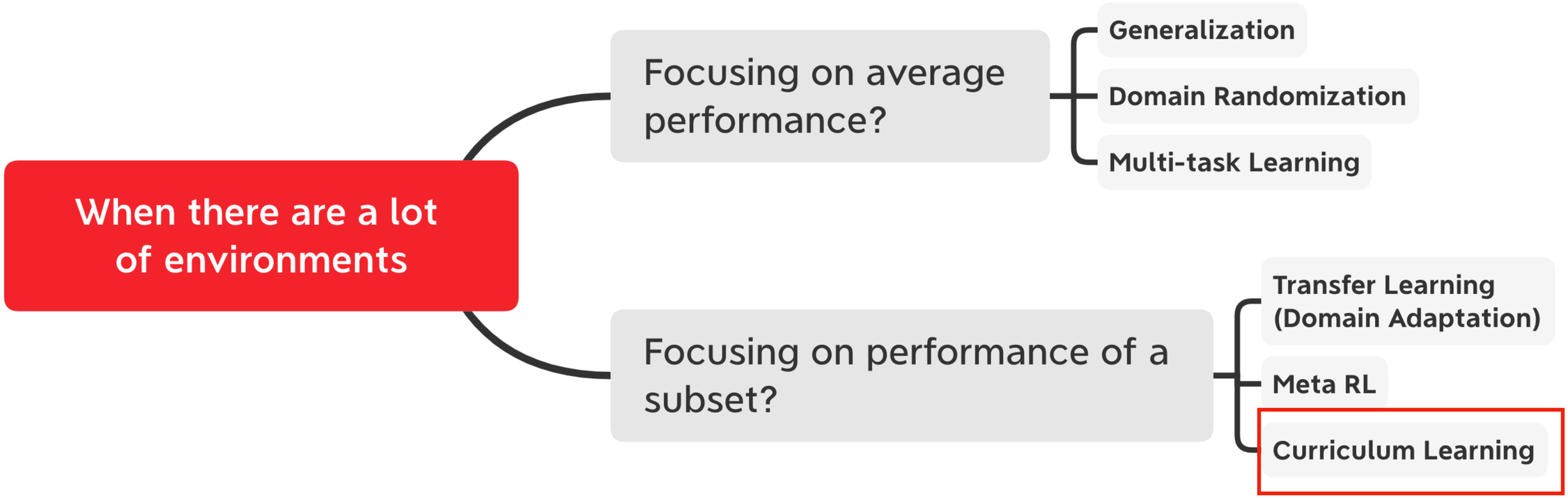
Quickly Adaptation!





Overview

What if we have a set (or a distribution) of environments?

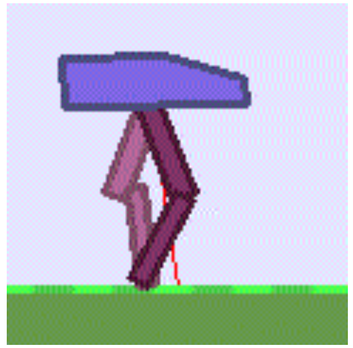


Curriculum Learning

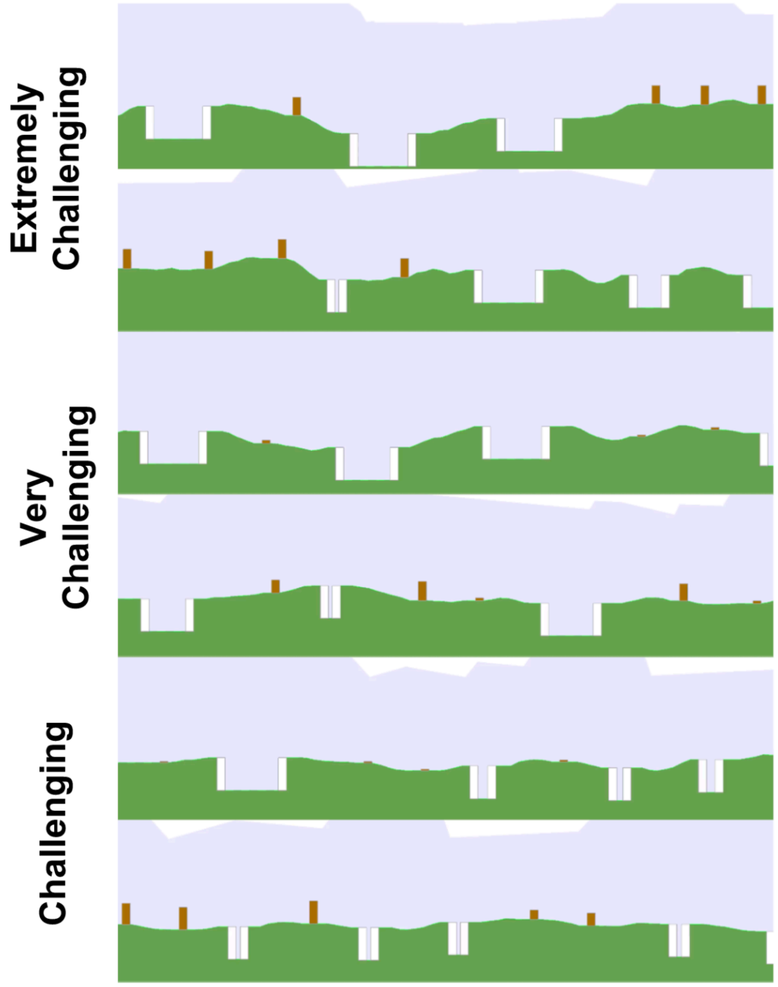
- We, as human beings, learn through solving easy tasks then hard tasks.
- *Curriculum Learning* provides a series of tasks with different difficulties.

Curriculum Learning

BipedalWalker-v3

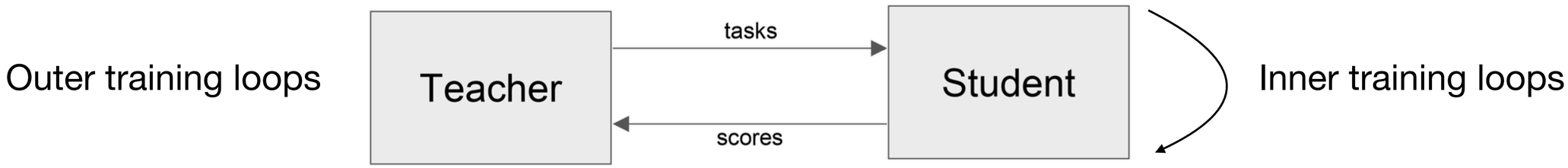


- Key Points:
  - How to define the curriculum (the difficulty of tasks)?
  - How to define the mastery (when to switch tasks)?
  - How to avoid catastrophic forgetting and training instability (how to mix tasks)?



Paired Open-Ended Trailblazer (POET)

Teacher-Student Curriculum Learning

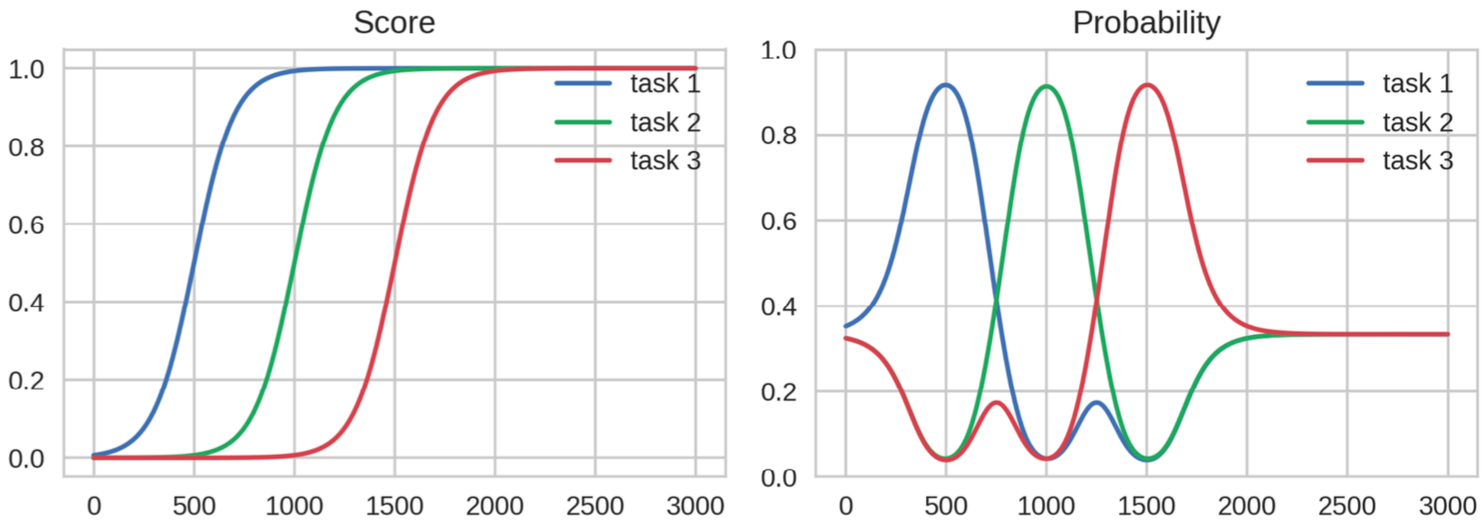


- Student: Arbitrary RL algorithms, learns from given tasks
- Teacher: A policy that provides tasks to student based on its performance
  - **What is the objective?** Maximize the performance of all tasks.
  - **How to define its input, performance?**
  - **How to define its output, tasks?**
  - **How to train?**

Teacher-Student Curriculum Learning

- For **reinforcement learning**, suppose there are **N possible tasks** and only **1 task is training** for student at each "Teacher time step".
- Teacher is an RL agent:
  - **observation**: episode reward of student in all running tasks $o_t = (r_t^{(1)}, \ldots, r_t^{(N)})$
  - **action**: the probability of each task $a_t = (p_t^{(1)}, \ldots, p_t^{(N)})$
  - **reward**: the change of reward of current task: $x_t^{(i)} - x_{t'}^{(i)}$, t' denotes the last step that (i) task is sampled

Teacher-Student Curriculum Learning



Ideal curves:
the tasks gradually be solved
the probability of tasks change over time.

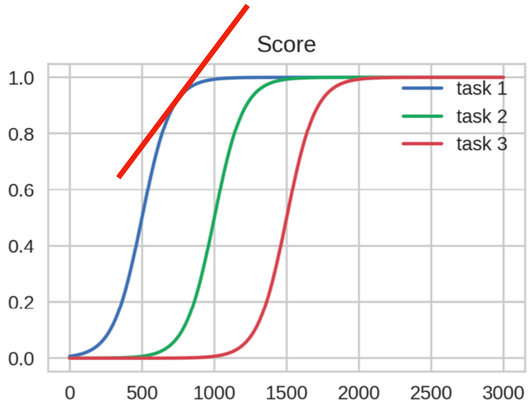
But the reward is noisy!

- Issues:
  - The reward is noisy
  - We need to balance the exploration and exploitation of Teacher! We need to try new tasks.

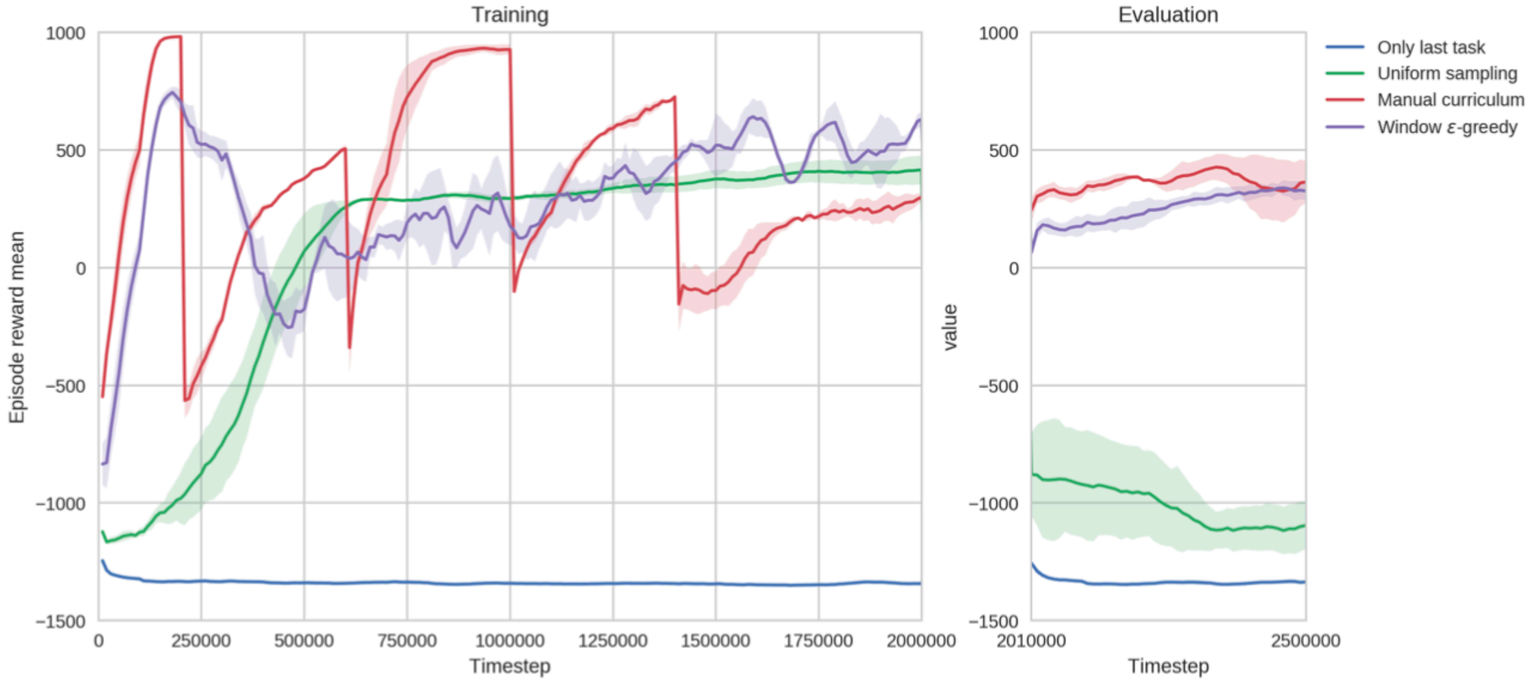
Teacher-Student Curriculum Learning

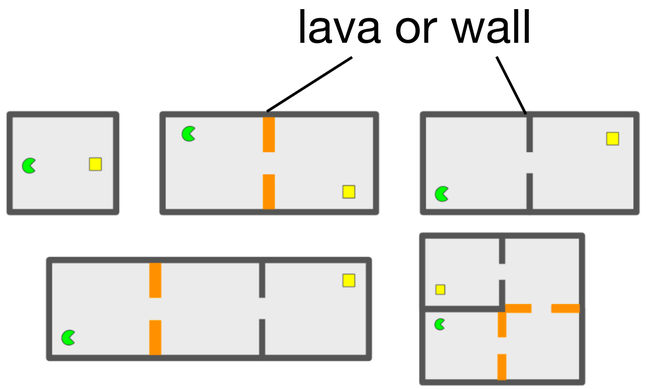
- How to estimate the learning progress? (Note: r_t denotes the "teacher reward", not "student reward")
  - **Online algorithm**: Update Q function with $Q_{t+1}(a_t) = \alpha r_t + (1-\alpha)Q_t(a_t)$, use ϵ-greedy to explore.
  - **Naive algorithm**: Train student for each task K times, compute the average performance improvement.
  - **Window algorithm**: Use a FIFO buffer to store the performance and the time step of each task, compute the average performance improvement.
  - **Sampling algorithm**: For each task, randomly choose a reward from the last K rewards, and choose the task with highest teacher reward.

We want to know the slope!



Teacher-Student Curriculum Learning

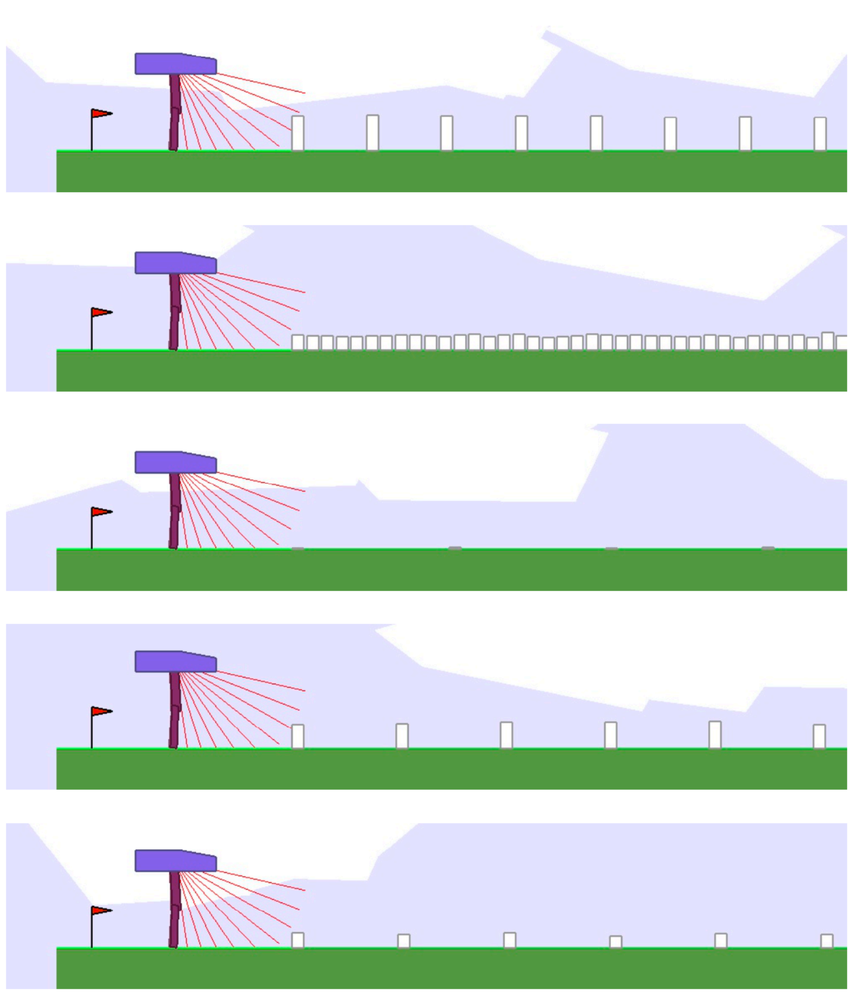


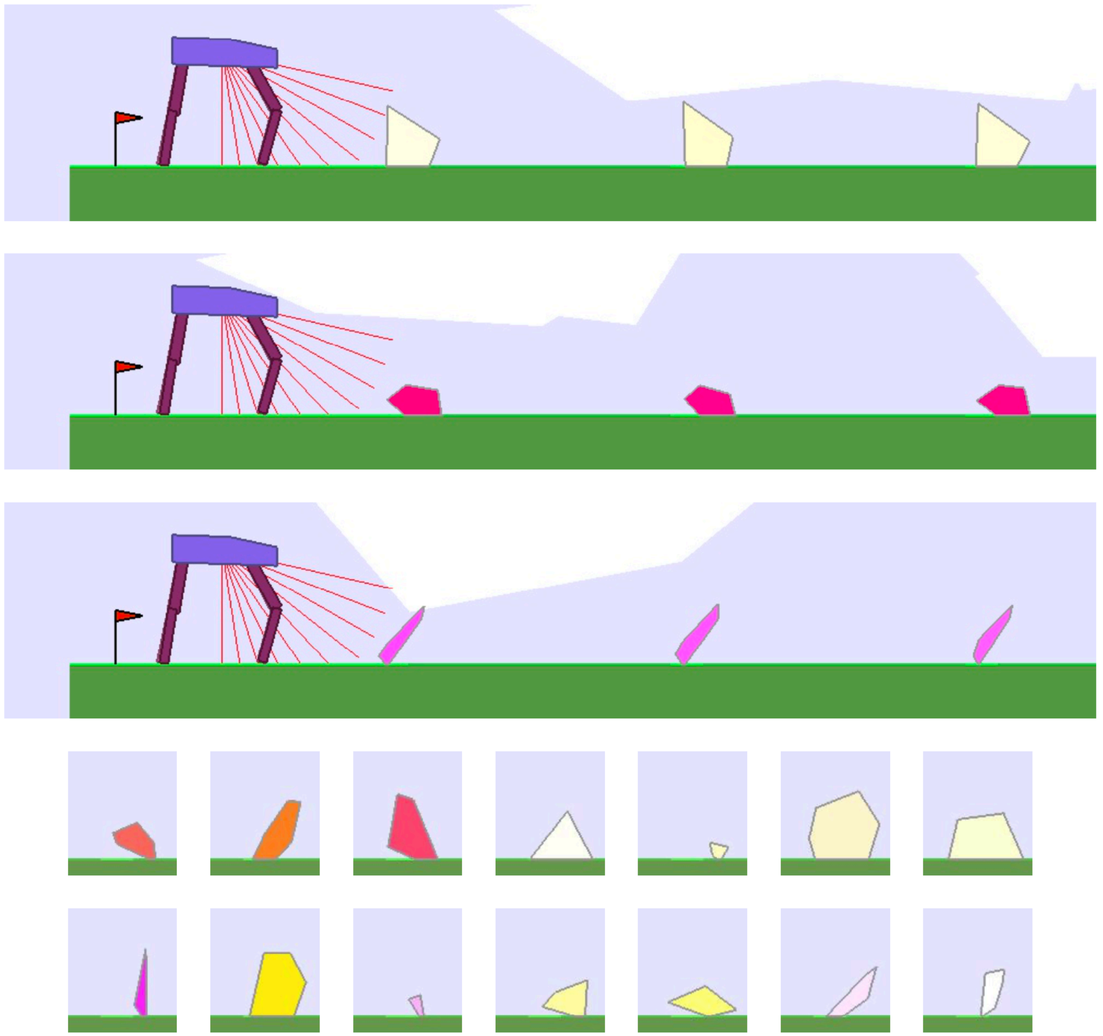


5-step curriculum

- **Window method + Epsilon greedy exploration** can match **manual curriculum**, and much better than training on **only last task**.

Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments



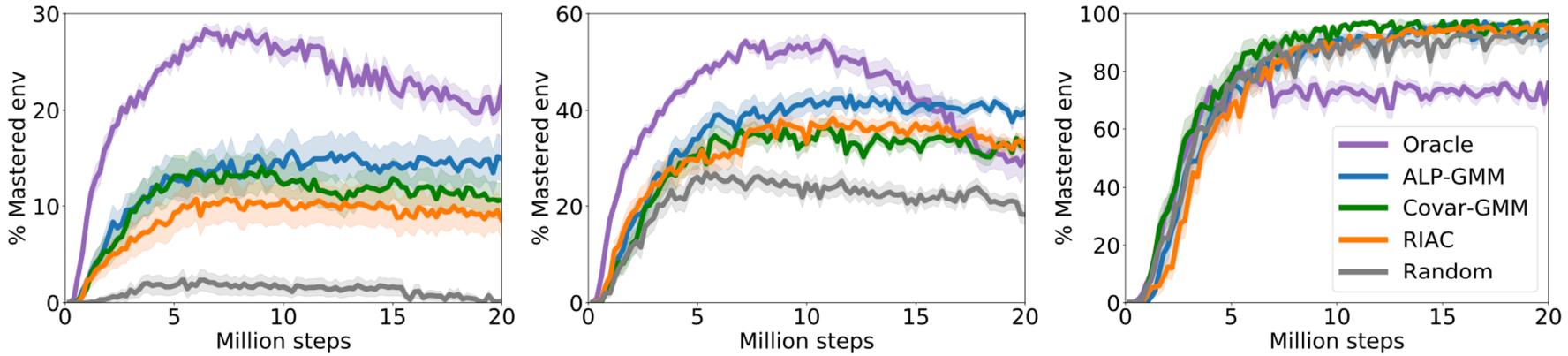


Procedurally Generated Bipedal Walker

Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments

- Environment parameter space is continuous, not only N choices.
- Teacher is an Gaussian Mixture Model, not a Q function.
- **Absolute Learning Progress (ALP)**
  - ALP = $|r_{new} - r_{old}|$
  - Use ALP as the utility function to fit GMM

Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments



(a) Short agents (b) Default agents (c) Quadrupedal agents

- **ALP-GMM** method matches **oracle** performance and outperform **RIAC** algorithm

Overview

What if we have a set (or a distribution) of environments?