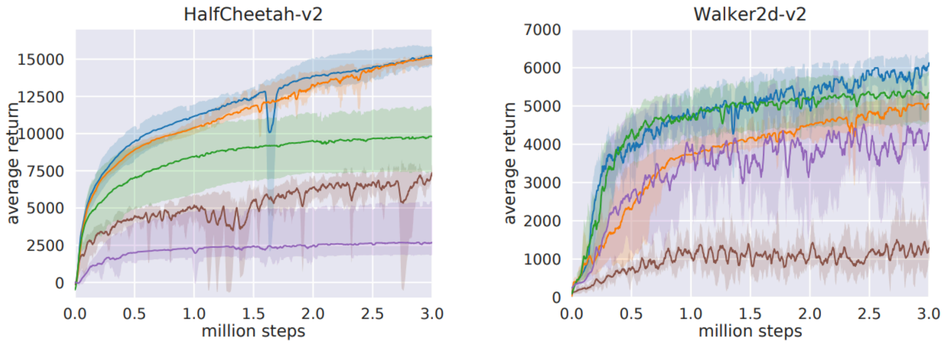
Model-based RL / Meta-RL

Until now…

- We learn value function methods and policy gradient methods.
  - Value function methods focus on estimating the values of states.
  - Policy gradient methods focus on improving the current policy.
- Q. Which methods show stable, monotonic improvements? Policy Gradient methods
- Q. Which methods show better sample efficiency? Value Function methods

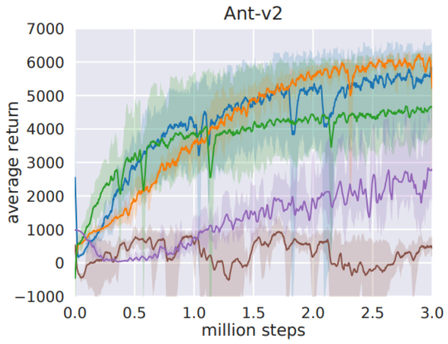
Motivation

- How can we achieve a lot better sample efficiency?

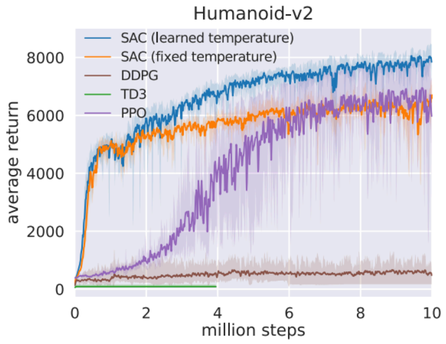


(a) HalfCheetah

(b) Walker2d



(c) Ant

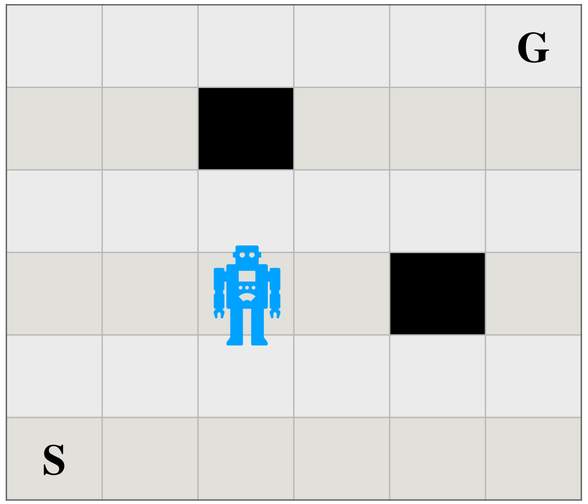


(d) Humanoid

Haarnoja et al. “Learning to Walk via Deep Reinforcement Learning”, 2020

Idea #1: Learning the dynamics

- Model-based RL tries to understand the dynamic (=transition function) and leverage it for learning.
  - One analogy is a “mental practice”.
  - Seems intuitive in the grid world. How about robotics?



Idea #2: Learning to learn

- If we experienced similar task before, we will be able to learn a lot faster.

Quiz

- Can you give the example of model-based RL / meta-RL in your favorite sports?

Model-based RL

Simple RL Algorithm

- While not converge:
  - Run a policy $\pi_\theta(a_t, s_t)$ to collect the data $\mathcal{D} = \{(s, a, s')_i\}$
  - Update the policy parameters θ

The policy can be an explicit neural network, or implicitly defined by a value function.

Extension to Model-based RL

- Run a random policy $\pi_\theta(a_t, s_t)$ to collect the initial data $\mathscr{D} = \{(s, a, s')_i\}$
- While not converge:
  - Learn a transition model $f(s, a)$ to minimize $\sum |f(s_i, a_i) - s_i'|^2$
  - Plan actions using the learned model f.
  - Execute those actions and add the data $\mathscr{D} = \{(s, a, s')_i\}$

Extension to Model-based RL

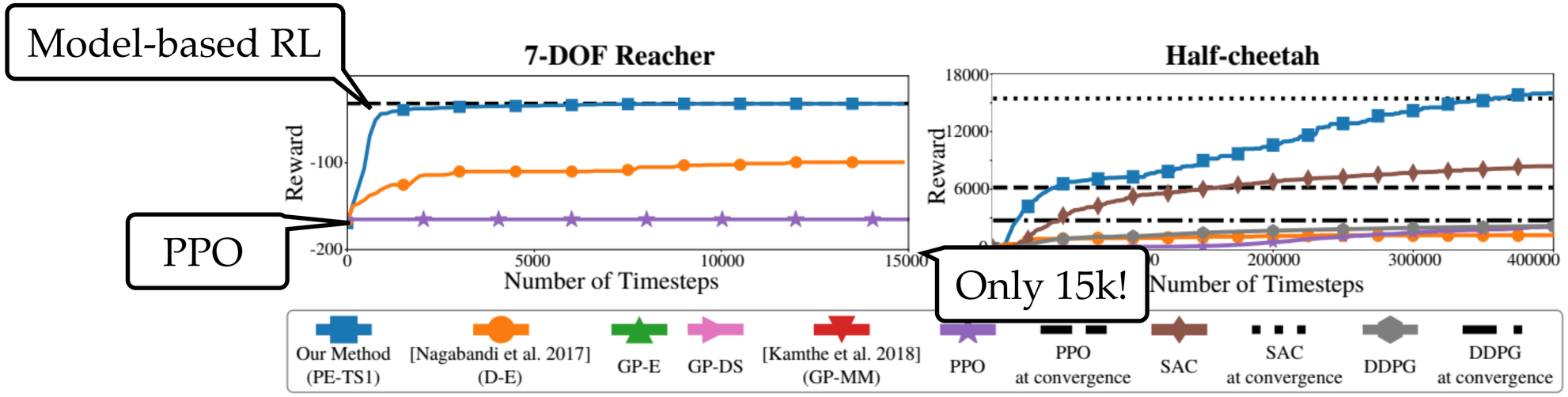
- Run a random policy $\pi_\theta(a_t, s_t)$ to collect the initial data $\mathcal{D} = \{(s, a, s')_i\}$
- While not converge:
  - Learn a transition model $f(s, a)$ to minimize $\sum |f(s_i, a_i) - s_i'|^2$
  - Plan actions using the learned model f.
  - Execute those actions and add the data $\mathcal{D} = \{(s, a, s')_i\}$

Will it work?

- Yes! We can view this as the expressive version of classical system identification.

- Achieves a great sample efficiency.

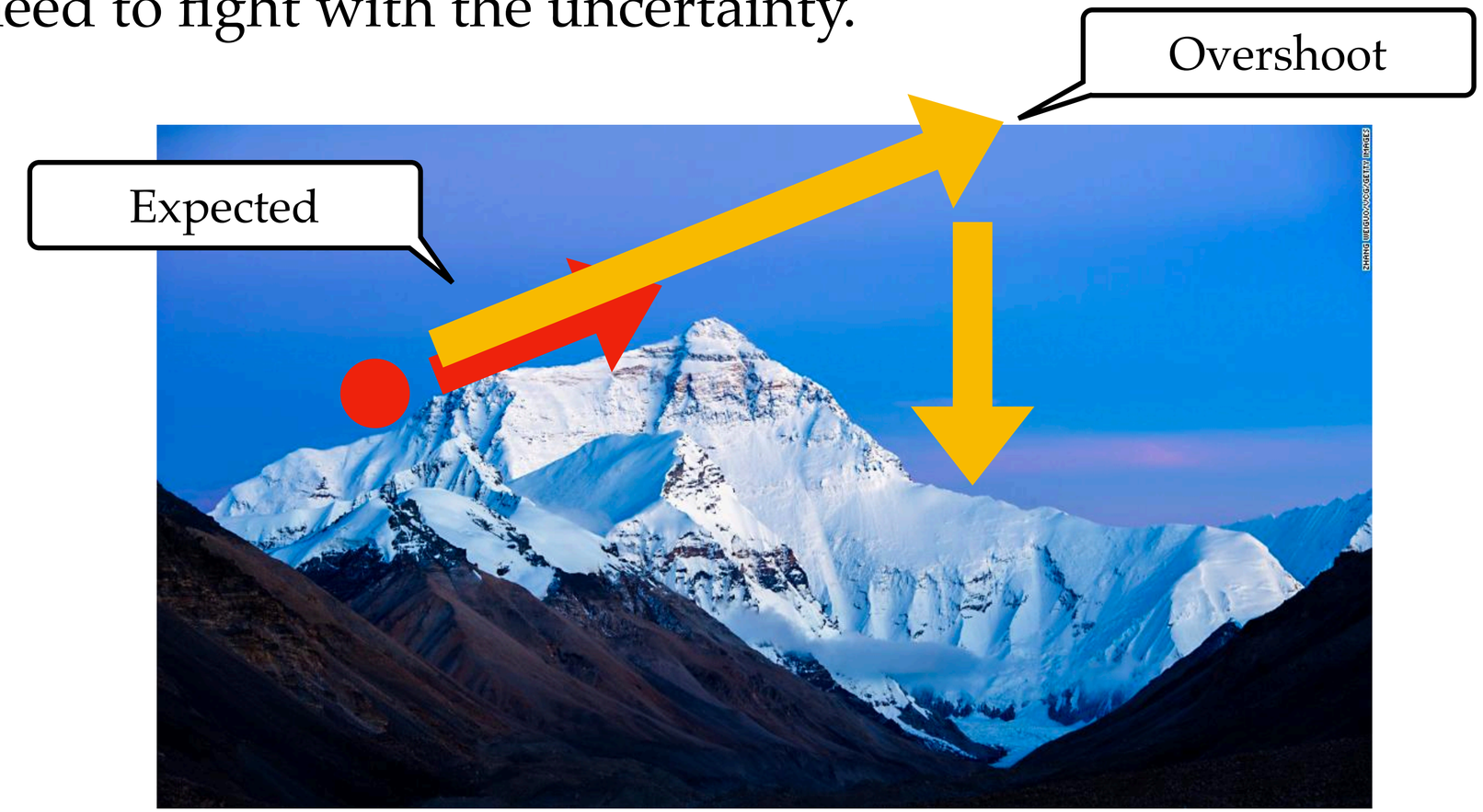
Identify system parameters from real-world data, such as mass and inertia.



Chua et al. “Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models”. 2018

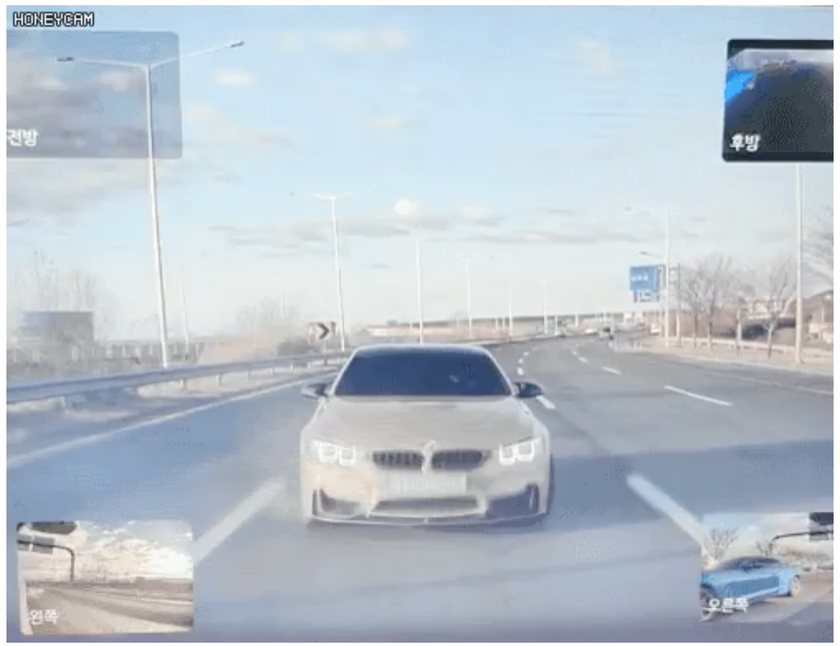
Will it work?

- No! The learned model can guess wrong and make a wrong plan.
  - We need to fight with the uncertainty.



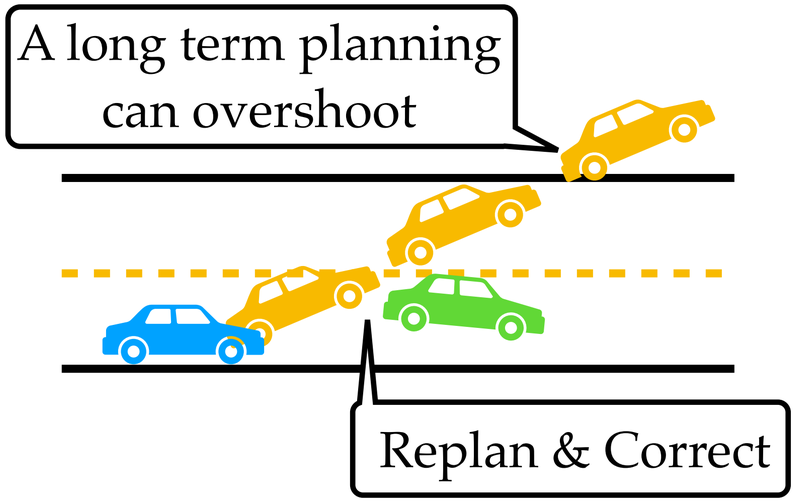
Key Challenge: Uncertainty

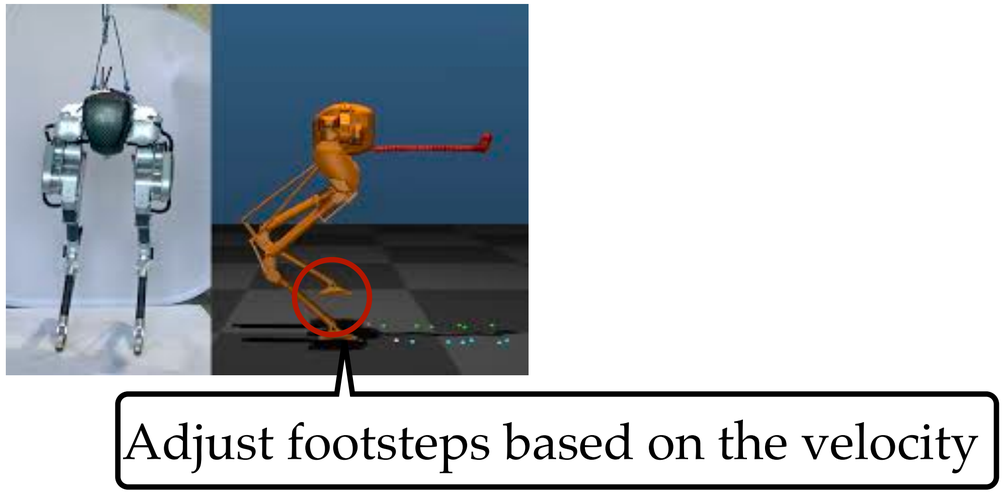
- Model-error is inevitable and causes uncertainty.
- If it starts to see some new states, the error will quickly be accumulated over time and exacerbated.



Model-predictive Control

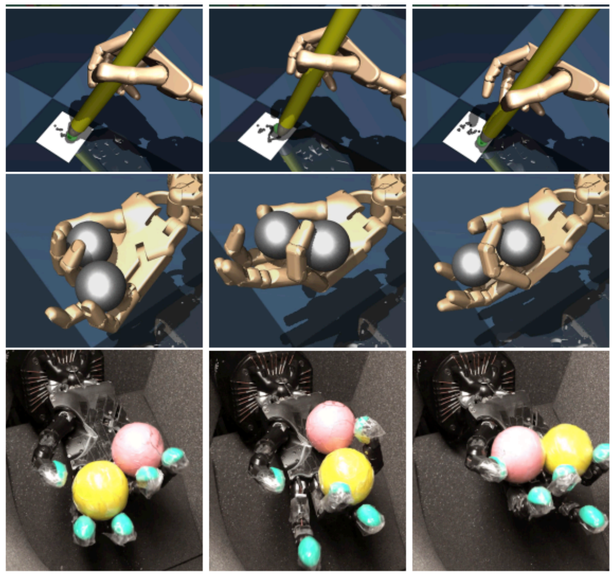
- While not converge:
  - Learning a transition model $f(s, a)$ to minimize $\sum |f(s_i, a_i) - s_i'|^2$
  - Plan actions using the learned model f.
  - Execute the "first" action and add (s, a, s') to the data $\mathcal{D}$.





Ensemble Approach

- We can train a set of models to reduce the variance.
  - Before: $J(\theta) = \Sigma_{t=1}^{T} r(s_t, a_t)$ where $s_{t+1} = f(s_t, a_t)$
  - After: $J(\theta) = \Sigma_{i=1}^{N} \Sigma_{t=1}^{T} r_i(s_t, a_t)$ where $s_{t+1,i} = f_i(s_{t,i}, a_t)$



Nagabandi et al. “Deep Dynamics Models for Learning Dexterous Manipulation”, 2019

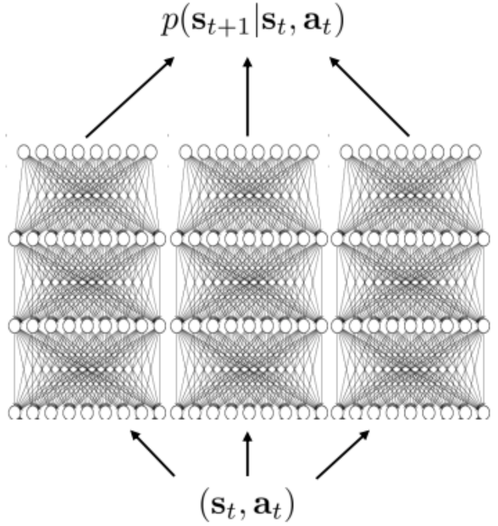


Image credit to Prof. Levine at UCB

Shorter Horizon Rollouts

- A longer horizon can be also problematic.

- We generate a lot of shorter trajectories instead of a longer one.

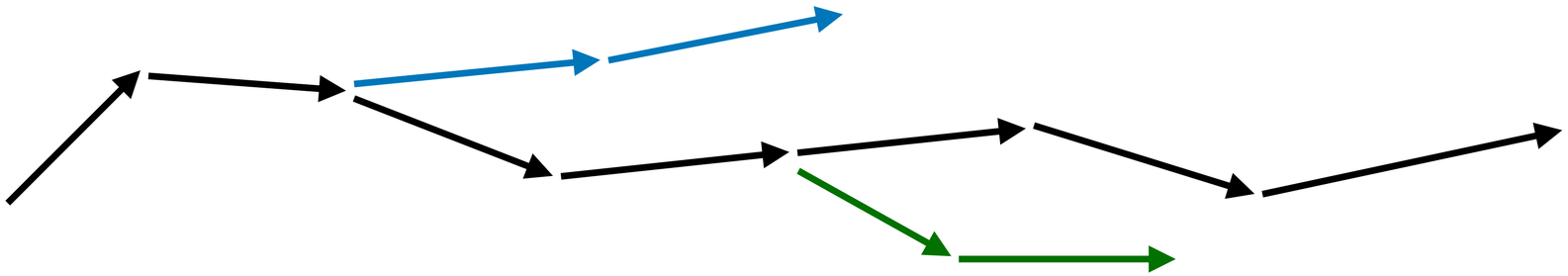
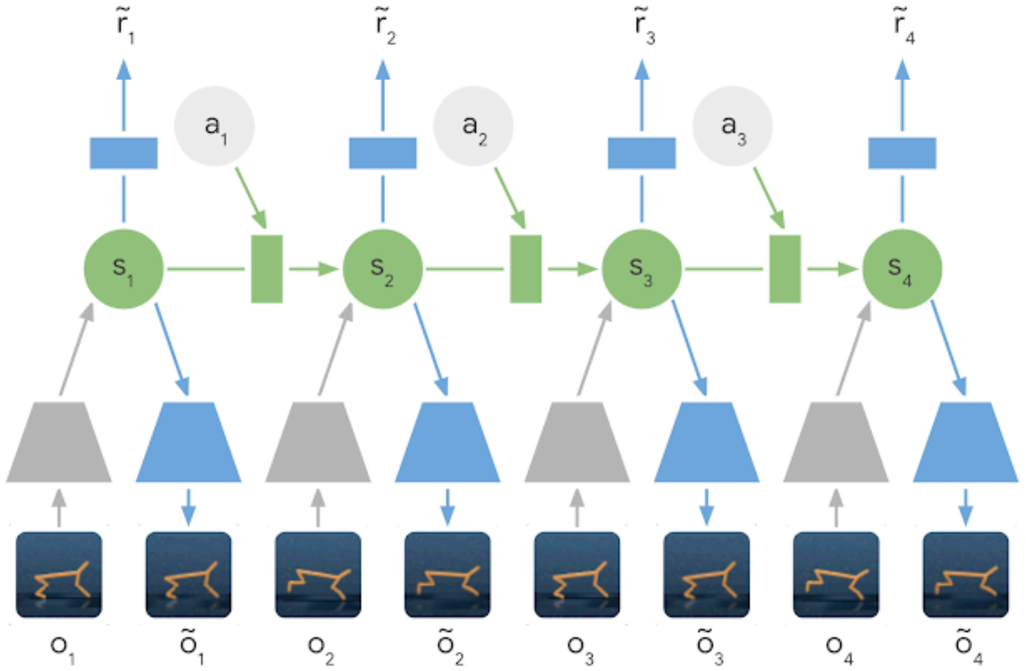


Image-based Model

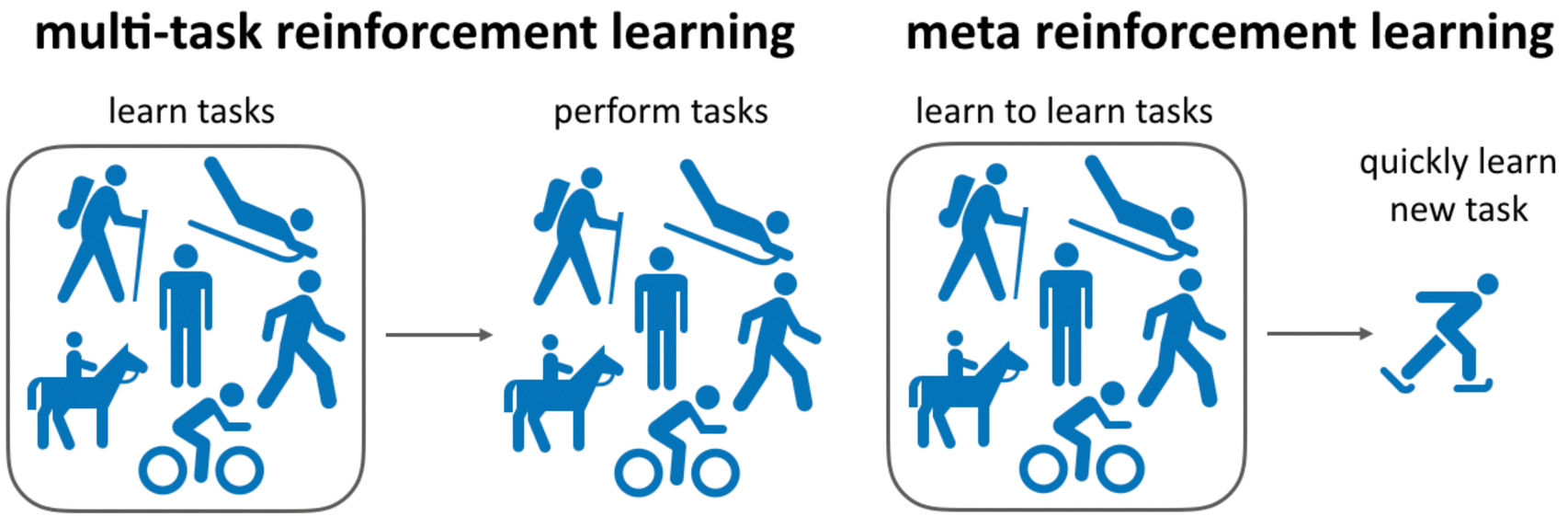
- We can magically learn a model in an image space.
  - … by modeling a compact latent space dynamics.



Meta-RL

Meta Learning

- If you learned similar tasks before, you can figure out how to learn faster.
- Meta learning = “Learning to learn”
- Related to multi-task learning. What is difference?

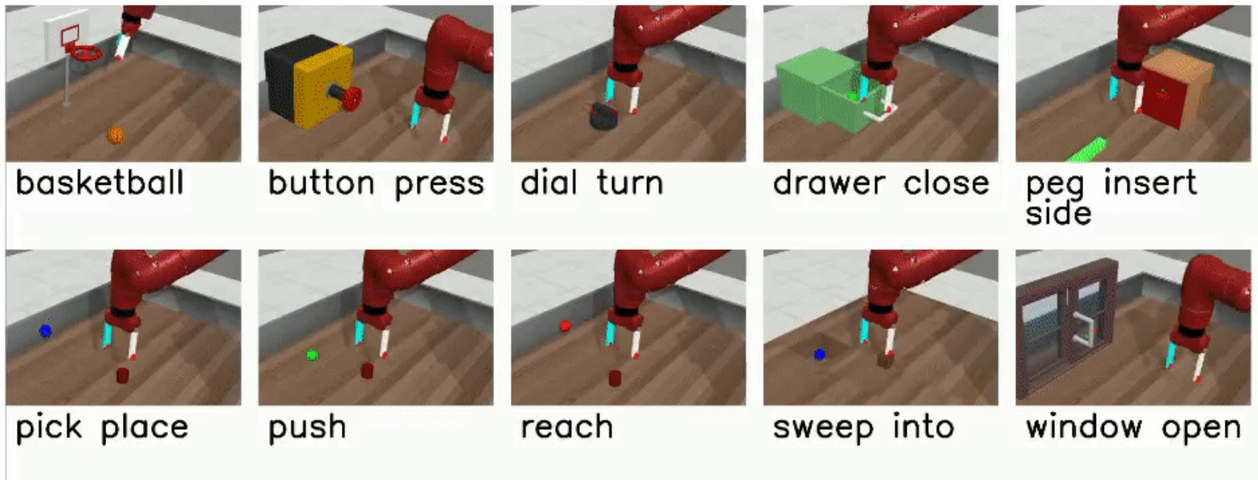


meta-world.github.io

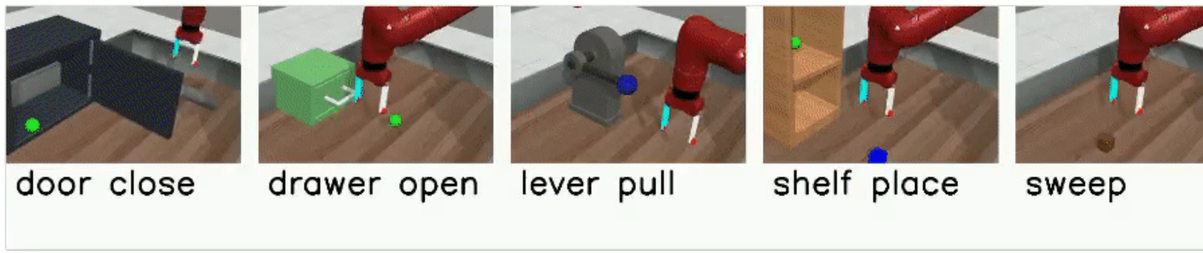
Brainstorming

- What information can shared across different tasks?

Train



Test



Meta-learning in Supervised Learning

- Supervised learning: $f(x) \mapsto y$
- Supervised meta-learning: $f(\mathscr{D}^{tr}, x) \mapsto y$

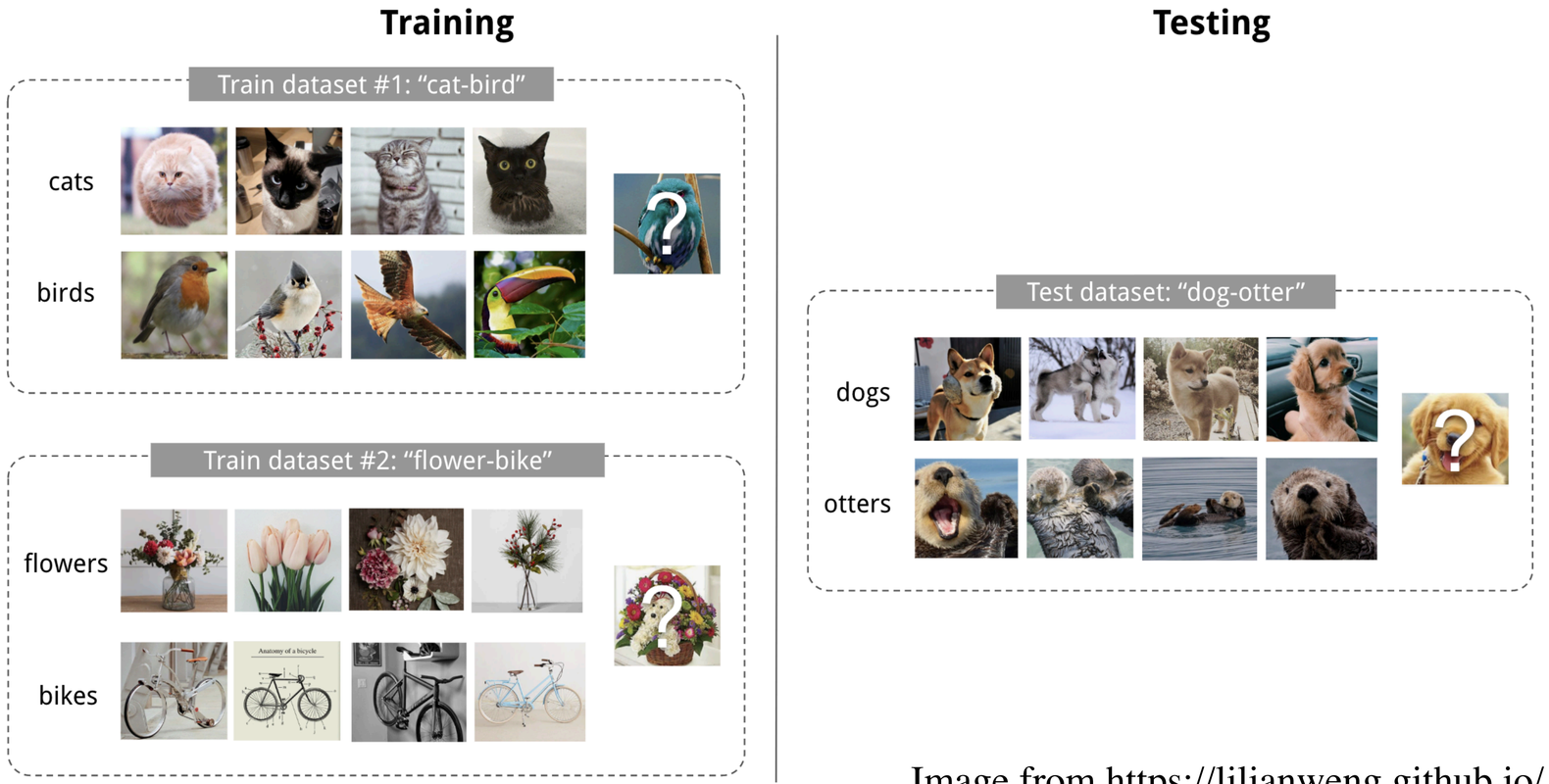


Image from https://lilianweng.github.io/

Formulation

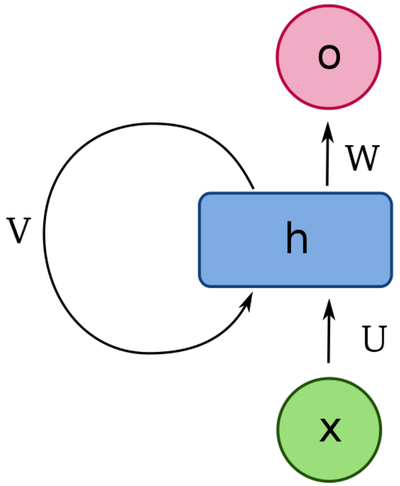
- Learning: $\theta^* = \arg\min L(\theta, D) = f_{learn}(D)$
- Meta-learning: $\theta^* = \arg\min \Sigma_i L(\phi_i, D_i)$ where $\phi_i = f_{learn}(\theta, D_i)$
- RL: $\theta^* = \arg\max E_\theta[R(\tau)] = f_{RL}(M)$ — MDP formulation
- Meta RL: $\theta^* = \arg\max \Sigma_i E_{\phi_i}[R(\tau)]$ where $\phi_i = f_{RL}(\theta, M_i)$ — MDP for task i

Quiz

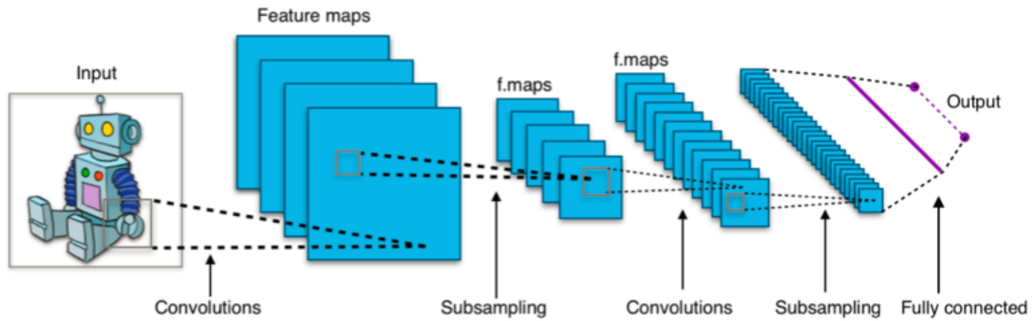
- Which architecture will be useful for meta learning?

  A. Recurrent Neural Network (memory)

  B. Convolutional Neural Network (spatial relation)



RNN



CNN

Approaches

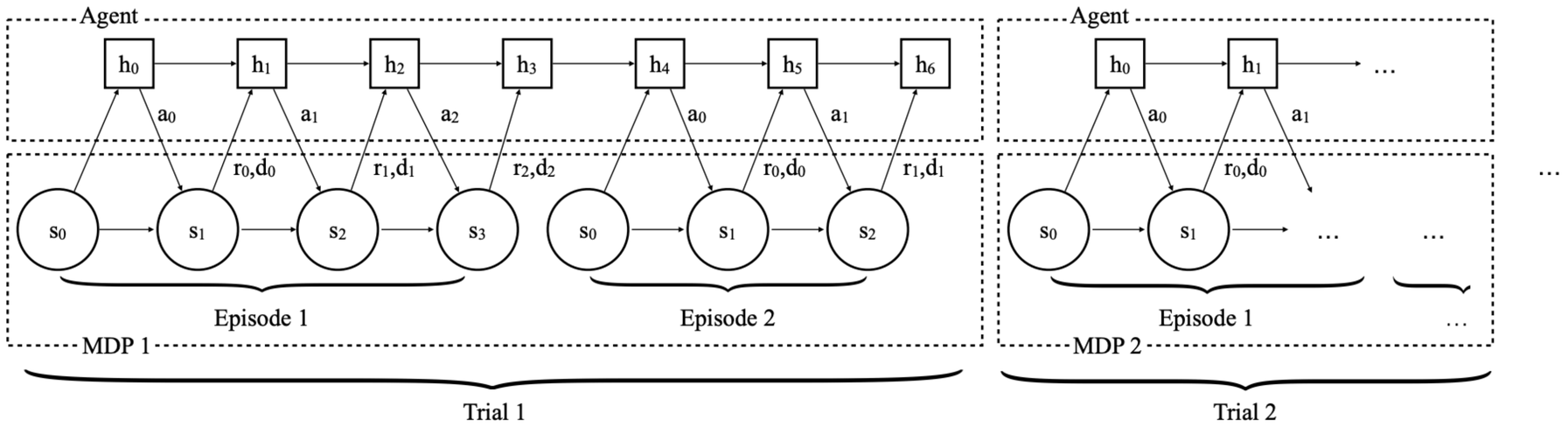
- Adaptation via Context Variables (RNN)
- Adaptation via Gradient Descent
- Adaptation via Simple Optimization

Meta-RL with recurrent policies

- Let's recall the formulation: what if f_{RL} infers "context" from the episodes?
  - RL: $\theta^* = \arg\max E_\theta[R(\tau)] = f_{RL}(M)$
  - Meta RL: $\theta^* = \arg\max \Sigma_i E_{\phi_i}[R(\tau)]$ where $\phi_i = f_{RL}(\theta, M_i)$
- We can model the "context" via recurrent neural network!
  - Note that the context is given in multi-task RL: ex) walking directions, target location, …

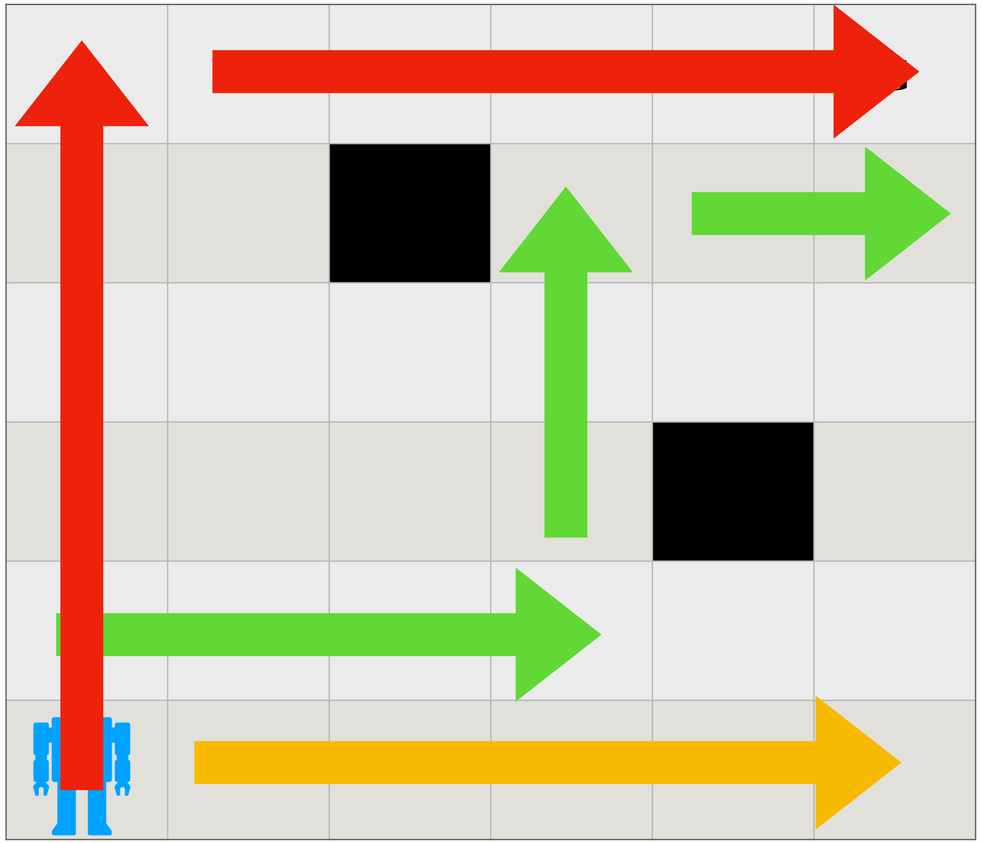
Meta-RL with recurrent policies

- You can simply train a recurrent policy on the given problems.
  - The key is not to reset between episodes: only reset between meta-episodes.



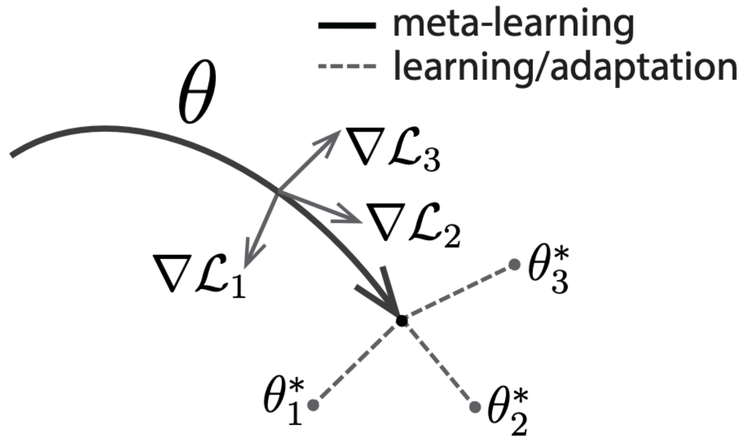
Duan et al. “RL2 : FAST REINFORCEMENT LEARNING VIA SLOW REINFORCEMENT LEARNING”, 2017

Example: Maze Navigation



Meta-RL with gradient descent

- Let's define the adaptation process f_{RL} as a single step of gradient.
  - $f_{RL}(M_i, \theta) = \theta + \alpha \nabla_\theta J_i(\theta)$
- We will put a policy parameter θ to a position that can be quickly adapt to different tasks within a single gradient step.



Finn et al. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks", 2017

Meta-RL with gradient descent

Algorithm 1 Model-Agnostic Meta-Learning

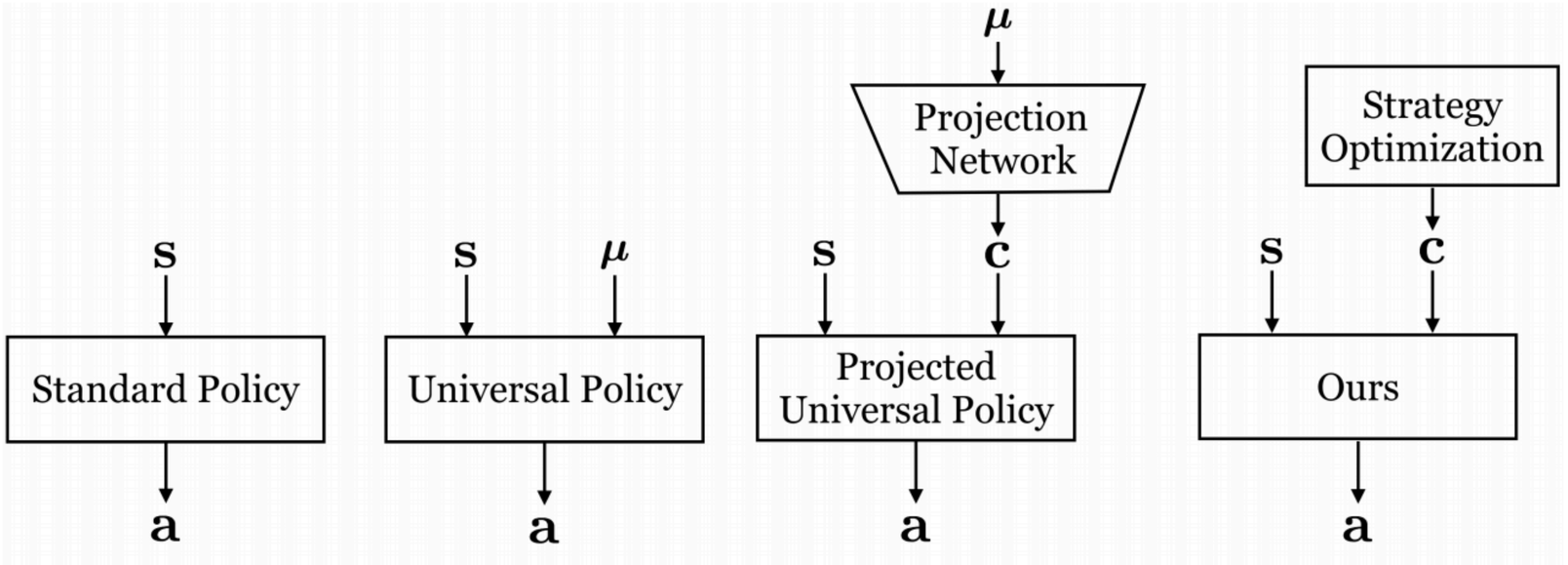
Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
1: randomly initialize θ
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4: **for all** $\mathcal{T}_i$ **do**
5: Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
6: Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
7: **end for**
8: Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
9: **end while**

Meta-RL with gradient descent

- A few notes on Model-agnostic Meta Learning (MAML)
  - MAML can be applied to both supervised and reinforcement learning.
  - MAML can be expensive: why? (Hint: f_{RL} already has a gradient)
    - MAML requires the computation of Hessian.
    - Supported by Tensorflow / you can approximate it.

Meta-learning with Simple Optimization

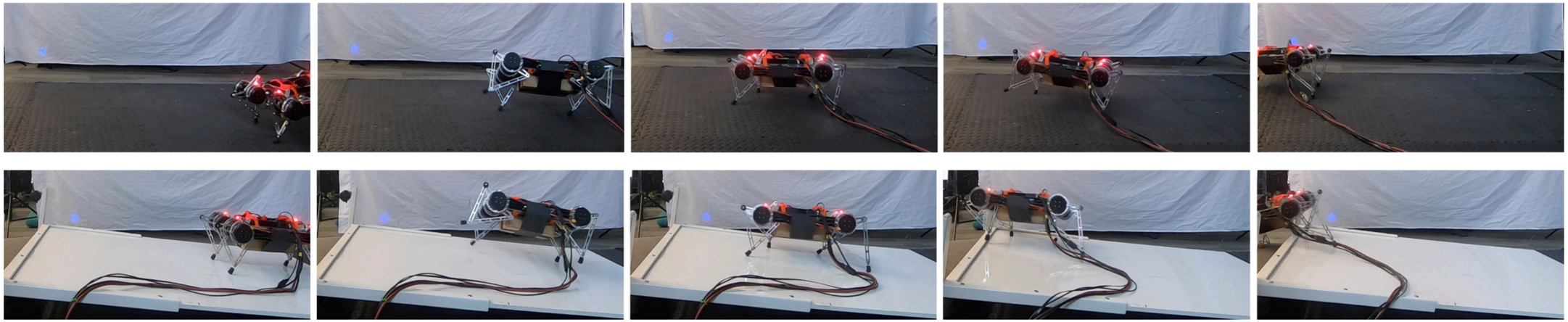
- Similarly, we treat f_{RL} as the optimization of context variables
  - Latent variables are “knobs” for tuning behaviors.
  - We can use a simple optimization, such as Bayesian optimization.



Yu et al. “Learning Fast Adaptation with Meta Strategy Optimization”, 2020

Meta-learning with Simple Optimization

- Trained with Augmented Random Search (gradient free!).
- Works well, if you have a good representation of context variables.



Summary

- Model-based RL leverages the learned dynamics model for better planning.
- Meta RL aims to model fast-adaptation between episodes.
- (+) Both approaches can achieve far better sample efficiency.
- (-) Both approaches often do not work in real-world problems.