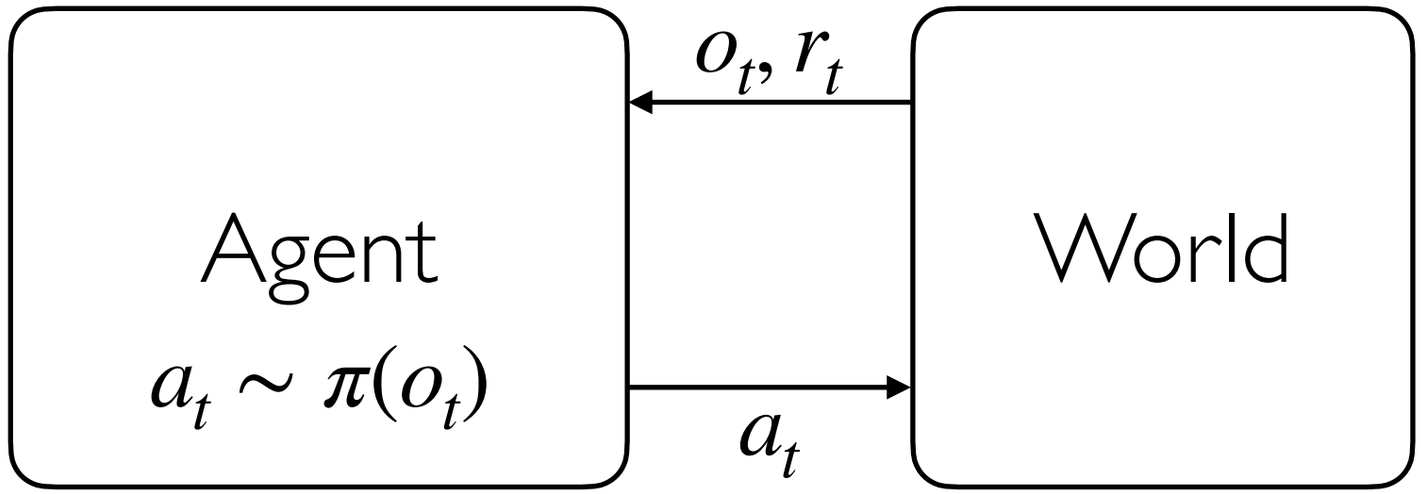
# Learning Models for RL

Saurabh Gupta

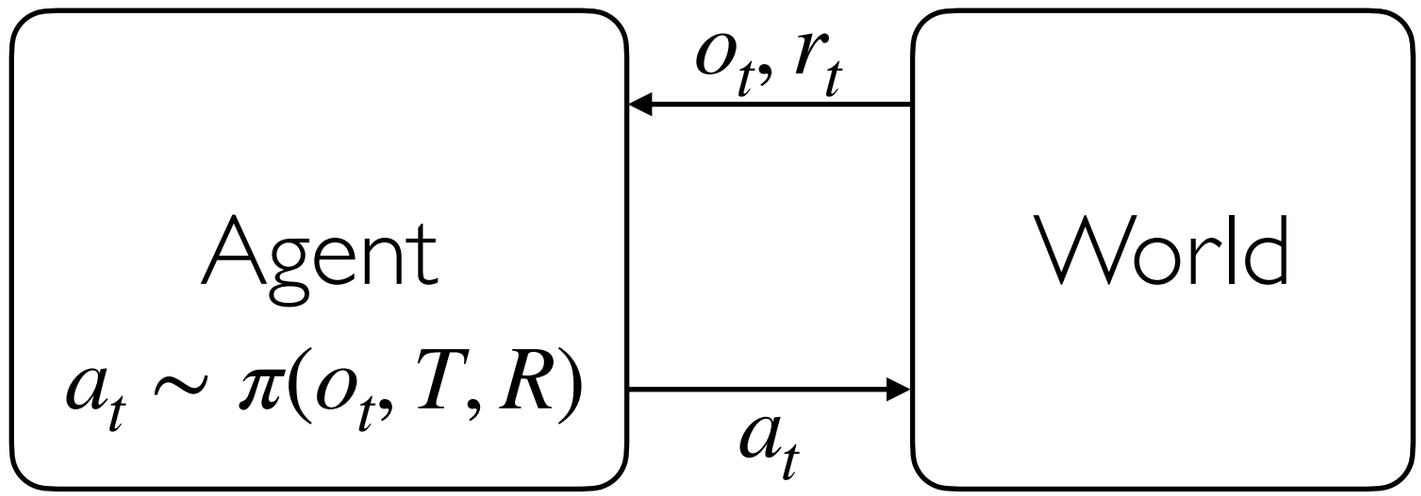
# Solving MDPs

**Policy:** $a_t \sim \pi(o_t)$

Most General Case




More Specific Case




Fully Observed System $o_t = s_t$

Known Transition Function $s_{t+1} \sim T(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

# Solving a RL Problem

Better Reward Signals

Better Optimization

Convert into a Supervised Training Problem

Solve a Related but Supervision-rich Problem

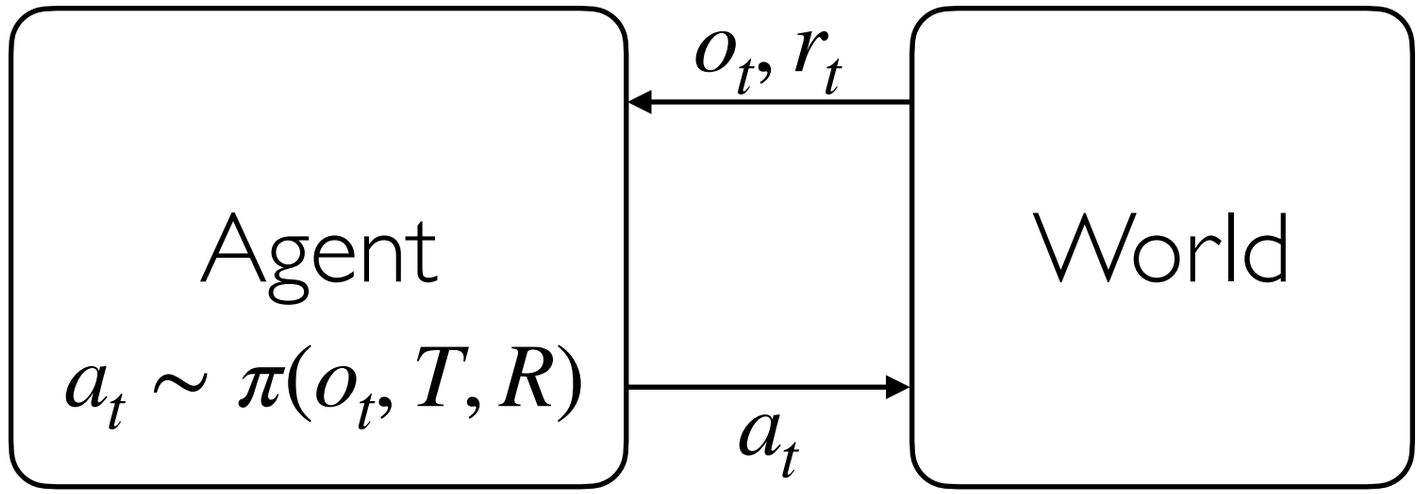
Build Models and Plan with Them

Model-free RL with sparse rewards

Known reward, known model. Model-based RL

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL



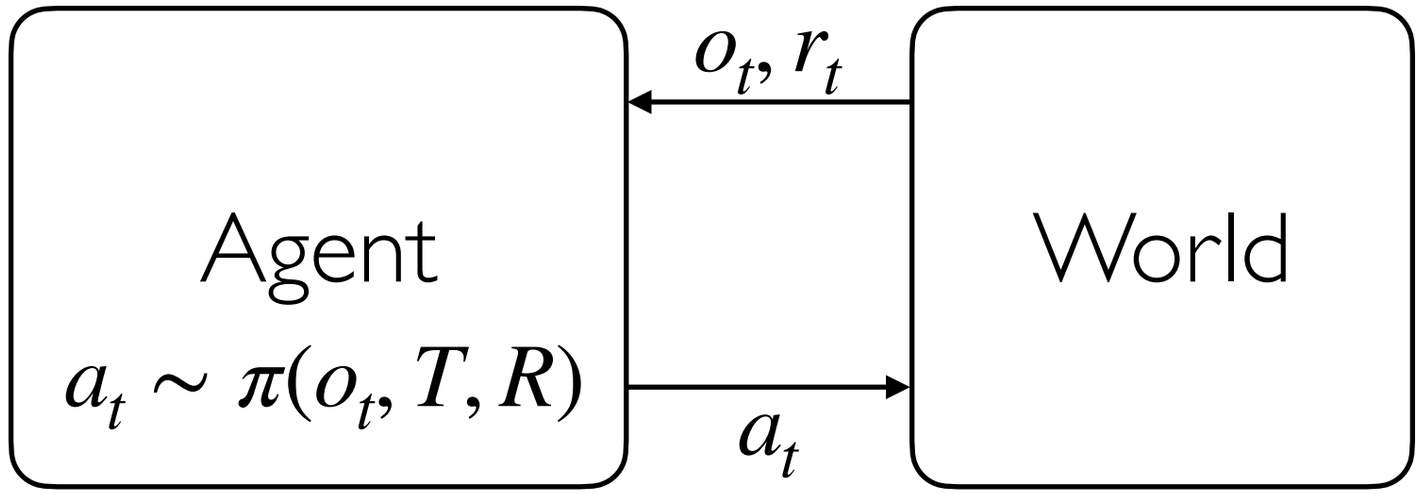

Fully Observed System $o_t = s_t$

Known Transition Function $s_{t+1} \sim T(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL




Fully Observed System $o_t = s_t$

~~Known Transition Function $s_{t+1} \sim T(s_t, a_t)$~~

Learn the Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

# Build Models and Plan with Them

Relax Assumptions For Model-Based RL

Learn the Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

1. Collect data for learning the model:
    1. Initialize system and execute random actions
    2. Collect dataset of triples $(s_{t+1}, s_t, a_t)$
2. Fit a *function* $\hat{T}$ that predicts $s_{t+1}$ given $s_t$ and $a_t$

    Gaussian Process

    Neural Network
3. Plan using function $\hat{T}$

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL

Learn the Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

| | Optimizer | Random Shooting | First Order |
|---|---|---|---|
| Function $\hat{T}$ | | | |
| Gaussian Process | | | PILCO |
| Neural Network | | PETS | Deep PILCO |

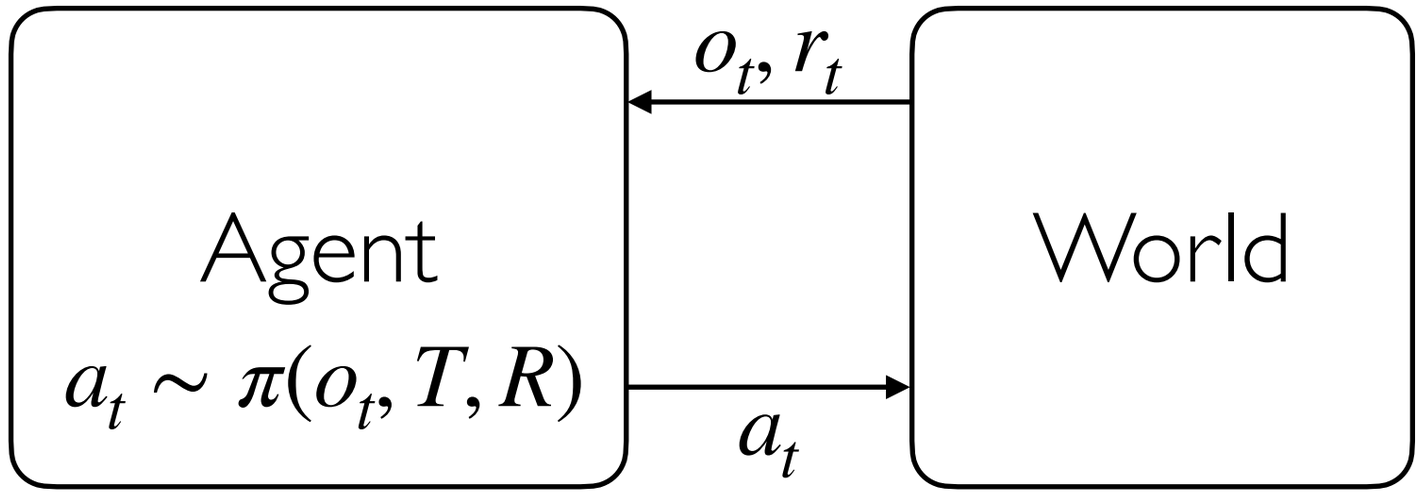

**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

**[PETS]** K. Chua et al. **Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models.** NIPS 2018

**[Deep PILCO]** Y. Gal et al. **Improving PILCO with Bayesian neural network dynamics models.** ICMLW 2016.

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL




Fully Observed System $o_t = s_t$

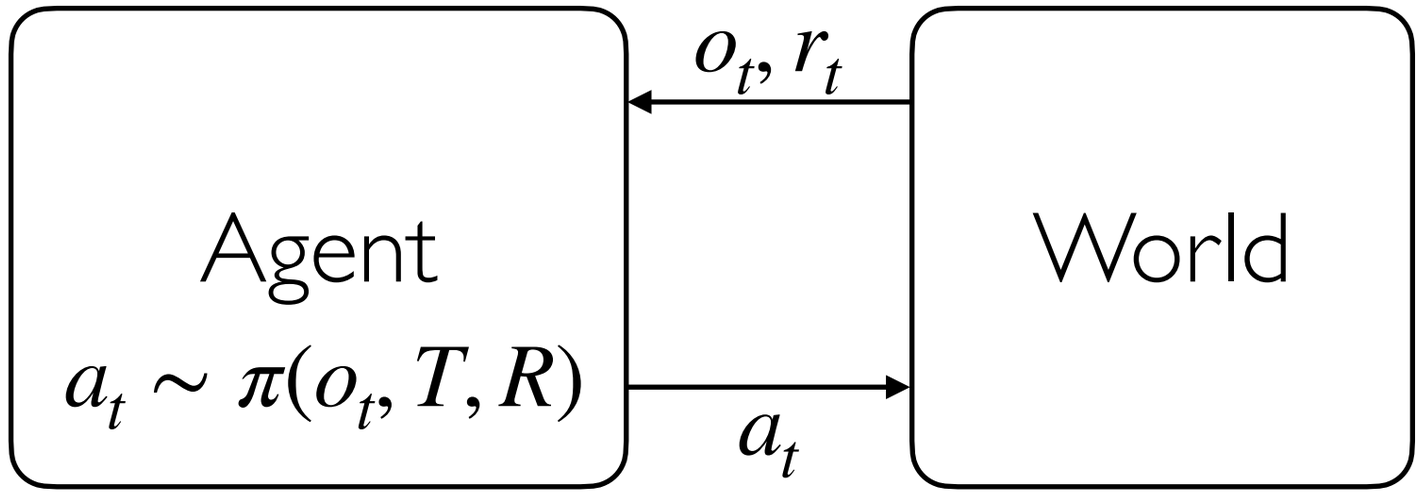
~~Known Transition Function $s_{t+1} \sim T(s_t, a_t)$~~

Learn the Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

Known Reward Function $R(s_{t+1}, s_t, a_t)$

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL




~~Fully Observed System~~ ~~$o_t = s_t$~~

~~Known Transition Function~~ ~~$s_{t+1} \sim T(s_t, a_t)$~~

~~Learn the Transition Function~~ ~~$s_{t+1} \sim \hat{T}(s_t, a_t)$~~

Learn Transition Function (using observations) $o_{t+1} \sim \hat{T}(o_t, a_t)$

~~Known Reward Function~~ ~~$R(s_{t+1}, s_t, a_t)$~~

Reward Function (using observations) $R(o_{t+1}, o_t, a_t)$

# Build Models and Plan with Them

Relax Assumptions For Model-Based RL

Learn Transition Function (using observations) $o_{t+1} \sim \hat{T}(o_t, a_t)$

1. Collect data for learning the model:
    1. Initialize system and execute random actions
    2. Collect dataset of triples $(o_{t+1}, o_t, a_t)$
2. Fit a *function* $\hat{T}$ that predicts $o_{t+1}$ given $o_t$ and $a_t$

~~Gaussian Process~~

Neural Network

3. Plan using function $\hat{T}$

# Build Models and Plan with Them

## Relax Assumptions For Model-Based RL

Learn the Transition Function $s_{t+1} \sim \hat{T}(s_t, a_t)$

| | Optimizer | Random Shooting | First Order |
|---|---|---|---|
| Function $\hat{T}$ | | | |
| Gaussian Process | | | PILCO |
| Neural Network | | PETS | Deep PILCO |


**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

**[PETS]** K. Chua et al. **Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models.** NIPS 2018

**[Deep PILCO]** Y. Gal et al. **Improving PILCO with Bayesian neural network dynamics models.** ICMLW 2016.

# PILCO

**Algorithm 1** PILCO

**init:** Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
Apply random control signals and record data.
**repeat**
  Learn probabilistic (GP) dynamics model, see Sec. 2.1, using all data.
  Model-based policy search, see Sec. 2.2–2.3.
  **repeat**
    Approximate inference for policy evaluation, see Sec. 2.2: get $J^\pi(\theta)$, Eqs. (10)–(12), (24).
    Gradient-based policy improvement, see Sec. 2.3: get $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$, Eqs. (26)–(30).
    Update parameters $\theta$ (e.g., CG or L-BFGS).
  **until** convergence; **return** $\theta^*$
  Set $\pi^* \leftarrow \pi(\theta^*)$.
  Apply $\pi^*$ to system (single trial/episode) and record data.
**until** task learned

**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

# Gaussian Processes

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right)$$

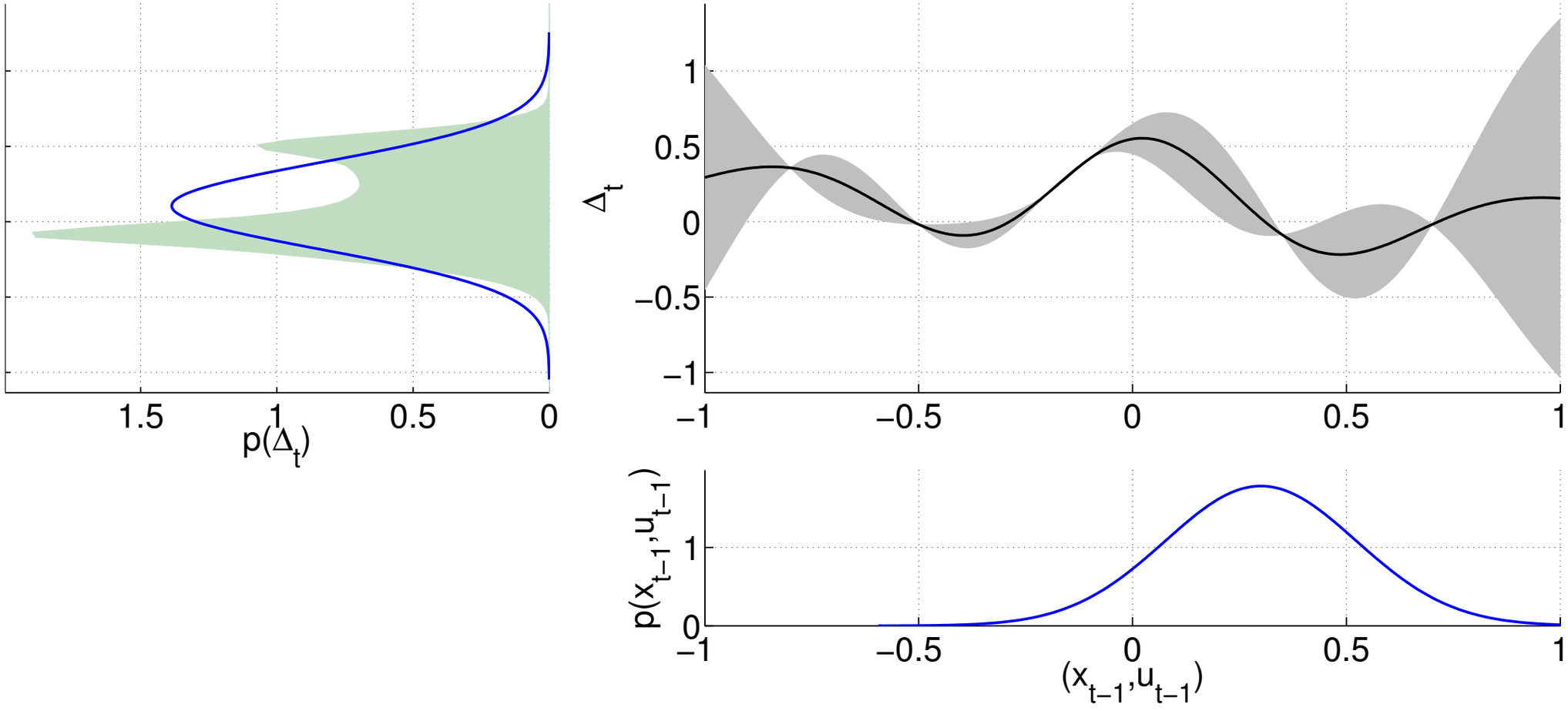
$$\mathbf{f}_* | X_*, X, \mathbf{f} \sim \mathcal{N}\big(K(X_*,X)K(X,X)^{-1}\mathbf{f}, \\ K(X_*,X_*) - K(X_*,X)K(X,X)^{-1}K(X,X_*)\big)$$

http://chifeng.scripts.mit.edu/stuff/gp-demo/

# Policy Evaluation

$$\mathbb{E}_{\mathbf{x}_t}\left[c(\mathbf{x}_t)\right] = \int c(\mathbf{x}_t)\mathcal{N}\left(\mathbf{x}_t \,|\, \mu_t, \mathbf{\Sigma}_t\right) \mathrm{d}\mathbf{x}_t \,,$$

Given $p(x_0)$ and policy $\pi(\theta)$, want to compute $p(x_t)$.



# Policy Improvement

$$\mathbb{E}_{\mathbf{x}_t}\left[c(\mathbf{x}_t)\right] = \int c(\mathbf{x}_t)\mathcal{N}\left(\mathbf{x}_t \,|\, \mu_t, \mathbf{\Sigma}_t\right) \mathrm{d}\mathbf{x}_t \,,$$

Compute gradient with respect to $\theta$
using chain rule.

# PILCO

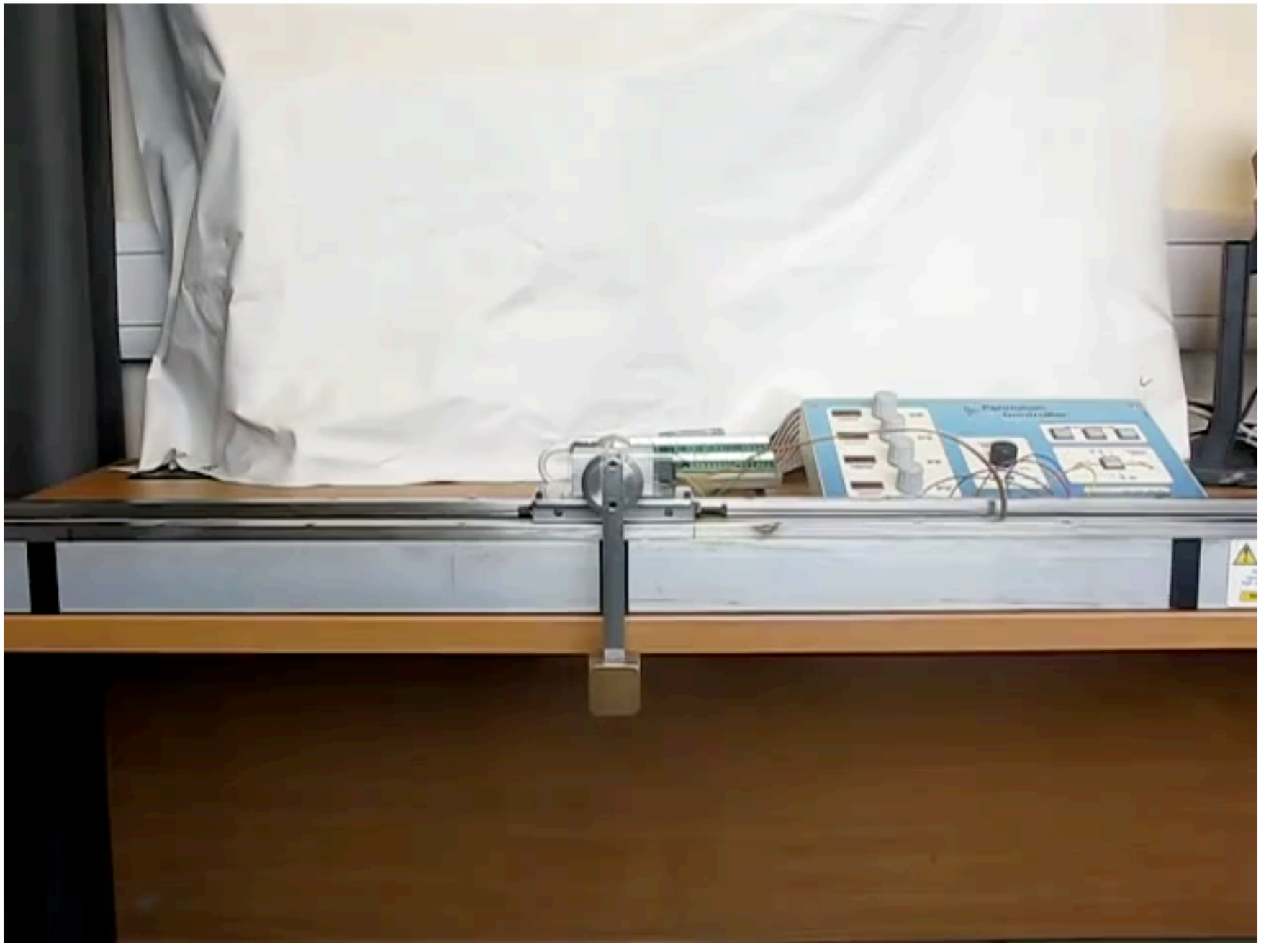
**Algorithm 1** PILCO

**init:** Sample controller parameters $\theta \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
Apply random control signals and record data.
**repeat**
  Learn probabilistic (GP) dynamics model, see Sec. 2.1, using all data.
  Model-based policy search, see Sec. 2.2–2.3.
  **repeat**
    Approximate inference for policy evaluation, see Sec. 2.2: get $J^\pi(\theta)$, Eqs. (10)–(12), (24).
    Gradient-based policy improvement, see Sec. 2.3: get $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$, Eqs. (26)–(30).
    Update parameters $\theta$ (e.g., CG or L-BFGS).
  **until** convergence; **return** $\theta^*$
  Set $\pi^* \leftarrow \pi(\theta^*)$.
  Apply $\pi^*$ to system (single trial/episode) and record data.
**until** task learned

**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

# Build Models and Plan with Them

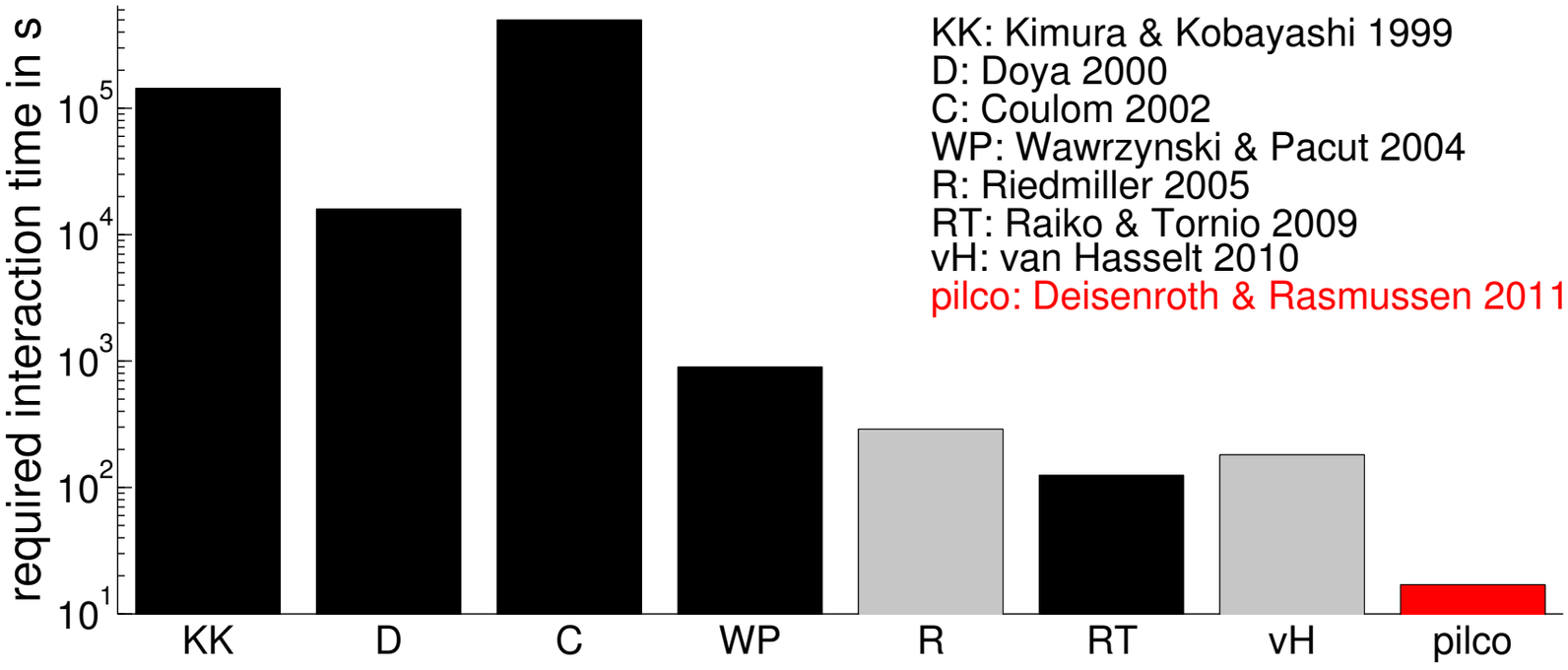
## PILCO - Inverting a pendulum



# PILCO

*Table 1.* PILCO's data efficiency scales to high dimensions.

| | cart-pole | cart-double-pole | unicycle |
|---|---|---|---|
| state space | $\mathbb{R}^4$ | $\mathbb{R}^6$ | $\mathbb{R}^{12}$ |
| # trials | $\leq 10$ | 20–30 | $\approx 20$ |
| experience | $\approx 20\,\mathrm{s}$ | $\approx 60\,\mathrm{s}$–$90\,\mathrm{s}$ | $\approx 20\,\mathrm{s}$–$30\,\mathrm{s}$ |
| parameter space | $\mathbb{R}^{305}$ | $\mathbb{R}^{1816}$ | $\mathbb{R}^{28}$ |



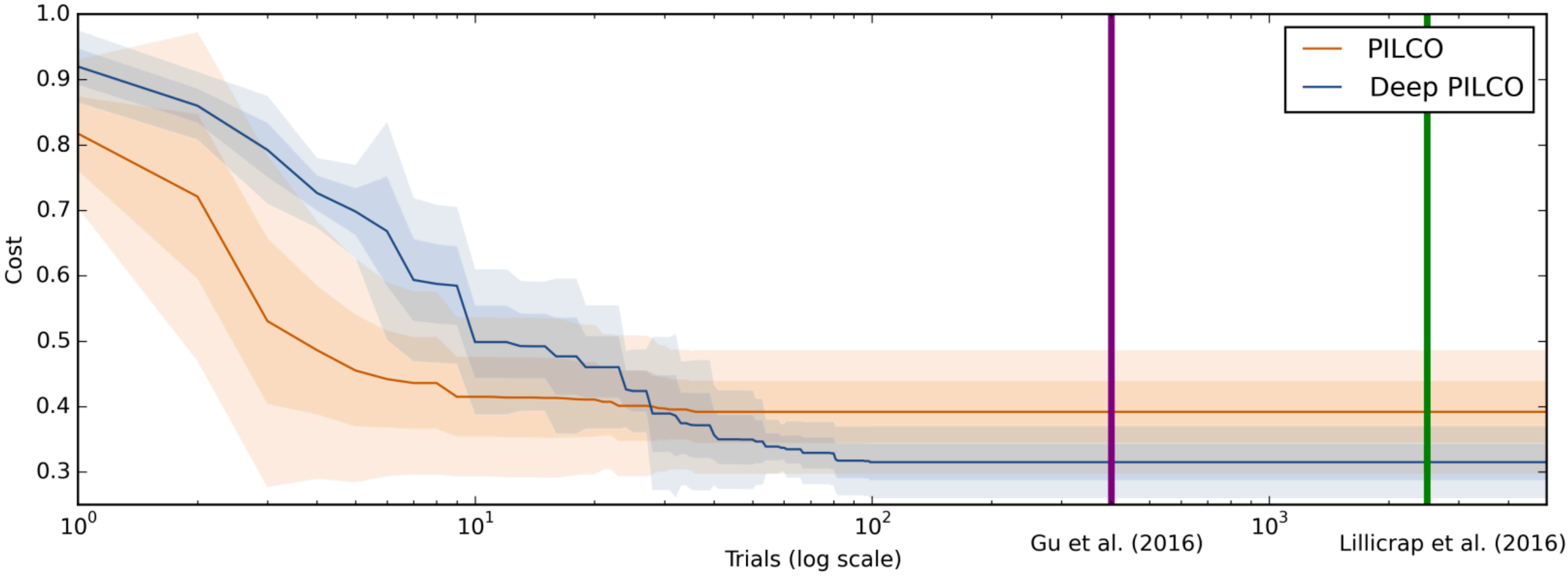

**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

# Build Models and Plan with Them

## PILCO - Inverting a pendulum

**Improving PILCO with Bayesian Neural Network Dynamics Models**

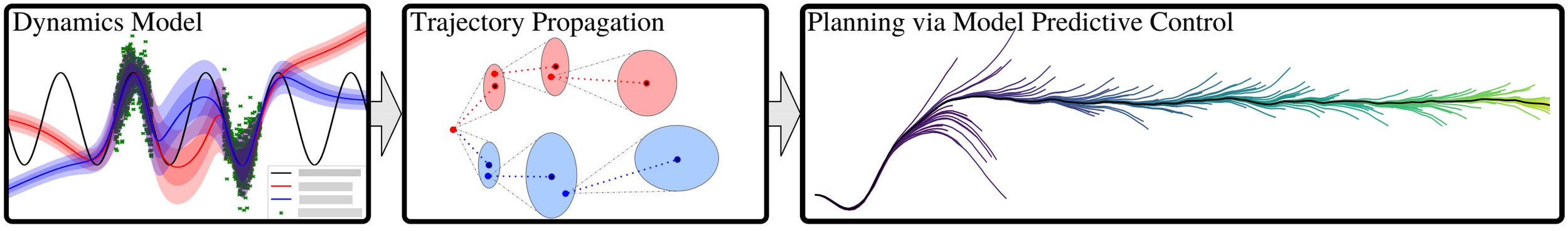
Yarin Gal and Rowan Thomas McAllister and Carl Edward Rasmussen[1]




**[PILCO]** M. Deisenroth et al. **PILCO: A Model-based and Data-Efficient Approach to Policy Search.** ICML 2011

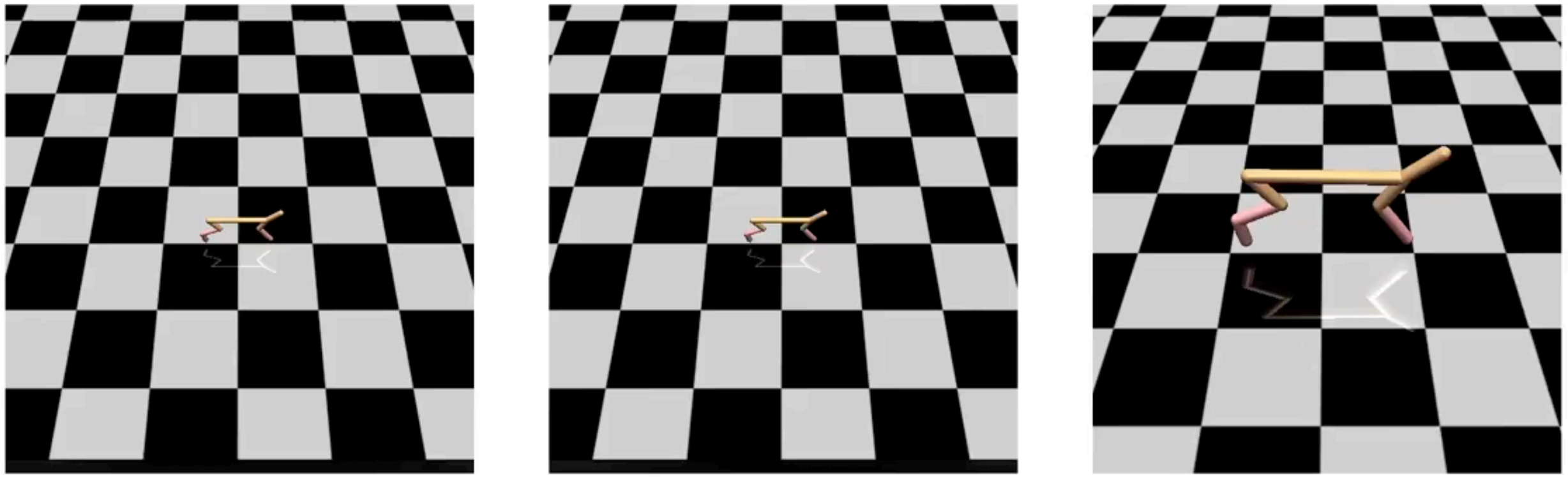
# PETS

- Employ Neural Network based probabilistic models for Model-based RL
- Use an ensemble of probabilistic models
- Act at test time using CEM method



# Build Models and Plan with Them

## Probabilistic Ensembles with Trajectory Sampling (PETS)




**[PETS]** K. Chua et al. **Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models.** NIPS 2018

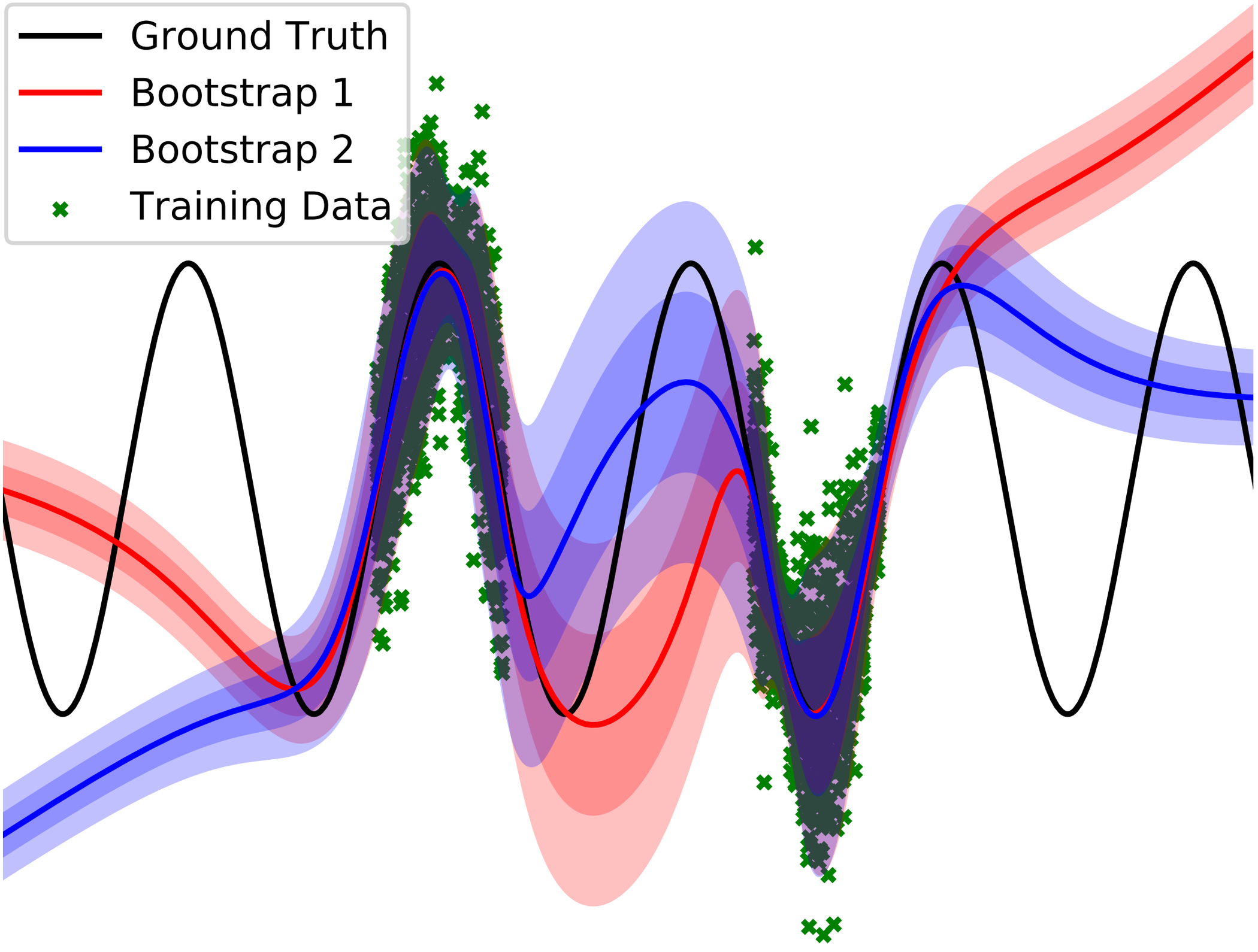
# Uncertainties

- Aleatoric uncertainty (inherent stochasticity)
- Epistemic uncertainty (lack of data)

Table 1: Model uncertainties captured.

| **Model** | **Aleatoric uncertainty** | **Epistemic uncertainty** |
|---|---|---|
| *Baseline Models* | | |
| Deterministic NN (D) | No | No |
| Probabilistic NN (P) | Yes | No |
| Deterministic ensemble NN (DE) | No | Yes |
| Gaussian process baseline (GP) | Homoscedastic | Yes |
| *Our Model* | | |
| Probabilistic ensemble NN (PE) | **Yes** | **Yes** |

# Uncertainties



# Planning

- Model predictive control

$$\boldsymbol{a}_{t:t+T} \doteq \{\boldsymbol{a}_t, \ldots, \boldsymbol{a}_{t+T}\}$$

$$\arg\max_{\boldsymbol{a}_{t:t+T}} \sum_{\tau=t}^{t+T} \mathbb{E}_{\tilde{f}}[r(\boldsymbol{s}_\tau, \boldsymbol{a}_\tau)].$$

- Cross entropy method

1. *Choose an initial parameter vector* $\widehat{\mathbf{v}}_0$. *Let* $N^{\mathrm{e}} = \lceil \varrho N \rceil$. *Set* $t = 1$ *(level counter).*

2. *Generate* $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\mathrm{iid}} f(\cdot; \widehat{\mathbf{v}}_{t-1})$. *Calculate the performances* $S(\mathbf{X}_i)$ *for all* $i$, *and order them from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t$ *be the sample* $(1-\varrho)$*-quantile of performances; that is,* $\widehat{\gamma}_t = S_{(N-N^{\mathrm{e}}+1)}$.

3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and solve the stochastic program*

$$\max_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} \ln f(\mathbf{X}_k; \mathbf{v}) \,. \tag{9}$$

   *Denote the solution by* $\widehat{\mathbf{v}}_t$.

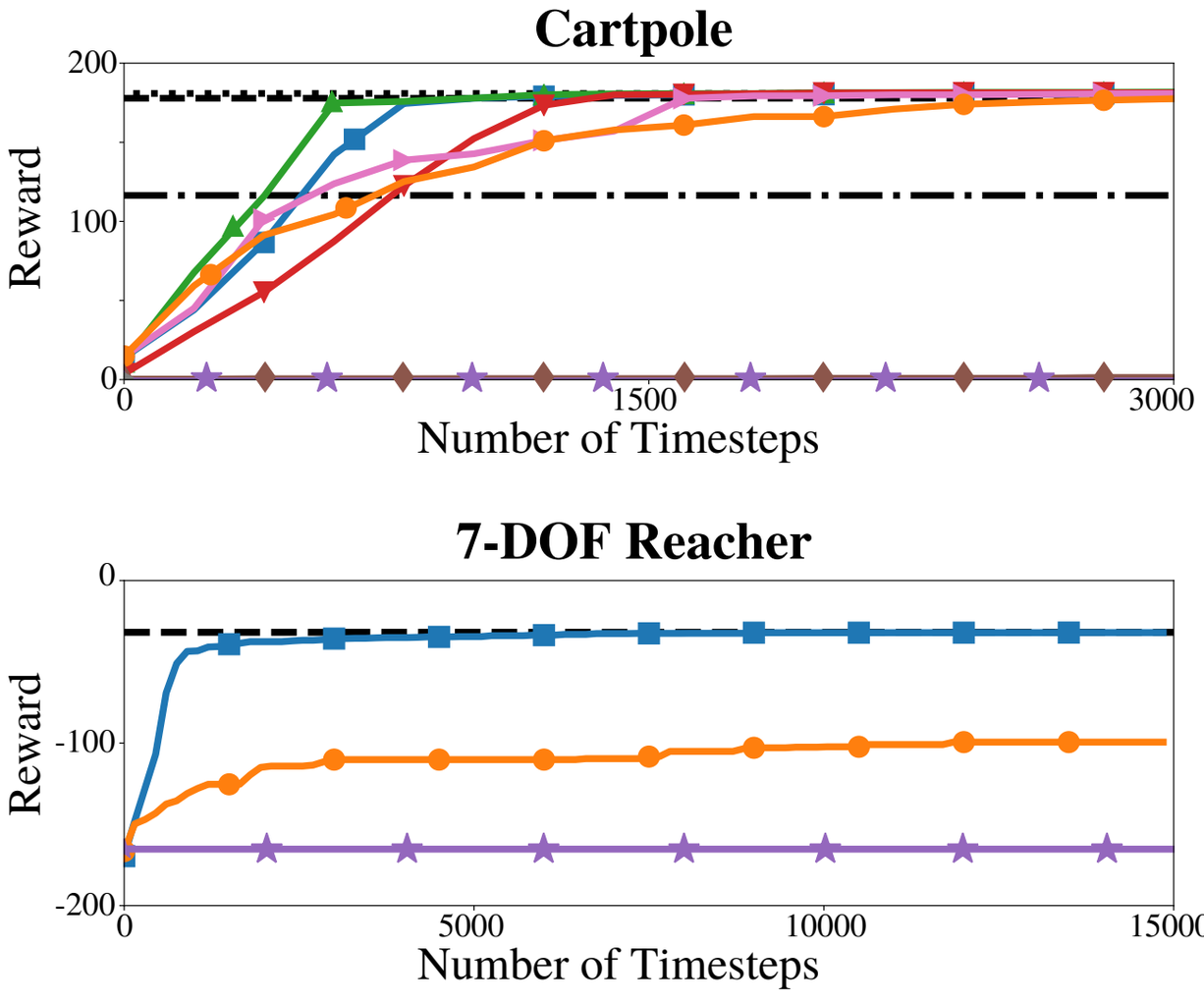
4. *If some stopping criterion is met, stop; otherwise, set* $t = t+1$, *and return to Step 2.*

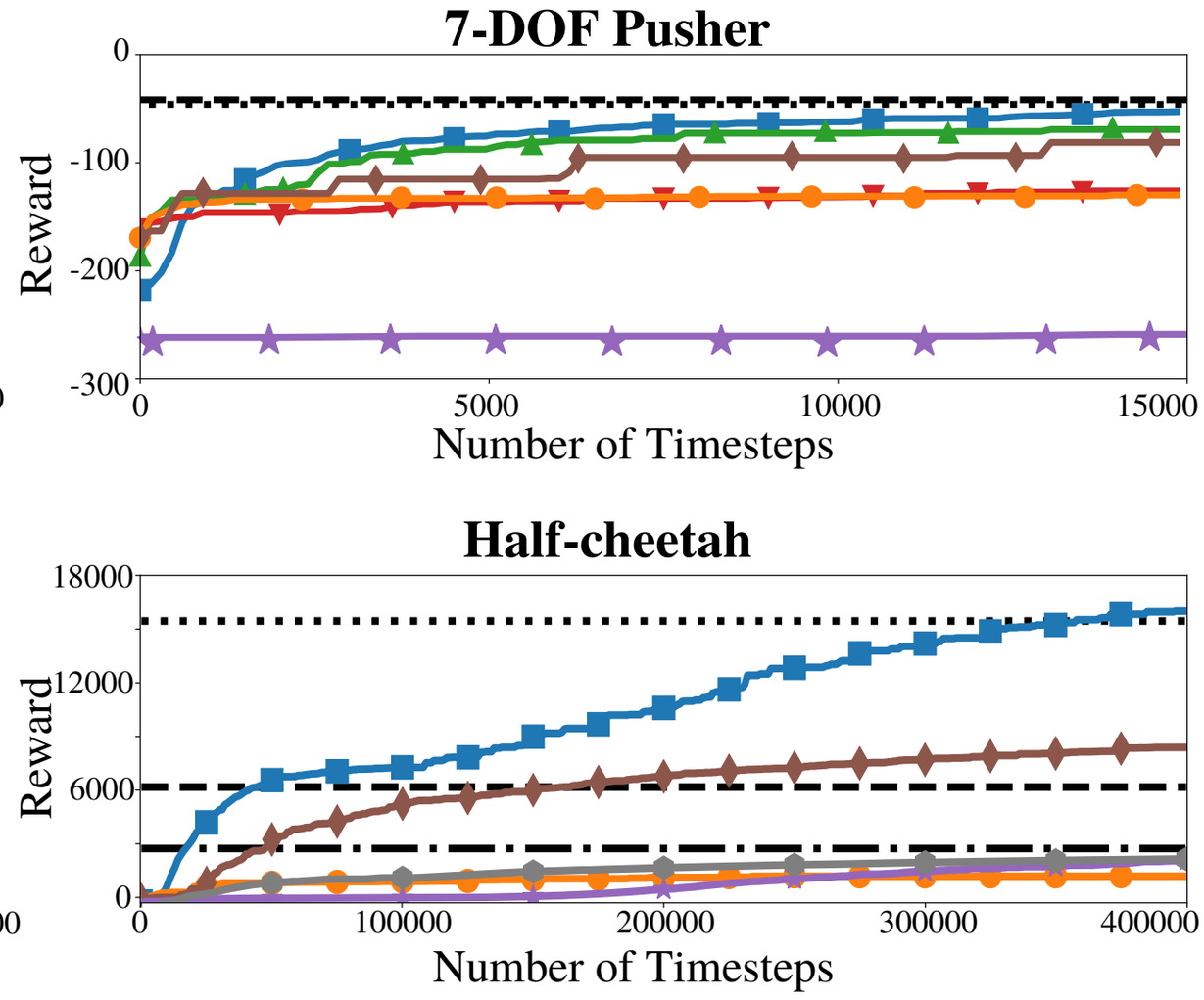
# PETS

**Algorithm 1** Our model-based MPC algorithm '*PETS*':

Initialize data $\mathbb{D}$ with a random controller for one trial.
**for** Trial $k = 1$ to $K$ **do**
    Train a *PE* dynamics model $\widetilde{f}$ given $\mathbb{D}$.
    **for** Time $t = 0$ to TaskHorizon **do**
        **for** Actions sampled $\boldsymbol{a}_{t:t+T} \sim \text{CEM}(\cdot)$, 1 to NSamples **do**
            Propagate state particles $\boldsymbol{s}_{\tau}^{p}$ using *TS* and $\widetilde{f}\big|\{\mathbb{D}, \boldsymbol{a}_{t:t+T}\}$.
            Evaluate actions as $\sum_{\tau=t}^{t+T} \frac{1}{P} \sum_{p=1}^{P} r(\boldsymbol{s}_{\tau}^{p}, \boldsymbol{a}_{\tau})$
        Update CEM$(\cdot)$ distribution.
    Execute first action $\boldsymbol{a}_{t}^{*}$ (only) from optimal actions $\boldsymbol{a}_{t:t+T}^{*}$.
    Record outcome: $\mathbb{D} \leftarrow \mathbb{D} \cup \{\boldsymbol{s}_{t}, \boldsymbol{a}_{t}^{*}, \boldsymbol{s}_{t+1}\}$.

# Experiments










# Experiments

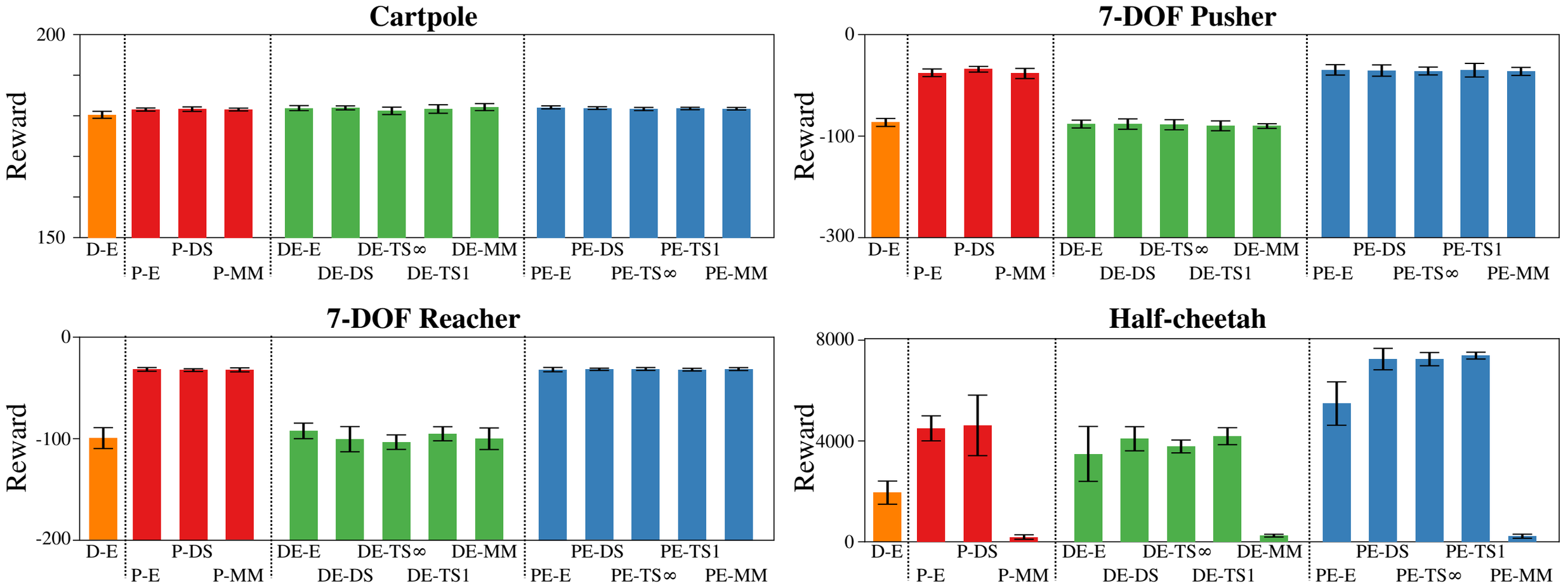



Figure 4: Final performance for different tasks, models, and uncertainty propagation techniques. The model choice seems to be more important than the technique used to propagate the state/action space. Among the models the ranking in terms of performance is: $PE > P > DE > D$. A linear model comparison can also be seen in Appendix A.10.

Thank you