Transfer from Simulation to Real World through Learning Deep Inverse Dynamics Model

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaermba

OpenAI, San Francisco, CA, USA

Presentor: Hao-Wei Lee

Develop Control Policy for a System

If you have a robot. To find a good way to control it, you can either:

- Peform reinforcement learning during the robot operation.
  - takes higher cost and time.
- Perform reinforcement learning on a simulation of the robot.

Learn Policies from Simulation?

- Policies learned from simulation usually cannot be used directly.
- Simulation often captures only high level trajectories, ignoring details of physical properties.
- Can we transfer learned policy from simulation to real world?

Transfer Learning of Policy

- Policies are found by simulation instead of real world.
- Use neural network to map learned policy in source environment (simulation) to target environment (real world).
- Transfer good policies in one simulation to many other real world environments.

Variables in Environments

Each environment has its own:

- State Space S: $s \in S$ are states of the environment.
- Action Space A: $a \in A$ are actions can be take.
- Observation Space O: o(s) is the observation of environment in state s
- System Forward Dynamic: $T(s, a) = s'$, determine new state s' given action and previous state

Deep Inverse Dynamic Model

- $\tau_{-k:}$: Trajectory: $\{o\}$most recent k observations and k-1 actions of target environment.
- π_{source}: Good enough policy in source environment.
- ϕ: Inverse dynamics is a neural network that maps source policy to target policy.

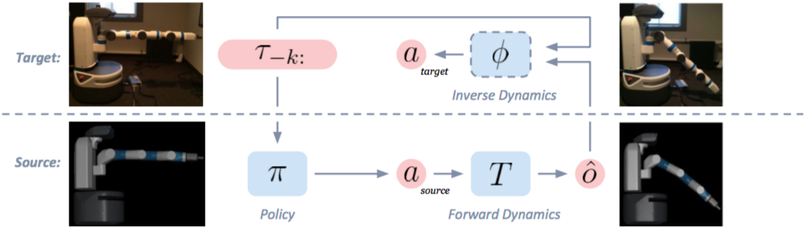
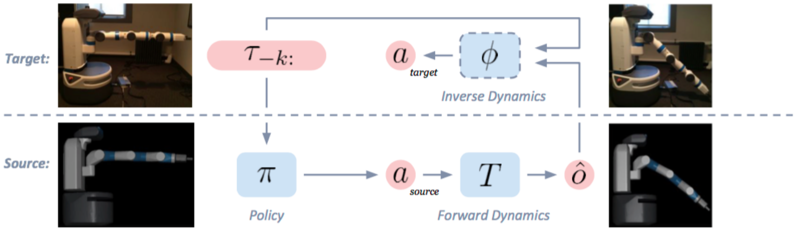


Figure:

Deep Inverse Dynamic Model

1. Compute source action $a_{source} = \pi_{source}(\tau_{-k:})$ according to target trajectory.
2. Observe the next state given $\tau_{-k:}$ and a_{source}: $\hat{o}_{next} = o(T_{source}(\tau_{-k:}, a_{source}))$
3. Feed $\hat{o}_{next}$ and $\tau_{-k:}$ to Inverse dynamics that produce a_{target}



Training of Inverse Dynamics Neural Network I

- Given trajectory of previous k time step and the desired observation o_{k+1}, the network output action that leads to desired observation
  $\phi : (o_0, a_0, o_1, \ldots, a_{k-1}, o_k, o_{k+1}) \rightarrow a_k$
- Training data are obtained by preliminary inverse dynamics model ϕ and prelimiary policy π_{target} of target environment
- Diversity of training data can be achieved by adding noise to predefined actions

Architecture of Inverse Dynamic Neural Network

- input: previous k observations, previous $k-1$ actions, desired observation for next time step
- output: the action that leads to desired observation
- Hidden layer: two fully connected hidden layer with 256 unit followed by ReLU activation function.

Simulation 1 to Simulation 2 Transfer I

- The experiments are performed on Simulators that can change conditions of it's environment.
- The source and target environment are basically the same model except gravity or motor noise
- The following four models are used for simulation.

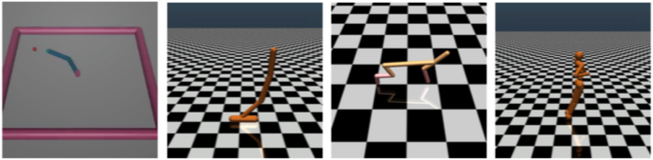
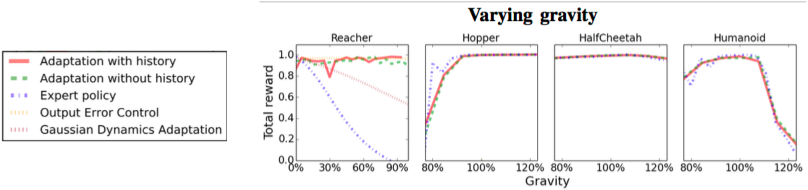


Figure: From left to right are Reacher, Hopper, Half-cheetah, and Humanoid

Simulation 1 to Simulation 2 Transfer II

Variation of Gravity



Simulation 1 to Simulation 2 Transfer III

Variation of Motor Noise

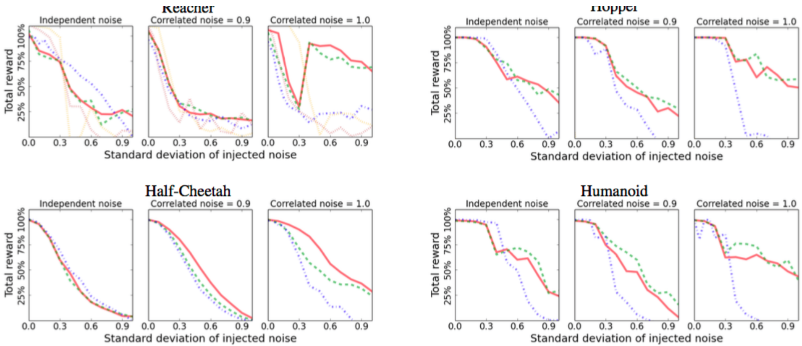


Figure:

Simulation to Real Transfer

- The real evironment is a physical Fetch Robot.
- The groundtruth is the observation obtained by directy apply reinforcement learning on the robot.
- The baseline to compare with is a PD controller.

Method \ Task	Swings limited with a bungee cord
Our method	$3.72\% \pm 0.020\%$
PD controller	$4.49\% \pm 0.050\%$

Figure: The discrepancy between observations on transferred policy and ground truth is measured.

Conclusion

- The method succefully adapt complex control policies to real world.
- obsrvation in source and target environment are assume the same, which are not always true.
- The method can also be applied to the simulation that actions cannot be seen.