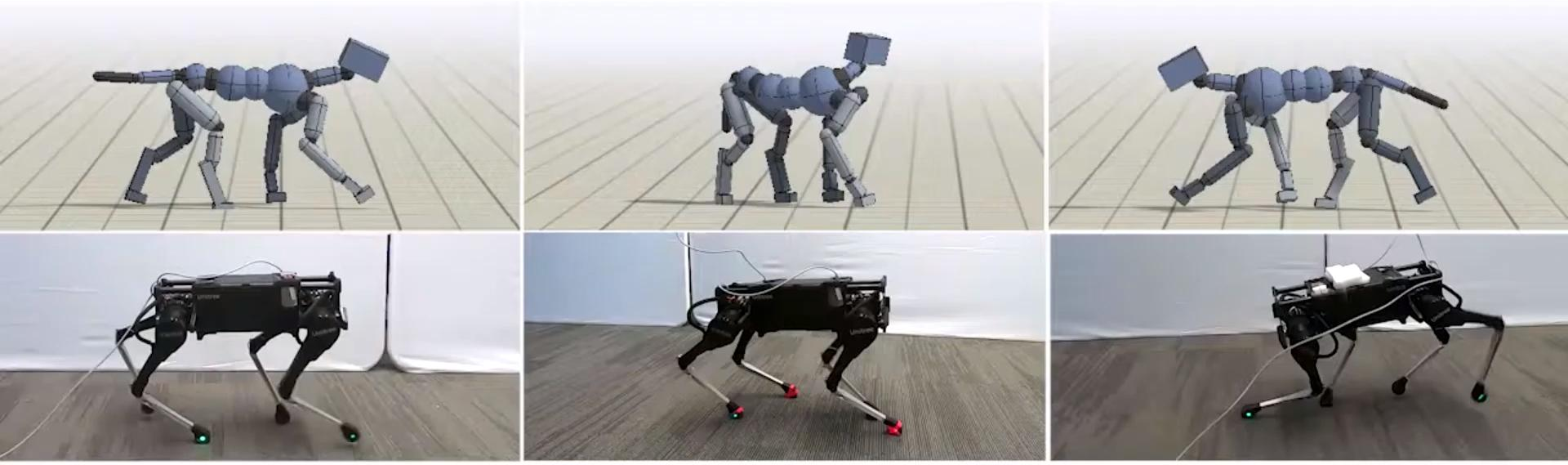
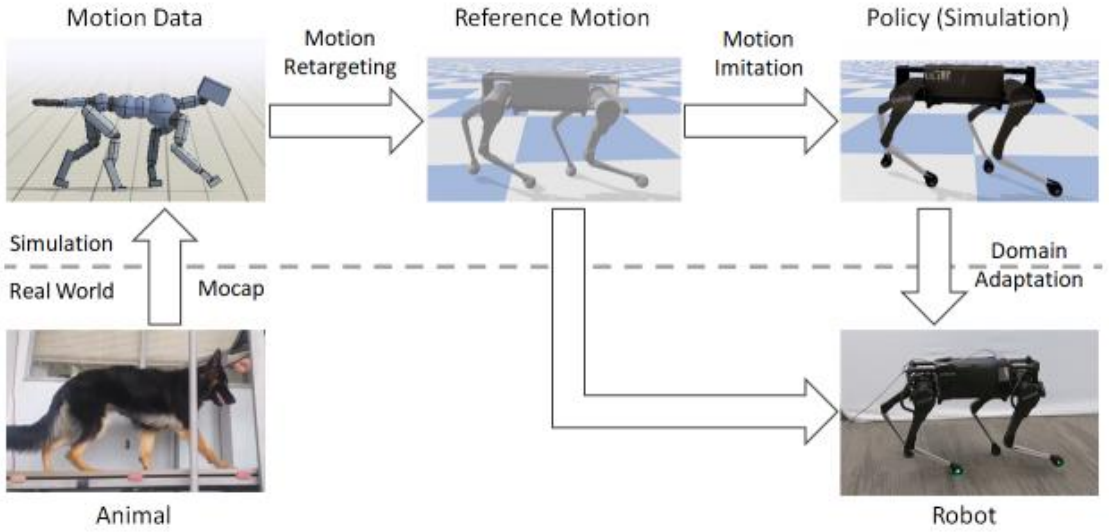
Learning Agile Robotic Locomotion Skills by Imitating Animals

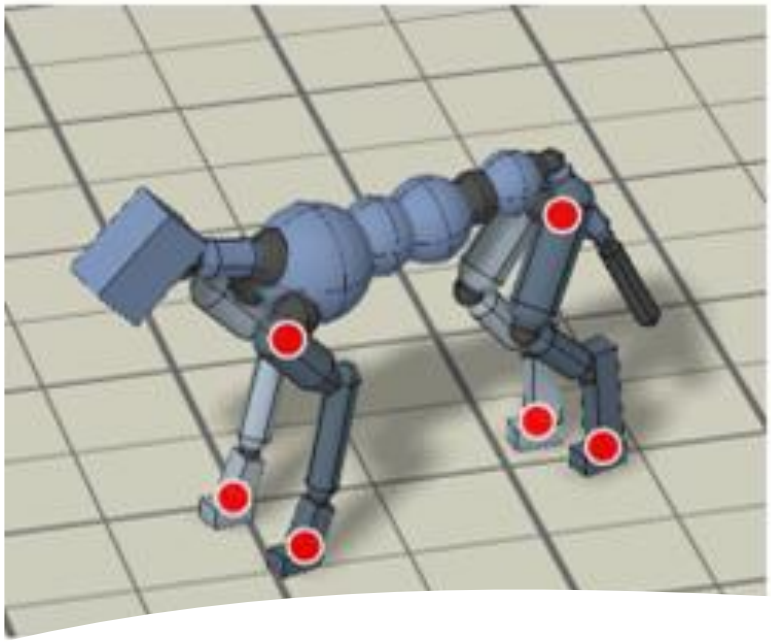


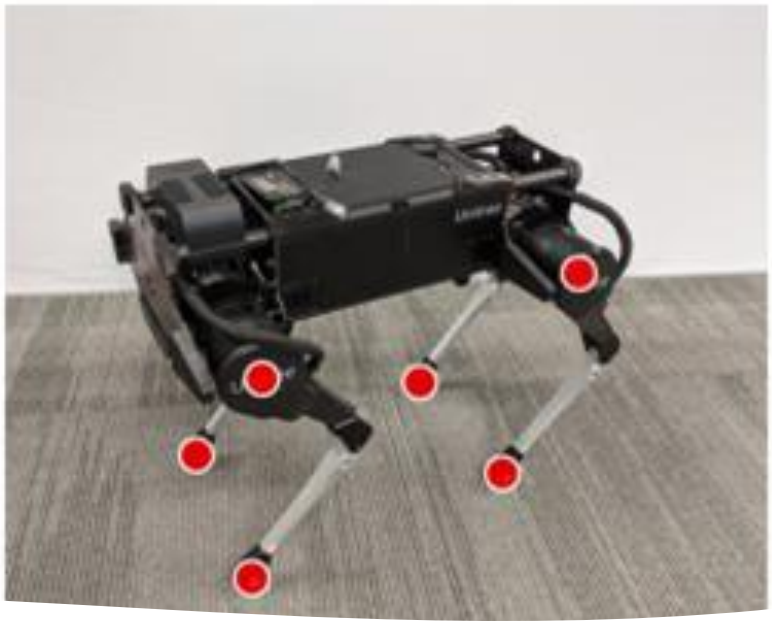
Paper ID: 21

Three-Stage Learning Framework

- 1. Motion Retargeting: it adapts the animal's motion to the robot's body using inverse kinematics.
- 2. Motion Imitation: Trains a simulated robot with RL to reproduce the retargeted motion, using domain randomization for robustness.
- 3. Domain Adaptation: Transfers the learned policy to the real robot and fine-tunes it using a latent dynamics model for efficient real-world adaptation.







Stage 1: Motion Retargeting

- Source: motion capture of a real dog (= mocap)
- Problem: dog and robot have different body structures
- Solution: use inverse kinematics to align keypoints (hips, feet, torso) and create a reference motion trajectory *(1)*
- Produces a robot-feasible version of the motion

(1) $$\arg\min_{\mathbf{q}_{0:T}} \sum_t \sum_i ||\hat{\mathbf{x}}_i(t) - \mathbf{x}_i(\mathbf{q}_t)||^2 + (\bar{\mathbf{q}} - \mathbf{q}_t)^T \mathbf{W} (\bar{\mathbf{q}} - \mathbf{q}_t)$$

Stage 2: Motion Imitation

- Policy (π) input:
  - robot state (IMU, joint angles, velocities) *(1)*
  - goal (future poses) *(2)*
- Output: target joint commands
- Reward encourages tracking of joint angles, velocities, end-effector positions, and root motion *(3)*
- Training performed entirely in simulation

(1) $\mathbf{s}_t = (\mathbf{q}_{t-2:t}, \mathbf{a}_{t-3:t-1})$

(2) $\mathbf{g}_t = (\hat{\mathbf{q}}_{t+1}, \hat{\mathbf{q}}_{t+2}, \hat{\mathbf{q}}_{t+10}, \hat{\mathbf{q}}_{t+30})$

(3)

$$r_t = w^{\mathrm{p}} r_t^{\mathrm{p}} + w^{\mathrm{v}} r_t^{\mathrm{v}} + w^{\mathrm{e}} r_t^{\mathrm{e}} + w^{\mathrm{rp}} r_t^{\mathrm{rp}} + w^{\mathrm{rv}} r_t^{\mathrm{rv}}$$

$$r_t^{\mathrm{p}} = \exp\left[-5 \sum_j ||\hat{\mathbf{q}}_t^j - \mathbf{q}_t^j||^2\right].$$

(...)

$$r_t^{\mathrm{rp}} = \exp\left[-20||\hat{\mathbf{x}}_t^{\mathrm{root}} - \mathbf{x}_t^{\mathrm{root}}||^2 - 10||\hat{\mathbf{q}}_t^{\mathrm{root}} - \mathbf{q}_t^{\mathrm{root}}||^2\right]$$

$$r_t^{\mathrm{rv}} = \exp\left[-2||\hat{\dot{\mathbf{x}}}_t^{\mathrm{root}} - \dot{\mathbf{x}}_t^{\mathrm{root}}||^2 - 0.2||\hat{\dot{\mathbf{q}}}_t^{\mathrm{root}} - \dot{\mathbf{q}}_t^{\mathrm{root}}||^2\right]$$

Stage 3: Domain Adaptation

- Challenge: Simulation ≠ real world (differences in friction, mass, latency)

Done in 3 steps:

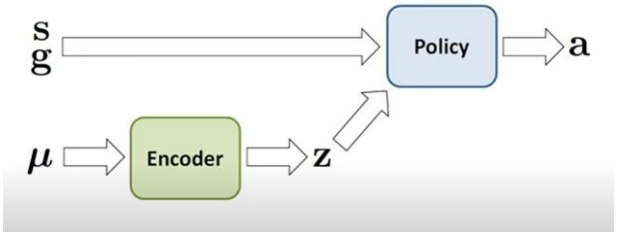
- Domain randomization: vary parameters during training for robustness *(μ)*
- Latent-space adaptation:
  - Learn compact dynamics representation *(z)*
  - AWR: fine-tune on real robot with ~50 trials

(μ matrix and Encoder are only for simulation)

(μ)

Parameter	Training Range	Testing Range
Mass	$[0.8, 1.2]$ × default value	$[0.5, 2.0]$ × default value
Inertia	$[0.5, 1.5]$ × default value	$[0.4, 1.6]$ × default value
Motor Strength	$[0.8, 1.2]$ × default value	$[0.7, 1.3]$ × default value
Motor Friction	$[0, 0.05]$ Nms/rad	$[0, 0.075]$ Nms/rad
Latency	$[0, 0.04]$ s	$[0, 0.05]$ s
Lateral Friction	$[0.05, 1.25]$ Ns/m	$[0.04, 1.35]$ Ns/m

(z)



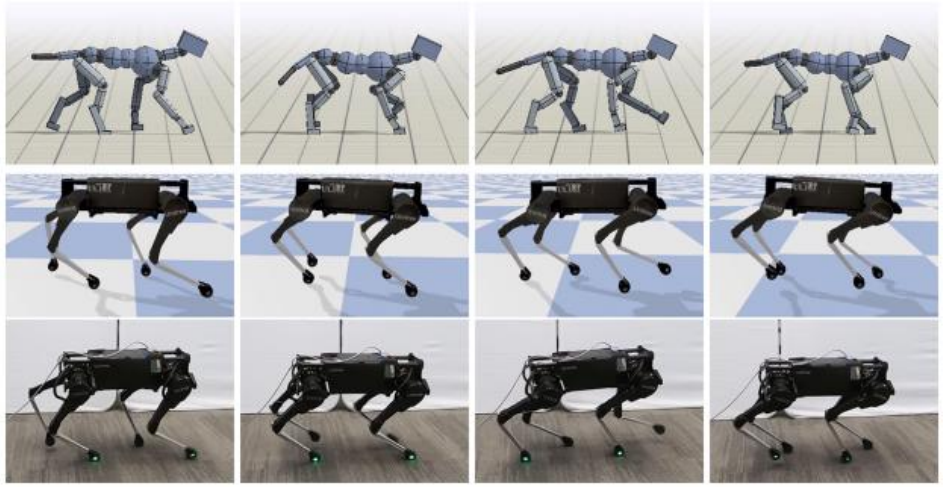
Experimental Evaluation

Setup

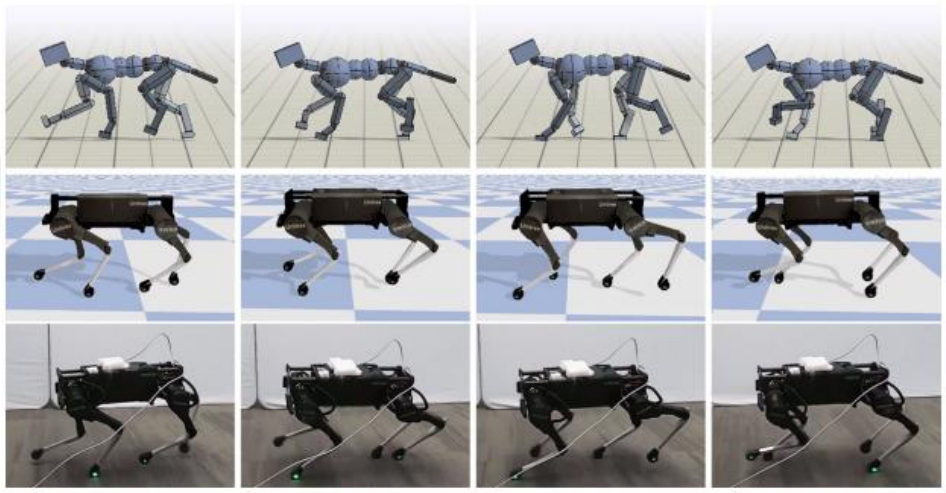
- Simulations done in PyBullet.
- Real tests done on Laikago, an 18-DOF quadruped robot.
- Motion data taken from real dogs + artist-created animations.

Training

- Each policy trained with Proximal Policy Optimization **(PPO)** for 200 million samples.
- Adaptation done on the physical robot using ~50 trials, each 5–10 seconds.



(a) Dog Pace



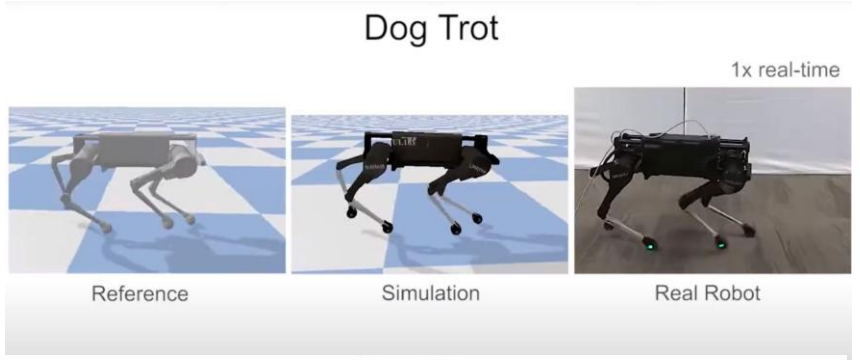
(b) Dog Backwards Trot

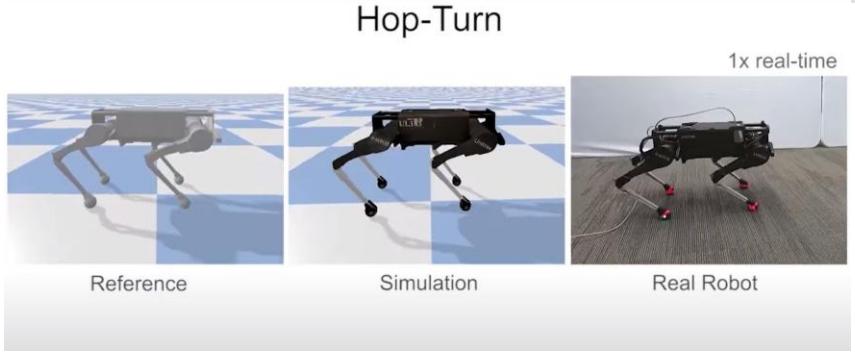
Learned Skills

The robot successfully learned:

- Dog-inspired gaits like pace, trot and backward trot.
- Then some moves were “artist animated” such as the side steps, turns, and hop-turns (aerial 90° turn midair).
- Possible to create “fun” motions like the “running man” dance.

For this last one the system had difficulties imitating challenging motions.

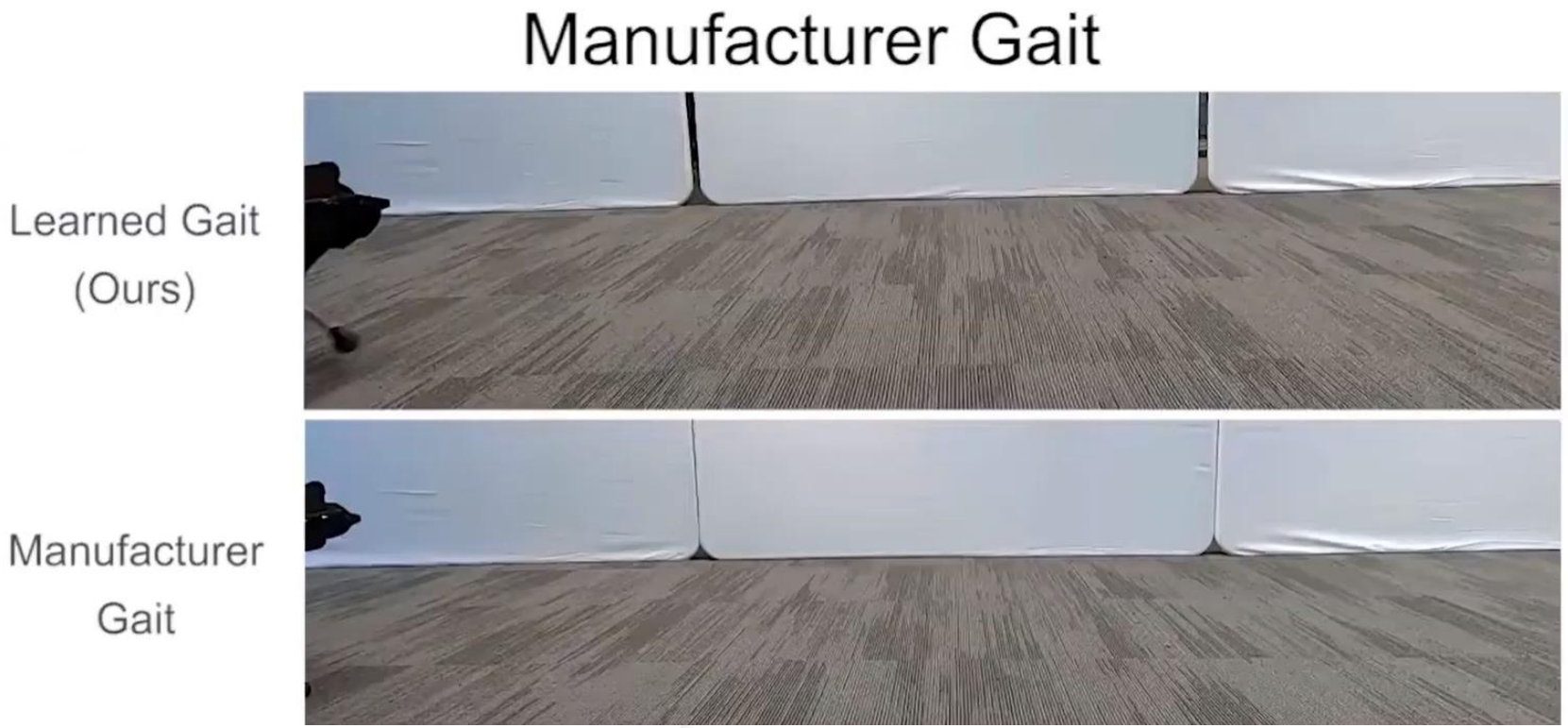




Results

Learned gaits were *faster and more natural* than the manufacturer's built-in controller.

Example: manufacturer's top speed = 0.84 m/s **vs.** learned trot = 1.08 m/s.



Domain Adaptation Results

Three policy types were compared:

1. **No Rand (baseline)**: trained with fixed dynamics.
2. **Robust** : trained with domain randomization only.
3. **Adaptive**: trained with randomization + latent adaptation.

Findings:

- Adaptive policies performed best both in simulation and real world.
- They maintained balance longer and successfully reproduced complex motions.
- Non-adaptive policies failed under slight dynamic mismatches.

Domain Adaptation Results

Figures 5–9 show that:

- Adaptive policies achieved higher normalized returns.
- Adaptation improved stability and consistency.
- Adaptation converged in few real-world episodes (<50).

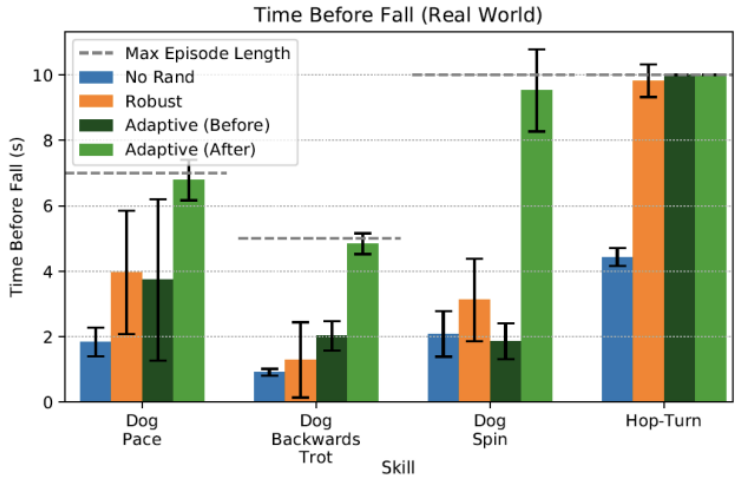


Fig. 7. Comparison of the time elapsed before the robot falls when deploying various policies in the real world. The adaptive policies are often able to maintain balance longer than the other baselines policies, and tend to reach the max episode length without falling.

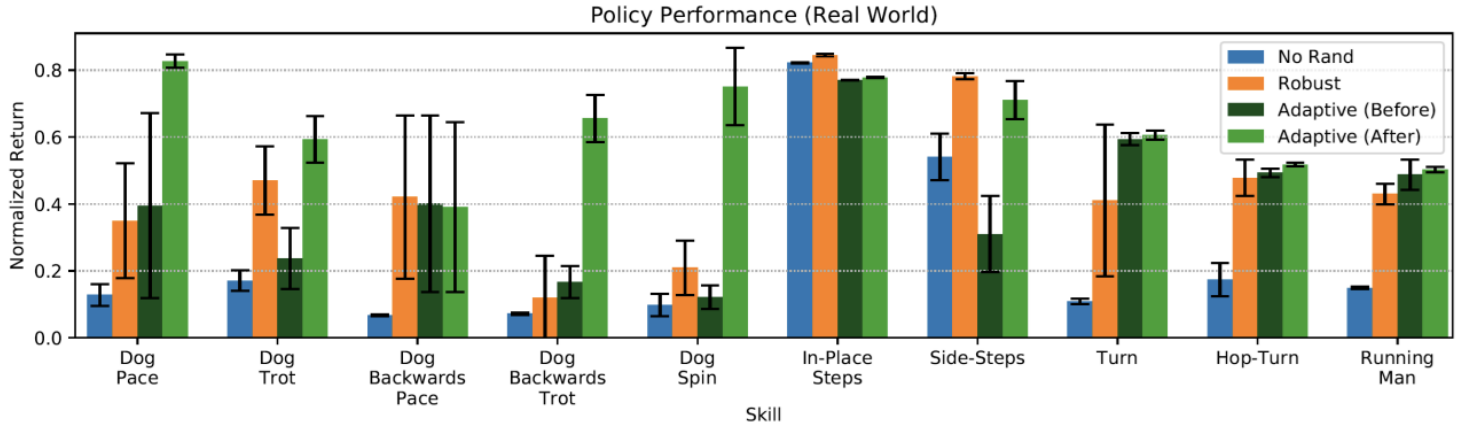


Fig. 5. Performance statistics of imitating various skills in the real world. Performance is recorded as the average normalized return between [0, 1]. Three policies initialized with different random seeds are trained for each combination of skill and method. The performance of each policy is evaluated over 5 episodes, for a total of 15 trials per method. The adaptive policies outperform the non-adaptive policies on most skills.

Domain Adaptation Results

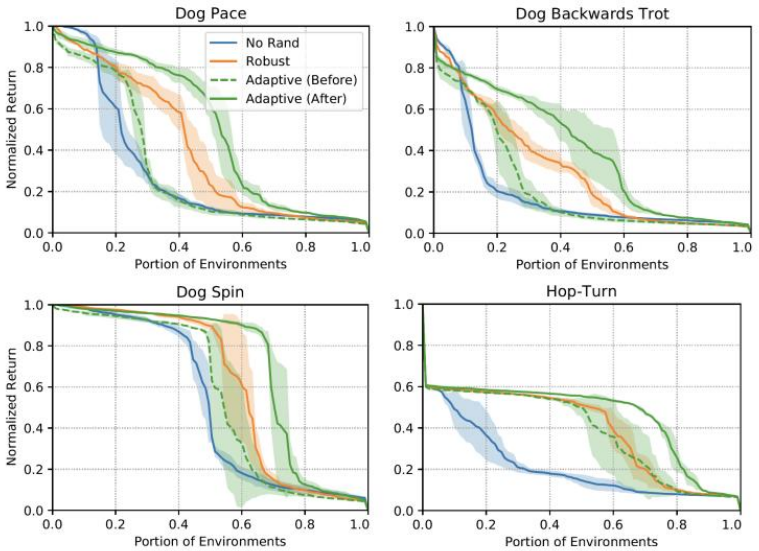


Fig. 8. Performance of policies in 100 simulated environments with different dynamics. The vertical axis represents the normalized return, and the horizontal axis records the portion of environments in which a policy achieves a return higher than a particular value. The adaptive policies achieve higher returns under more diverse dynamics than the non-adaptive policies.

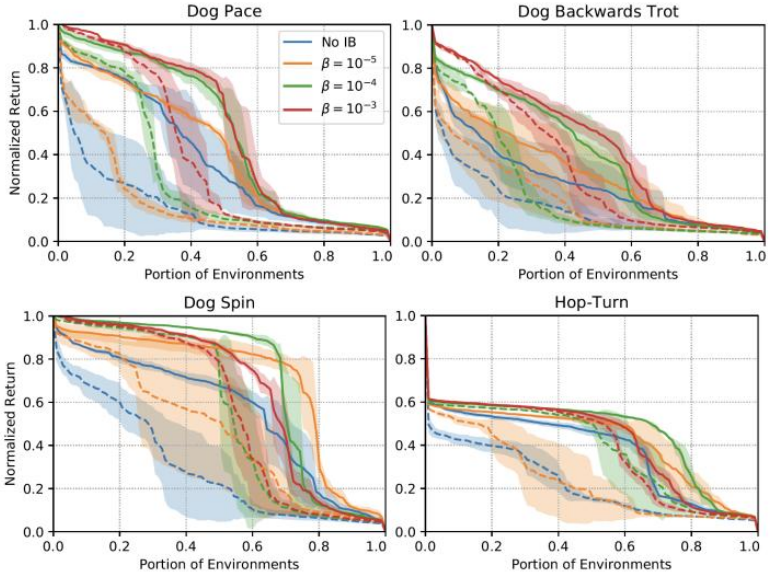


Fig. 10. Performance of adaptive policies trained with different coefficients β for the information penalty. "No IB" corresponds to policies trained without an information bottleneck. The dotted lines represent performance before adaptation, and the solid lines represent after adaptation.

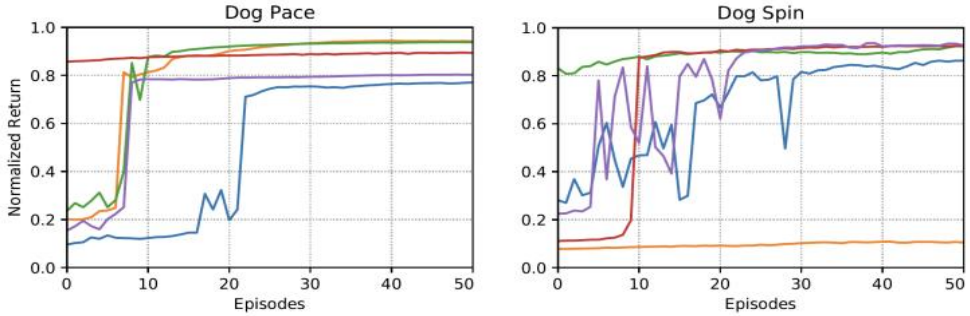


Fig. 9. Learning curves of adapting policies to different simulated environments using the learned latent space. The policies are able to adapt to new environments in a relatively small number of episodes.

Discussion & Future Work

Contributions (Pros):

- First system enabling a real quadruped to imitate animal locomotion.
- No need hand-tuned reward per skill.
- Successful sim-to-real transfer with minimal real-world trials.

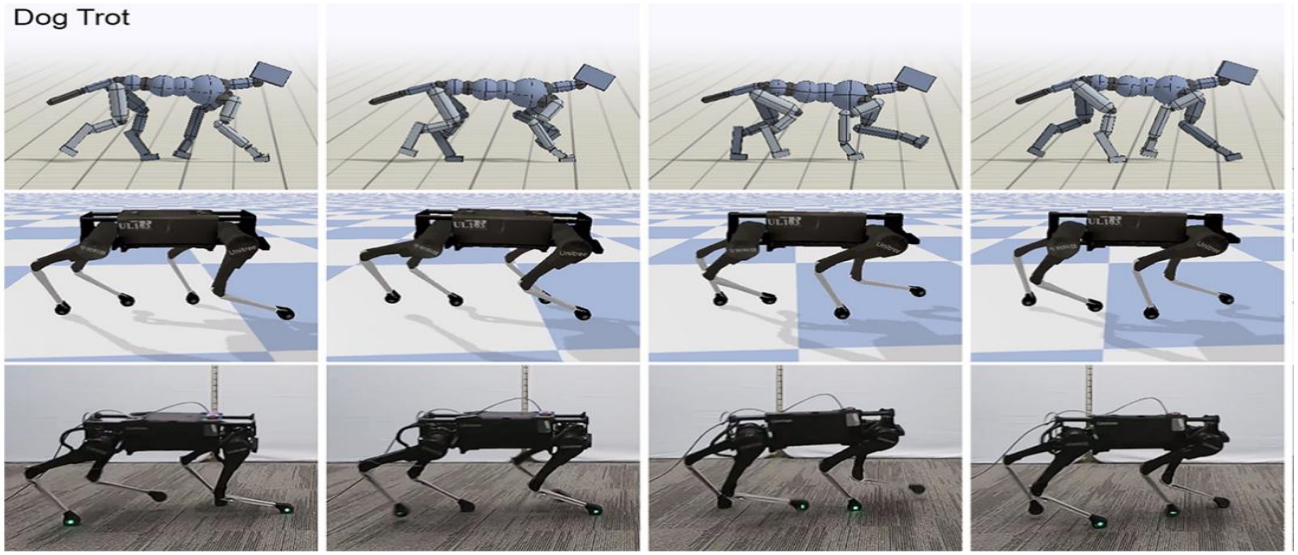
Limitations (Cons):

- Could not yet reproduce very dynamic behaviors (e.g., running or big jumps) due to hardware and algorithmic limits.
- Learned motions are still less stable than expertly hand-tuned controllers.

Future directions:

- Learn from video data (not just motion capture).
- Improve hardware robustness.
- Develop more efficient adaptation for faster real-world deployment.

Overall, this paper is a big step toward truly agile, animal-like robotic locomotion!!!



Thank you for listening!