Learning and Transfer of Modulated Locomotor Controllers

Agenda

Motivation and Problem

Method

Experiments

Positive takeaways

Critiques

Discussion

Problem Overview

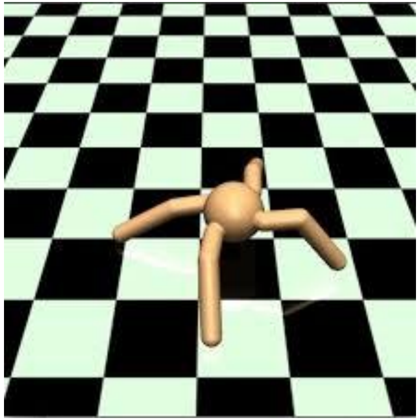
Motivation

Meaningful Reinforcement Learning is difficult and often not readily solvable by many pure RL approaches

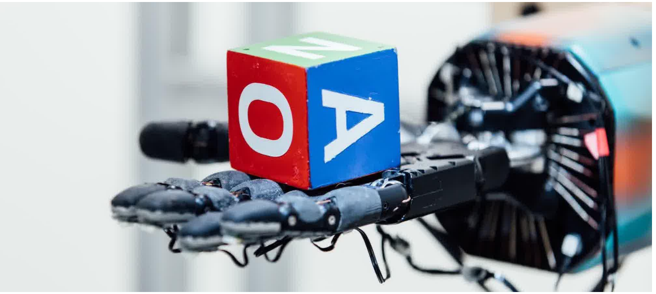
- Long horizons
- Large state and action spaces
- Sparse rewards
- Lengthy training

State of RL

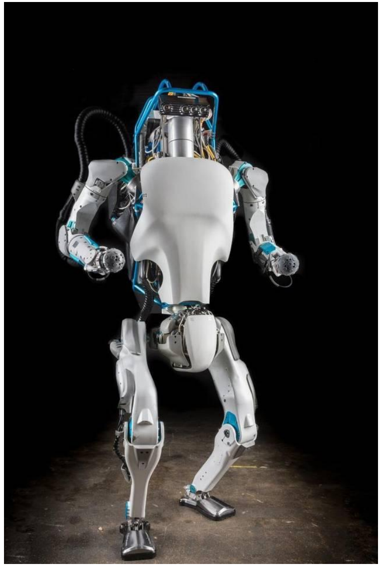
< 1 day exploration
8 dof



100 years of exploration
20 dof



???? of exploration
> 30 dof



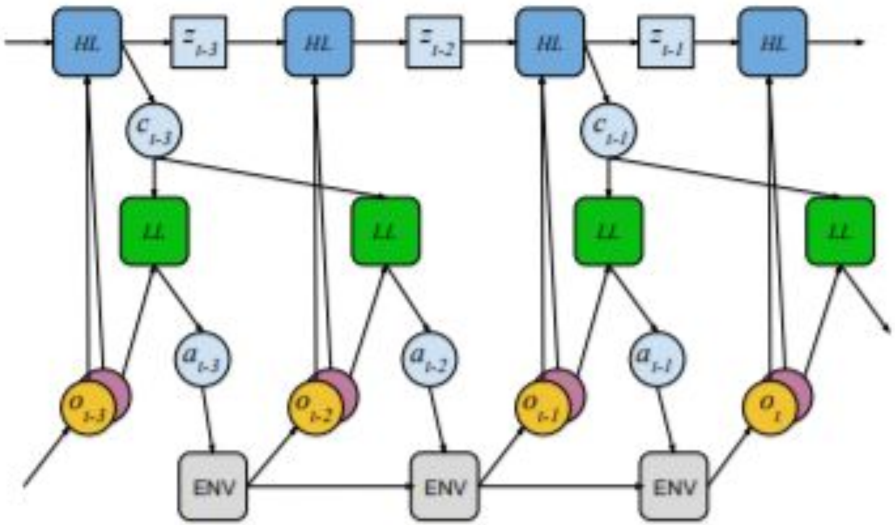
Problem

Enable efficient policy learning on difficult sparse reward tasks where pure reinforcement learning fails.

- Efficient exploration of large state and action spaces.
- Meaningful policy pre-training for transfer to new tasks.
- Learn useful action primitives for complex tasks to be leveraged by a higher-level decision-making module

Method

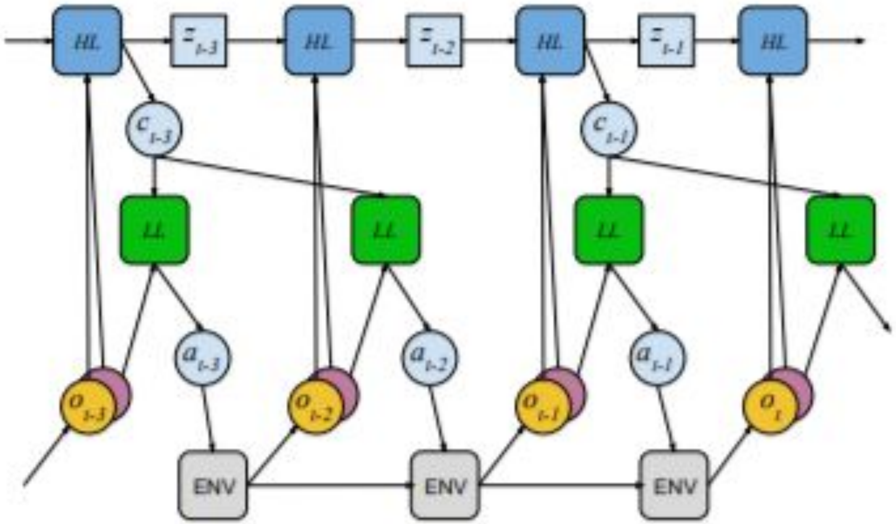
Architecture



Agent consists of high level and low level controllers

Architecture

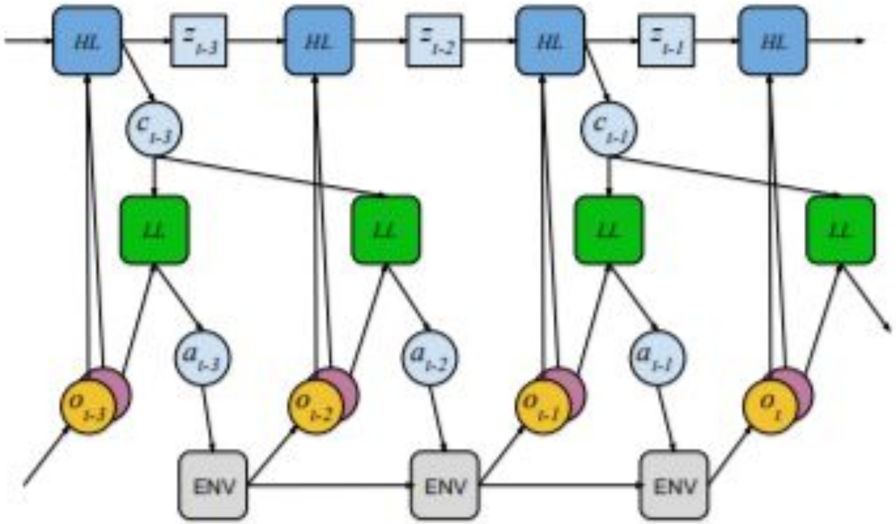
- High level controller modulates the low level controllers via c_t
- High level controllers are relearned for every task (pretraining and all transfer tasks)
- The High level controller operates on a different time scale
  - Updates to c_t do not occur every step
- Both controllers have access to proprioceptive information (eg. joint angles)
- High level controller has access to task-specific information (eg. location of goal)
  - Available to low level controllers via c_t information bottleneck
    - Encourages domain invariance



Architecture

- Low level controllers parameterize actions as samples from a gaussian distribution
- Proprioceptive information (o^P) and high level controller states c_t are used to compute mean and variance

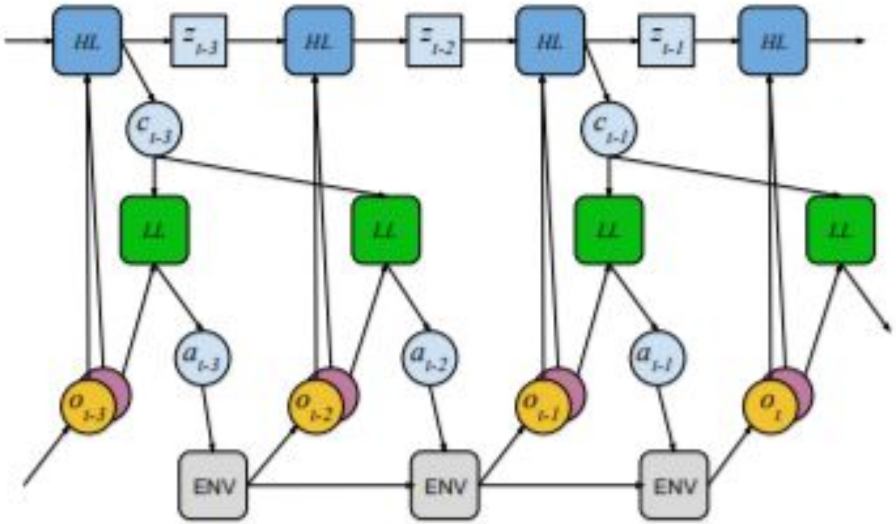
$$
\begin{aligned}
(\mu, \sigma) &= F_L(o^P, c) \\
a &\sim \mathcal{N}(\cdot|\mu, \sigma^2).
\end{aligned}
$$



Architecture

- High level controllers use the full state o^F (proprioceptive and task-specific features) to compute an LSTM hidden state z_t
- c_t is only updated every K timesteps as a function of the current z_t

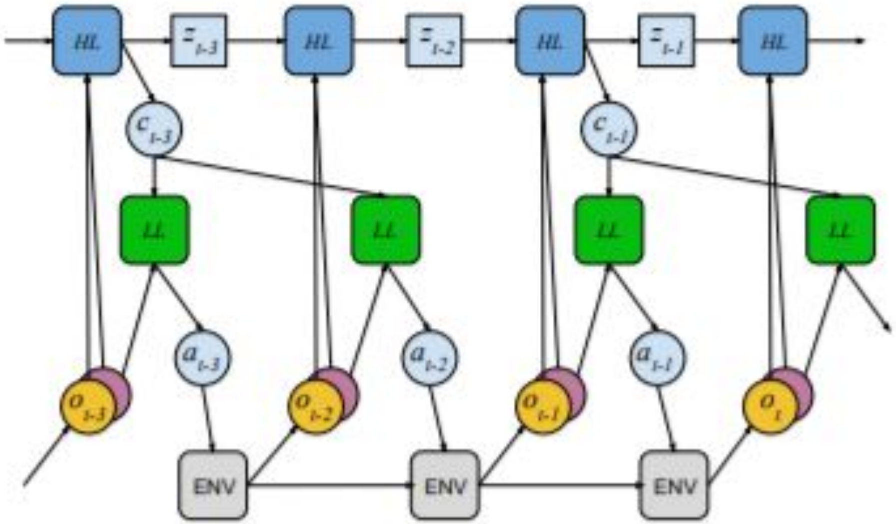
$$
\begin{aligned}
z_t &= f_H(o_t^F, z_{t-1}) \\
c_t &= g_H(z_{\tau(t)}) \\
\tau(t) &= \lfloor (t-1)/K \rfloor K + 1
\end{aligned}
$$



Architecture

- g_H can be a deterministic function or it can also be a gaussian as well
- It is unclear however if c_t is sampled once the first time it's updated, or is sampled at every timestep t

$$
\begin{aligned}
z_t &= f_H(o_t^F, z_{t-1}) \\
c_t &= g_H(z_{\tau(t)}) \\
\tau(t) &= \lfloor (t-1)/K \rfloor K + 1
\end{aligned}
$$



Architecture -- How to Train Your Model

- Reparameterization Trick
  - Allows backprop through random sampling.
  - Randomness at 2 levels!

$$x \sim \mathcal{N}(\mu, \sigma)$$

$$x = \mu + \sigma y, \quad y \sim \mathcal{N}(0, 1)$$

- Advantages for policy gradient
  - Reduce variance
  - How much better/worse than average is this transition
- Generalized Advantage Estimation
  - Balance tradeoff between bias and variance

$$A(s, a) = Q(s, a) - V(s)$$

Experiments

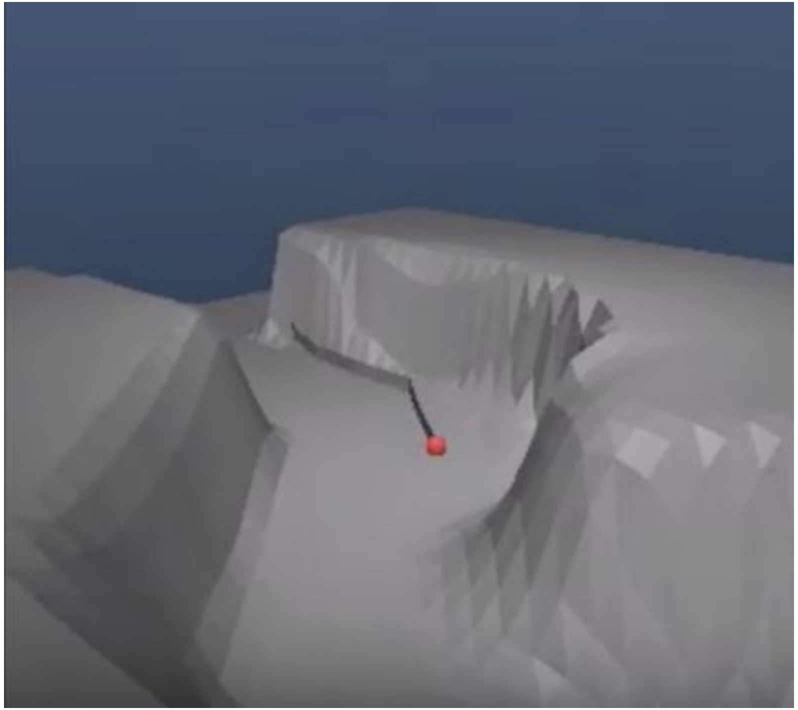
General Experiment Setup

1) Pretrain on some simple task with dense reward
   a) Analyze low level controller behavior by sampling random noise for c_t
2) Replace high level controller and provide new task-specific features
3) Train on a task with sparse reward
4) Compare results of pretrained agents wrt learning from scratch
5) Profit $$$

Snake

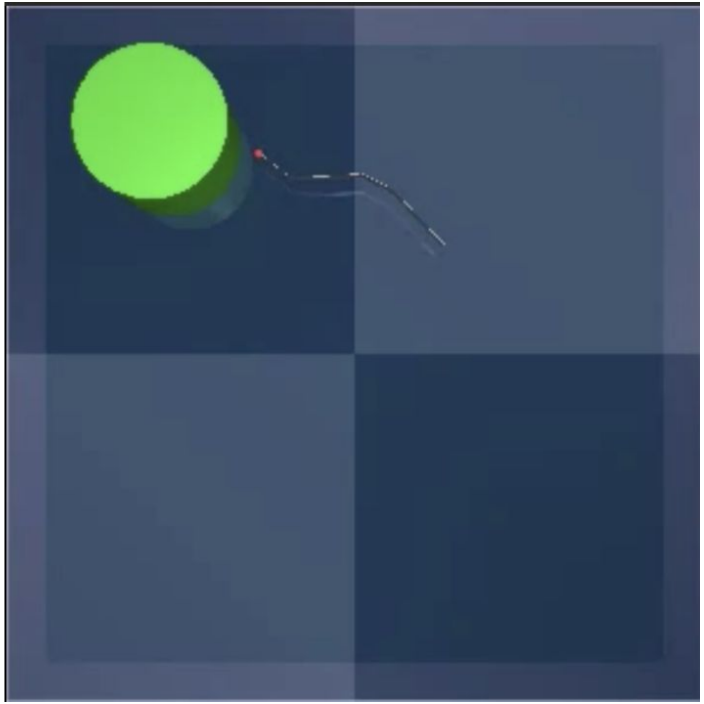
Setup

- The first experiment is run on a 16-dimensional swimming snake.
  - Low-level controllers: joint angles, angular velocities, and velocities of 6 segments in local coordinate frames
  - Tasks revolve around reaching a target point



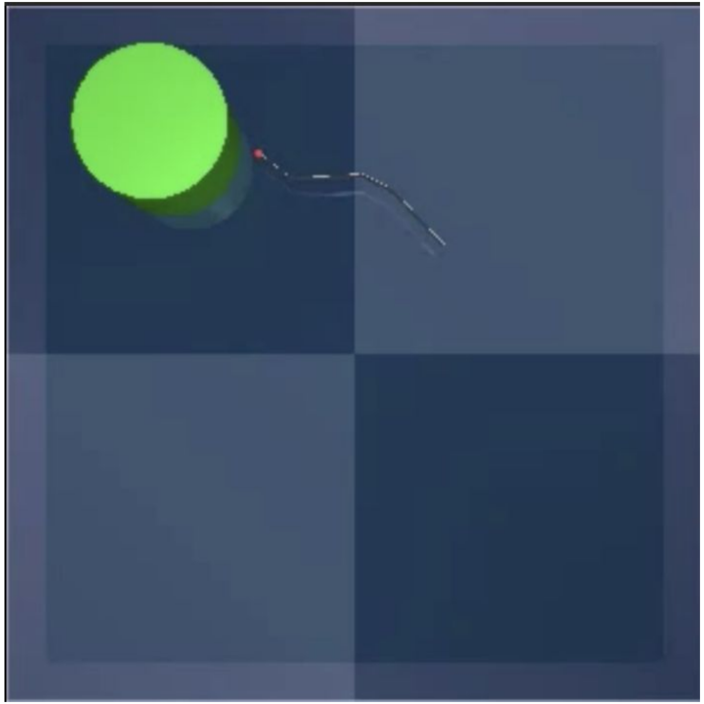
Pre-training task

- Swim towards a fixed target over 300 timesteps
  - Provisional (temporary) high-level controller is also exposed to an egocentric position of the target
- Reward function is dense: negative distance to target
- Modulation: low-level controller is updated every K = 10 time steps



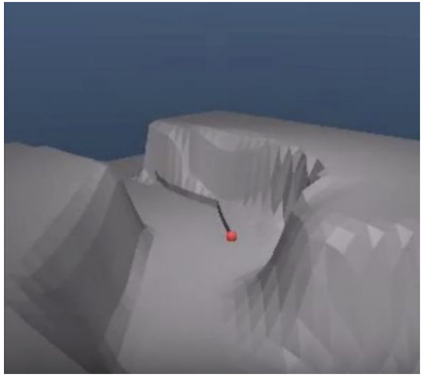
Transfer task 1: Target-seeking

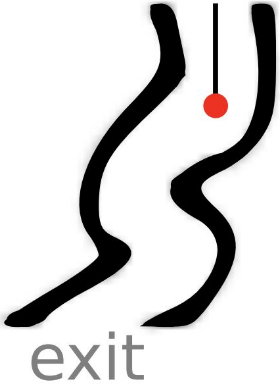
- Reward function is sparse: snake is rewarded if its head reaches the target
- Snake only sees target if within 120 deg. field of vision
- Needs to learn to turn around and swim toward target--snake and target are both randomly initialized
- Episode lasts 800 timesteps



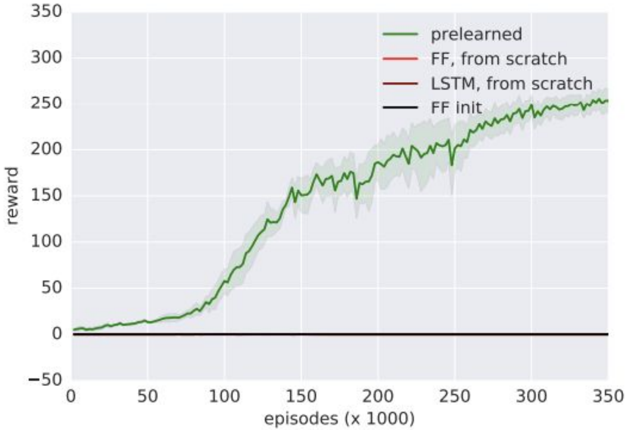
Transfer task 2: Canyon-traversal

- Reward function is sparse: snake is rewarded if its head goes through the end of the canyon
    - 3000-timestep limit
- Canyon walls provide constraints, as well as possibly impair vision

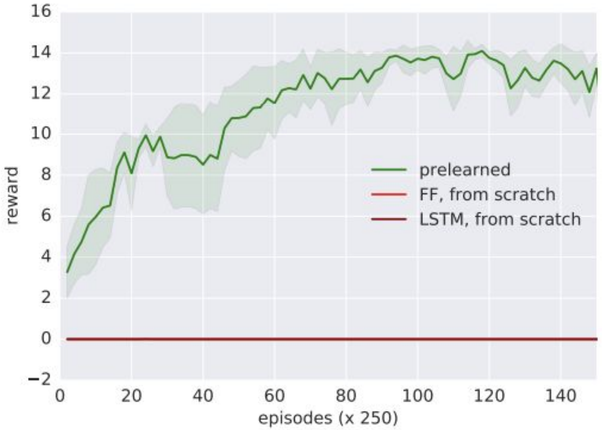




Snake results

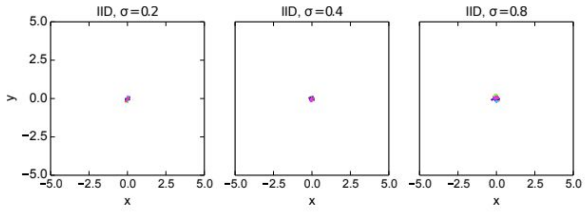


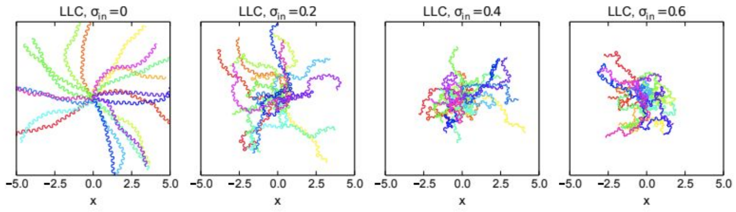
(a) target-seek



(b) canyon traversal

Choosing the action distribution to be a diagonal, zero-mean Gaussian can lead to poor results!





Quadruped

Quadruped Pre-training: Fixed Target Seeking

Fixed target Seeking

- Task-specific features: relative x,y position of goal target
- Dense reward: negative distance to target

Quadruped Task 1: Fixed Target Seeking

Fixed target Seeking

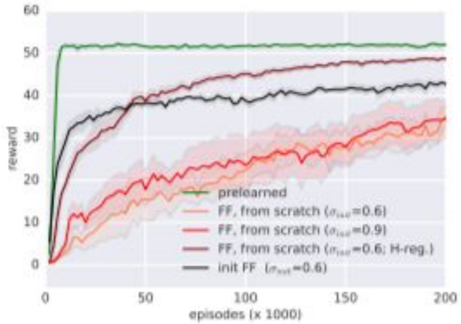
- Task-specific features: relative x,y position of goal target
- Sparse reward: torso is within the green area
- Task difficulty modulated by start distance from target

Quadruped Task 2: Soccer

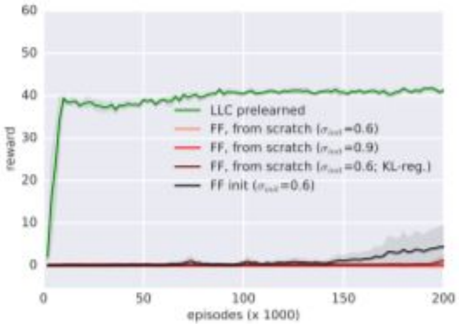
Fixed target Seeking

- Task-specific features: Velocity of ball, and relative distance from ball and goal
- Sparse reward: ball crosses goal zone
- Task difficulty
  - V1: ball starts between quadruped and goal
  - V2: ball starts behind quadruped

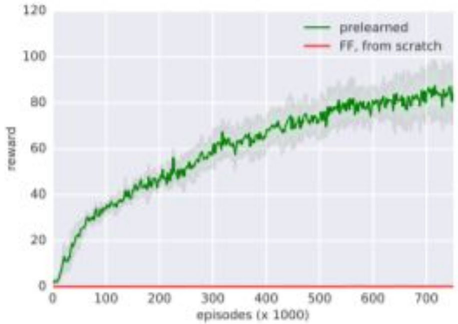
Quadruped results



(a) target-seek (easy)



(b) target-seek (hard)



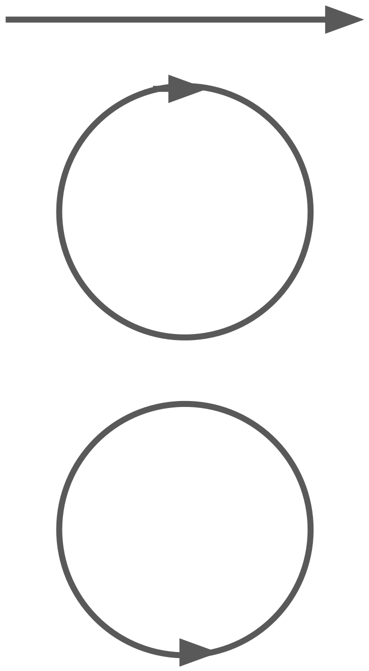
(c) soccer

Humanoid

Humanoid Pretraining: Path Following

Fixed target Seeking

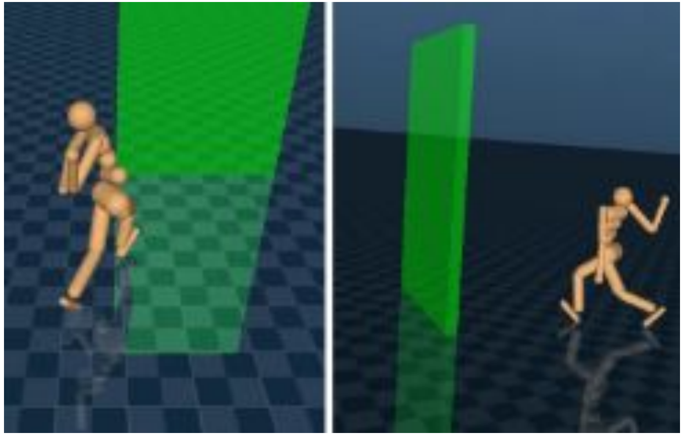
- Task-specific features: relative x,y position of goal target
- Dense reward: quadratic penalty for being off the path



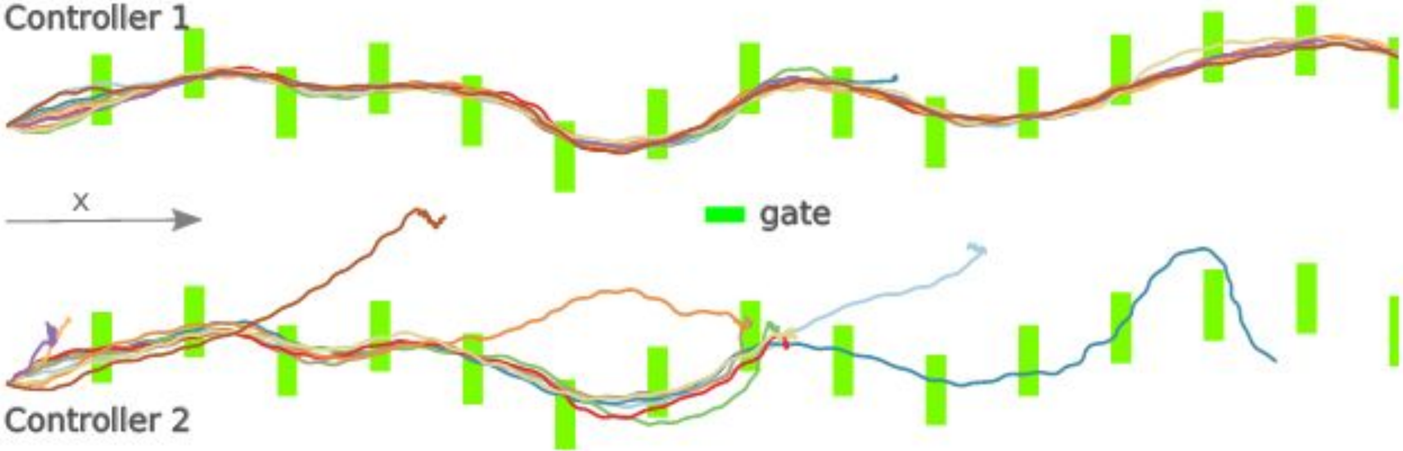
Humanoid Task: Slalom

Path following with waypoints

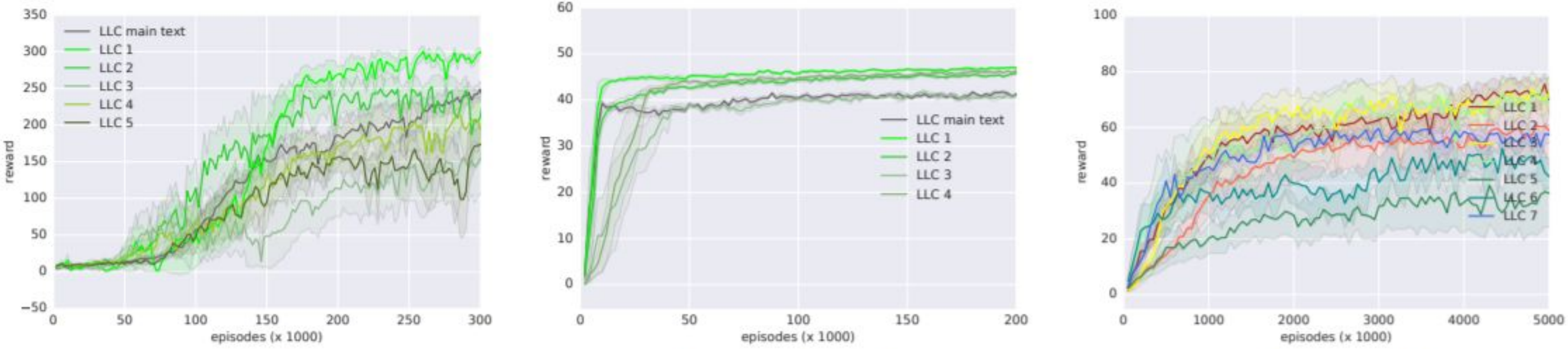
- Task-specific features: relative position and orientation of next waypoint
- Sparse reward: +5 when agent passes a waypoint
- Terminal state if a waypoint is missed



Humanoid: Results



Low Level Controller Variability



(a) Snake (b) Quadruped (c) Humanoid

Run to Run variability for different seeds

Positive Takeaways

Takeaways

Novel network architecture demonstrating use of latent model for compositional policies.

- Many subsequent works use similar ideas (eg. Multiplicative Compositional Policies)
- Can transfer from tasks with dense rewards to tasks with sparse rewards
- Show convergence on complicated and high DOF tasks
- Exploration in a hierarchical model might have better properties

Paper is concise, straight-to-the-point, and well-organized. The performance results demonstrate clearly the efficacy of the approach.

Critiques

Room For Improvement

- Not entirely clear on environment / reward design (e.g., snake)
- No training information for experiment replication
- Why not more ablation studies?
    - Frequency of modulation, instead of just K=1, K=10
    - Size of networks
    - Different distributions for exploration

Discussion

- Does the information bottleneck really help domain invariance? Can useful additional signal be utilized effectively by the LLCs without it?
- How does one extend this pretraining idea to more complex regimes and settings where it is not obvious how to create a simple/solvable pretraining task that is nearby in task space?
    - One component of this being: how can we create dense reward functions for “trivial” tasks in real-world settings that can be used for more desirable tasks?
- Why is the Gaussian distribution chosen for exploration, rather than Zipfian or other distribution?

Future Work

- Curriculum learning to pretrain locomotors that get progressively better
- Unsupervised Meta Learning to construct pretraining tasks that lead to better downstream transfer
- *N* hierarchical layers instead of two for more complex tasks
    - Greater modularity, extensibility