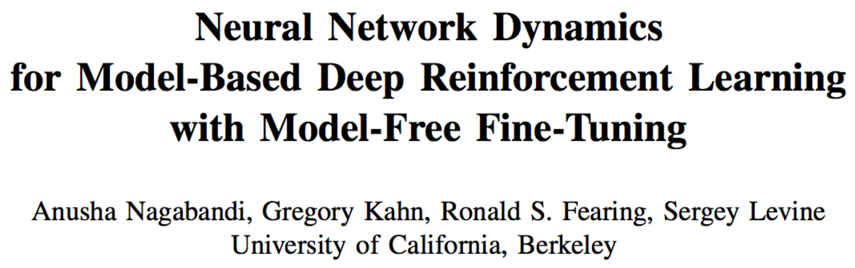
Model-based Reinforcement Learning

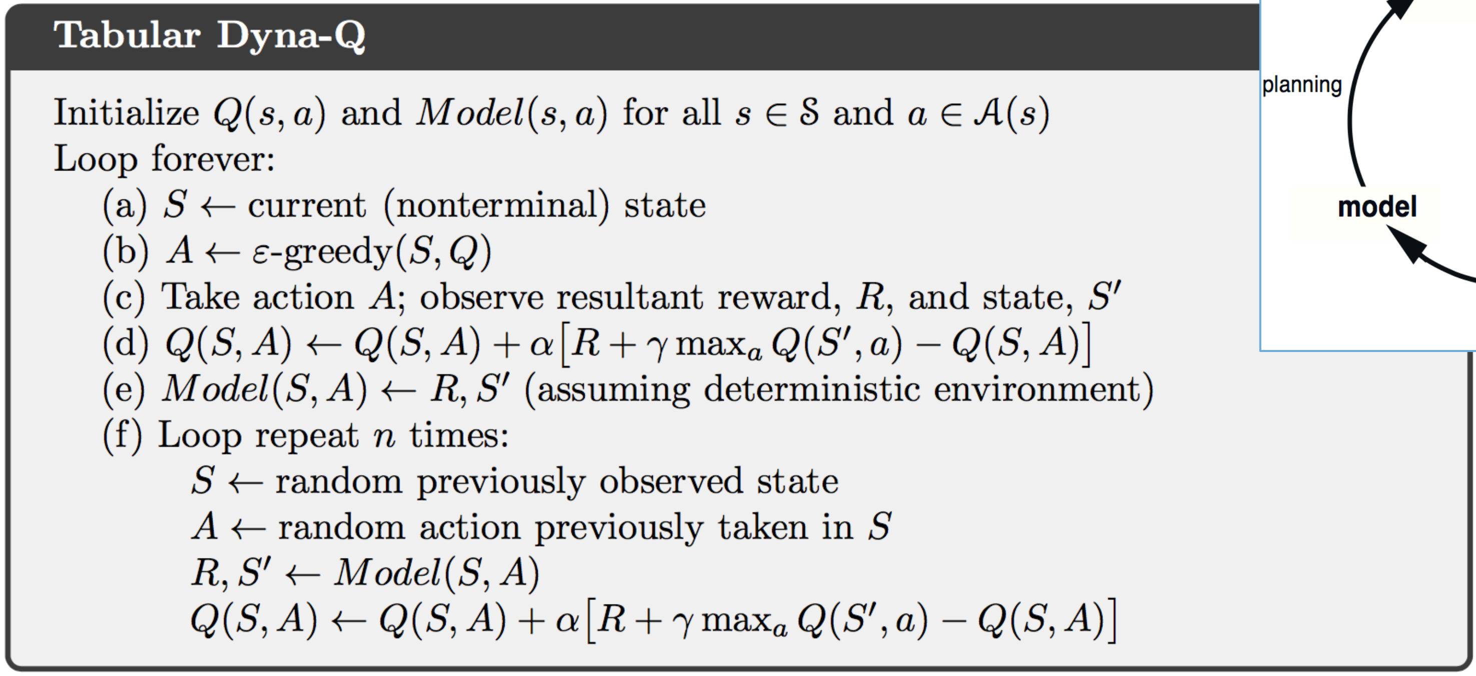
- one example



Neural Network Dynamics
for Model-Based Deep Reinforcement Learning
with Model-Free Fine-Tuning

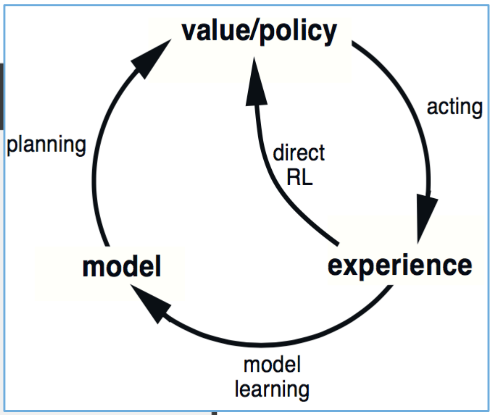
Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, Sergey Levine
University of California, Berkeley

Basic Model-based RL



Tabular Dyna-Q

```
Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Loop forever:
    (a) $S \leftarrow$ current (nonterminal) state
    (b) $A \leftarrow \varepsilon$-greedy$(S,Q)$
    (c) Take action $A$; observe resultant reward, $R$, and state, $S'$
    (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
    (f) Loop repeat $n$ times:
            $S \leftarrow$ random previously observed state
            $A \leftarrow$ random action previously taken in $S$
            $R, S' \leftarrow Model(S,A)$
            $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
```



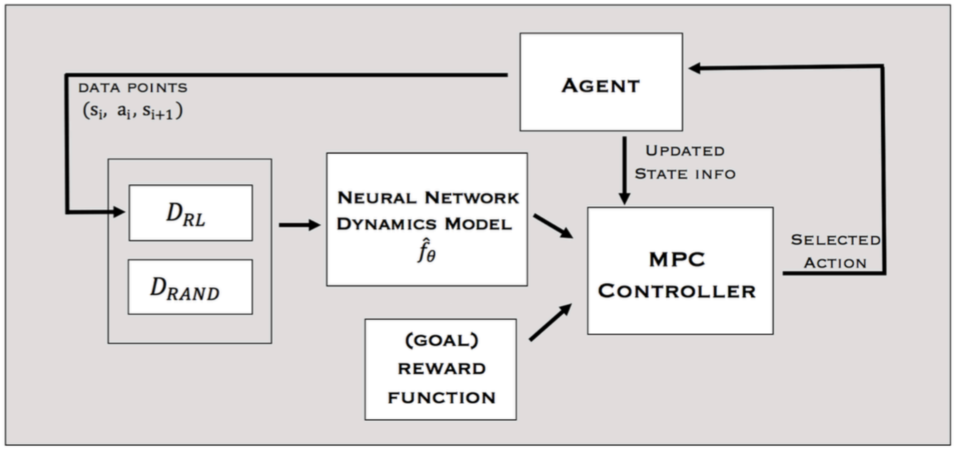
[Sutton, p164]

Algorithm 1 Model-based Reinforcement Learning

```
1: gather dataset $\mathcal{D}_{\text{RAND}}$ of random trajectories
2: initialize empty dataset $\mathcal{D}_{\text{RL}}$, and randomly initialize $\hat{f}_\theta$
3: for iter=1 to max_iter do
4:     train $\hat{f}_\theta(\mathbf{s}, \mathbf{a})$ by performing gradient descent on Eqn. 2,
       using $\mathcal{D}_{\text{RAND}}$ and $\mathcal{D}_{\text{RL}}$
5:     for $t = 1$ to $T$ do
6:         get agent's current state $\mathbf{s}_t$
7:         use $\hat{f}_\theta$ to estimate optimal action sequence $\mathbf{A}_t^{(H)}$
           (Eqn. 4)
8:         execute first action $\mathbf{a}_t$ from selected action sequence
           $\mathbf{A}_t^{(H)}$
9:         add $(\mathbf{s}_t, \mathbf{a}_t)$ to $\mathcal{D}_{\text{RL}}$
10:    end for
11: end for
```

use of dynamics:

$$\hat{\mathbf{s}}_{t+1} = \mathbf{s}_t + \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t).$$



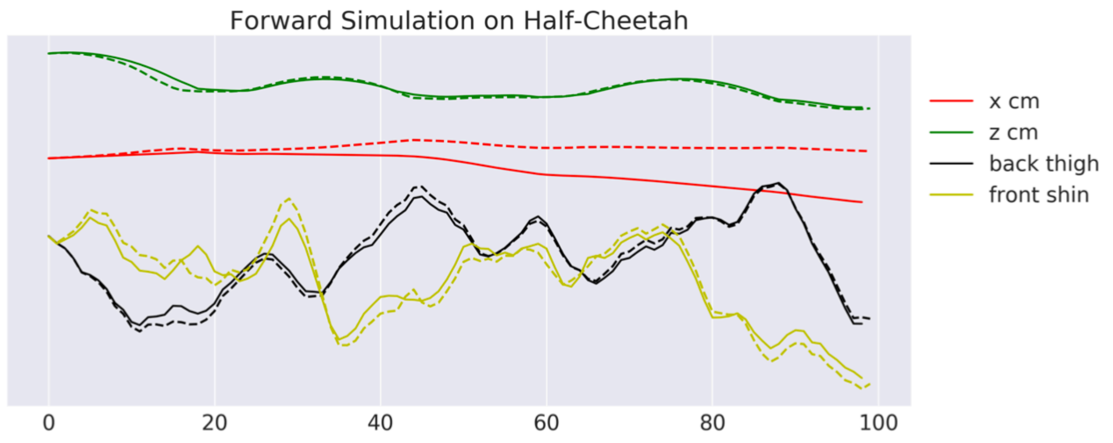


Fig. 5: Given a fixed sequence of controls, we show the resulting true rollout (solid line) vs. the multi-step prediction from the learned dynamics model (dotted line) on the half-cheetah agent. Although we learn to predict certain elements of the state space well, note the eventual divergence of the learned model on some state elements when it is used to make multi-step open-loop predictions. However, our MPC-based controller with a short horizon can succeed in using the model to control an agent.

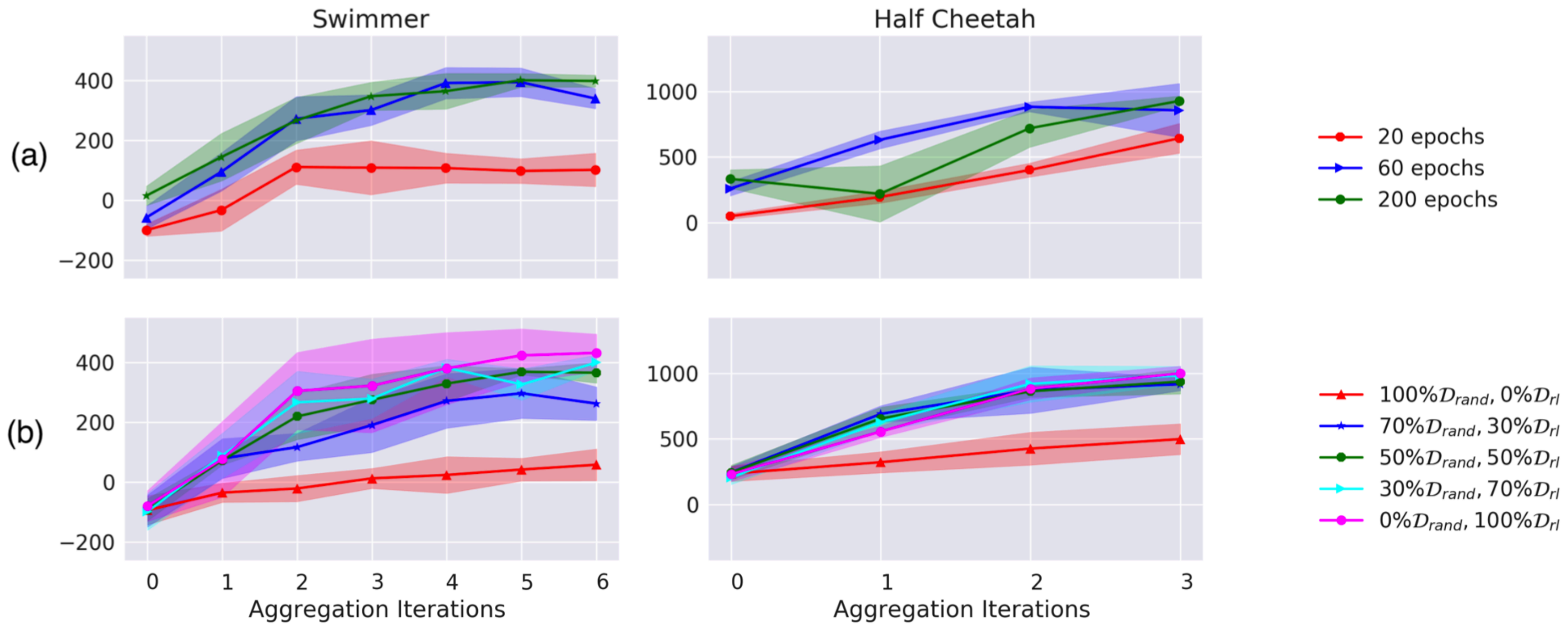


Fig. 6: Analysis of design decisions for our model-based reinforcement learning approach. (a) Training steps, (b) dataset training split, (c) horizon and number of actions sampled, (d) initial random trajectories. Training for more epochs, leveraging on-policy data, planning with medium-length horizons and many action samples were the best design choices, while data aggregation caused the number of initial trajectories that have little effect.

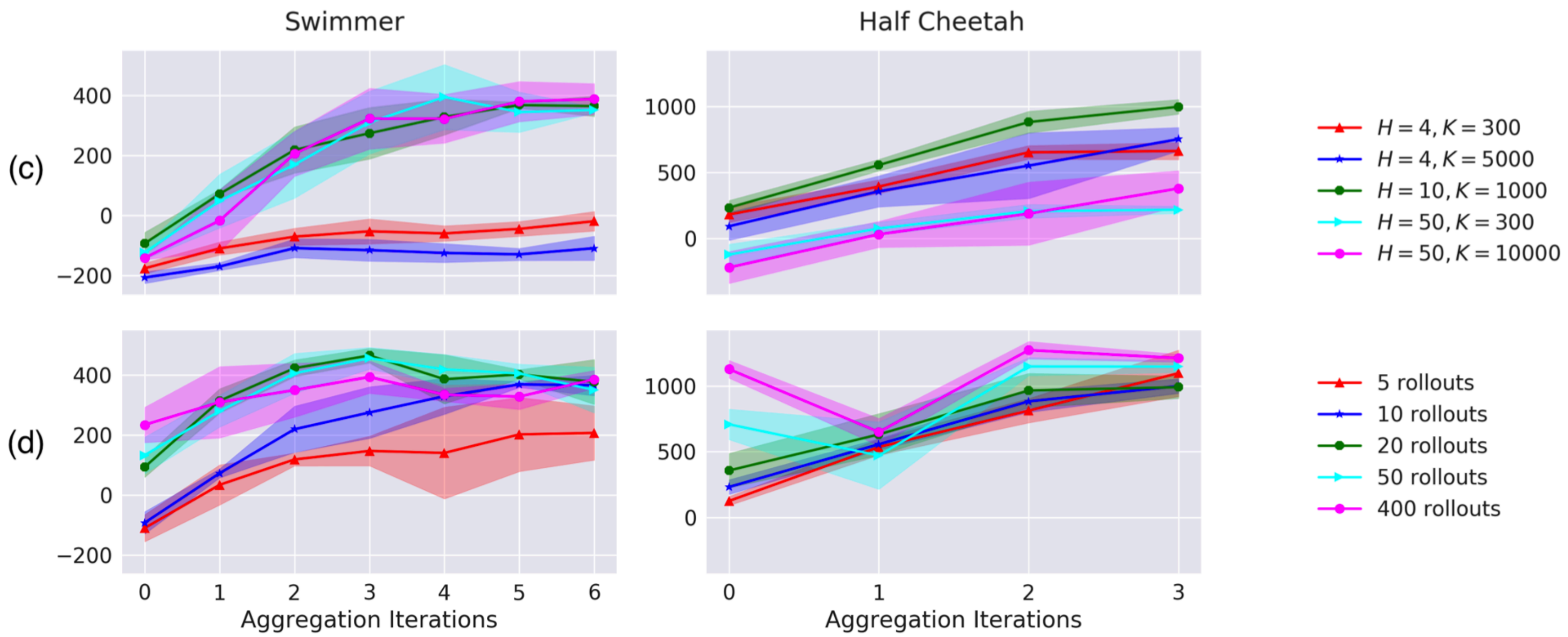


Fig. 6: Analysis of design decisions for our model-based reinforcement learning approach. (a) Training steps, (b) dataset training split, (c) horizon and number of actions sampled, (d) initial random trajectories. Training for more epochs, leveraging on-policy data, planning with medium-length horizons and many action samples were the best design choices, while data aggregation caused the number of initial trajectories that have little effect.

Model-based + Model-free

- train with model-based MPC method, as described
- imitation learning
  - intialize a learned policy using behavior cloning
  - iterative policy training   (DAGGER)
    - perform on-policy roll-outs
    - query MPC “expert” for the “true” actions for the visited states
- fine-tune using model-free RL, e.g., policy gradient with TRPO

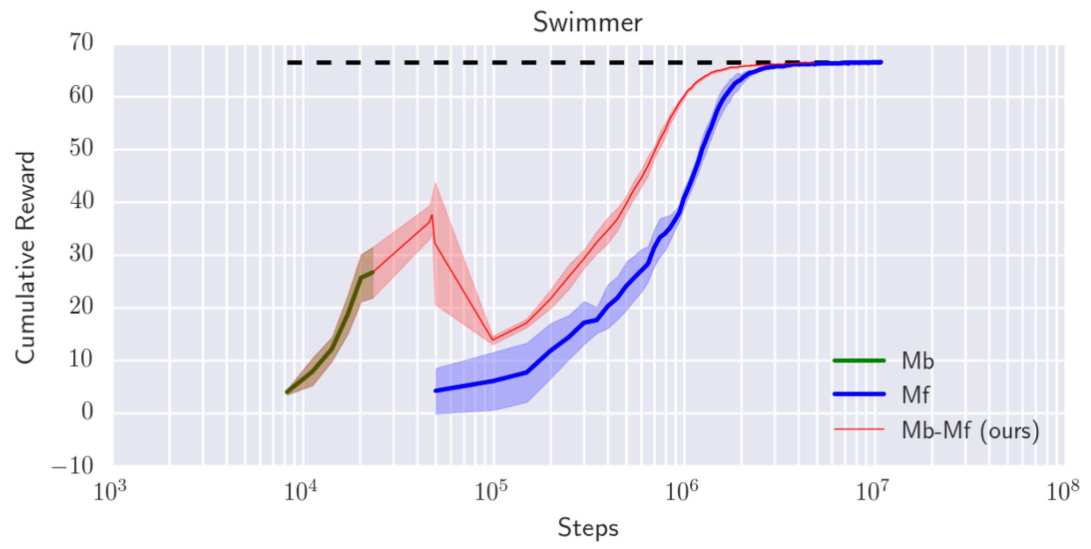
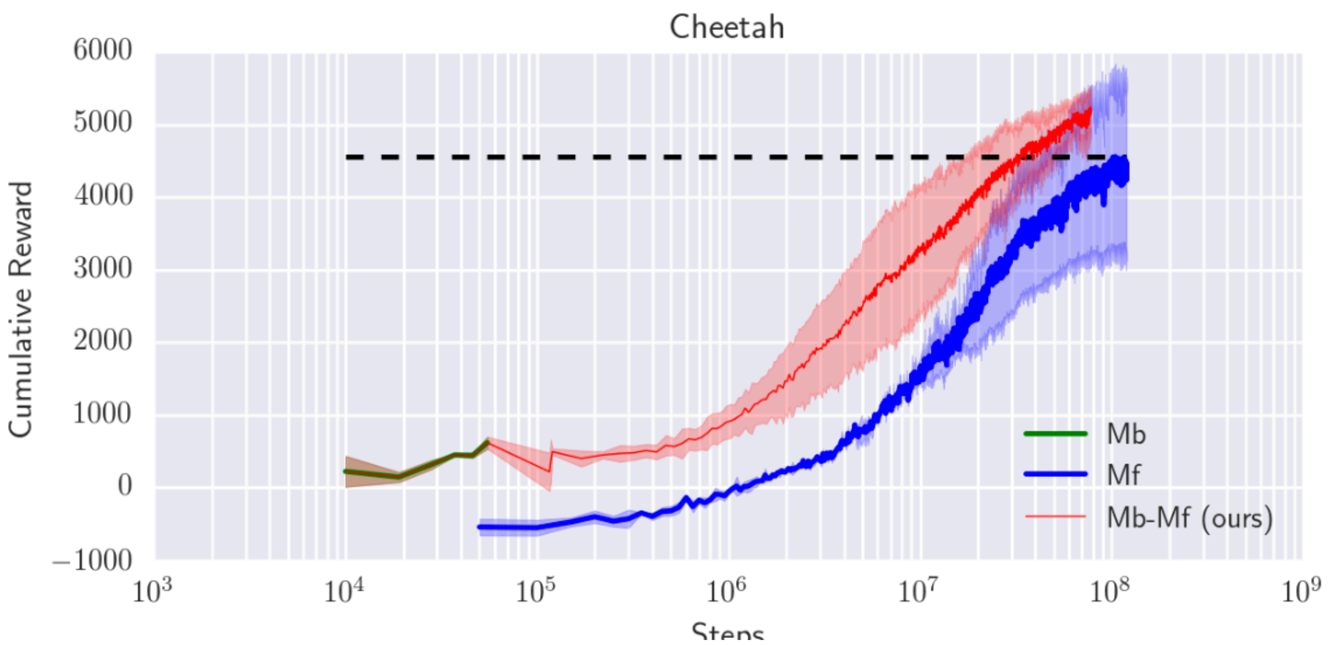
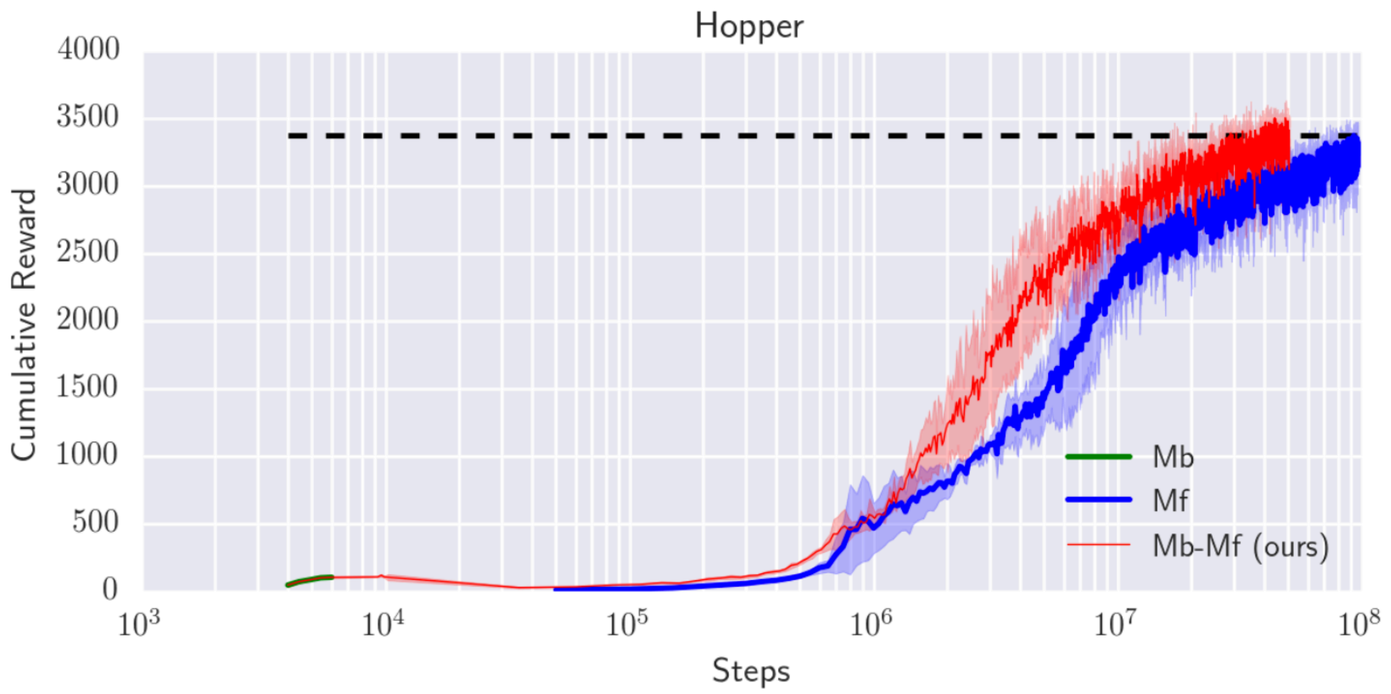
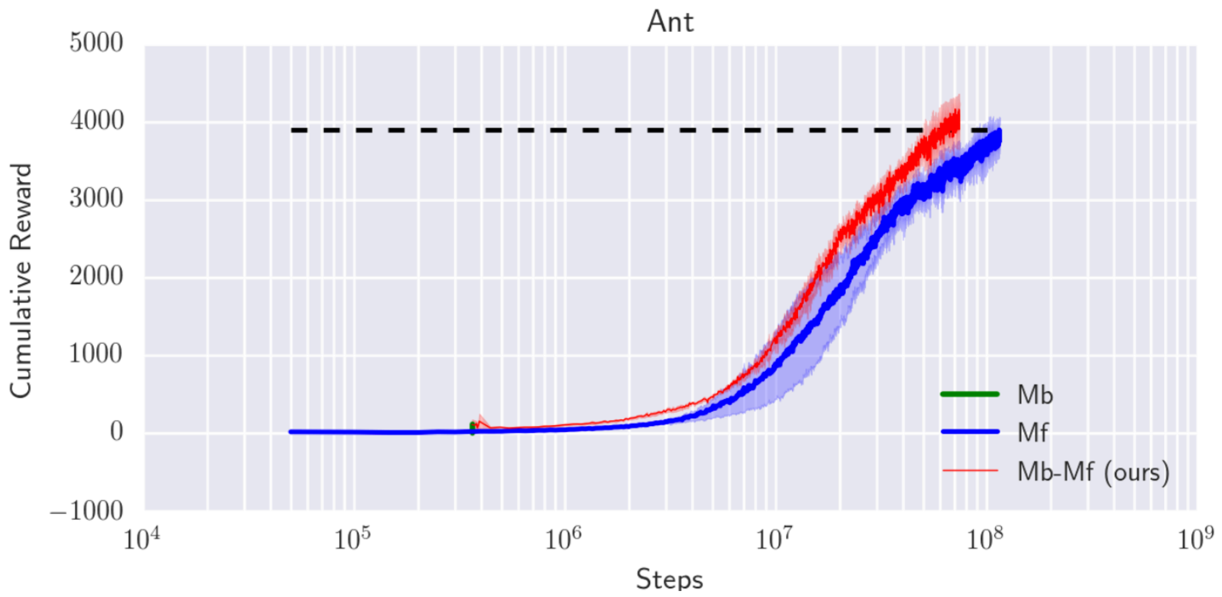


Fig. 7: Using the standard Mujoco agent's reward function, our model-based method achieves a stable moving-forward gait for the swimmer using 20× fewer data points than a model-free TRPO method. Furthermore, our hybrid Mb-Mf method allows TRPO to achieve its max performance 3× faster than for TRPO alone.



Cheetah
Cumulative Reward
6000
5000
4000
3000
2000
1000
0
−1000
10^3
10^4
10^5
10^6
10^7
10^8
10^9
Steps
Mb
Mf
Mb-Mf (ours)


Hopper
Cumulative Reward
4000
3500
3000
2500
2000
1500
1000
500
0
10^3
10^4
10^5
10^6
10^7
10^8
Steps
Mb
Mf
Mb-Mf (ours)


Ant
Cumulative Reward
5000
4000
3000
2000
1000
0
−1000
10^4
10^5
10^6
10^7
10^8
10^9
Steps
Mb
Mf
Mb-Mf (ours)