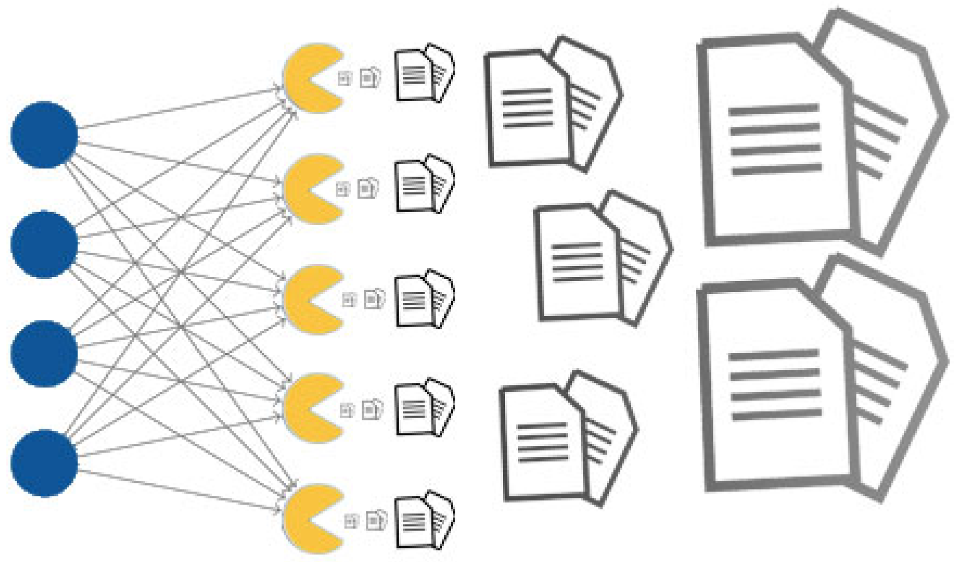
# Deep RL with a Handful of Trials

## Obtaining Data-Efficiency with Bayesian Neural Network Dynamics Models

**Yarin Gal, Rowan McAllister, Carl Rasmussen** {yg279,rtm26,cer54}@cam.ac.uk

UNIVERSITY OF CAMBRIDGE

## Problem

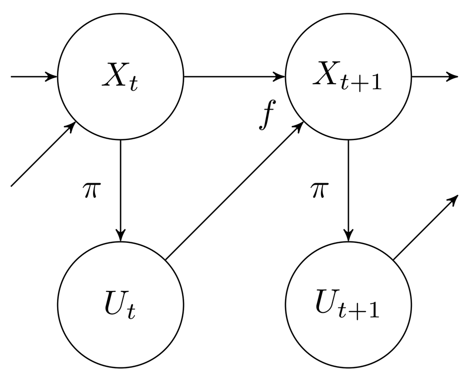


In many tasks, data efficiency is **critical**; but deep reinforcement learning is inherently data **inefficient**.

**How can we get deep RL to be data efficient?**

## Possible Solution: PILCO

**PILCO** = data-efficient RL framework exploiting **probabilistic dynamics models**




- With **dynamics models** agents can **generalise knowledge** on system dynamics to unobserved states
- But selecting a **single dynamics model** from a large plausible set would lead to **model bias**
  - state prediction after many time steps is random noise
- **Dynamics uncertainty is crucial**; it focuses policy optimisation towards policy changes that are **more certain** to have effects
- PILCO avoids model bias by **considering all plausible dynamics** models in prediction

**PILCO Algorithm**:

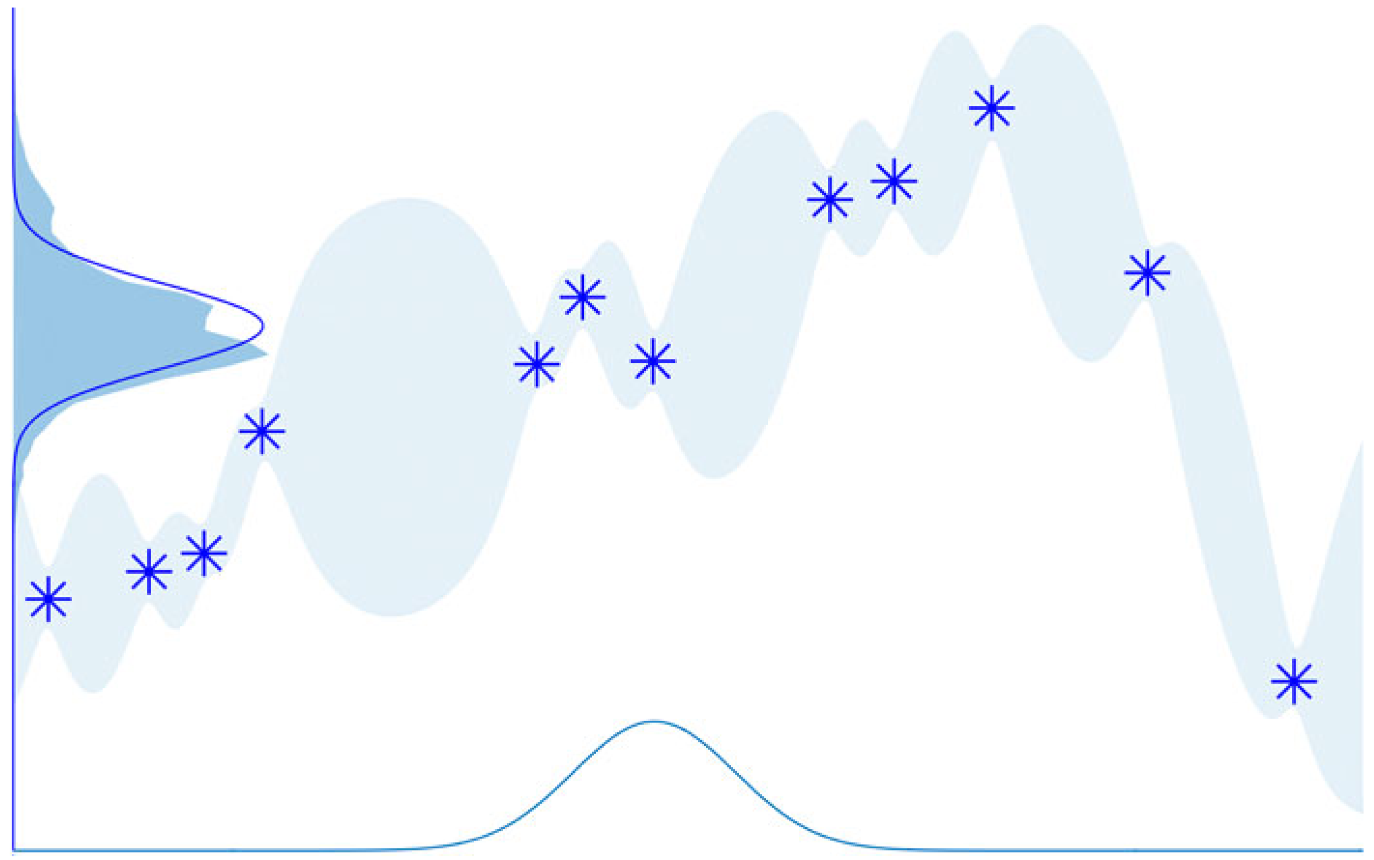
1: *Define* policy's functional form: $\pi : z_t \times \psi \to u_t$.
2: *Initialise* policy parameters $\psi$ randomly.
3: **repeat**
4: *Execute* system, record data.
5: *Learn* dynamics model.
6: *Predict* system trajectories from $p(X_0)$ to $p(X_T)$.
7: *Evaluate* policy: $J(\psi) = \sum_{t=0}^{T} \gamma^t \mathbb{E}_X[\text{cost}(X_t)|\psi]$.
8: *Optimise* policy: $\psi \leftarrow \text{argmin}_\psi J(\psi)$.
9: **until** policy parameters $\psi$ converge

PILCO uses Gaussian processes to model dynamics; but these don't scale to high dimensional observation spaces...

## Model Based Deep RL

**Main idea**: use the PILCO framework with deep neural network dynamics models. **But for this we need**...

1. **Output uncertainty**: dynamics model has to capture **model's ignorance** about system dynamics.
2. **Input uncertainty**: PILCO **propagates state distributions** (step 6) through dynamics model. Uncertain dynamics outputs are passed between time steps as uncertain inputs to dynamics model in following time steps.



Depiction of probabilistic dynamics model with input and output distributions

**Our approach**...

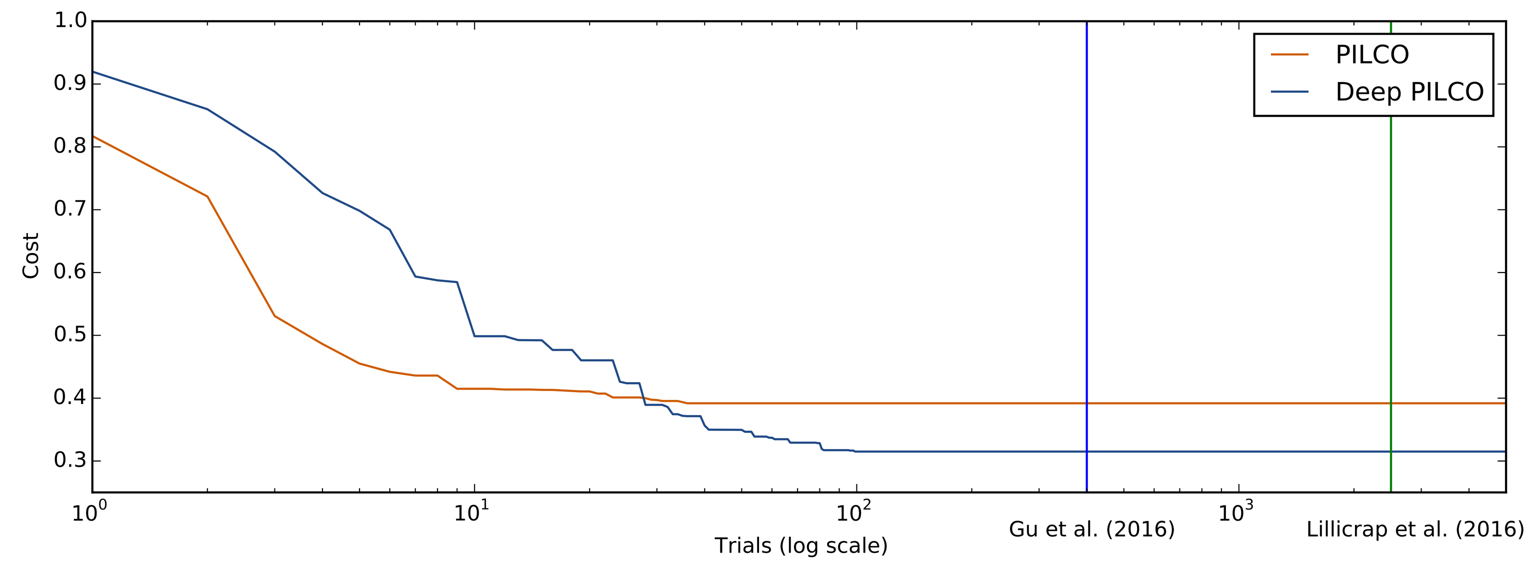
1. Output uncertainty: use **Bayesian neural networks** with dropout approximate inference, MC sampling to estimate uncertainty [Gal 2015]
2. Following RNN dropout [Gal 2015], sample mask and fix through time = **draw function realisation from belief over dynamics**
3. Input uncertainty: **propagate particles** through time [McHutchon, 2014], and moment-match output distribution every time step

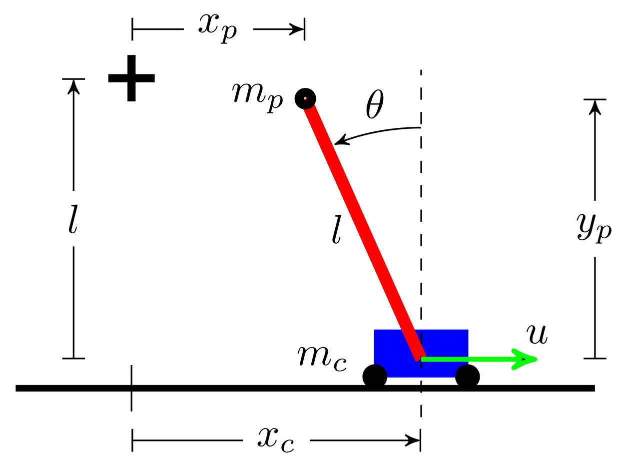
**Deep PILCO Algorithm** (adapting step 6 in PILCO Algorithm):

1: *Define* time horizon $T$.
2: *Initialise* set of $K$ particles $x_0^k \sim P(X_0)$.
3: **for** $k = 1$ to $K$ **do**
4: Sample BNN dynamics model weights $W^k$.
5: **end for**
6: **for** time $t = 1$ to $T$ **do**
7: **for** each particle $x_t^1$ to $x_t^K$ **do**
8: Evaluate BNN with weights $W^k$ and input particle $x_t^k$, obtain output $y_t^k$.
9: **end for**
10: Calculate mean $\mu_t$ and standard deviation $\sigma_t^2$ of $\{y_t^1, ..., y_t^K\}$.
11: Sample set of $K$ particles $x_{t+1}^k \sim \mathcal{N}(\mu_t, \sigma_t^2)$.
12: **end for**

## Proof of Concept

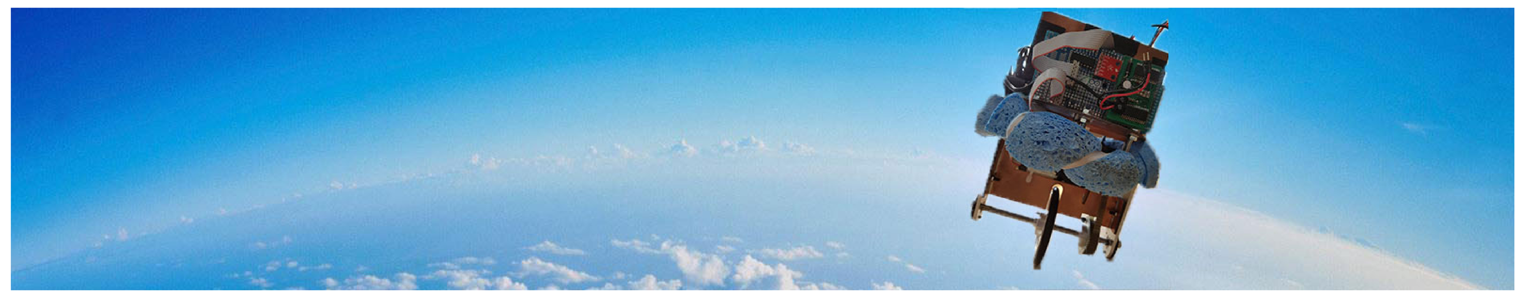
- Evaluated the ideas above on the **cartpole swing-up** benchmark




- Used *low-dimensional* state representation
- Improved from Lillicrap el al. [2015]'s thousands of trials and Gu el al. [2015]'s hundreds down to a **handful of trials**
- Close to PILCO's state-of-the-art data efficiency, obtaining **lower cost than PILCO**'s by modelling time dependence
- **Faster** running time and lower time complexity compared to PILCO

## What's Next



A series of **exciting challenges** extending this work:

- How do we model dynamics with **high dim. observation spaces**?
  - Need to predict next high dim. state
  - Current approaches train **auto-encoders** (AEs), and use decoder in dynamics model
  - But pre-training AEs **requires many observations** – too costly
  - And there's no need for the **AE decoding itself** for RL!
  - Is there a **better approach?**
- Better **uncertainty estimates**?
  - Dropout's uncertainty estimates are **cheap**
  - But can we get **improved estimates** and improve data efficiency?
- Capture **multi-modal state distributions**?
  - We **moment-matched** the output distribution to **simplify controller optimisation**
  - Can we **avoid** moment-matching?