Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning

Presenters: Mridul Khurana, Sammy Fellah, Shahwar Khursheed

VIRGINIA TECH

Outline

Introduction and Motivation

Model - Free vs Model - Based

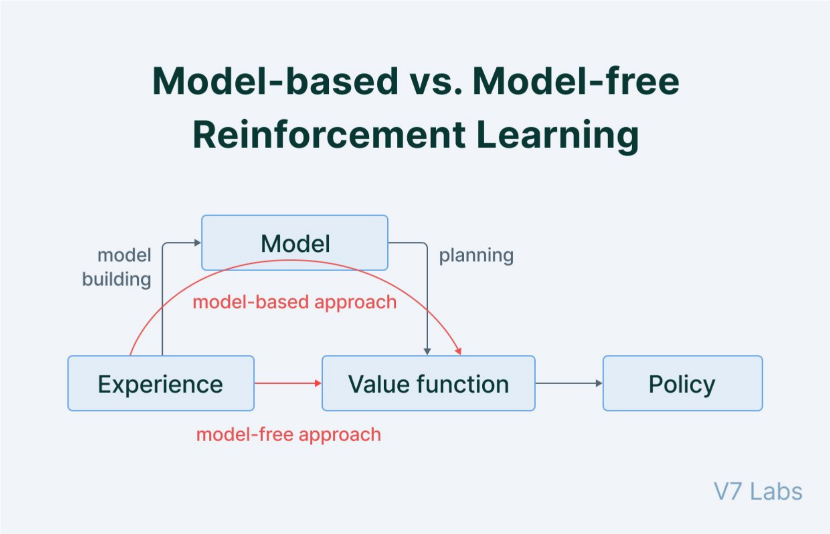


Image - https://www.v7labs.com/blog/deep-reinforcement-learning-guide

Model - Free

Model-free algorithms (ex. DQN, TRPO) are capable of learning a wide variety of tasks like:

- Atari games
- Complex locomotion tasks.

Problem ?

- They suffer from very high sample complexity i.e. require large number of samples to achieve good performance
- Hence, learning is difficult in real world

Model - Based

Model-based algorithms (ex. PILOC) learn more efficiently than model-free algorithms.

Problem ?

- Difficult to extend to high-capacity models like deep neural networks
- Uses simple function approximators or Bayesian models.

VIRGINIA TECH™

Model-based with Model-free Fine-tuning

The Model-based with Model-free fine-tuning (Mb-Mf) algorithm combines the benefits of both the algorithms.

- It uses a model-based learner using Model Predictive Control (MPC) to initialize a mode-free learner.
- Mb-Mf helps accelerate model-free learning and also helps achieving sample efficiency gains over existing model-free algorithms.

Model Predictive Control (MPC)

A model-based learner can be denoted as $f_\theta(s_t, a_t)$, a discrete-time dynamics function

Actions can be chosen by solving the optimization problem:

$$(\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}) = \arg \max_{\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}} \sum_{t'=t}^{t+H-1} \gamma^{t'-t} r(\mathbf{s}_{t'}, \mathbf{a}_{t'})$$

- At each time step, execute only the first action a_t from the sequence and the system evolves according to its dynamics.
- The process is repeated at each time step, with updated state information

Methodology

Model-based Reinforcement Learning

Given the model

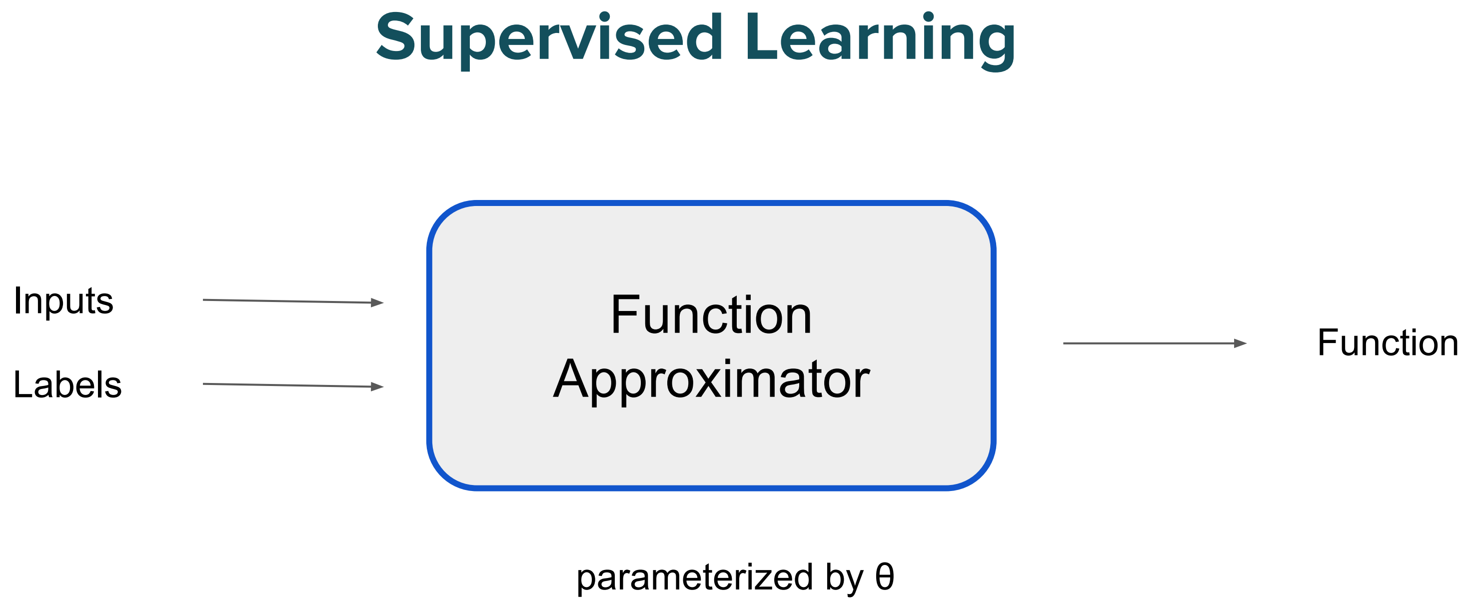
Learn the model

Model-based Reinforcement Learning

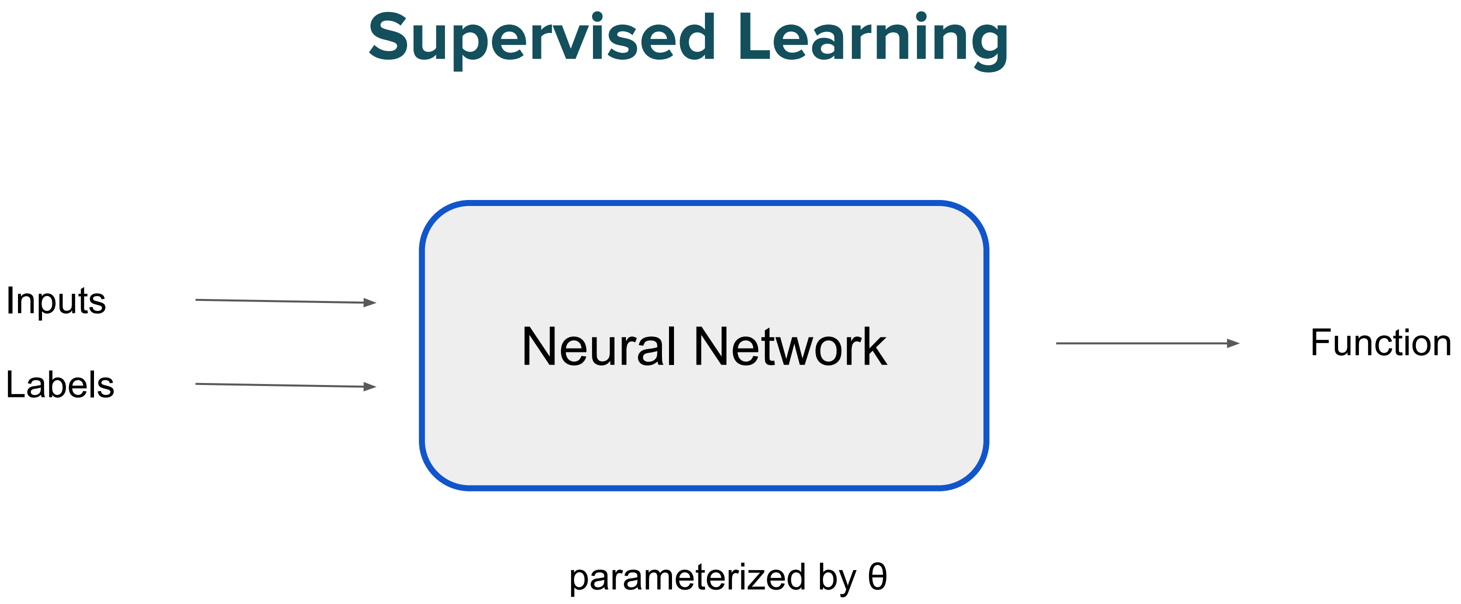
Given the model

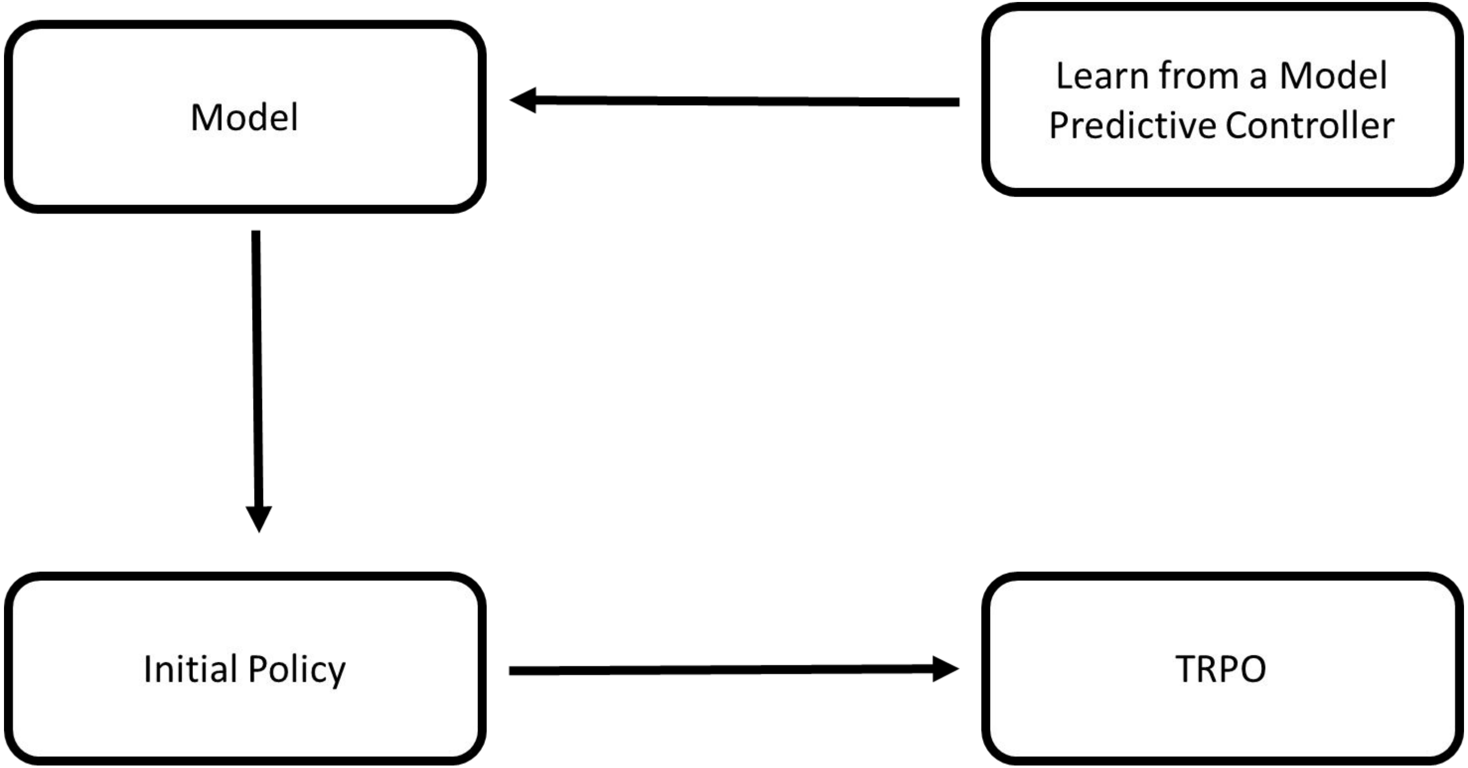
Learn the model

Learn the model

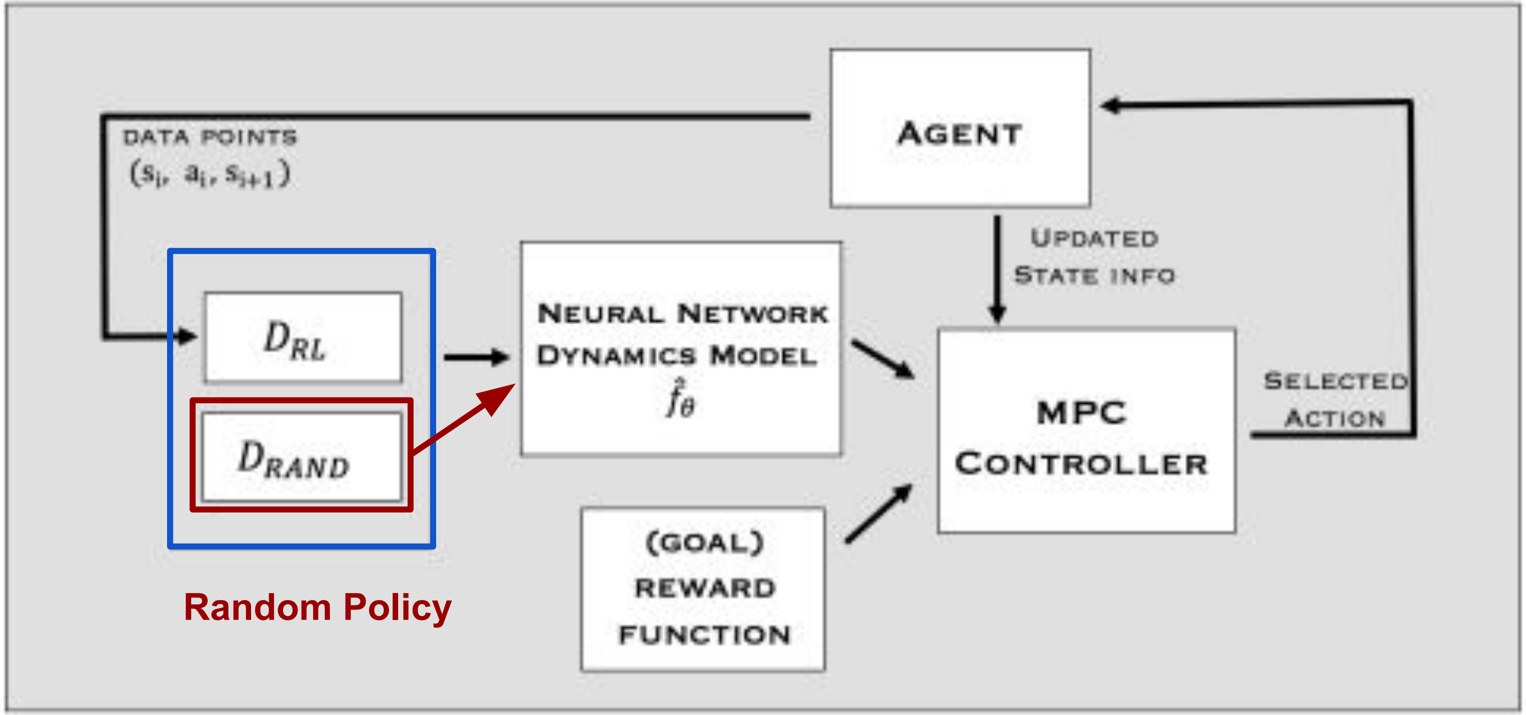


Learn the model

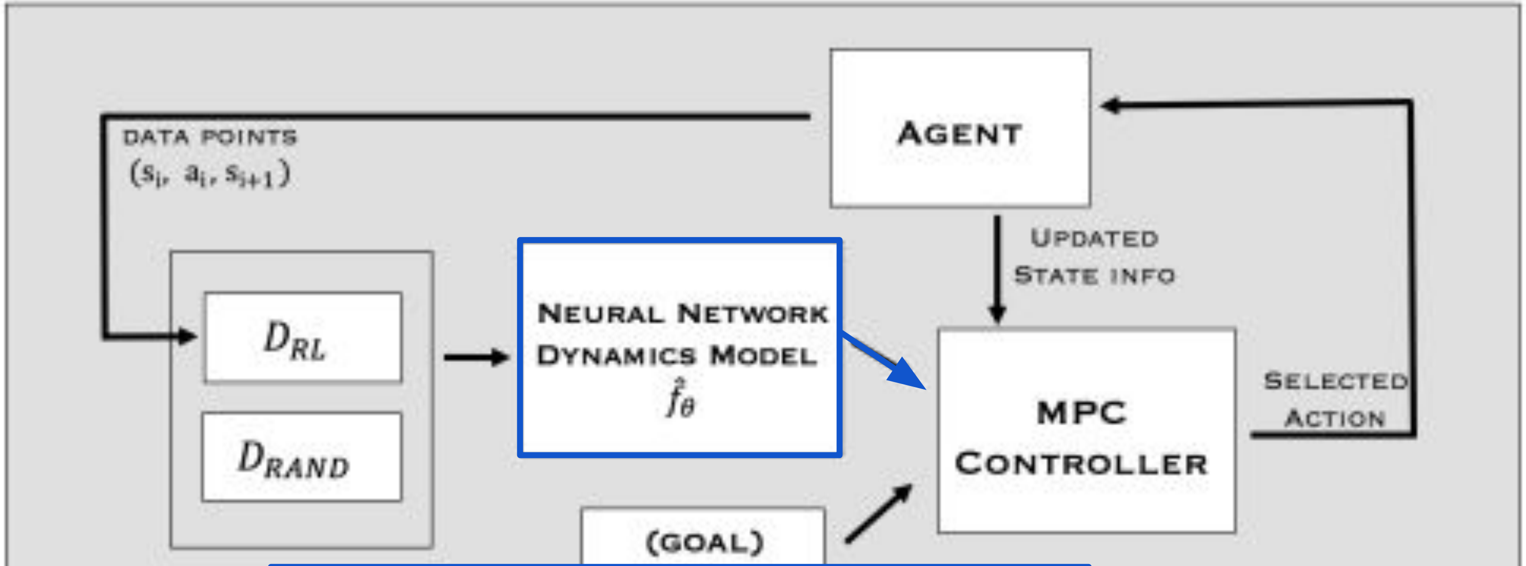



Model
Learn from a Model Predictive Controller
Initial Policy
TRPO

Model-based Reinforcement Learning



Model-based Reinforcement Learning

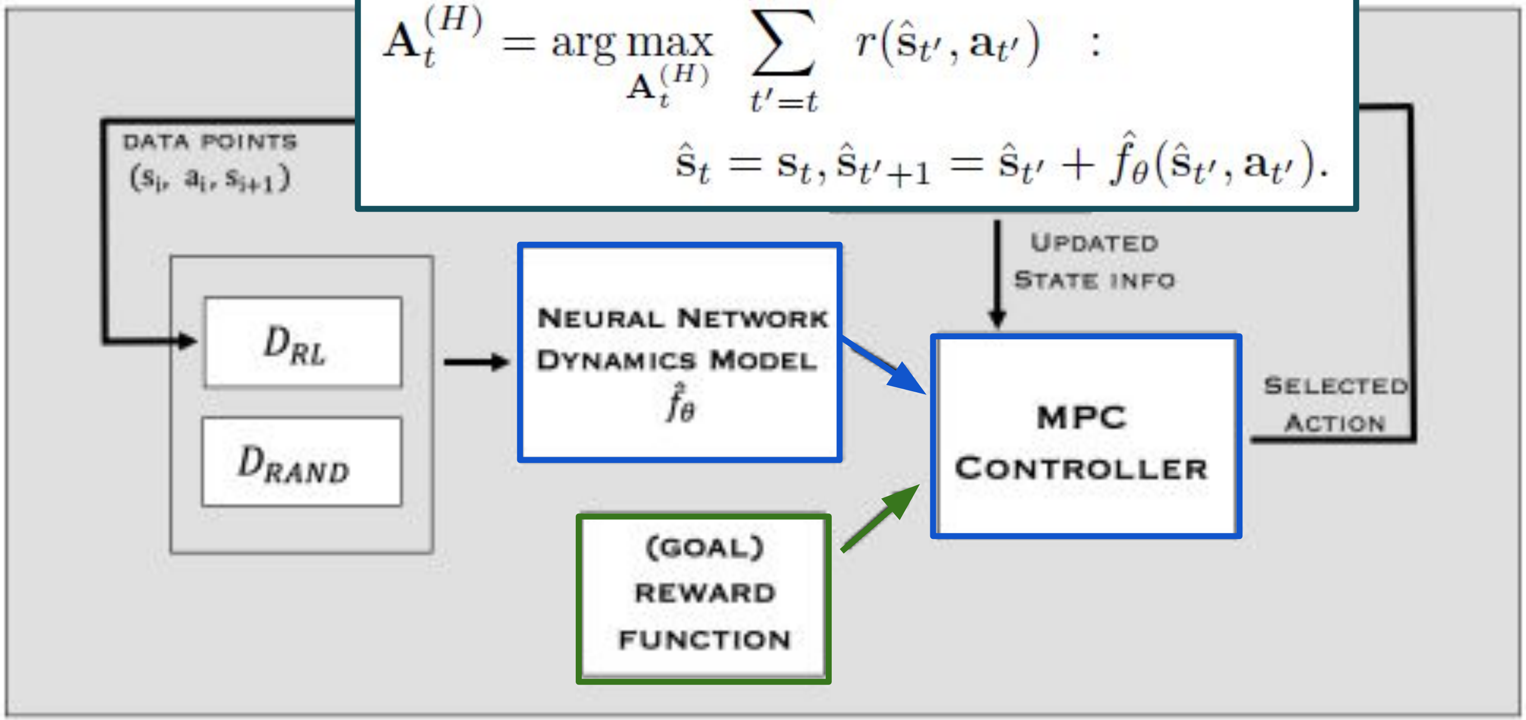


$$\mathcal{E}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \in \mathcal{D}} \frac{1}{2} \|(\mathbf{s}_{t+1} - \mathbf{s}_t) - \hat{f}_\theta(\mathbf{s}_t, \mathbf{a}_t)\|^2$$

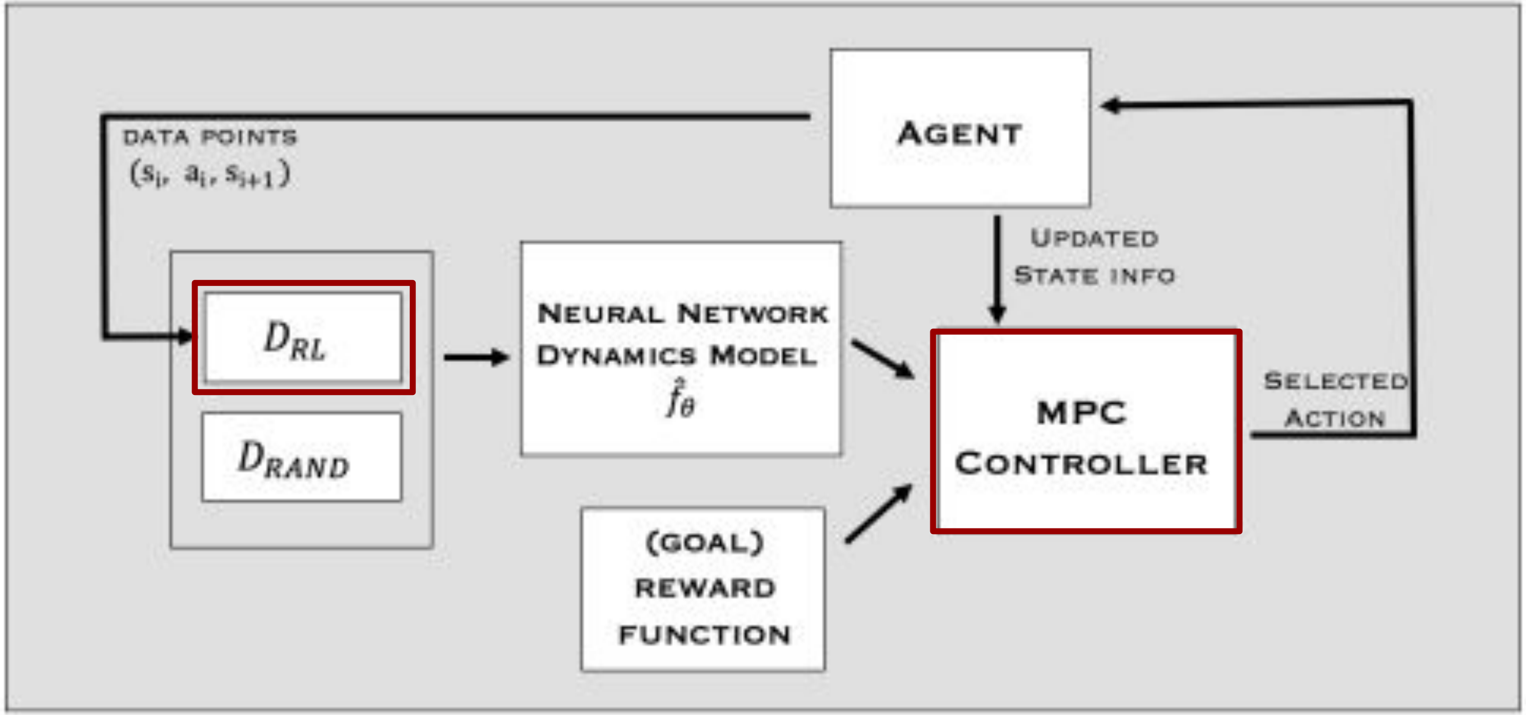
Model-based Reinforcement Learning

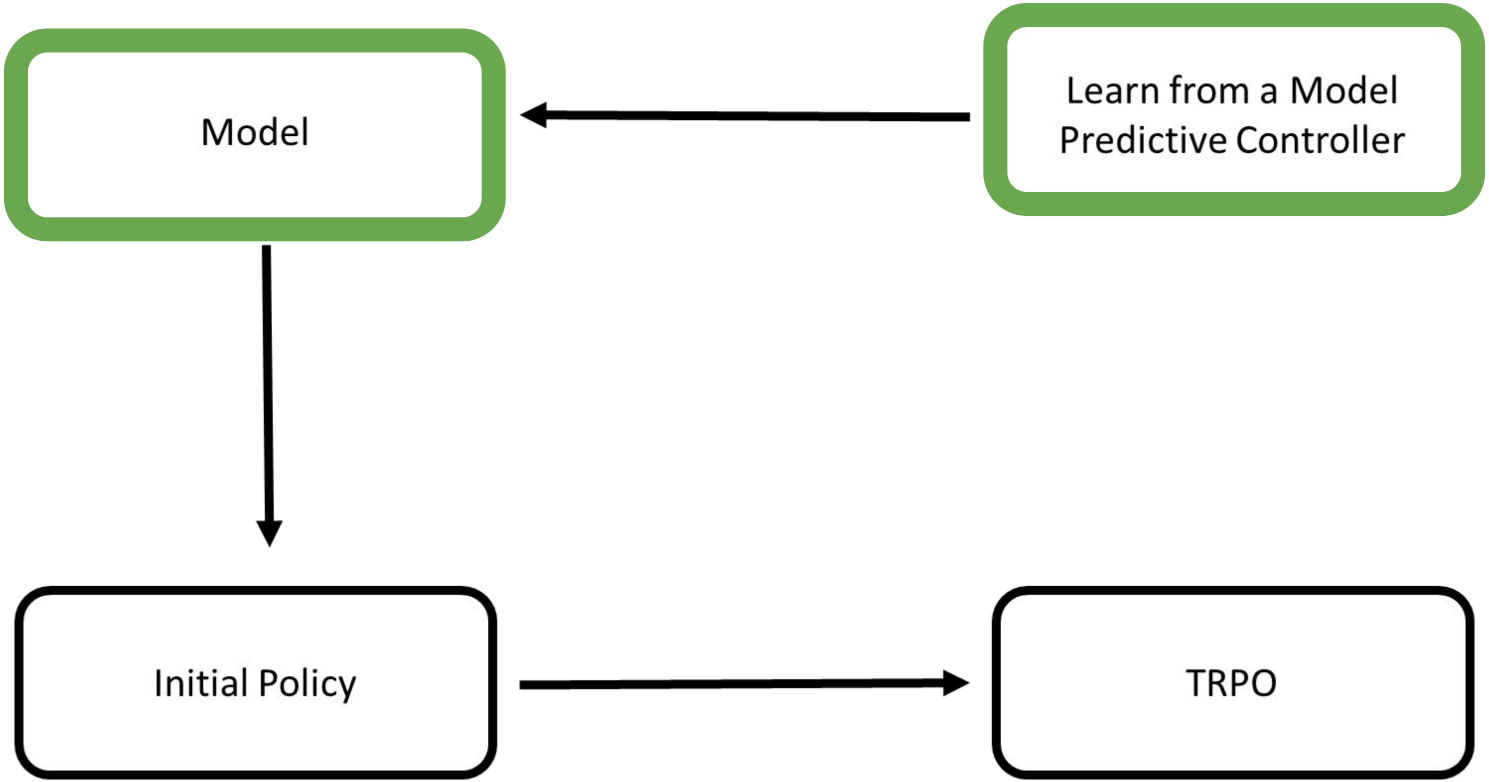
$$\mathbf{A}_t^{(H)} = \arg\max_{\mathbf{A}_t^{(H)}} \sum_{t'=t}^{t+H-1} r(\hat{\mathbf{s}}_{t'}, \mathbf{a}_{t'}) \quad :$$

$$\hat{\mathbf{s}}_t = \mathbf{s}_t, \hat{\mathbf{s}}_{t'+1} = \hat{\mathbf{s}}_{t'} + \hat{f}_\theta(\hat{\mathbf{s}}_{t'}, \mathbf{a}_{t'}).$$



Model-based Reinforcement Learning




Model
Learn from a Model
Predictive Controller
Initial Policy
TRPO

Extracting a policy

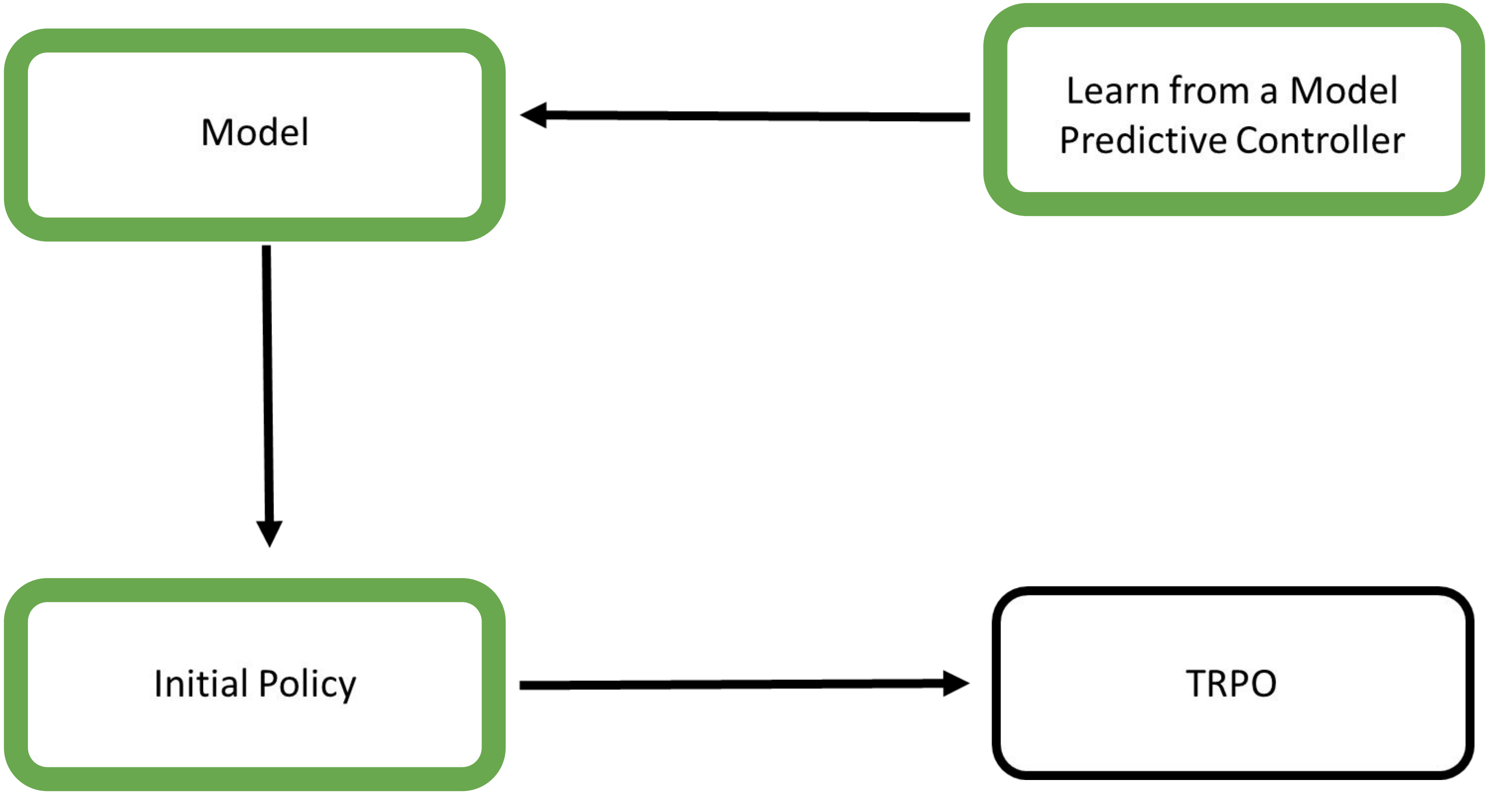
Collect "expert" trajectories from the MPC controller

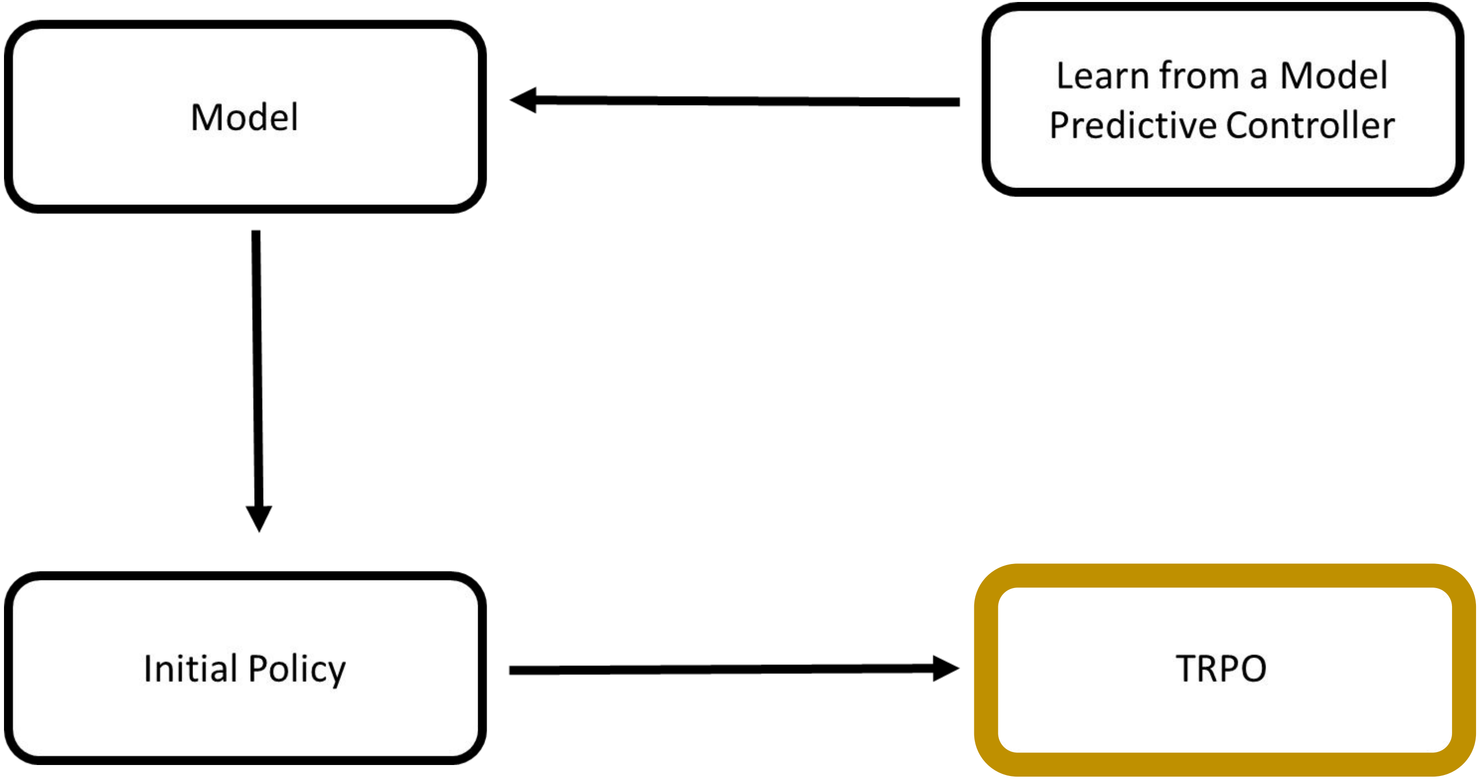
Get a policy based on these "expert" trajectories

$$\pi_\phi(\mathbf{a}|\mathbf{s}) \sim \mathcal{N}(\mu_\phi(\mathbf{s}), \Sigma_{\pi_\phi})$$

$$\min_\phi \frac{1}{2} \sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}^*} ||\mathbf{a}_t - \mu_\phi(\mathbf{s}_t)||_2^2$$

DAGGER


Model
Learn from a Model Predictive Controller
Initial Policy
TRPO



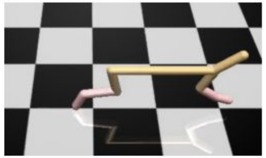
Model
Learn from a Model Predictive Controller
Initial Policy
TRPO

Experimental Results & Analysis

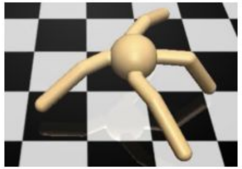
Evaluating Design Decisions for Model-Based Reinforcement Learning

A. Training steps
B. Dataset aggregation
C. Controller (H=Horizon, K=# random samples
D. Initial random trajectories

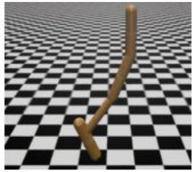
(a) Swimmer



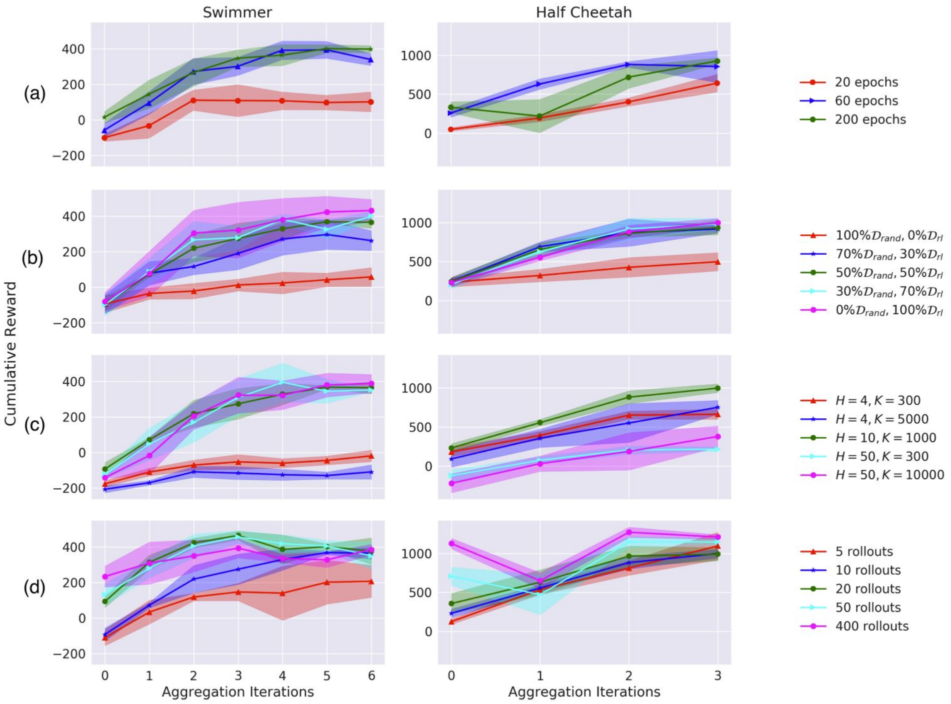
(b) Cheetah



(c) Ant



(d) Hopper



Explored design decisions on the swimmer and half cheetah agents on the locomotion task of running forward as quickly as possible

Trajectory Following with the Model-Based Controller

Experiments:

1. Model-based reinforcement learning on swimmer, ant, and half-cheetah
2. Dynamics model trained once with random initial trajectories
3. Model-based control of different tasks using learned model
4. Trajectory following task with reward function to track center of mass positions

Results:

1. Learned models capable of adapting to new tasks
2. Naive random-sampling controller effective with learned model
3. Model-based approach discovers gait without explicit instructions
4. Reward function penalizes perpendicular distance and encourages forward movement towards desired trajectory.

Model-based reinforcement learning algorithm *successfully* learns NN dynamics functions for complex simulated locomotion tasks using a small # of samples.

Mb-Mf Approach on Benchmark Tasks

Experiments:

1. Comparison of pure model-based approach and pure model-free method (TRPO) on standard benchmark locomotion tasks (swimmer, half-cheetah, hopper, ant)
2. OpenAI gym standard reward functions used for action selection
3. Pure model-based approach and pure model-free approach compared, and hybrid model-based plus model-free approach (Mb-Mf) implemented

Results:

1. Pure model-based approach quickly learns reasonable gait for all agents
2. Performance plateaus for hopper due to limited-horizon controller and insufficient reward signal
3. Hybrid Mb-Mf approach takes quickly learned gaits and performs model-free fine-tuning
4. 3-5x sample efficiency gains over pure model-free methods for all agents
5. Quick learning of gait for swimmer (20x faster than TRPO)