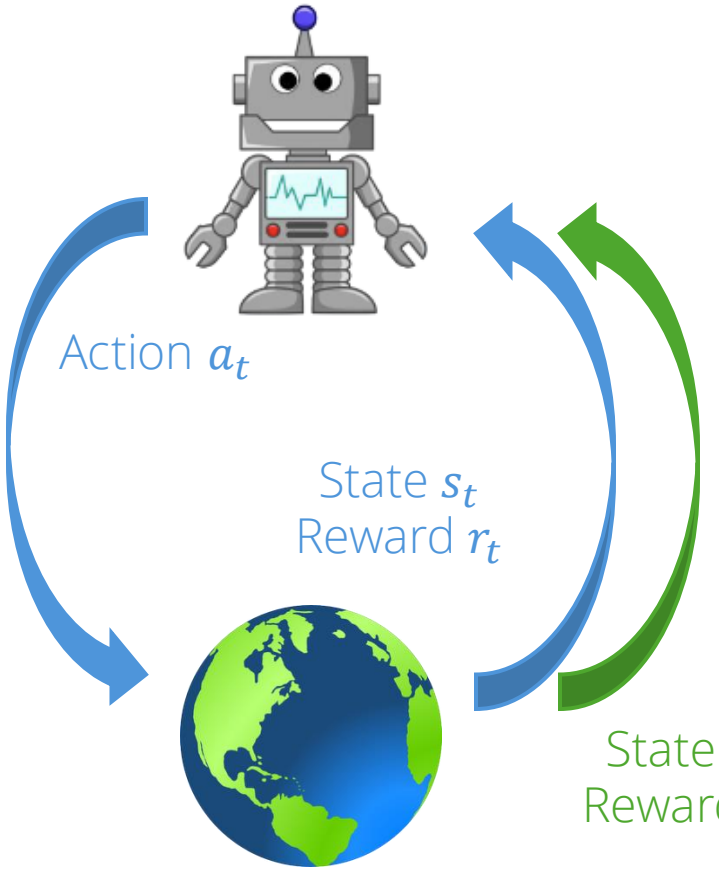
Robot Perception and Learning

Model Based RL

Tsung-Wei Ke

Fall 2025

Recap: the Objective of Reinforcement Learning



- A trajectory of interaction in the environment

$$\underbrace{p_\theta(\mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_T, \mathbf{a}_T)}_{p_\theta(\tau)} = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$$

The dynamic function of the environment

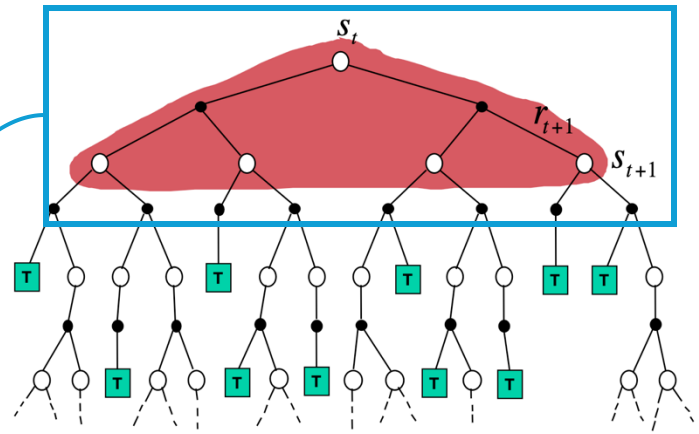
- Maximize the expected value of the cumulative sum of reward

$$\theta^\star = \arg\max_\theta E_{\tau \sim p_\theta(\tau)} \left[\sum_t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

Slide adapted from S. Levine

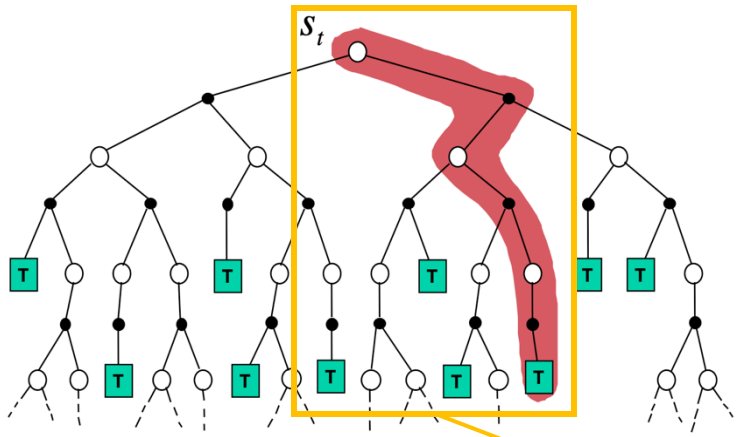
Why do We Use MC / TD, rather than DP? We Don't Know the Dynamic Function!

Dynamic Programming Backup

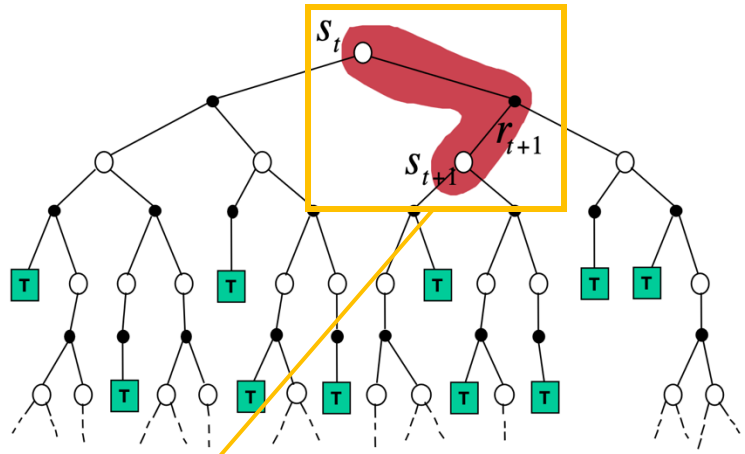


Simulate all possible trajectories, because the dynamic function is known

MC Backup



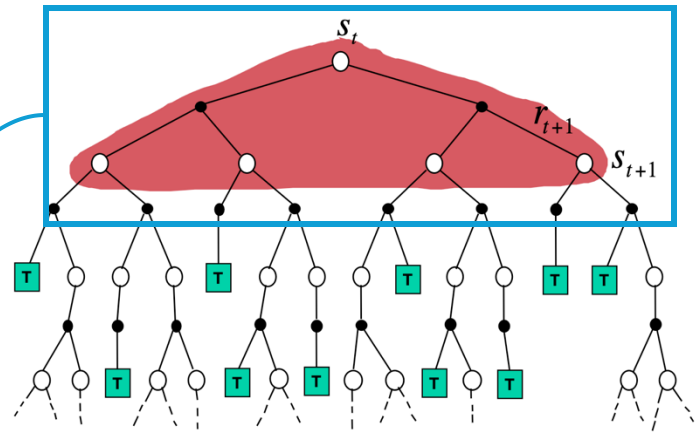
TD Backup



Sample a trajectory, because the dynamic function is unknown

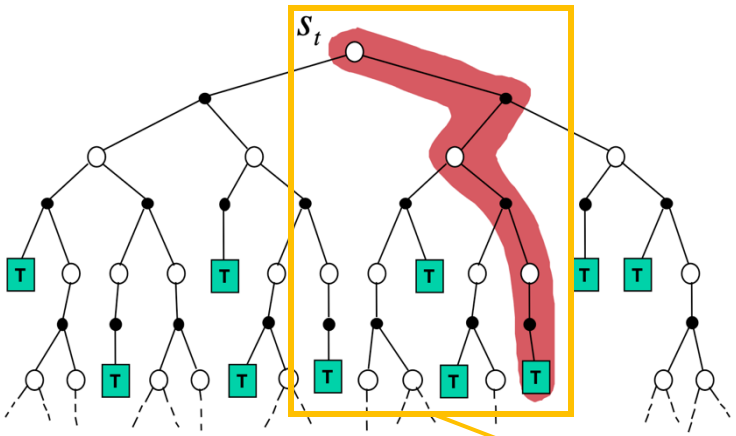
Why do We Use MC / TD, rather than DP? We Don't Know the Dynamic Function!

Dynamic Programming Backup

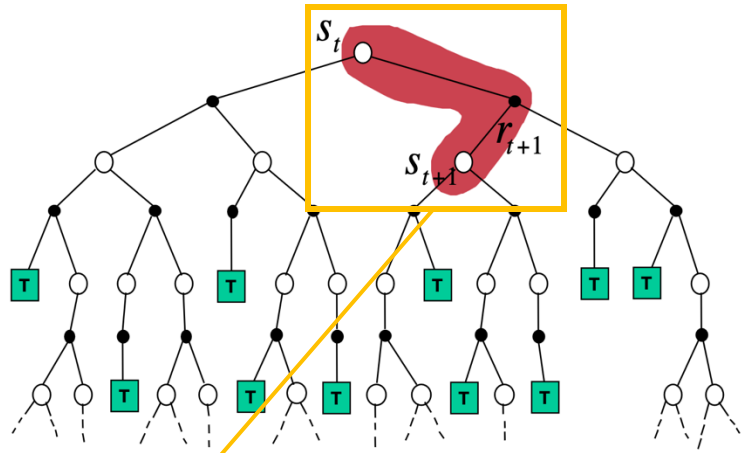


Simulate all possible trajectories, because the dynamic function is known

MC Backup



TD Backup

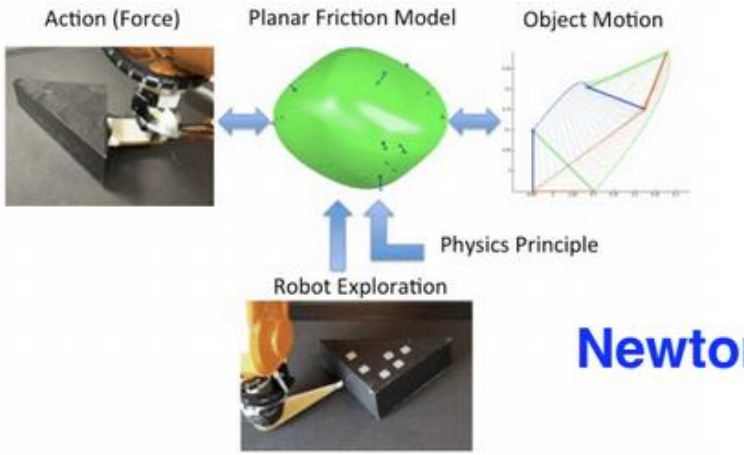


Sample a trajectory, because the dynamic function is unknown

- Policy Gradient: $\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \left(\sum_{t=0}^{T} r(s_t, a_t) - b \right) \right]$
- Actor Critic: $\nabla J(\theta) = E_{\tau \sim p_\theta(\tau)} \left[\left(\sum_{t=0}^{T} \nabla \log \pi_\theta(a_t | s_t) \right) \left(Q^\pi(s_t, a_t) - b \right) \right]$

What is a “Dynamics/World Model”?

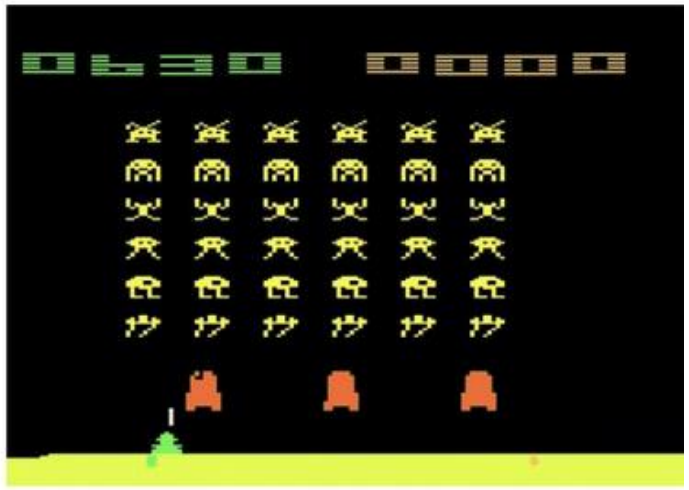
Occupancy map when navigating



Newton’s Physics when Pushing

Game Rules when playing

Helps us figure out what to do next



Slide credit P. Agrawal

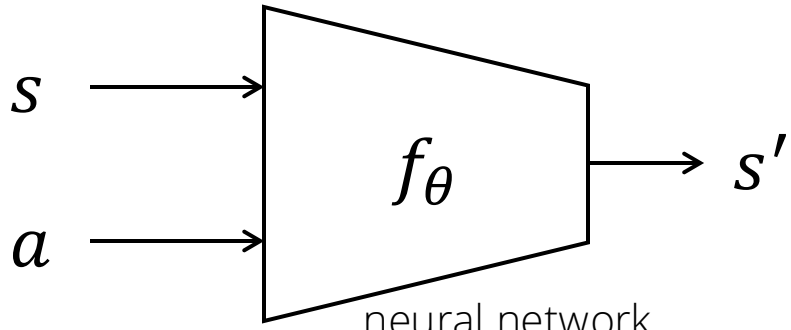
What is a “Dynamics / World Model”?

The real world: expensive to interact with

The simulator world: expensive to build and almost impossible to be identical to the real world

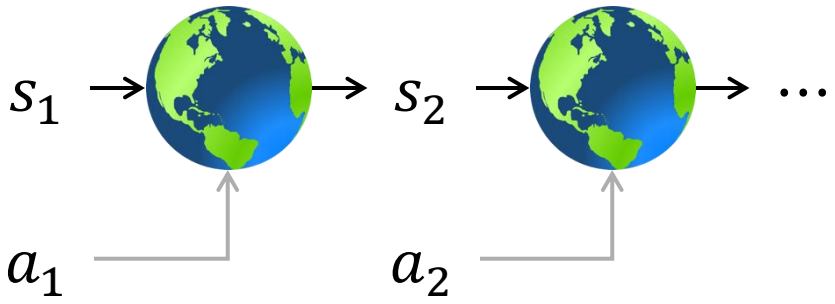
Isaac Sim by Nvidia

How to Model a "Dynamics / World Model"?

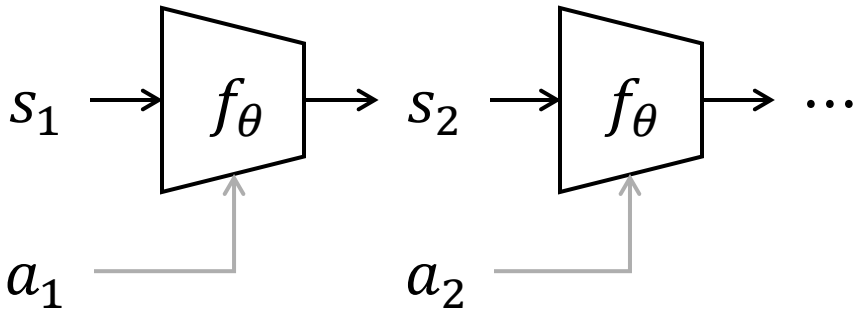


neural network,
decision tree,
gaussian process, …

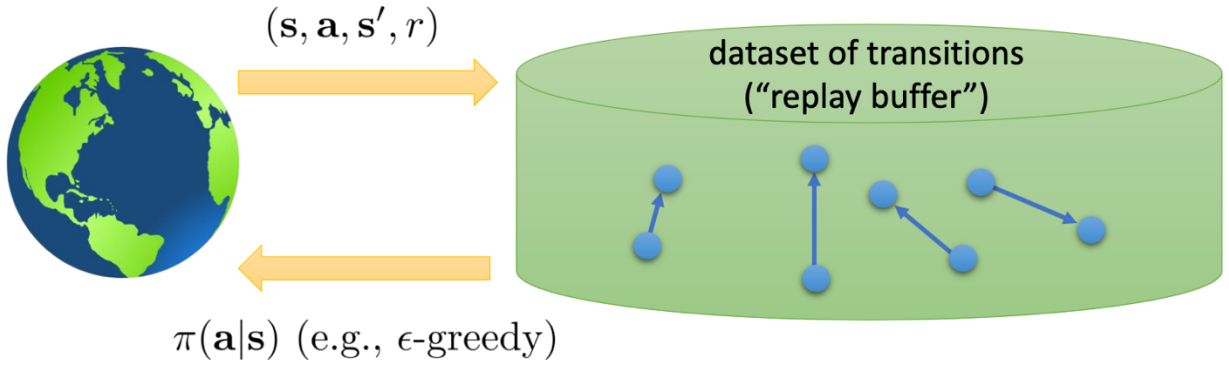
Execute to unroll a sequence of actions in the real world



Imagine to unroll a sequence of actions with a learned world model

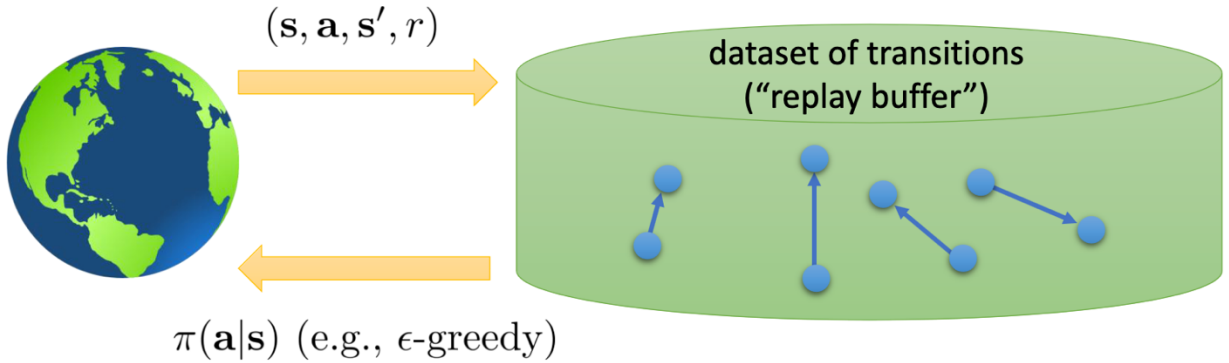


How to Learn a "Dynamics/World Model"?



- For value estimation, we learn value functions from the dataset of transitions:
- We can "re-use" the same dataset of transition for learning a world model.

How to Learn a "Dynamics/World Model"?



Algorithm:

1. Run base policy $\pi_0(a_t|s_t)$ (e.g. random policy) to collect $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$
2. (Training) Learn dynamics/world model:

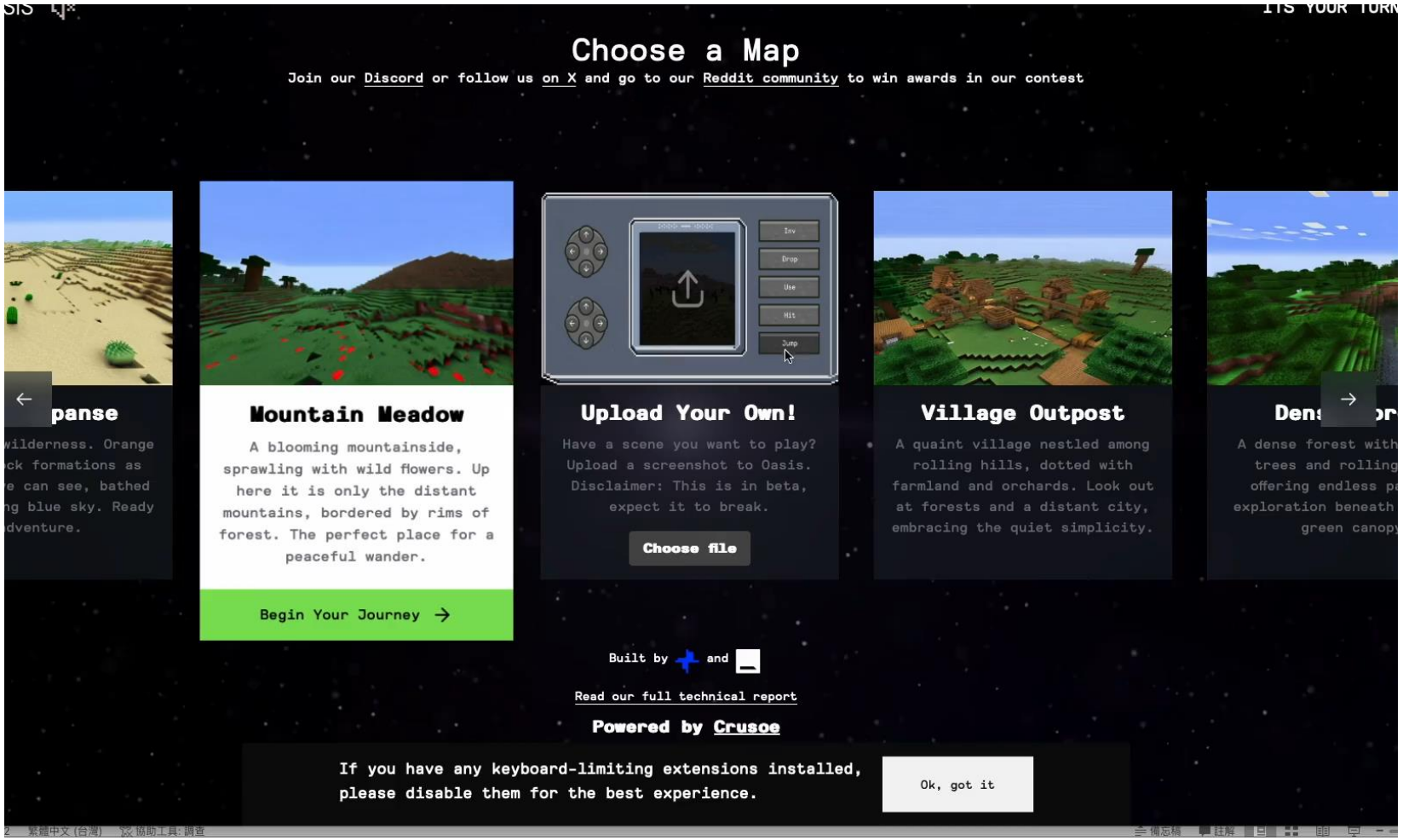
$$\min_{\theta} \mathbb{E}_{s_t, a_t, s_{t+1}}[\|s_{t+1} - f_\theta(s_t, a_t)\|]$$

3. (Testing) Plan an optimal sequence of actions $a^*_{0:T}$:

$$a^*_{0:T} = \max_{a_{0:T}} r(s_0, a_0) + r(f(s_0, a_0), a_1) + r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

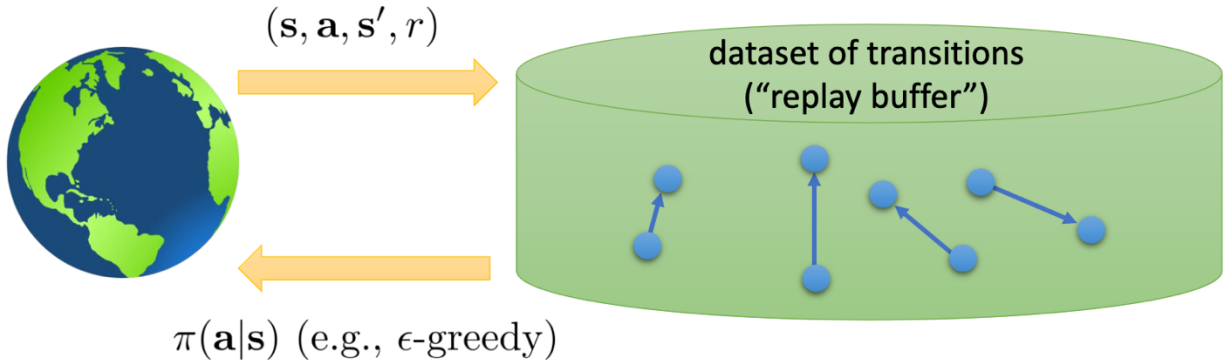
Slide adapted from S. Levine

But Learned Dynamics Models Still Have Out-of-Distribution Problems…



Oasis: A Universe in a Transformer. https://www.decart.ai/articles/oasis-interactive-ai-video-game-model

Improved the "Dynamics/World Model" with More Experience



Algorithm:

1. Run base policy $\pi_0(a_t|s_t)$ (e.g. random policy) to collect $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$
2. (Training) Learn dynamics/world model:

$$\min_{\theta} \mathbb{E}_{s_t, a_t, s_{t+1}}[\|s_{t+1} - f_\theta(s_t, a_t)\|]$$

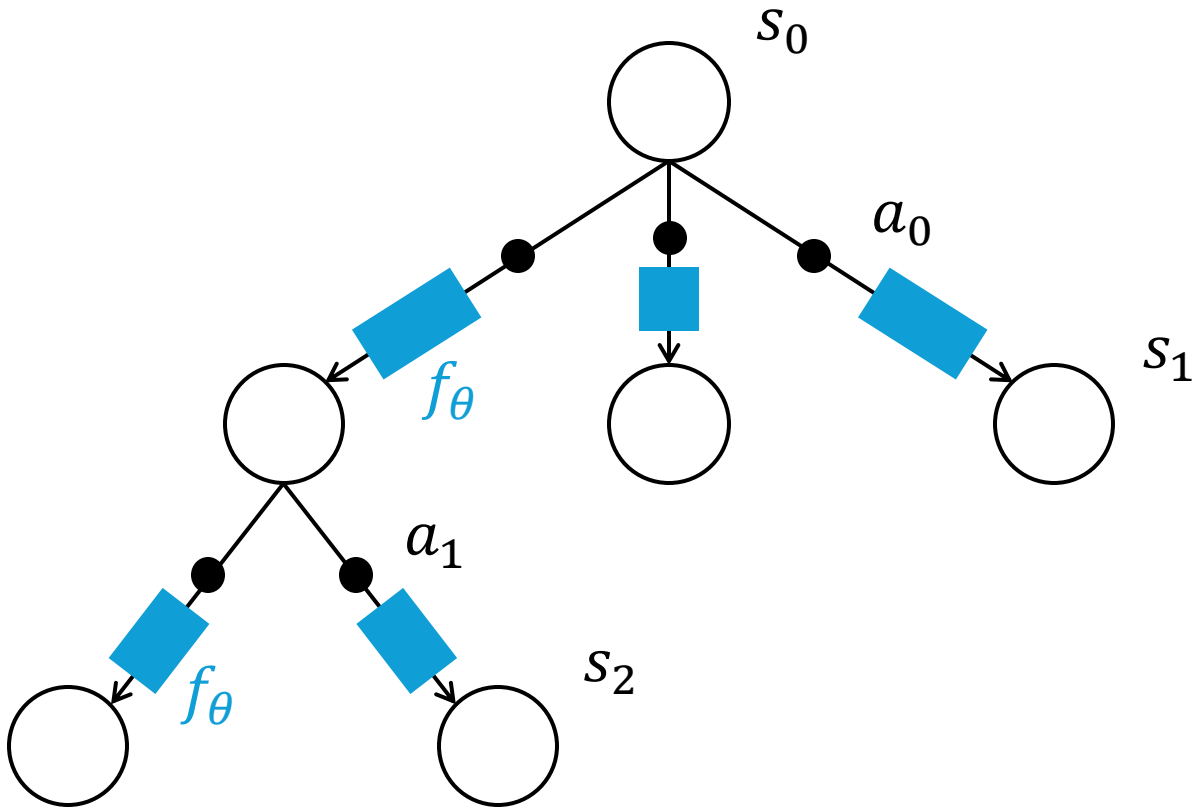
3. (Testing) Plan an optimal sequence of actions $a^*_{0:T}$:

$$a^*_{0:T} = \max_{a_{0:T}} r(s_0, a_0) + r(f(s_0, a_0), a_1) + r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

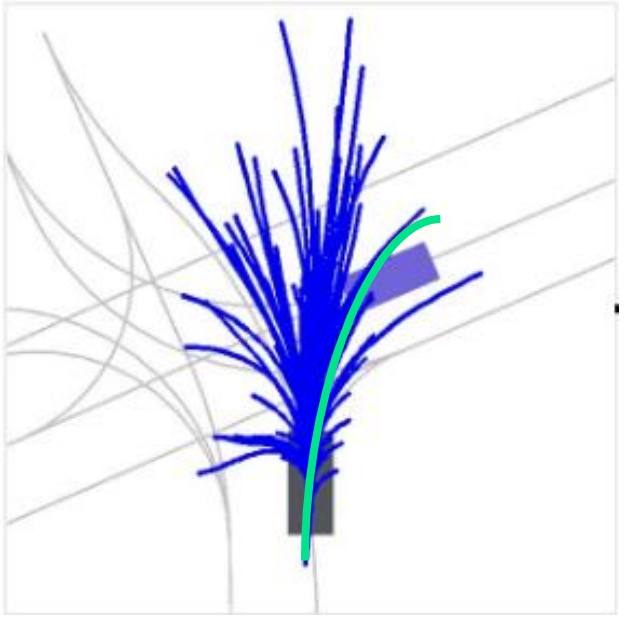
4. Execute the planned actions $a^*_{0:T}$ and add the resulting data $\{(s_t, a_t, s_{t+1})\}$ to $\mathcal{D}$

Slide adapted from S. Levine

What Can We Do with a Learned Dynamics Model? Plan/Search Optimal Action Sequences by Simulation



Random Search

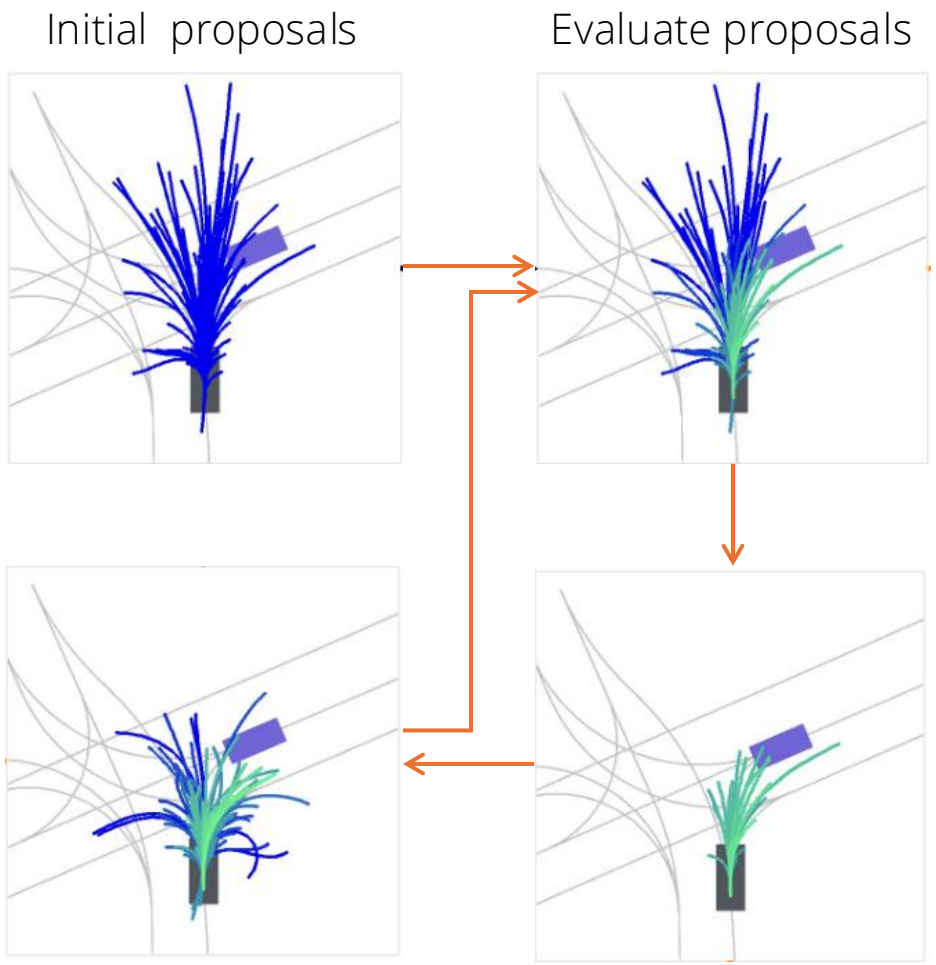


1. Sample N sequences of actions $a^1_{0:T}, a^2_{0:T}, \cdots, a^N_{0:T}$
2. Unroll each action sequence with the dynamics model f_θ
3. Evaluate each unrolled trajectory and pick the best one

- Need a lot of samples to obtain an optimal plan

Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous Driving and Zero-Shot Instruction Following. Yang et al.

Can We Search Smartly? Evolutionary Search



1. Sample N sequences of actions $a^1_{0:T}, a^2_{0:T}, \cdots, a^N_{0:T}$ from an initial distribution v_0
2. Unroll each action sequence with the dynamics model f_θ
3. Evaluate each unrolled trajectory and keep the top K sequences. Update the distribution to v_1 with the elite sequences.
4. Resample N sequences of actions $a^1_{0:T}, a^2_{0:T}, \cdots, a^N_{0:T}$ from the updated distribution v_1
5. Pick and execute the best action sequence

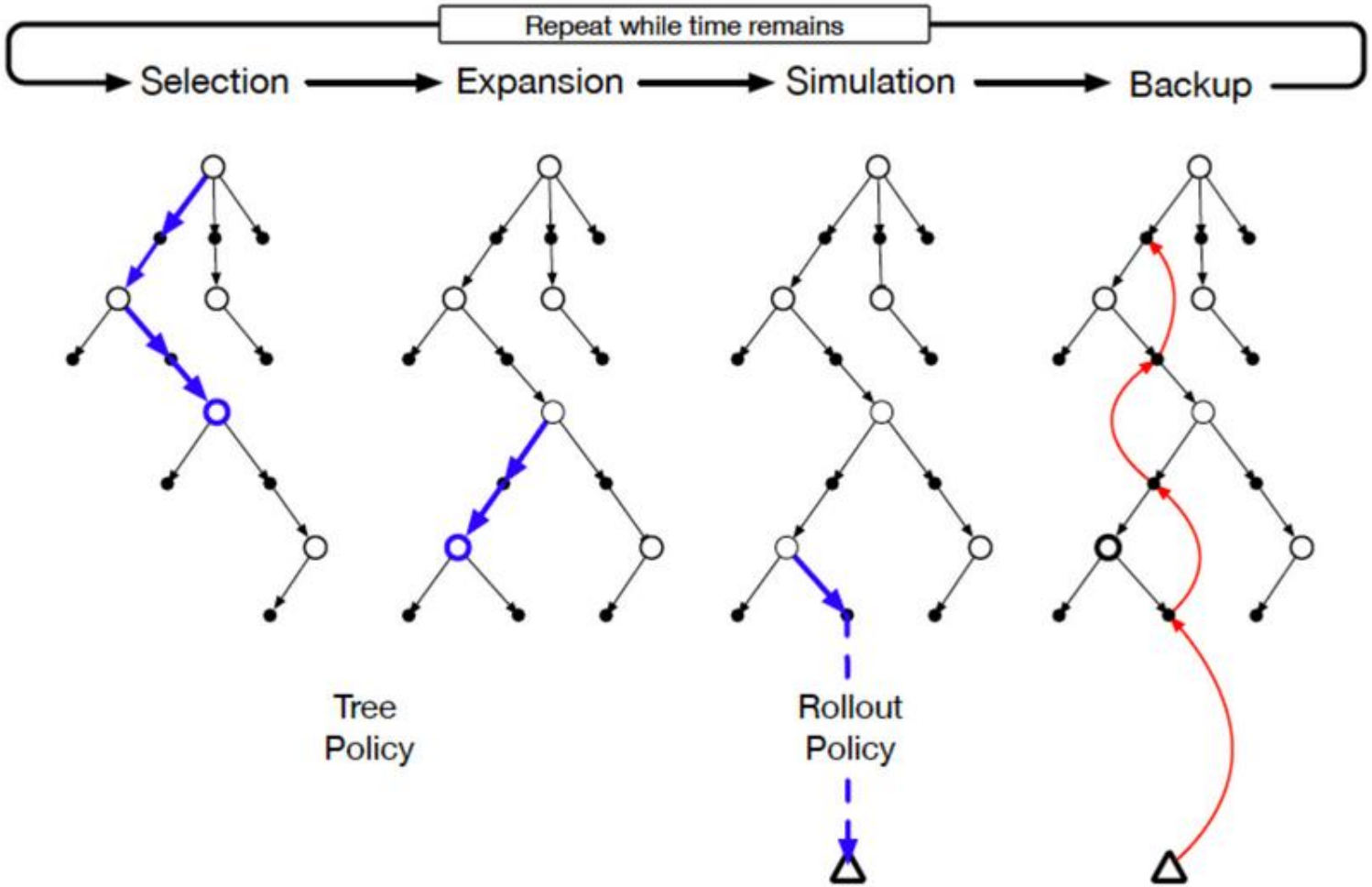
Diffusion-ES: Gradient-free Planning with Diffusion for Autonomous Driving and Zero-Shot Instruction Following. Yang et al.

An Example of Cross Entropy Method with Gaussian Distribution

```
// Initialize parameters
μ := -6
σ2 := 100
t := 0
maxits := 100
N := 100
Ne := 10
// While maxits not exceeded and not converged
while t < maxits and σ2 > ε do
    // Obtain N samples from current sampling distribution
    X := SampleGaussian(μ, σ2, N)
    // Evaluate objective function at sampled points
    S := exp(-(X - 2) ^ 2) + 0.8 exp(-(X + 2) ^ 2)
    // Sort X by objective function values in descending order
    X := sort(X, S)
    // Update parameters of sampling distribution via elite samples
    μ := mean(X(1:Ne))
    σ2 := var(X(1:Ne))
    t := t + 1
// Return mean of final sampling distribution as solution
return μ
```

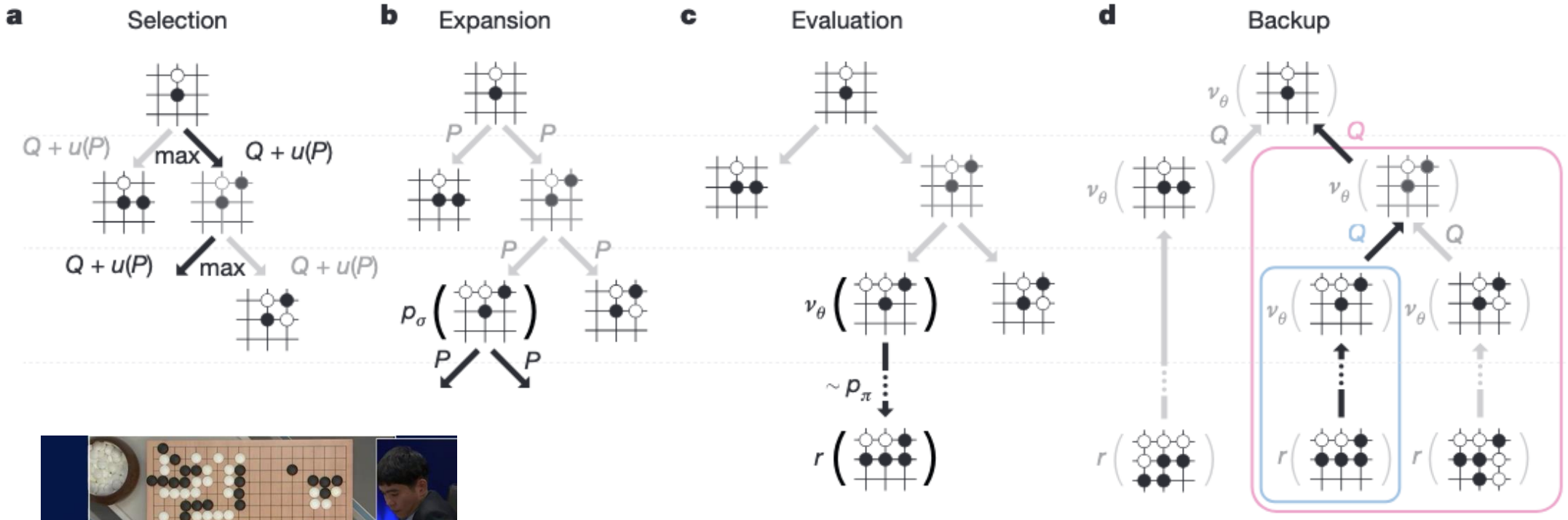
https://en.wikipedia.org/wiki/Cross-entropy_method

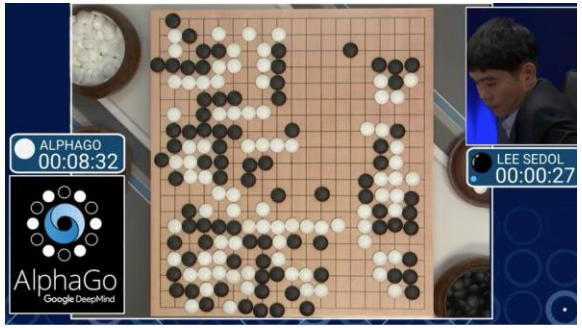
Can We Search Smartly? Monte-Carlo Tree Search



Slide credit K. Fragkiadaki

Can We Search Smartly? Monte-Carlo Tree Search



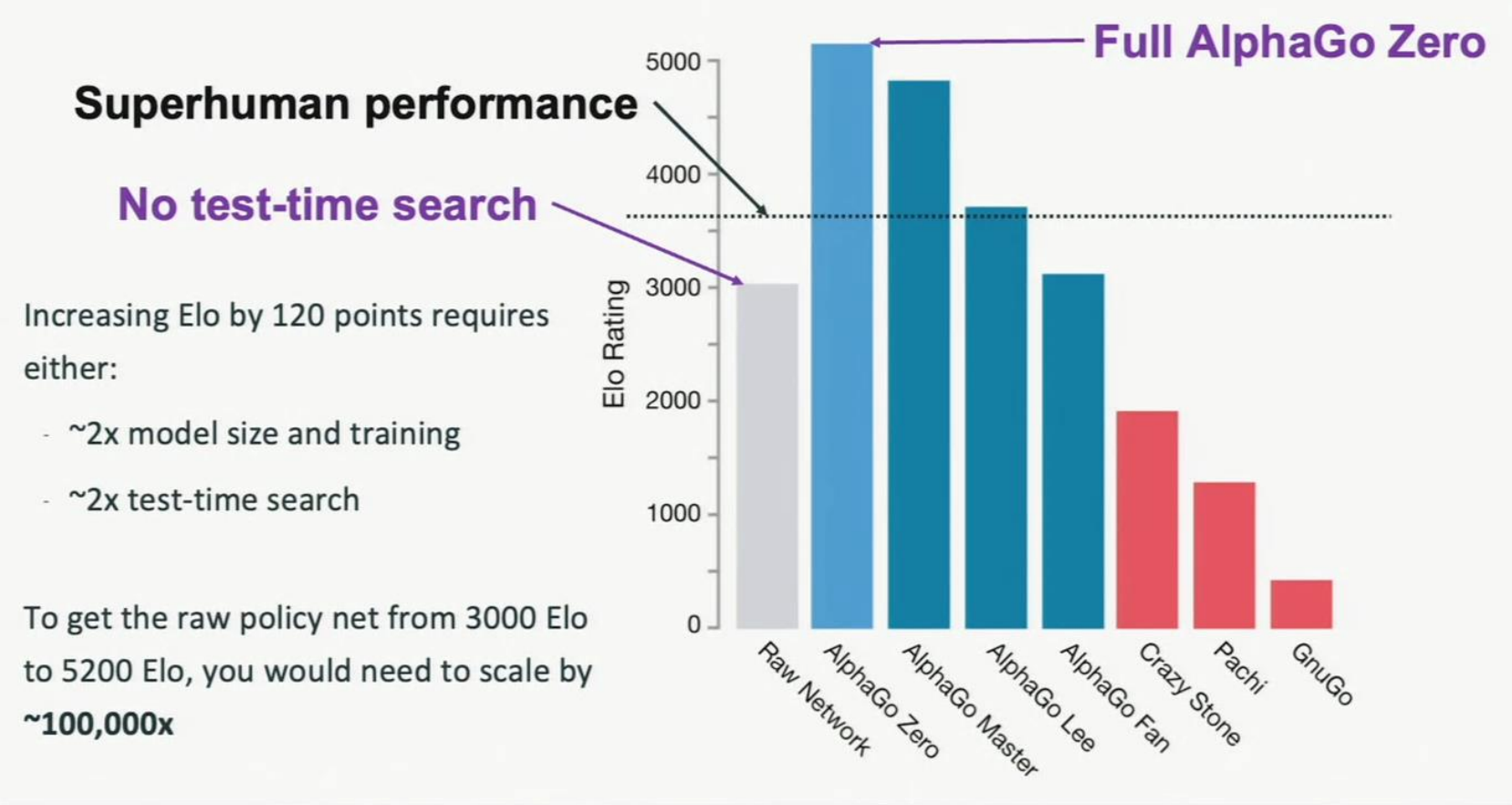


We'll go over Alpha Go today!

Mastering the game of Go with deep neural networks and tree search. Silver et al.

Why Do We Need Planning/Searching??

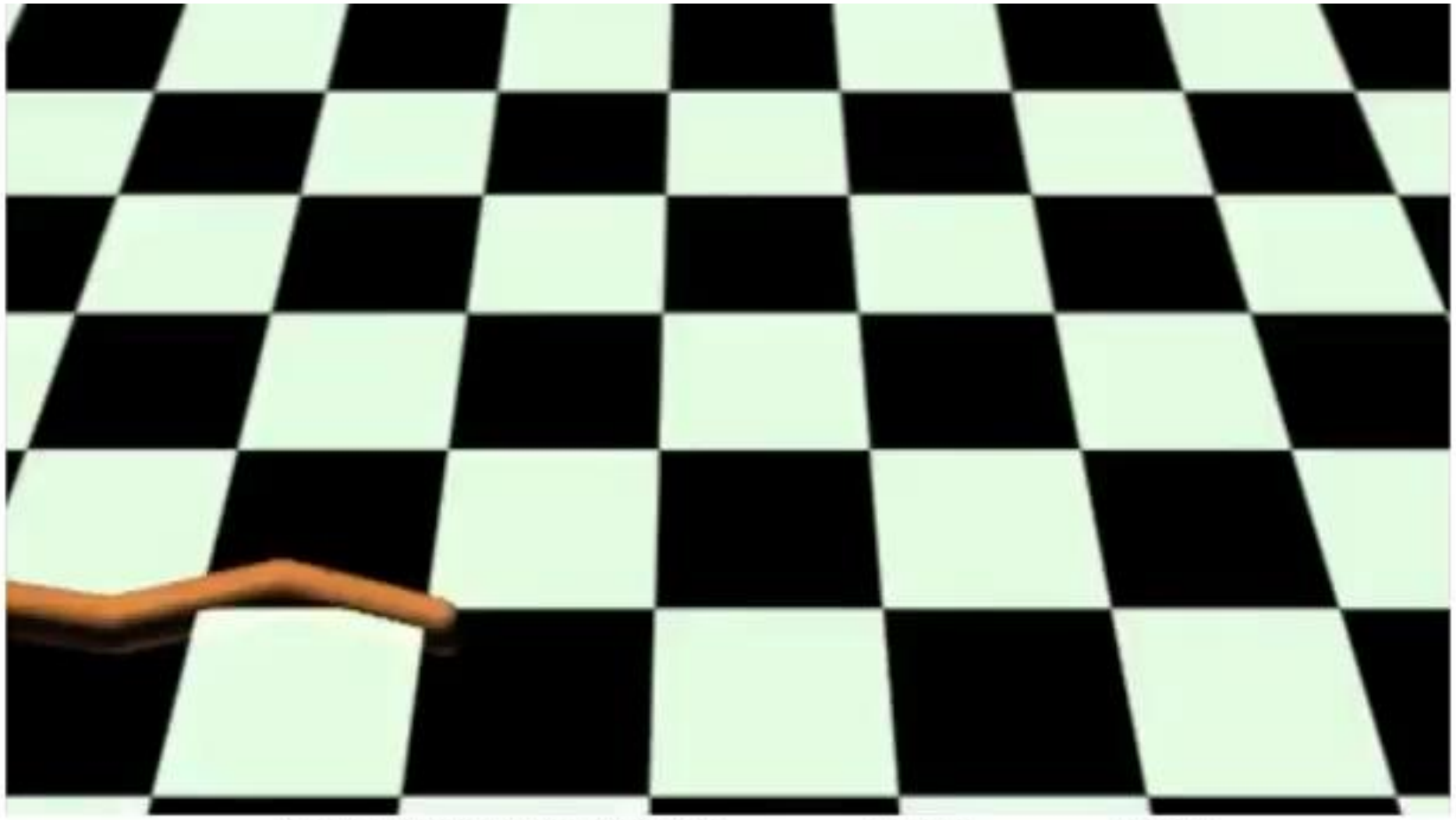
- Planning enables a sub-optimal policy to outperform a stronger policy w\o planning
- Planning facilitates learning of value functions



Slide credit Noam Brown: https://www.youtube.com/watch?v=eaAonE58sLU

Why Do We Need Planning/Searching?

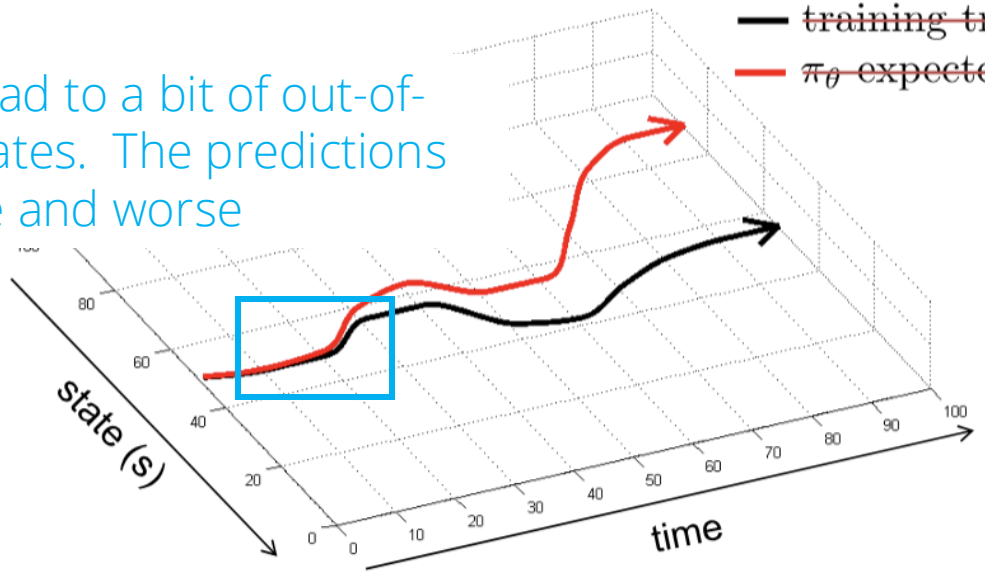
- Planning to achieve multiple (unseen) goals without re-training the policy



Neural network dynamics for model-based Deep RI with model-free finetuning. https://youtu.be/G7lXiuEC8x0?si=8cNF8Oa3_eMQ4Slz

But Learned Dynamics Models Still Have Error Accumulation Problem (Compounding Errors)…

Small errors lead to a bit of out-of-distribution states. The predictions become worse and worse



How quickly does error accumulate?

$$\mathcal{O}(\epsilon T^2)$$

Slide adapted from S. Levine

Solution 1: Augment the Training Data with Rollouts / Noise Perturbation

- Augment training with rollouts

$$\min_{\theta} \mathbb{E}_{s_t, a_t, s_{t+1}} [\| s_{t+1} - f_\theta(f_\theta(f_\theta(\dots), a_{t-2}), a_{t-1}), a_t) \|]$$

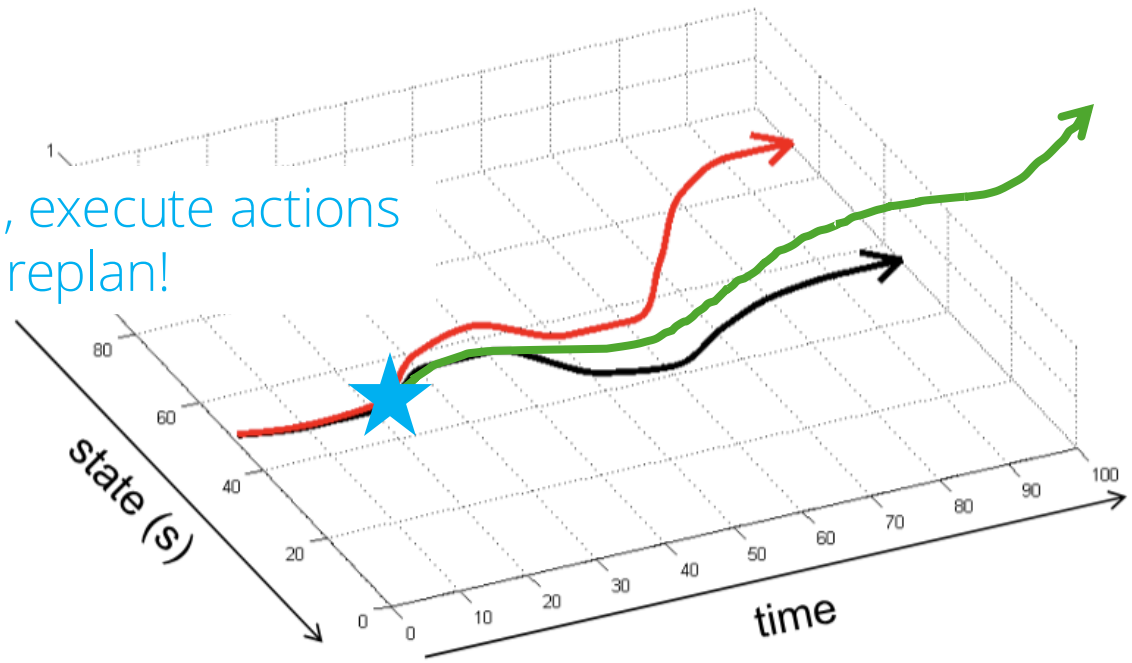
- Augment training with noise perturbation

$$\min_{\theta} \mathbb{E}_{s_t, a_t, s_{t+1}} [\| s_{t+1} - f_\theta(s_t + \epsilon, a_t) \|] \, , \epsilon \sim \mathcal{N}(0, 1)$$

Solution 2: Use the First Few Actions, where Errors are Still Small.

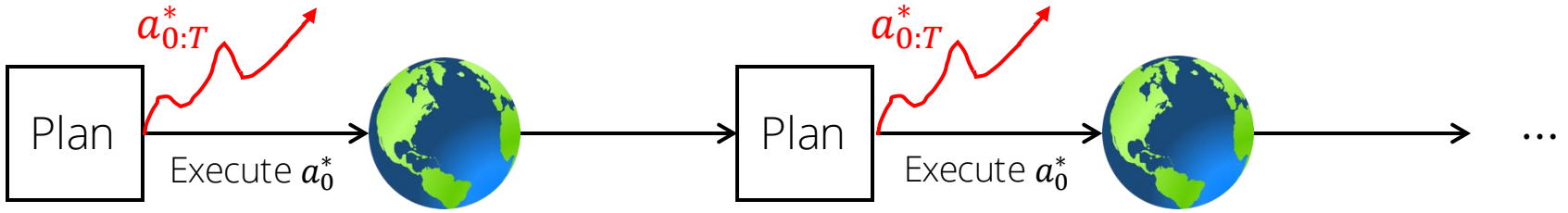
- Idea: Execute the first few actions, where the accumulation errors are still small, and then replan!

Errors are still small, execute actions until here and then replan!



Slide adapted from S. Levine

Model Predictive Control



Algorithm:

1. Run base policy $\pi_0(a_t|s_t)$ (e.g. random policy) to collect $\mathcal{D} = \{(s_t, a_t, s_{t+1})\}$
2. (Training) Learn dynamics/world model:

$$\min_\theta \mathbb{E}_{s_t,a_t,s_{t+1}}[\|s_{t+1} - f_\theta(s_t, a_t)\|]$$

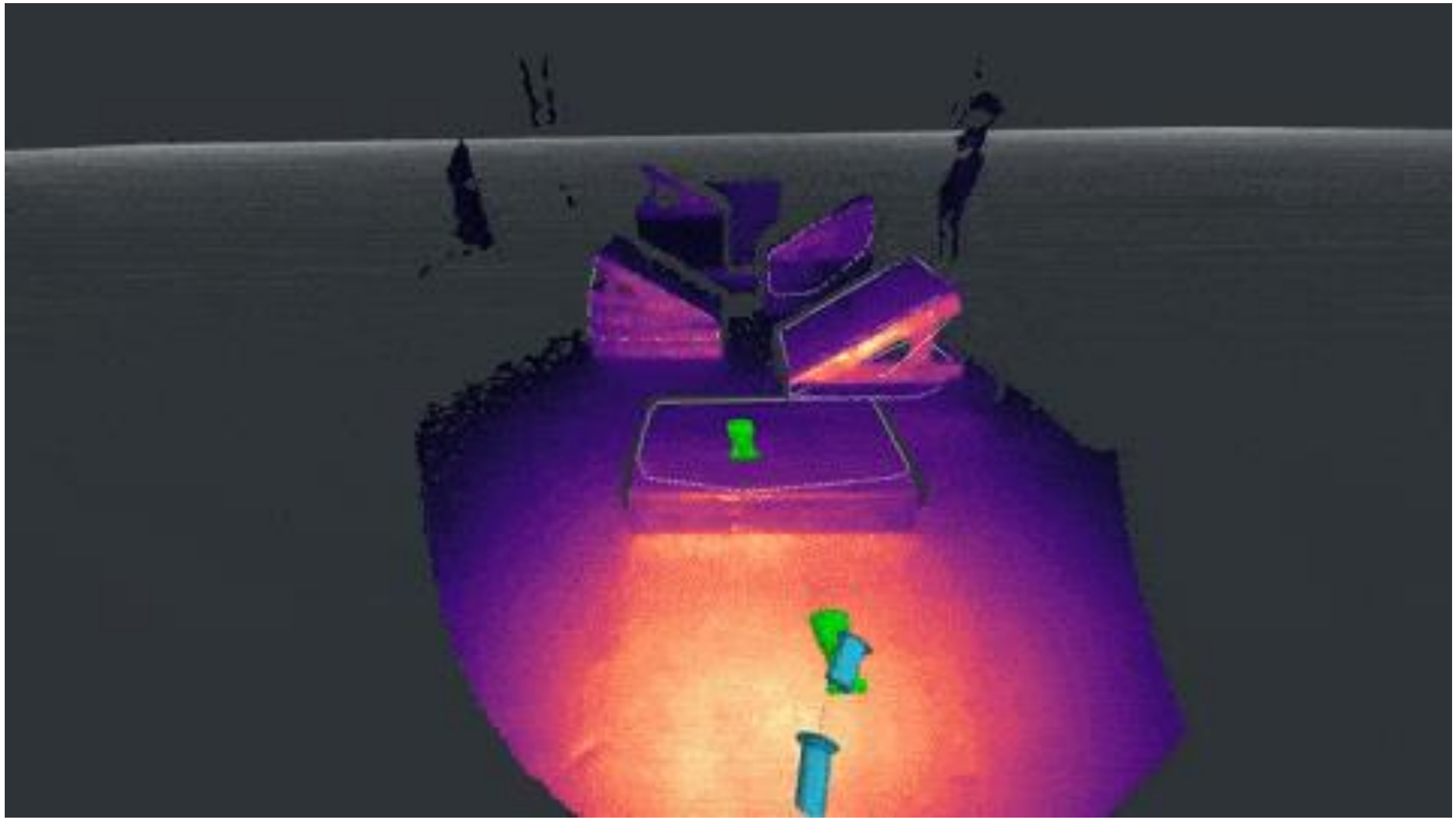
3. (Testing) Plan an optimal sequence of actions $a^*_{0:T}$:

$$a^*_{0:T} = \max_{a_{0:T}} r(s_0, a_0) + r(f(s_0, a_0), a_1) + r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

4. Execute the first planned action a^*_0 and observe resulting state s_1
5. add the resulting data $\{(s_t, a_t, s_{t+1})\}$ to $\mathcal{D}$

Slide adapted from S. Levine

Remember We've Talked About MPC in Optimal Control

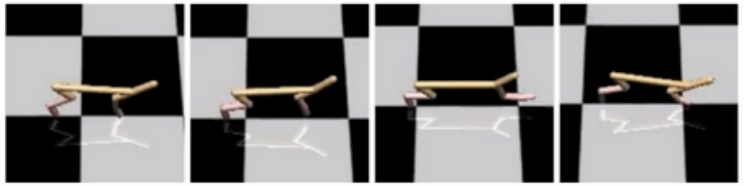


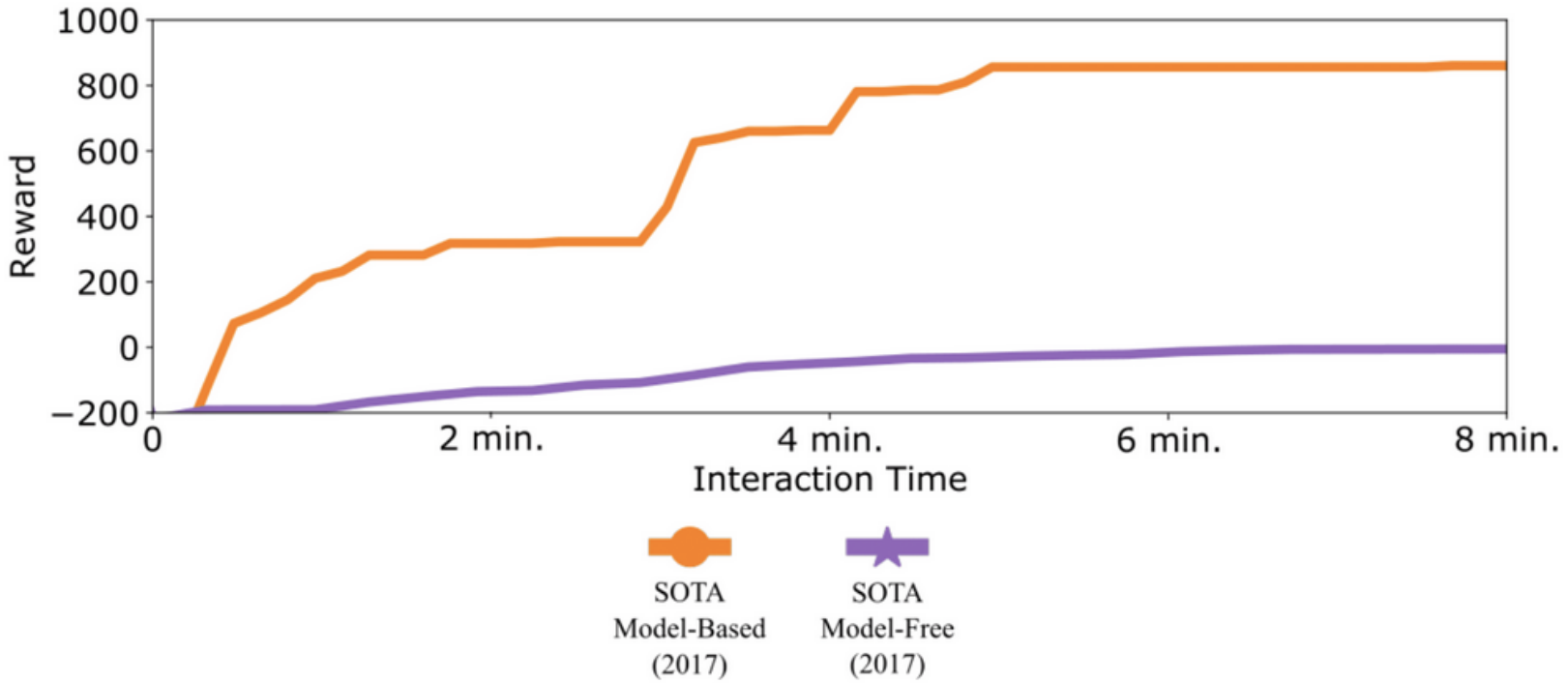
Video from Boston Dynamic
https://bostondynamics.com/blog/flipping-the-script-with-atlas/

What are Model-based RL Good For?

- Model-based RL methods are sample efficient, which require much less interaction than model-free RL methods

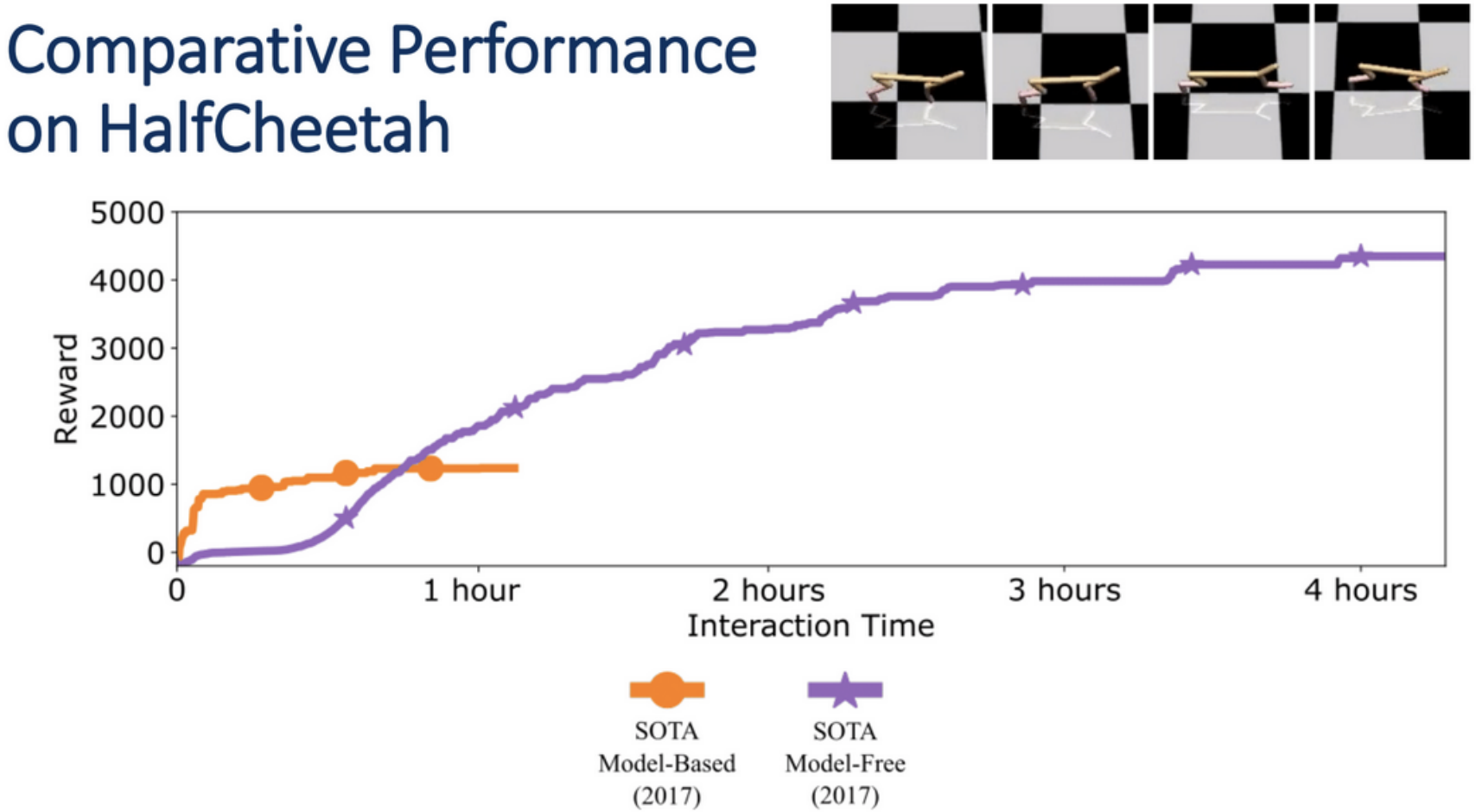
Comparative Performance on HalfCheetah





But, Pure Model-based RL Has Problems…

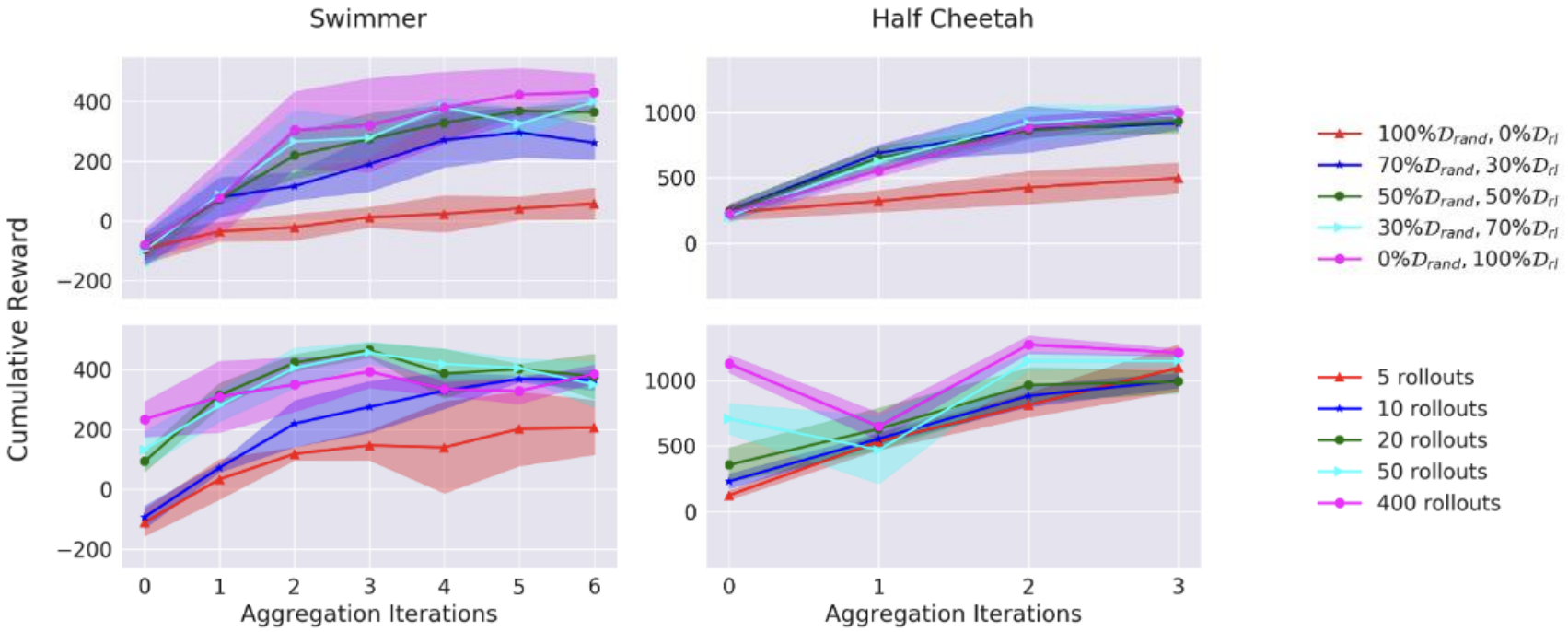
- Pure model-based RL methods are sub-optimal than model-free RL….



Slide credit K. Fragkiadaki

But, Pure Model-based RL Has Problems…

- For MPC to achieve optimality, the learned dynamic models need to know the state transitions of optimal action sequences
- MPC is sub-optimal when the learned dynamic models are trained with random policy rollouts



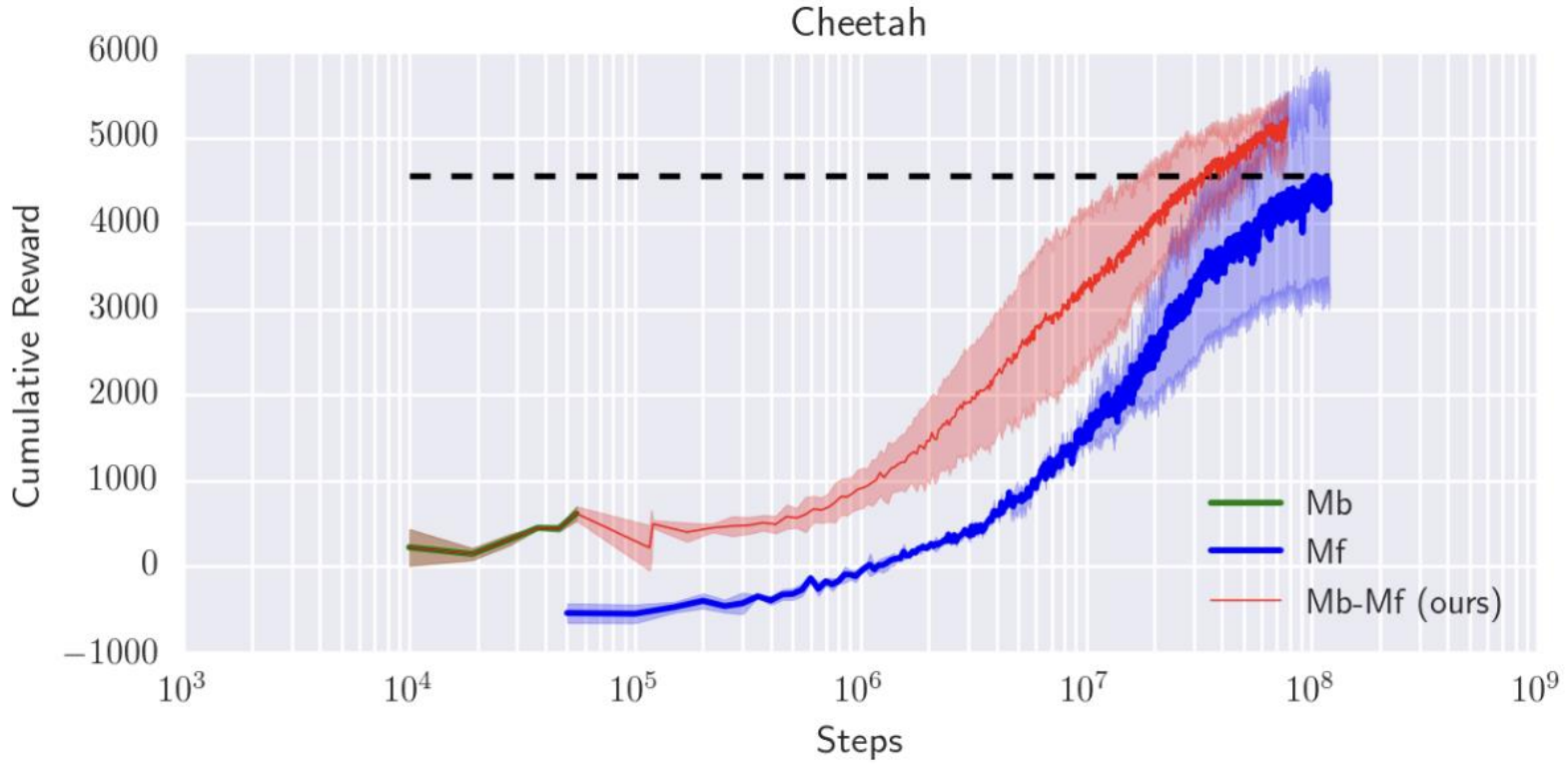
Neural network dynamics for model-based Deep RI with model-free finetuning. Nagaband et al.

Idea: Combine Model-based with Model-free RL

- Initialize the policy with DAGGER, using the rollouts $\mathcal{D}^*$ of MPC controllers as the training data

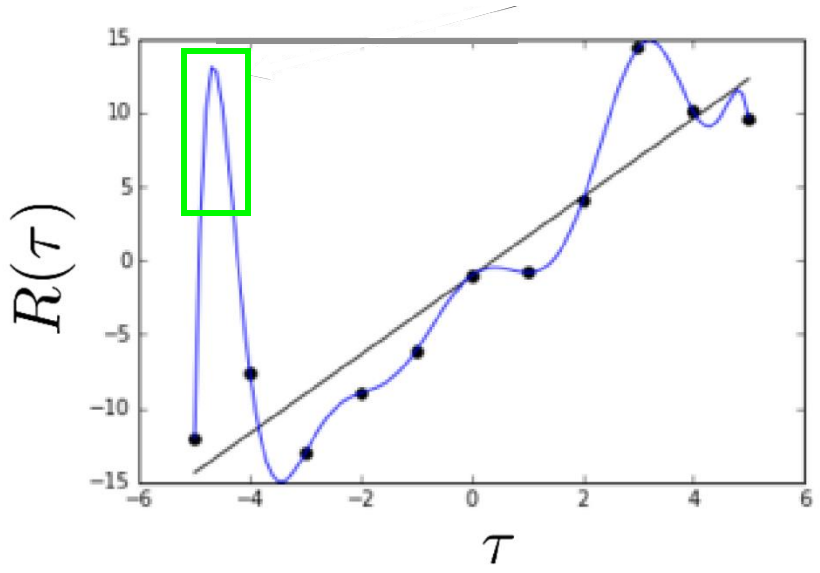
$$\min_{\phi} \frac{1}{2} \sum_{(\mathbf{s}_t, \mathbf{a}_t) \in \mathcal{D}^*} ||\mathbf{a}_t - \mu_\phi(\mathbf{s}_t)||_2^2,$$

- Fine-tun the policy with model-free RL (e.g. TRPO / PPO ...)

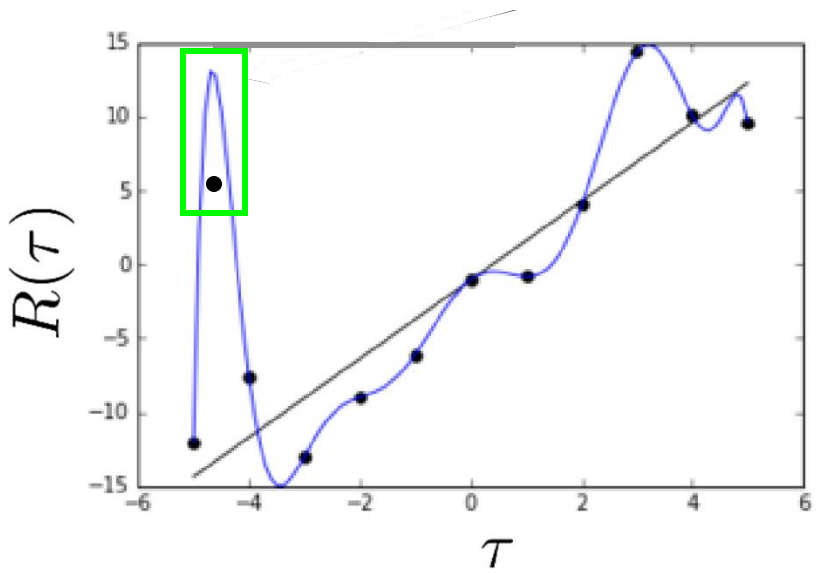


But Why Pure Model-based RL Doesn't Work?

High-variance reward trajectory induced by overfitting



High-variance reward trajectory induced by underfitting

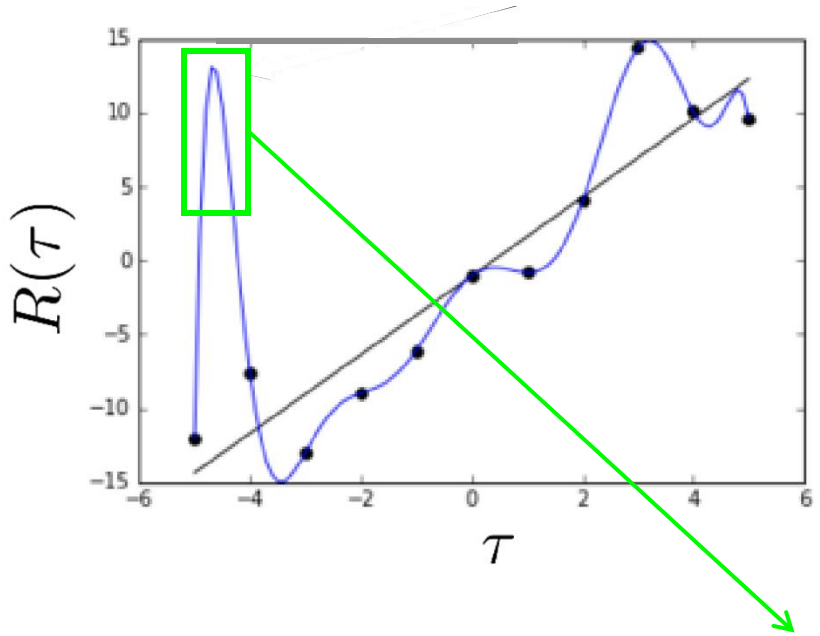


- Incorrect dynamics models lead to inaccurate reward estimates for planned rollouts

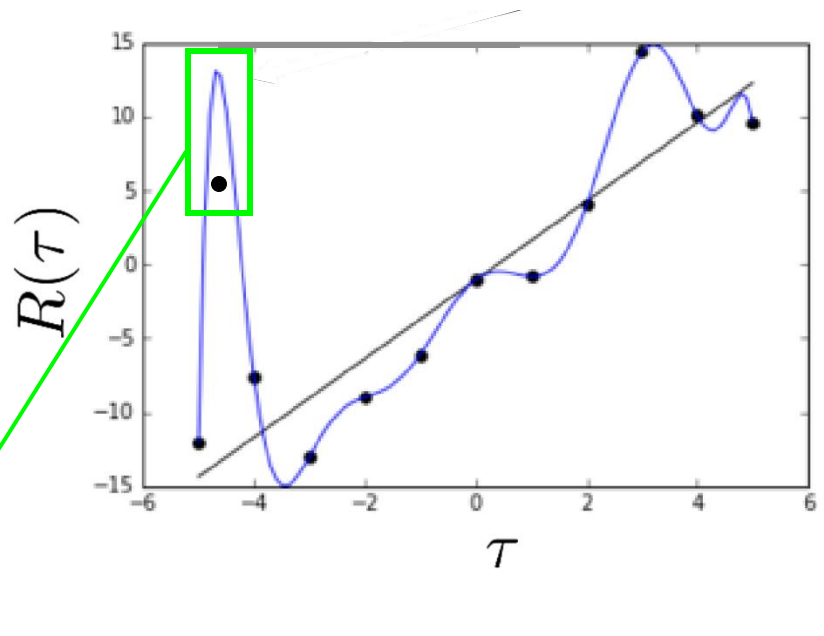
Slide adapted from S. Levine

But Why Pure Model-based RL Doesn't Work?

High-variance reward trajectory induced by overfitting



High-variance reward trajectory induced by underfitting



Model-based RL attempts to go to high-variance reward trajectory, which could have low expected reward instead.

Slide adapted from S. Levine

Idea: Account for Model Uncertainty

- Two types of uncertainty: (1) Epistemic uncertainty, (2) Aleatoric uncertainty

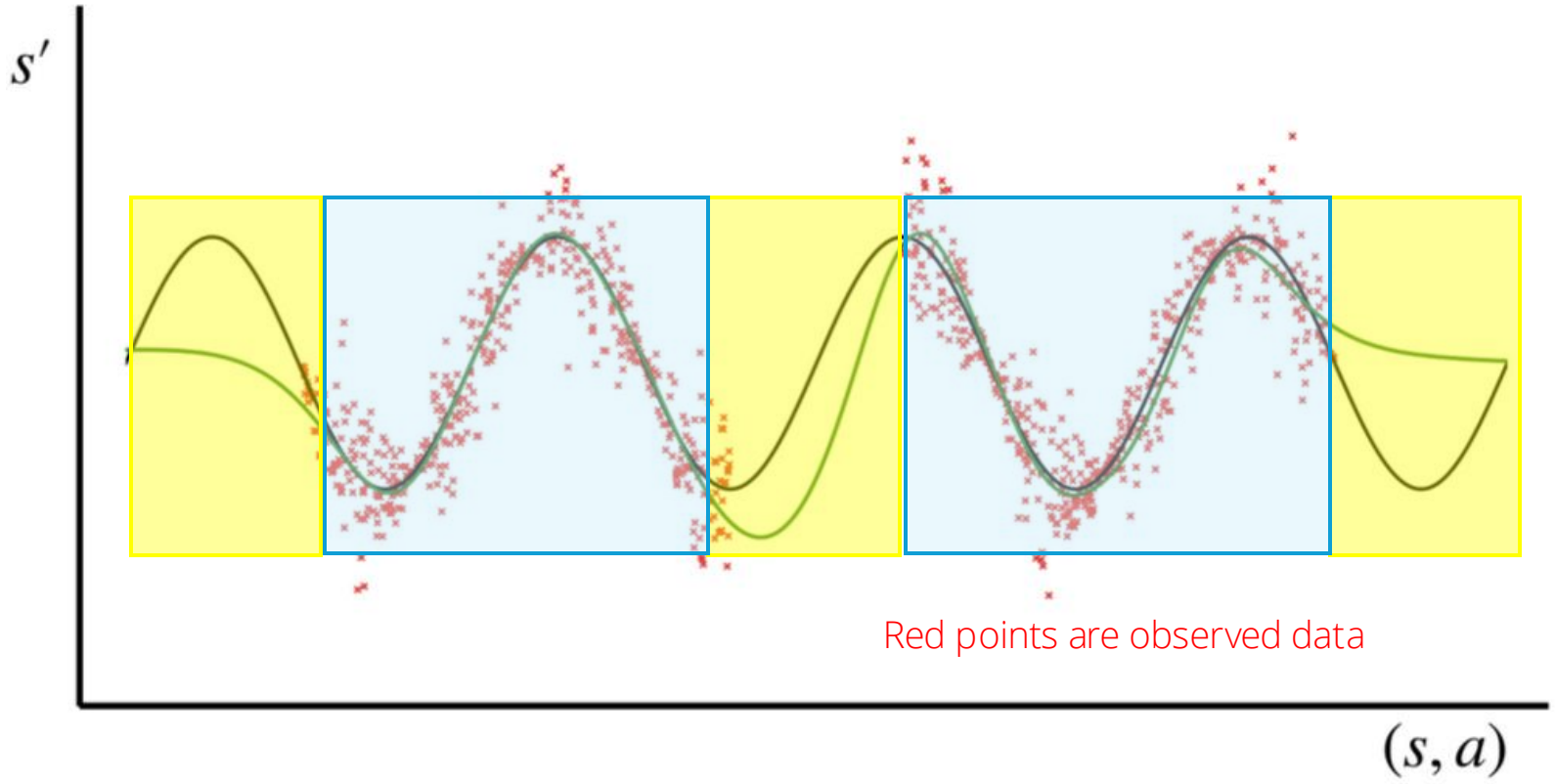


Image credit K. Fragkiadaki

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

Aleatoric Uncertainty

- Aleatoric uncertainty: arises from inherent stochasticity of a system, e.g. observation noise, process noise or multi-modality

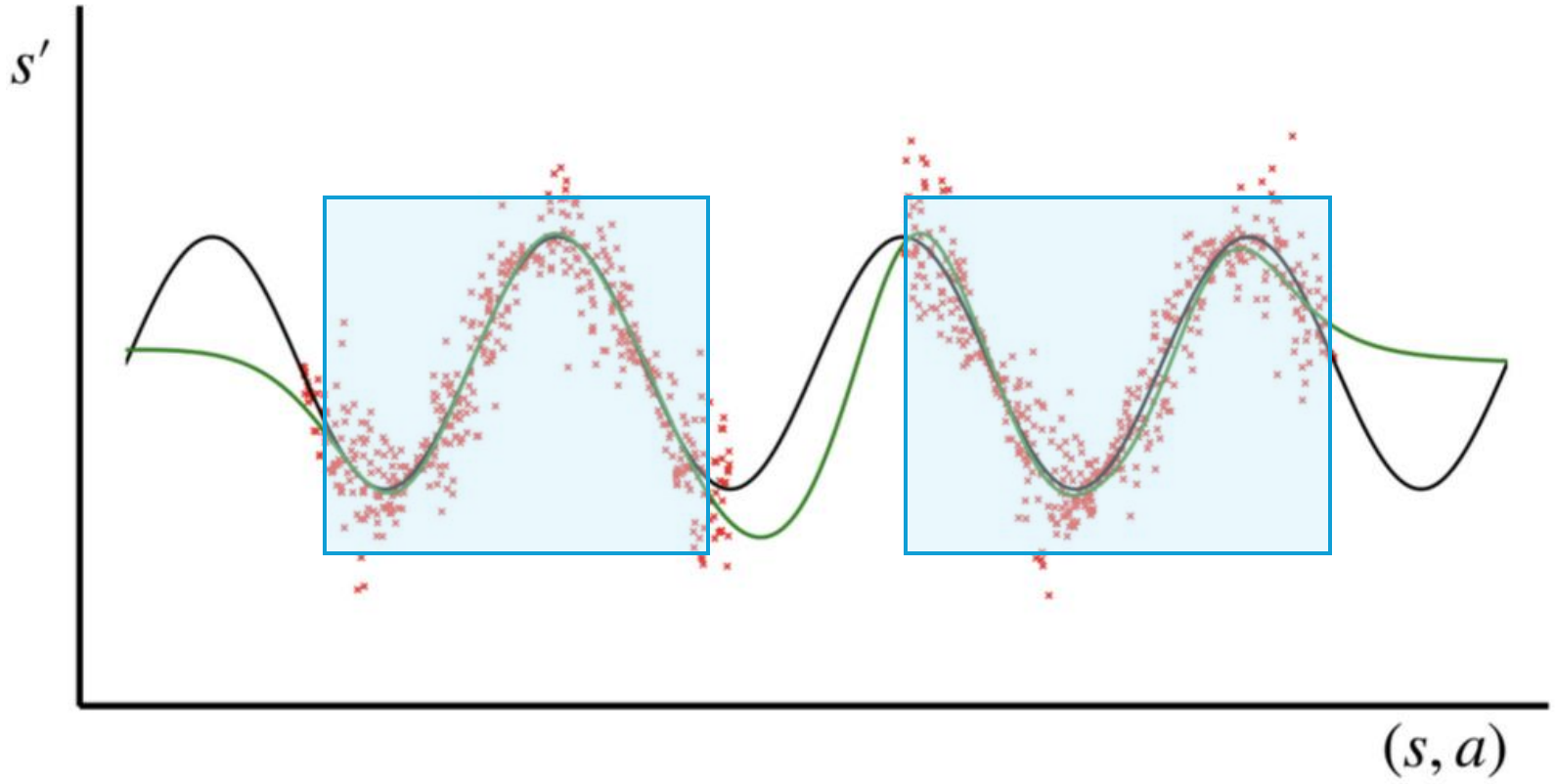


Image credit K. Fragkiadaki

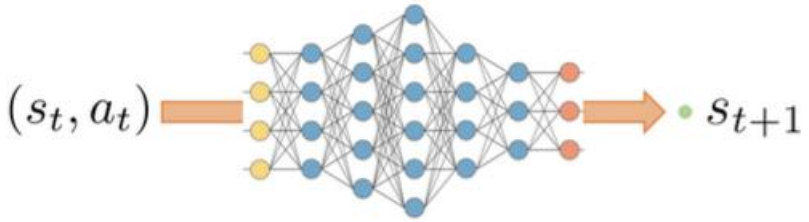
Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

Aleatoric Uncertainty

- Aleatoric uncertainty: arises from inherent stochasticity of a system, e.g. observation noise, process noise or multi-modality

- How to model aleatoric uncertainty? If the environment is stochastic:
  - ➢ Deterministic regression models fails to capture the output distribution

$$\mathcal{L}_\phi = \sum_{i=1}^{N} \|f(s_i, a_i; \phi) - s_i'\|$$



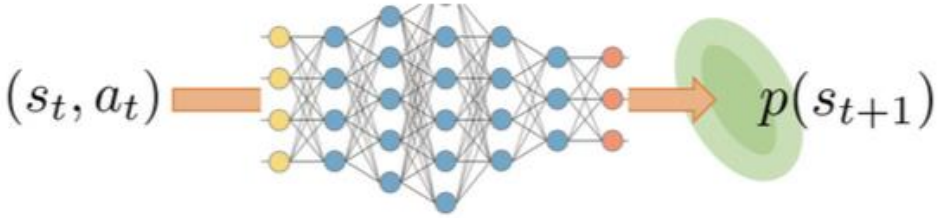
Aleatoric Uncertainty

- Aleatoric uncertainty: arises from inherent stochasticities of a system, e.g. observation noise, process noise or multi-modality

- How to model aleatoric uncertainty? If the environment is stochastic:
  - Deterministic regression models fails to capture the output distribution
  - Probabilistic models can capture the output distribution

- An example that captures Gaussian distribution:

$$p_\phi(s'|s,a) = \frac{\exp\left(-\frac{1}{2}(s'-\mu(s,a;\phi)^\top(\mathbf{\Sigma}(s,a;\phi))^{-1}(s'-\mu(s,a;\phi)\right)}{\sqrt{(2\pi)^d \det\mathbf{\Sigma}(s,a;\phi)}}$$



$$\mathcal{L}_\phi = -\frac{1}{N}\sum_{i=1}^{N}\log p_\phi(s_i'|s_i,a_i)$$

$$= \frac{1}{2}(s_i'-\mu(s_i,a_i;\phi))^T\mathbf{\Sigma}(s_i,a_i;\phi)^{-1}(s_i'-\mu(s_i,a_i;\phi)) + \frac{1}{2}\log(\det\mathbf{\Sigma}(s_i,a_i;\phi)) + \text{ const.}$$

Slide credit K. Fragkiadaki

Epistemic Uncertainty

- Epistemic uncertainty: subjective uncertainty about the dynamics function, due to a lack of sufficient data to uniquely determine the underlying system exactly
- The model is certain about the data, but we are not certain about the model

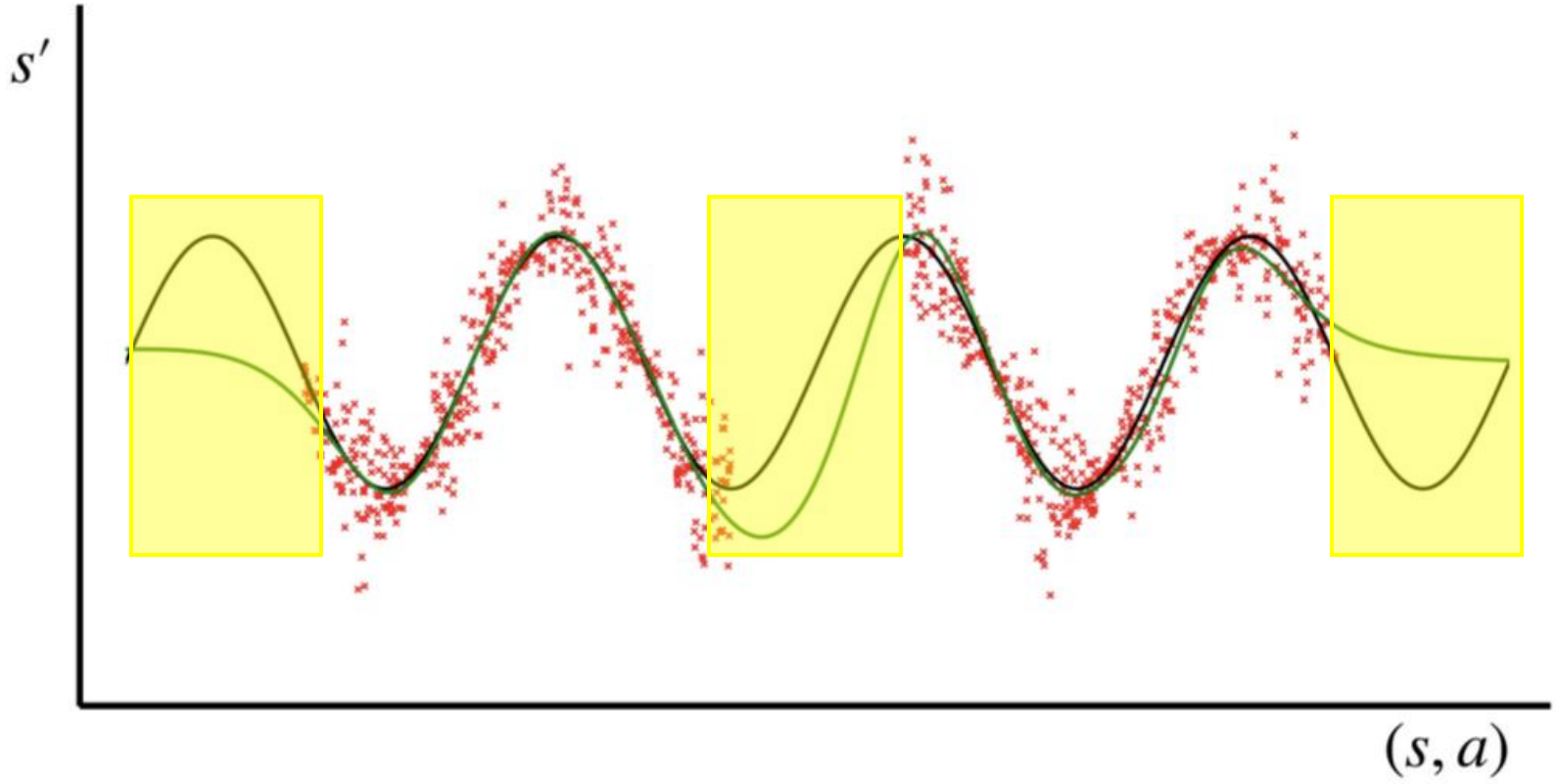


Image credit K. Fragkiadaki

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

How to Model Epistemic Uncertainty? A Bayesian Inference Method

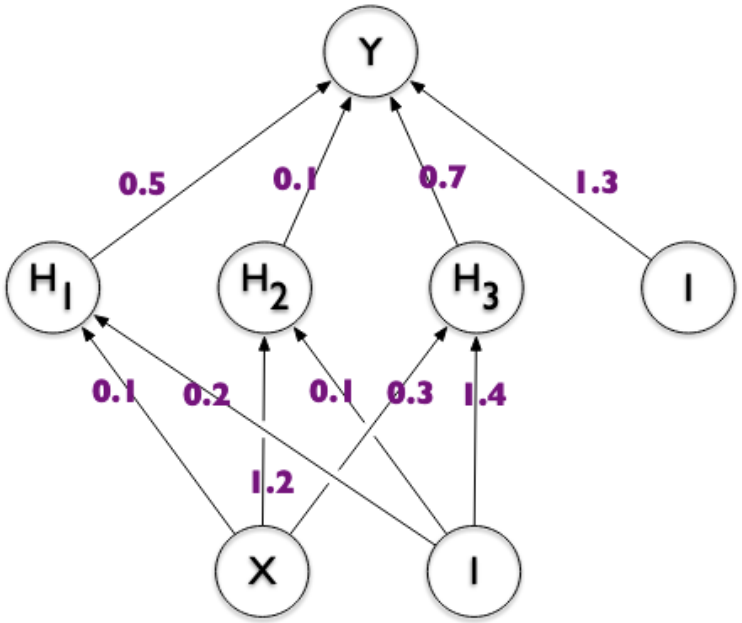
$$P(\text{ hypothesis } | \text{ data }) = \frac{P(\text{ hypothesis })P(\text{ data } | \text{ hypothesis })}{\sum_h P(h)P(\text{ data } | h)}$$

- Hypotheses here are weights for our learning model, i.e., weights of our neural networks that learns the transition dynamics
- The idea is to keep all the hypotheses that fit equally well the training set instead of committing to one, so that I can represent my uncertainty

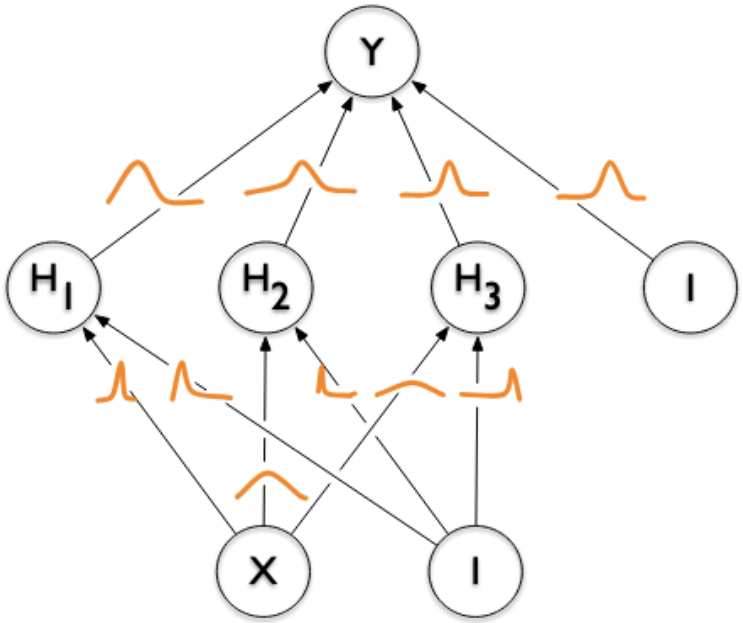
Slide credit K. Fragkiadaki

How to Model Epistemic Uncertainty? A Bayesian Inference Method

Network with fixed weights



Network with a distribution of weights

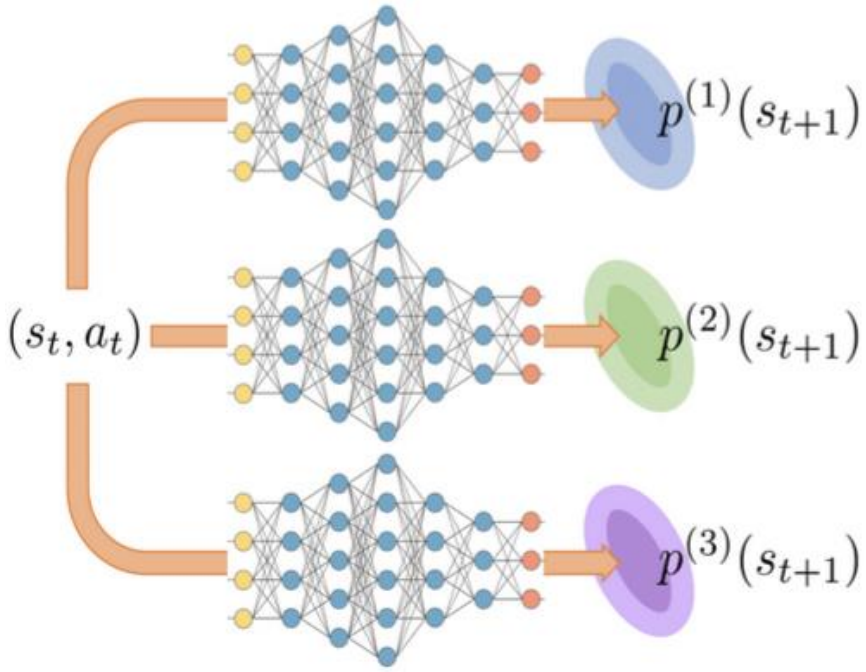


predict according to:

$$\int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta) p(\theta|\mathcal{D}) d\theta$$

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

How to Model Epistemic Uncertainty? An Ensemble-of-Bootstraps Method



- Model ensembles are good approximation of Bayesian Nets:

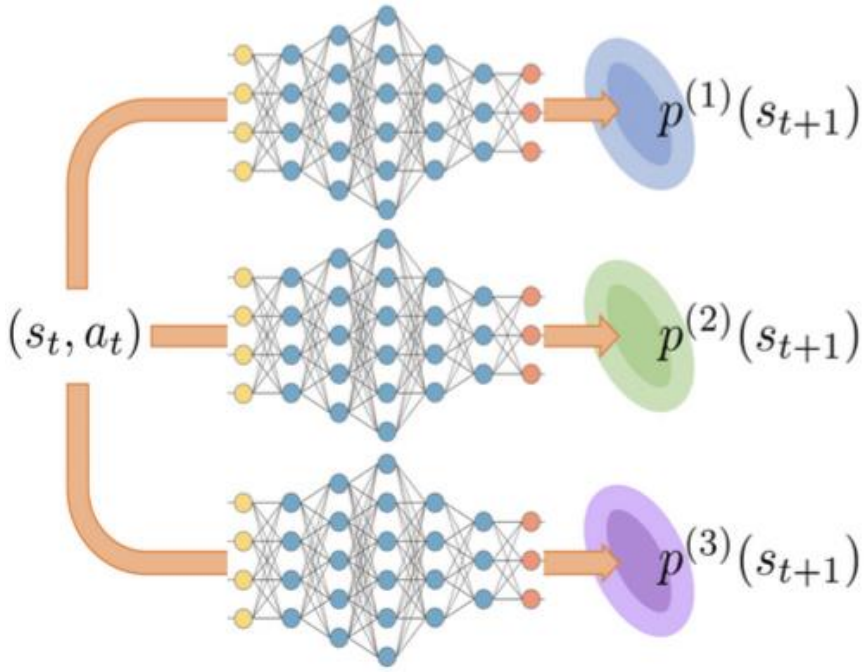
$$p(\theta|D) \approx \frac{1}{N}\Sigma_i\, \sigma(\theta_i)$$

$$\int p(s_{t+1}|s_t, a_t, \theta)p(\theta|D)d\theta \approx \frac{1}{N}\Sigma_i\, p(s_{t+1}|s_t, a_t, \theta_i)$$

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

Slide adapted from K. Fragkiadaki

How to Model Epistemic Uncertainty? An Ensemble-of-Bootstraps Method



- Model ensembles are good approximation of Bayesian Nets:

$$p(\theta|D) \approx \frac{1}{N}\Sigma_i \sigma(\theta_i)$$

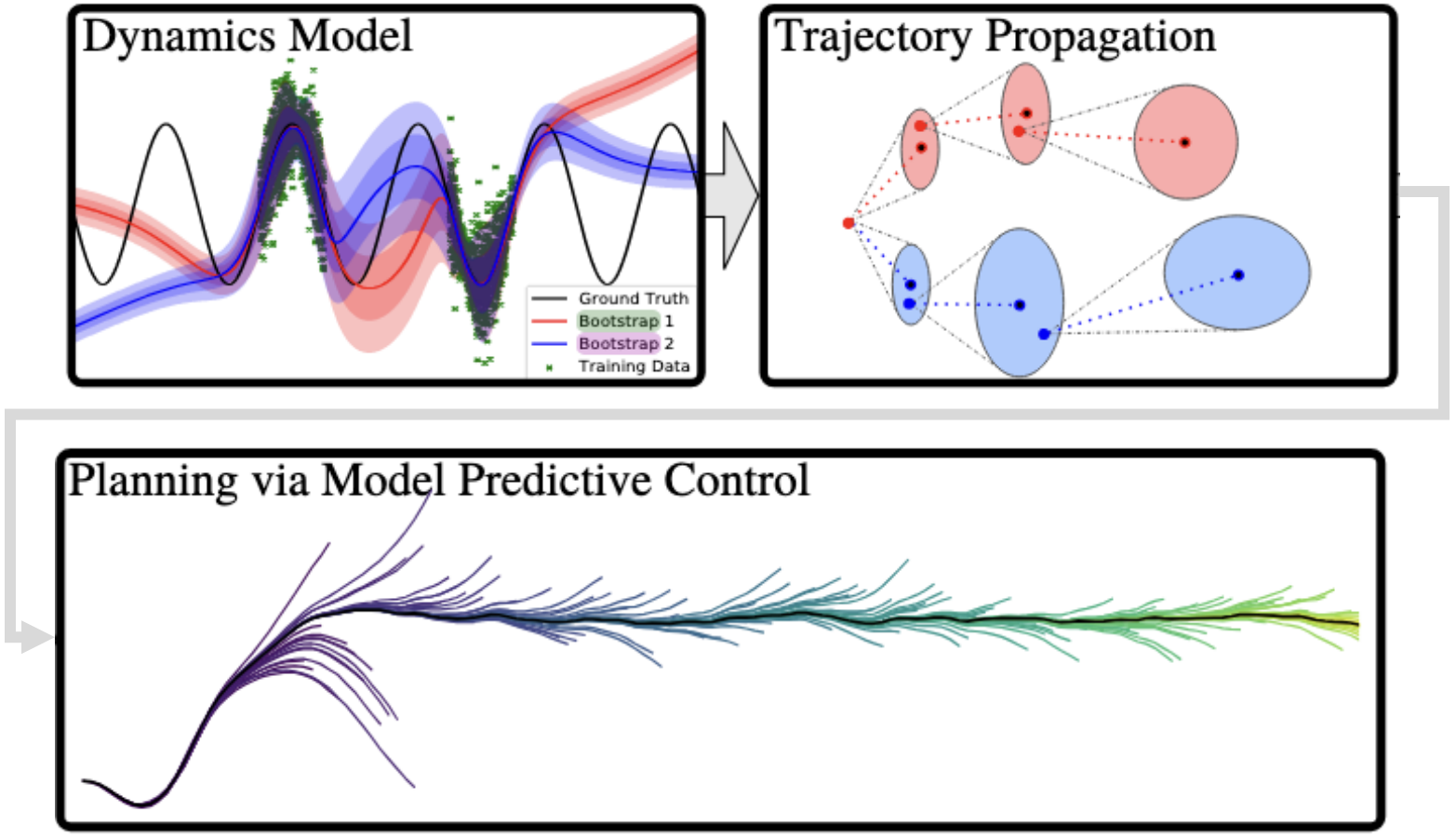
$$\int p(s_{t+1}|s_t, a_t, \theta)p(\theta|D)d\theta \approx \frac{1}{N}\Sigma_i p(s_{t+1}|s_t, a_t, \theta_i)$$

- Main idea: generate "independent" datasets for each "independent" model:
  - For seen data, they all agree (low-entropy prediction)
  - For unseen data, they disagree (high-entropy prediction)

Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

Slide adapted from K. Fragkiadaki

Unrolling an Ensemble of Bootstraps



Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

Unrolling an Ensemble of Bootstraps

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

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

Trajectory sampling: the strategy to sample bootstrap as the dynamics model

Results: Performant Model-Based RL

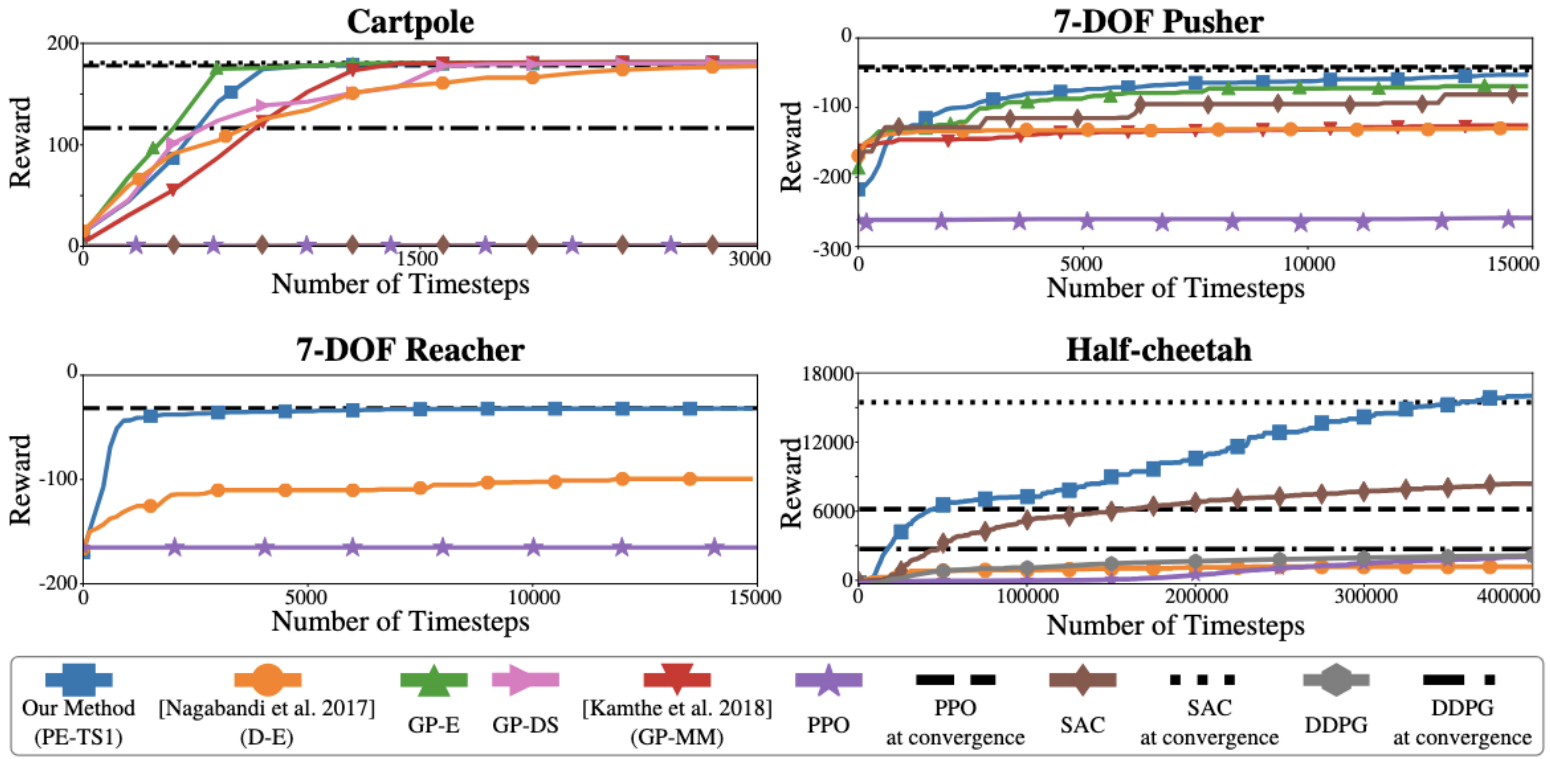
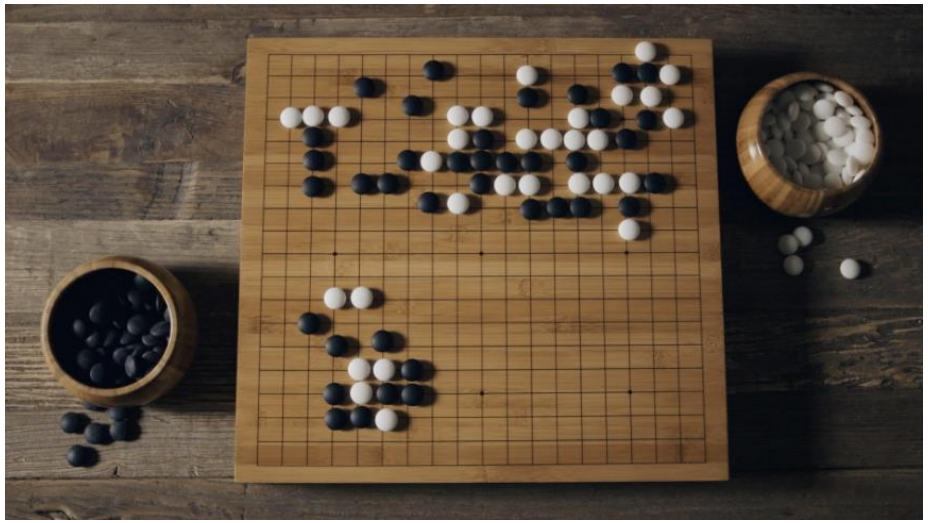


Figure 3: Learning curves for different tasks and algorithm. For all tasks, our algorithm learns in under 100K time steps or 100 trials. With the exception of Cartpole, which is sufficiently low-dimensional to efficiently learn a GP model, our proposed algorithm significantly outperform all other baselines. For each experiment, one time step equals 0.01 seconds, except Cartpole with 0.02 seconds. For visual clarity, we plot the average over 10 experiments of the maximum rewards seen so far.

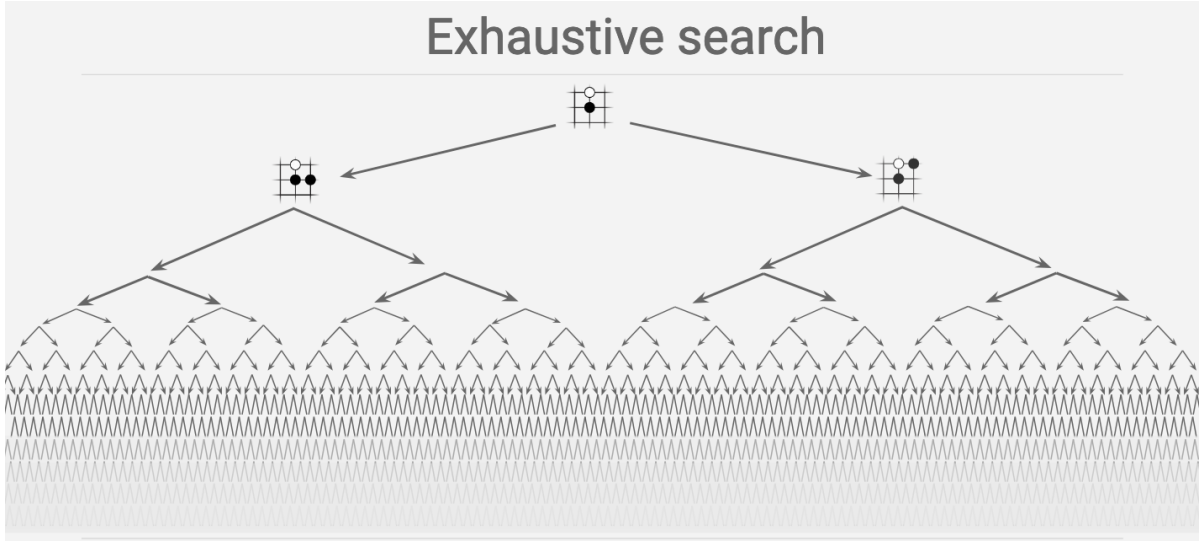
Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. Chua et al.

From AlphaGO (Model-free RL) to MuZero (Model-based RL)

Go is Hard…



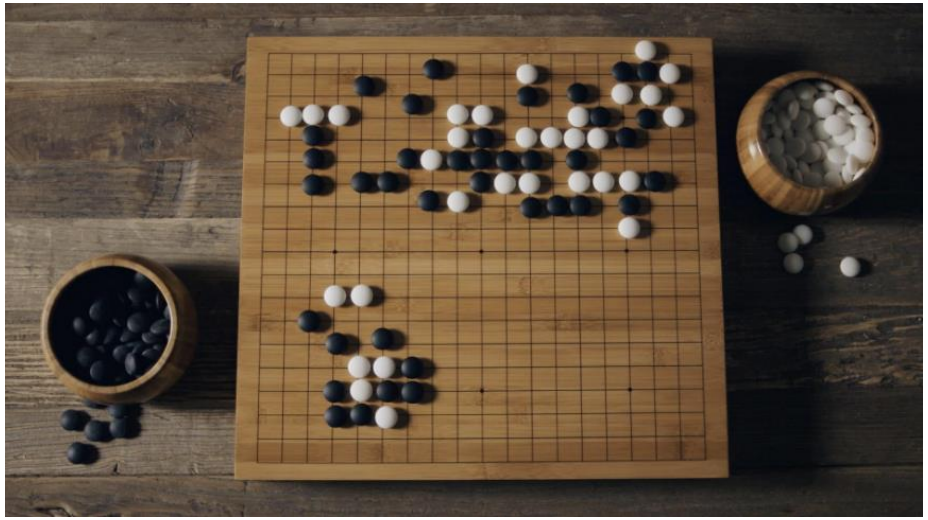
- Game tree complexity $= b^d$ (b: game breadth, d: game depth)
  - ➢ Search space is huge
  - ➢ "Impossible" for computers to evaluate who is winning



https://icml.cc/2016/tutorials/AlphaGo-tutorial-slides.pdf

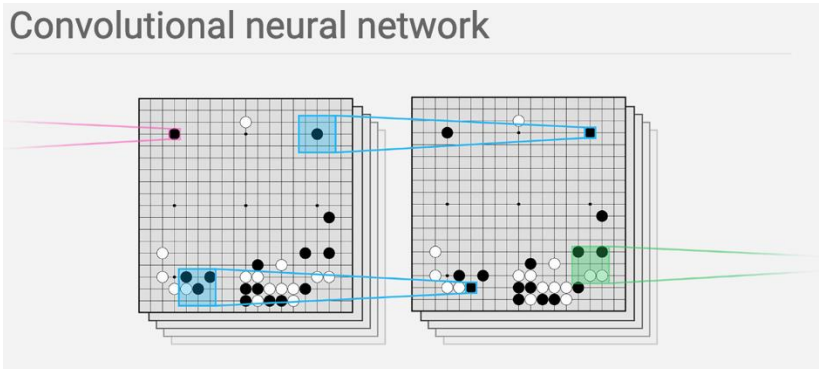
Mastering the game of Go with deep neural networks and tree search. Silver et al.

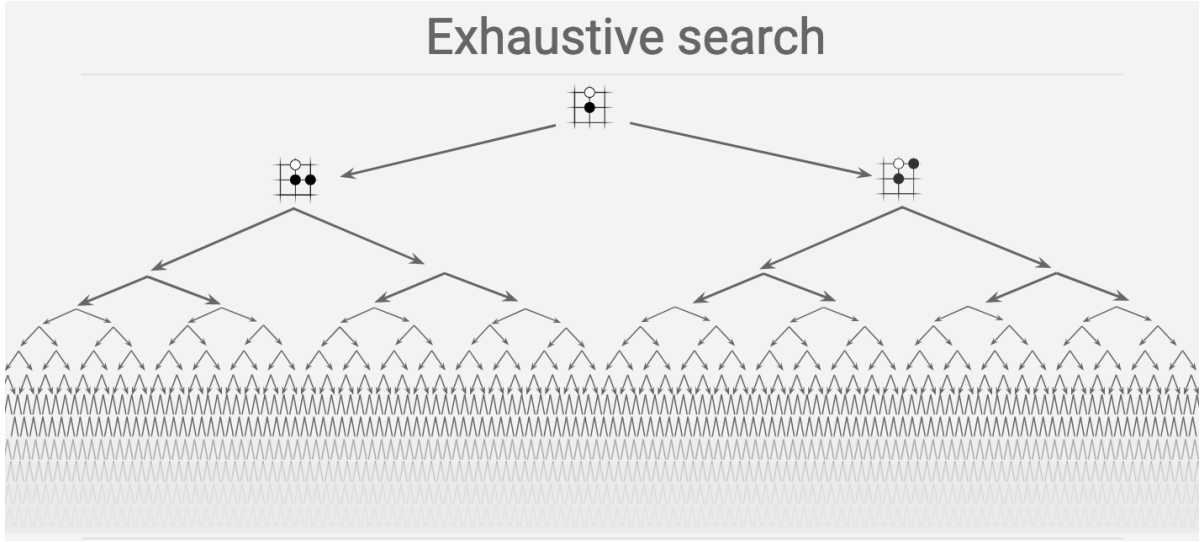
Go is Hard…



- Game tree complexity $= b^d$ (b: game breadth, d: game depth)
  - ➢ Search space is huge
  - ➢ "Impossible" for computers to evaluate who is winning

Visual encoder:





https://icml.cc/2016/tutorials/AlphaGo-tutorial-slides.pdf

Mastering the game of Go with deep neural networks and tree search. Silver et al.

Imitation Learning with Expert Demonstrations

Policy network: 12 layer convolutional neural network

Training data: 30M positions from human expert games (KGS 5+ dan)

Training algorithm: maximise likelihood by stochastic gradient descent

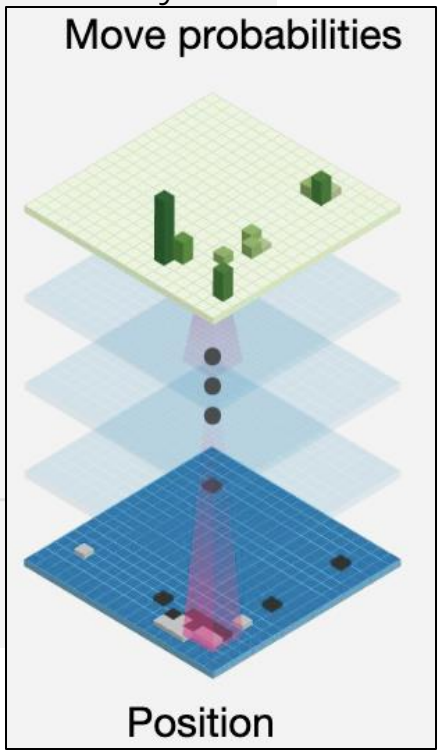
$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

Training time: 4 weeks on 50 GPUs using Google Cloud

Results: 57% accuracy on held out test data (state-of-the art was 44%)

Google DeepMind

Policy network:



Reinforcement Learning with Policy Gradient

Policy network: 12 layer convolutional neural network

Training data: games of self-play between policy network

Training algorithm: maximise wins z by policy gradient reinforcement learning

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma} z$$

z denotes if the current state leads to a winning game

Training time: 1 week on 50 GPUs using Google Cloud

Results: 80% vs supervised learning. Raw network ~3 amateur dan.

Google DeepMind

https://icml.cc/2016/tutorials/AlphaGo-tutorial-slides.pdf

Can We Do Better? Yes! MCTS + Policy/Value Networks

Value network: 12 layer convolutional neural network

Training data: 30 million games of self-play

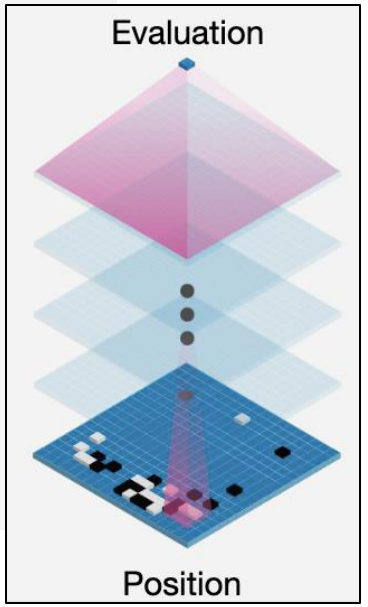
Training algorithm: minimise MSE by stochastic gradient descent

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

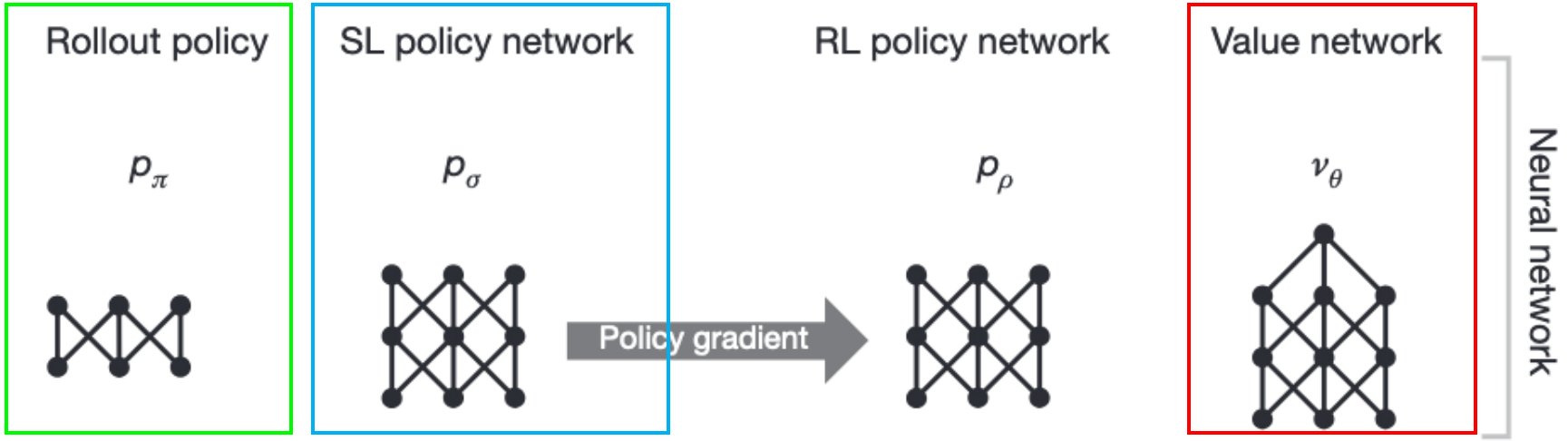
Training time: 1 week on 50 GPUs using Google Cloud

Results: First strong position evaluation function - previously thought impossible

Google DeepMind



Can We Do Better? Yes! MCTS + Policy/Value Networks



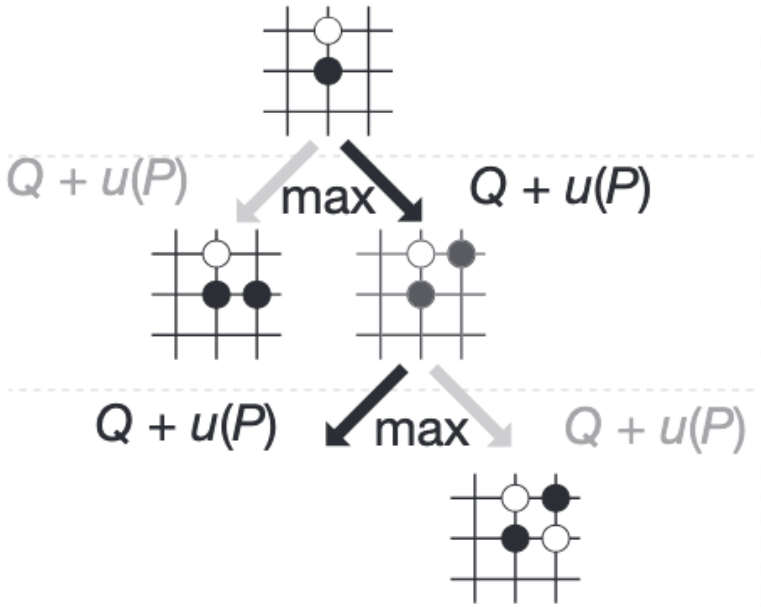
A cheap SL policy used to simulate the whole game during testing

A strong SL policy provide prior distribution of action at each state

Estimate if the current state will lead to "win"

Monte-Carlo Tree Search During Testing

Selection: selecting actions within the expanded tree



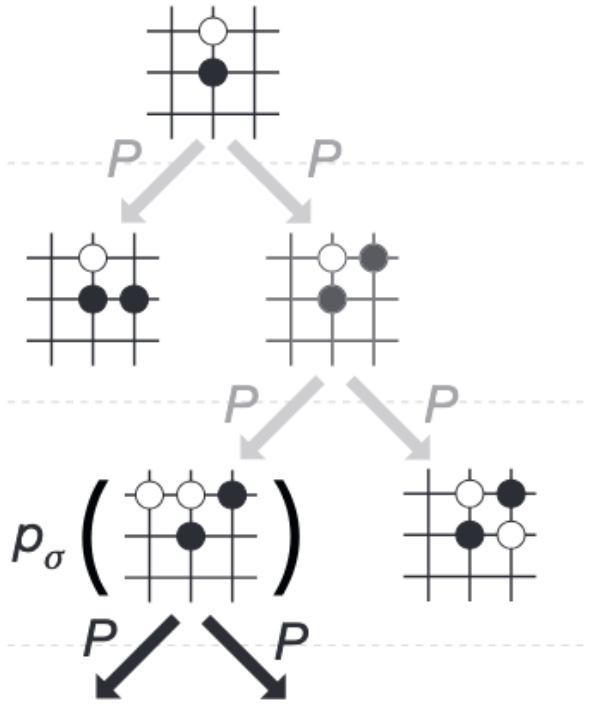
$$a_t = \underset{a}{\text{argmax}} \underbrace{Q\left(s_t, a\right)}_{\text{exploitation}} + \underbrace{c\sqrt{N(s_t)}\frac{\pi_\theta(a_t \mid s_t)}{1 + N(s_t, a_t)}}_{\text{exploration}}$$

- a_t - action selected at time step t from state s_t
- $Q\left(s_r, a\right)$ - average reward collected so far from MC simulations
- $\pi_\theta(a \mid s)$ - prior expert probability provided by the SL policy p_σ
- $N(s, a)$ - number of times we have taken action a from state s during MC simulations

Slide credit K. Fragkiadaki

Monte-Carlo Tree Search During Testing

Expansion: when reaching a leaf, play the action with highest score from p_σ

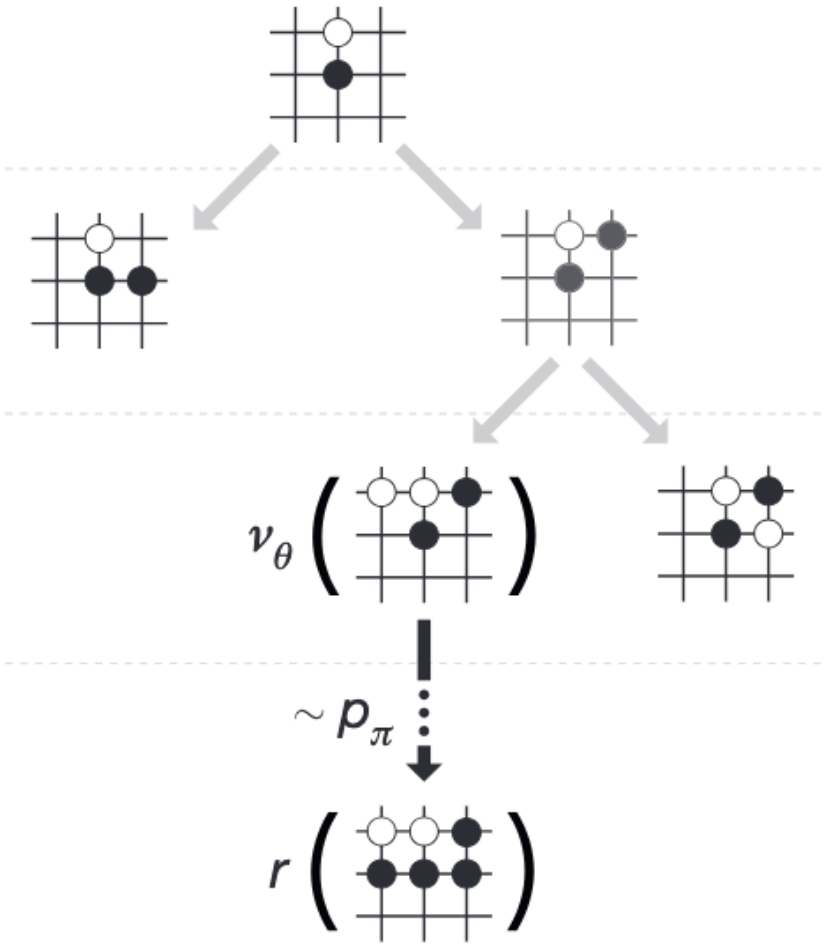


- When leaf node is reached, it has a chance to be expanded
- Processed once by SL policy network (p_σ) and stored as prior probs $P(s, a)$
- Pick child node with highest prior prob

Slide credit K. Fragkiadaki

Monte-Carlo Tree Search During Testing

Simulation/Evaluation: use the rollout policy to reach to the end of the game



- From the selected leaf node, run multiple simulations in parallel using the rollout policy
- Evaluate the leaf node as:

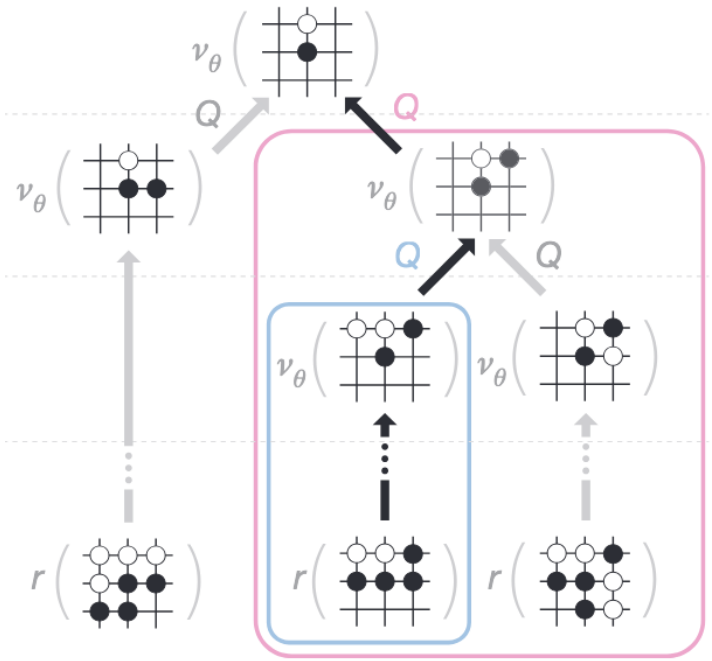
$$V(s_L) = (1-\lambda)v_\rho(s_L) + \lambda z_L$$

- v_ρ : value from the trained value function for board position s_L
- z_L : Reward from fast rollout p_x
  - Played until terminal step
- λ - mixing parameter

Slide credit K. Fragkiadaki

Monte-Carlo Tree Search During Testing

- Backup: update visitation counts and recorded rewards for the chosen path inside the tree



$$N(s,a) = \sum_{i=1}^{n} \mathbf{1}_{(s,a)\in\tau_i}$$

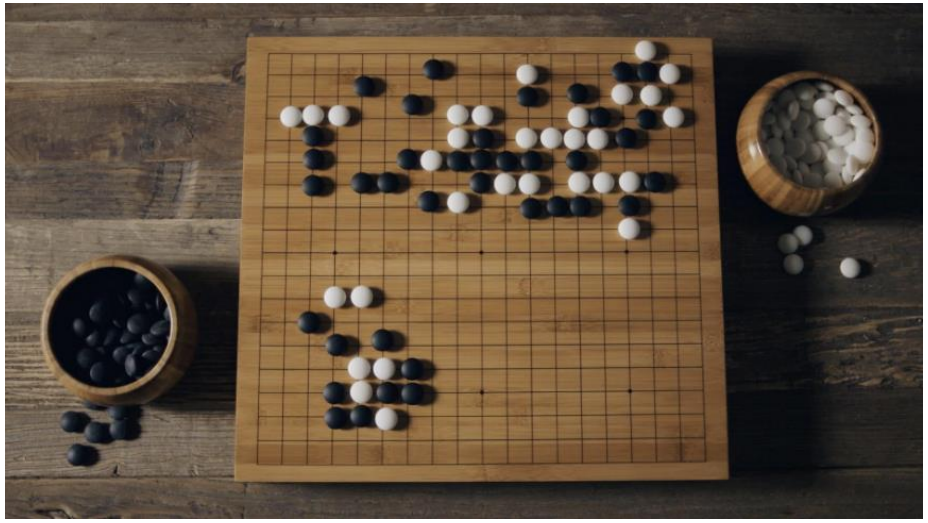
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} \mathbf{1}_{(s,a)\in\tau_i} V\left(s_L^i\right)$$

- Extra index is to denote the i simulation, n total simulations
- Update visit count and mean reward of simulations passing through node
- Once MCTS completes, *the algorithm chooses the most visited move from the root position.*

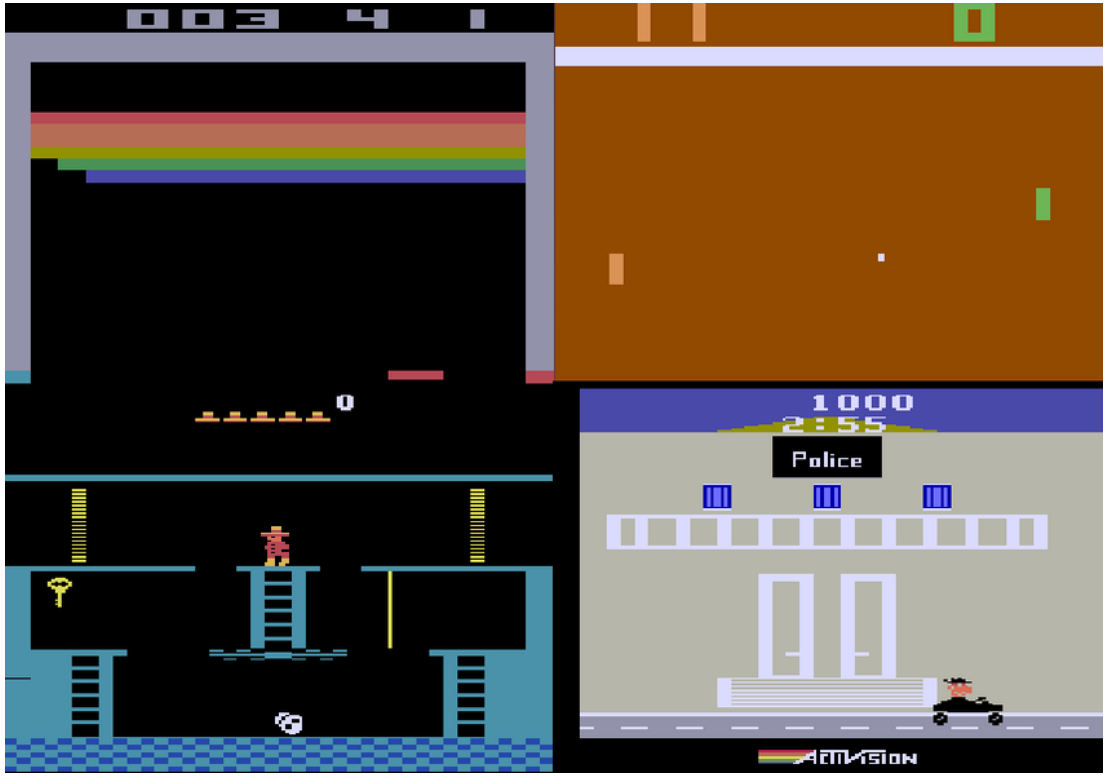
Slide credit K. Fragkiadaki

To Do Monte-Carlo Tree Search During Test Time, We Need to Simulate

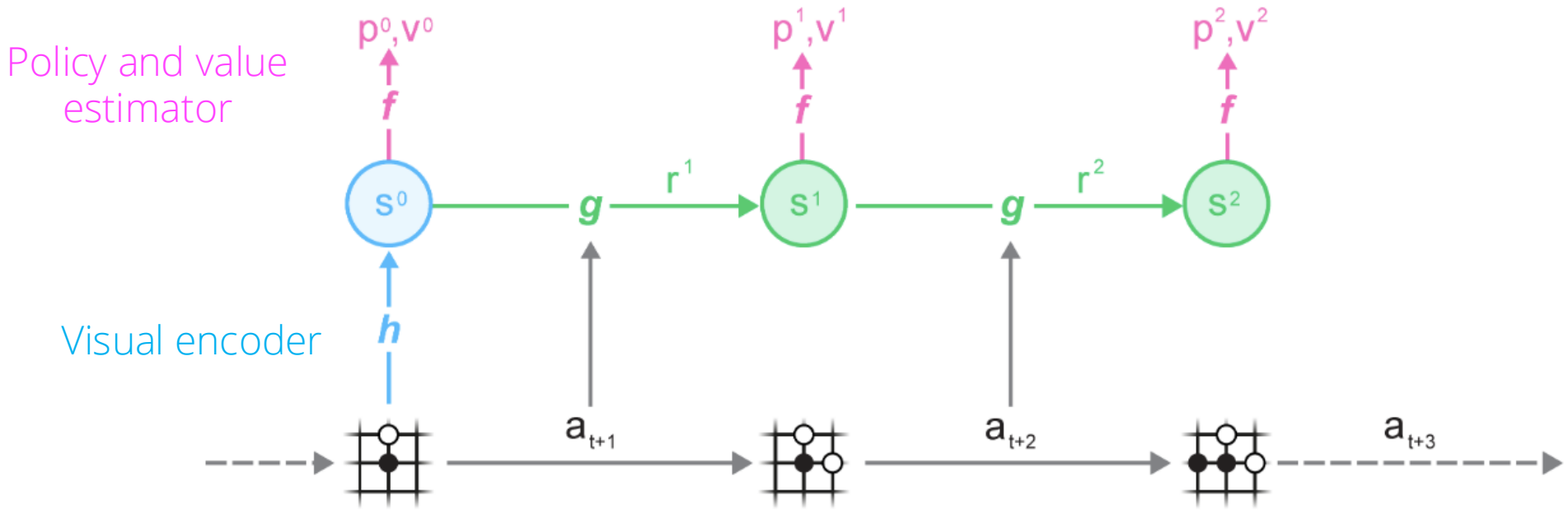
The rule is clear, trivial to build a simulator



The rule is complex, non-trivial to build a simulator

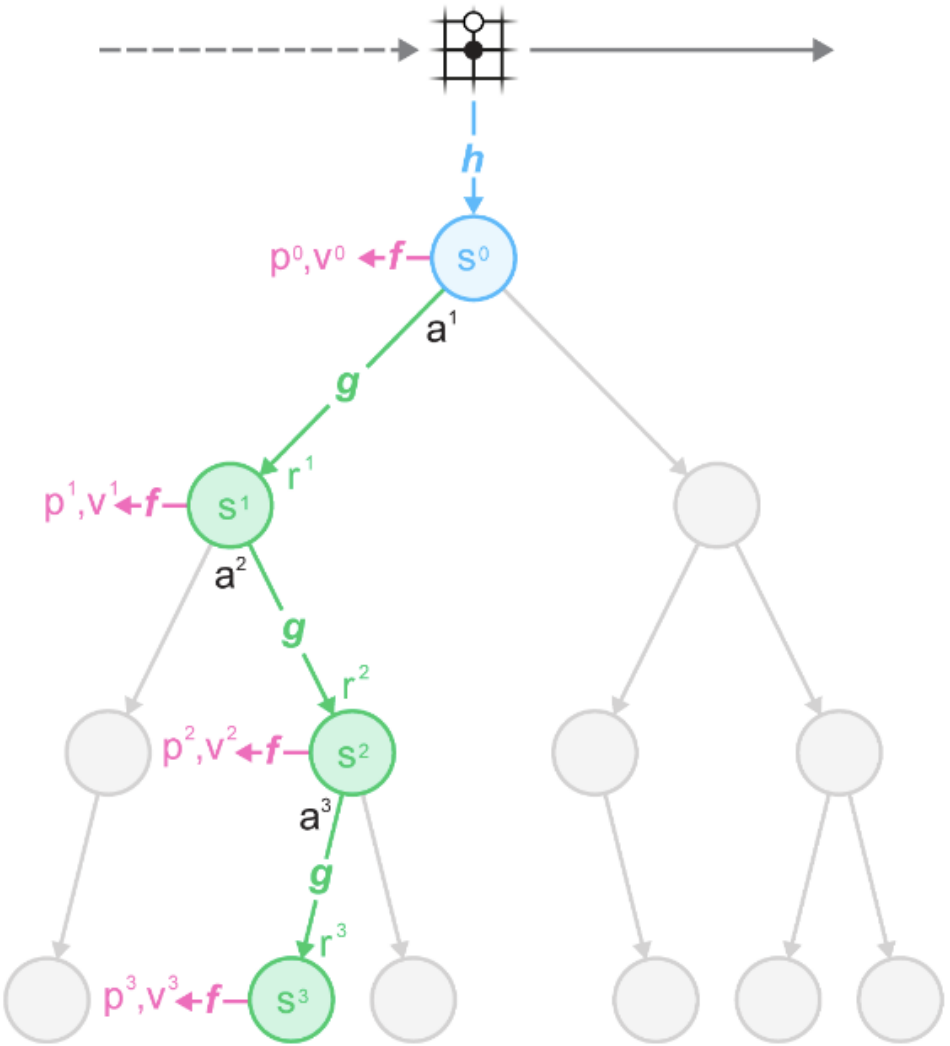


Learn a Dynamics Model for Different Games



- The dynamic model $r^t, s^t = g_\theta(s^{t-1}, a^t)$
- State s^t has no semantics of environment state attached to it, simply the hidden state of the overall model, and its sole purpose is to accurately predict relevant, future quantities: policies, values, and rewards

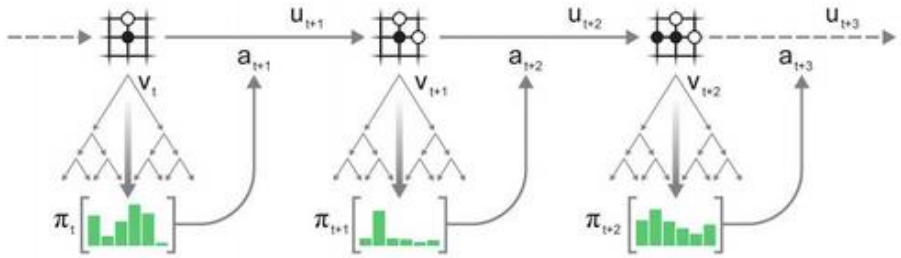
Use the Learned Dynamic Model for MCTS



Use MCTS to Improve Policy and Value Estimation

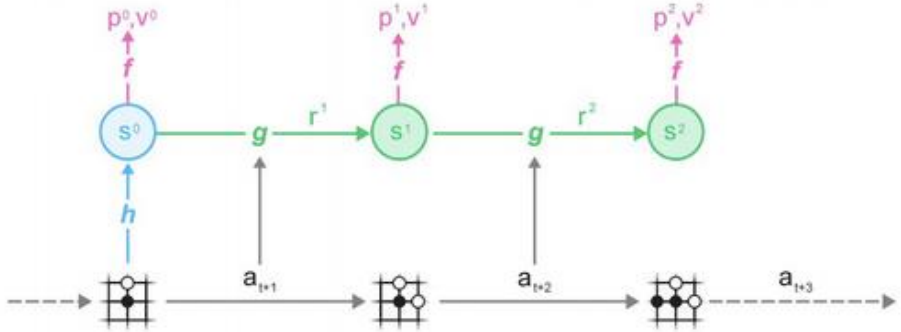
Policy MCTS Distillation

$$p_t \approx \pi_t$$



Reward

$$r_t \approx u_t$$

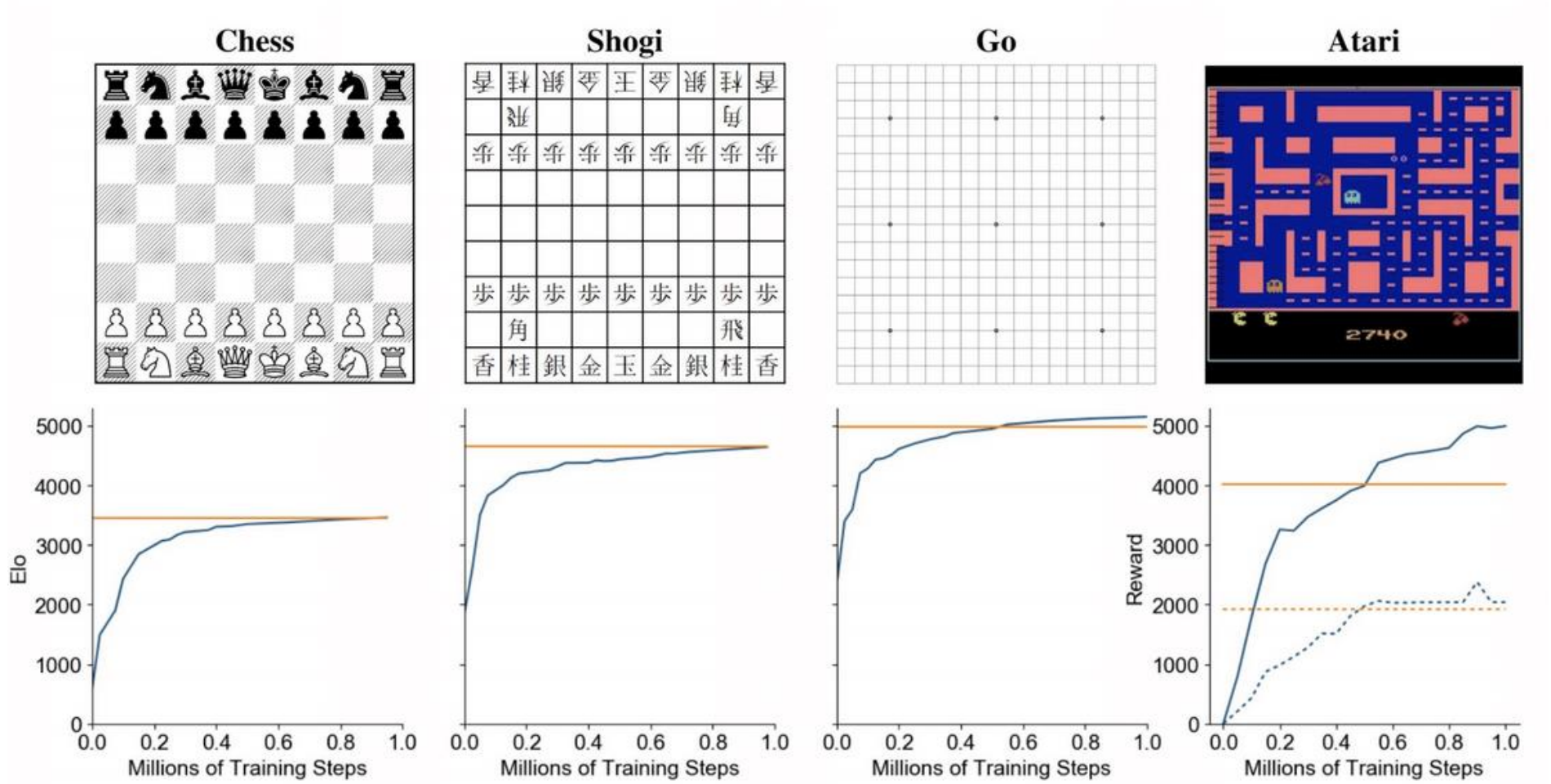


Value n-step bootstrapping

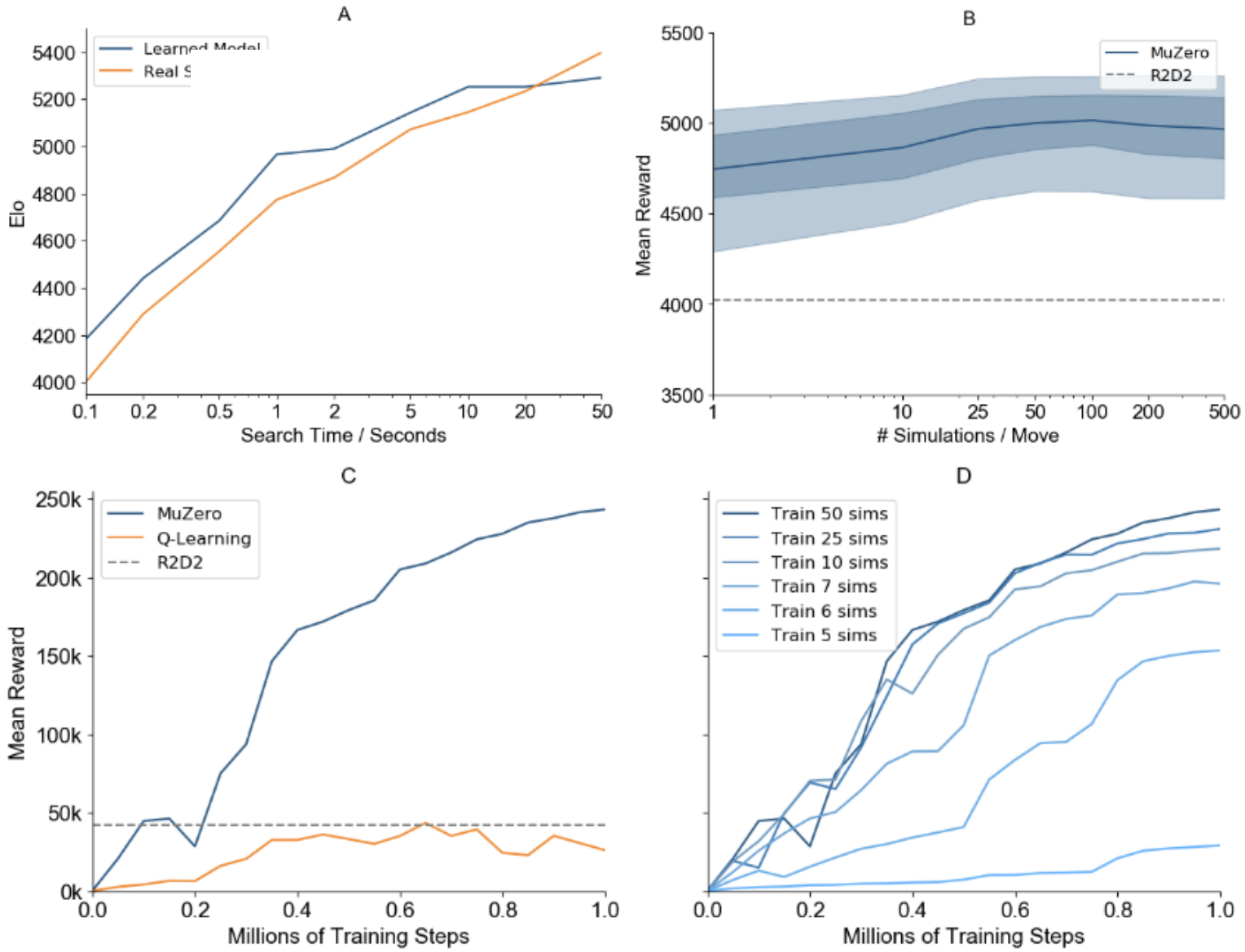
$$v_t \approx u_{t+1} + \gamma u_{t+2} + \ldots + \gamma^{n-1} u_{t+n} + \gamma^n \nu_{t+n}$$

Slide credit DeepMind

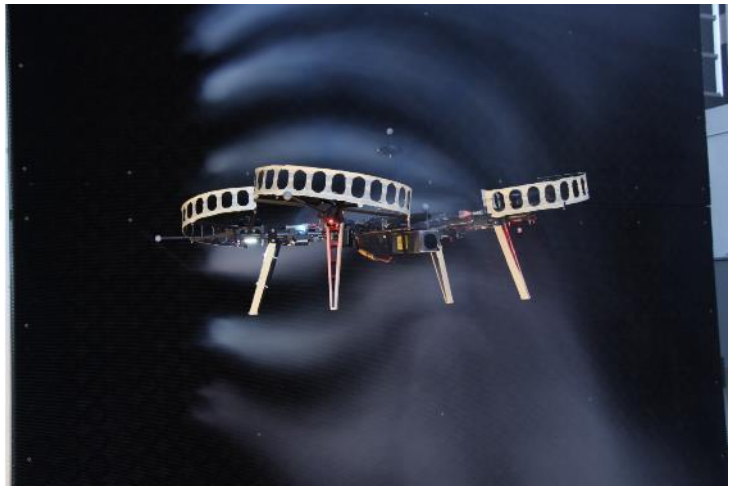
Results


Chess
Shogi
Go
Atari
2740
Elo
Reward
Millions of Training Steps
AlphaZero
MuZero
R2D2 mean
R2D2 median

MuZero is Much More Sample Efficient

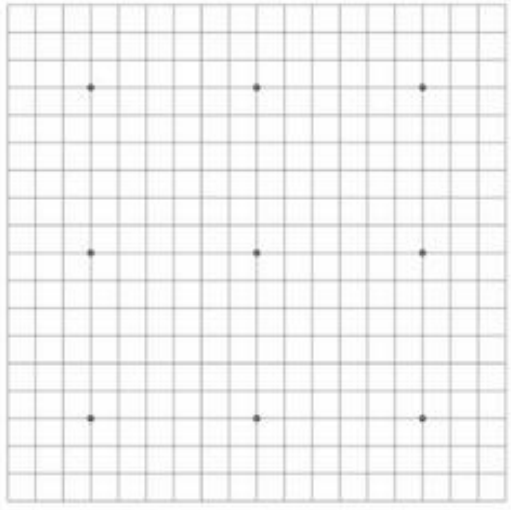


What if Real-world Model is not Stationary?



AnyCar to Anywhere Learning Universal Dynamics Model for Agile and Adaptive Mobility. Xiao et al.

v.s



Adaptive Model-Based RL

- The formulation of the dynamics model:

$$x_{t+1:t+H} \approx f_\theta^{\text{AnyCar}}\Big(\underbrace{\hat{x}_{t-K:t}, \hat{a}_{t-K:t-1}}_{\text{noisy state and action history}} , \underbrace{a_{t:t+H-1}}_{\text{future actions}} \Big),$$

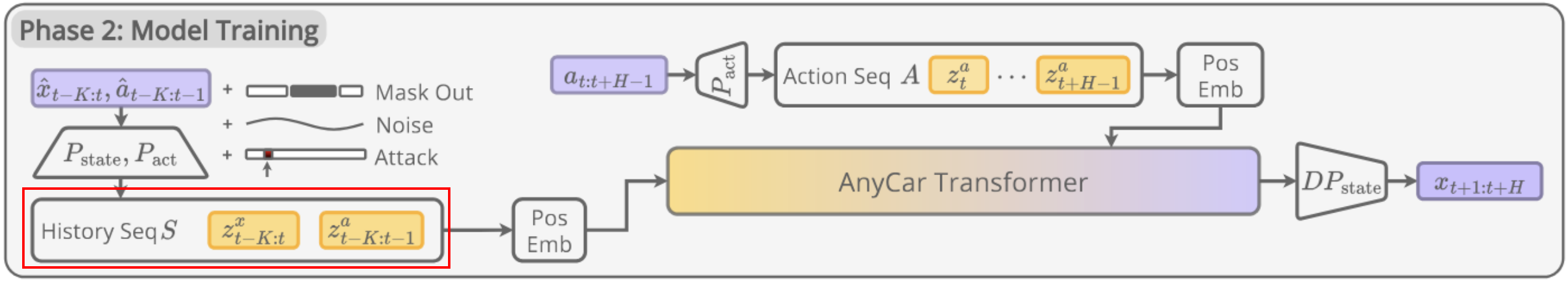
- The definition of states:

$$x_t \triangleq [p_t^x, p_t^y, \psi_t, \dot{p}_t^x, \dot{p}_t^y, \omega]$$

where (p_t^x, p_t^y) denotes the car position, ψ_t is the heading angle, $(\dot{p}_t^x, \dot{p}_t^y)$ denotes the velocity, and ω is the angular velocity

- The definition of actions a_t contain the throttle and steering angle

Adaptive Model-Based RL



Historical state-action pairs reveal the current dynamics of the environment

Learning the Dynamics Model from Existing Synthesized Demonstrations

- The simulation data generation has three sources of diversity:
  - Dynamics
  - Scenario
  - Controller
- Curriculum data generation:
  - Off-policy data: Pure pursuit controller for steering δ and a PD controller for throttle T , to track randomly synthesized reference track in simulation
  - On-policy data: NN-MPPI controller to track agile trajectories in simulation
  - Real-world data: NN-MPPI controller to track agile trajectories in the real world

Deployment

Phase 3: Deployment

1/16 Scale Car

20 cm

1/10 Scale Car

31 cm

w/ optional 3D-Printed Wheels

State Estimation Method

(a) MoCap

(b) ZED-VIO

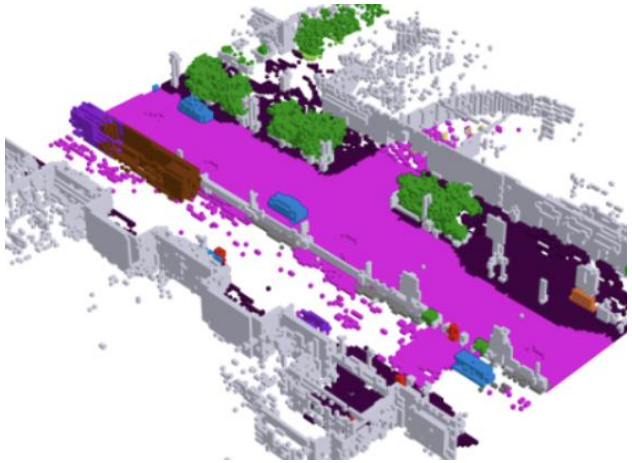
(c) LiDAR-SLAM

Adaptive Model-Based RL is Not Solved

- What if the state-action dynamic changes suddenly/accidentally (e.g. splitting water on the ground)?
- How about contact-rich manipulation?

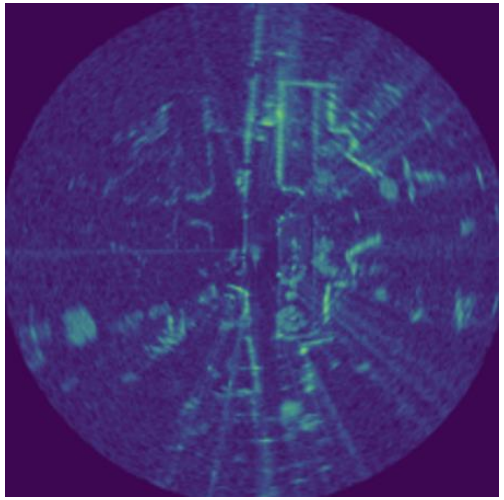
Usually We Have High-dimensional Sensory Input
What is the State Representations of the Dynamics Models?

Occupancy



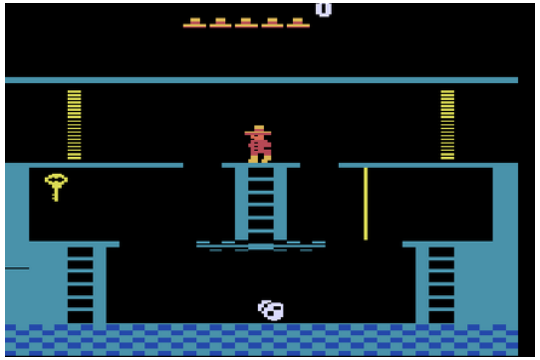
OccWorld: Learning a 3D Occupancy World Model for Autonomous Driving. Zhen et al.

Radar



CramNet: Camera-Radar Fusion with Ray-Constrained Cross-Attention for Robust 3D Object Detection. Hwang et al.

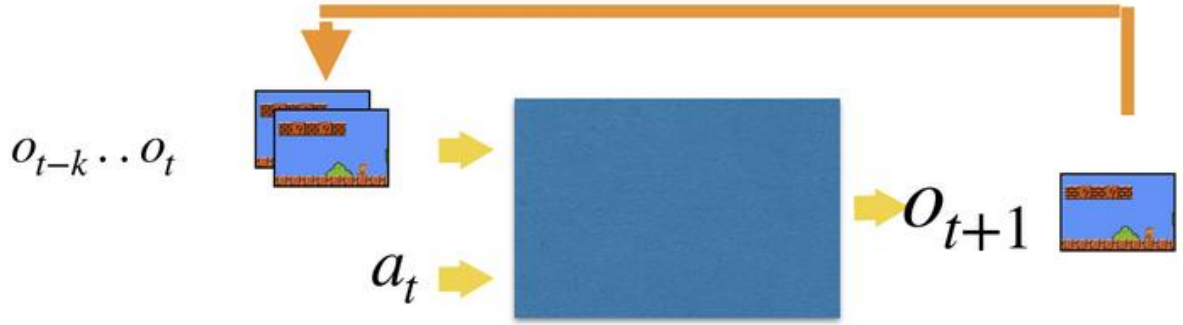
Image



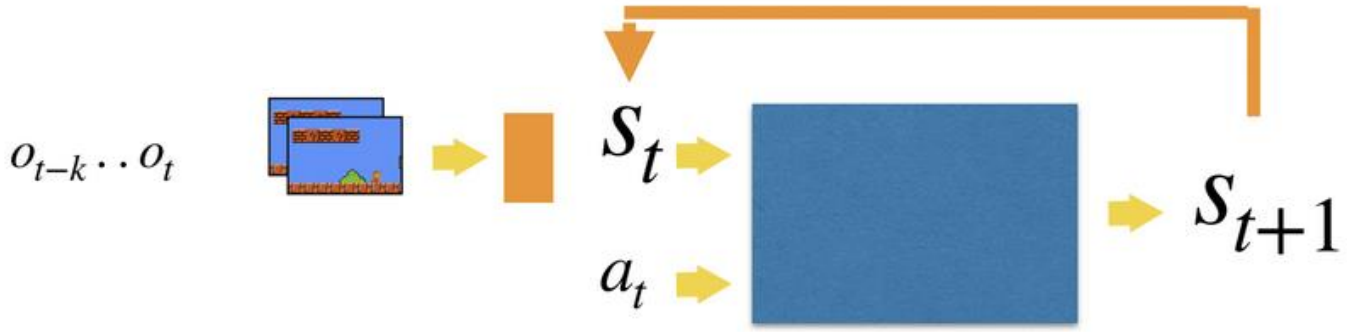
...

Usually We Have High-dimensional Sensory Input
What is the State Representations of the Dynamics Models?

Unrolling in the observation space: the model is trained to predict future observations.

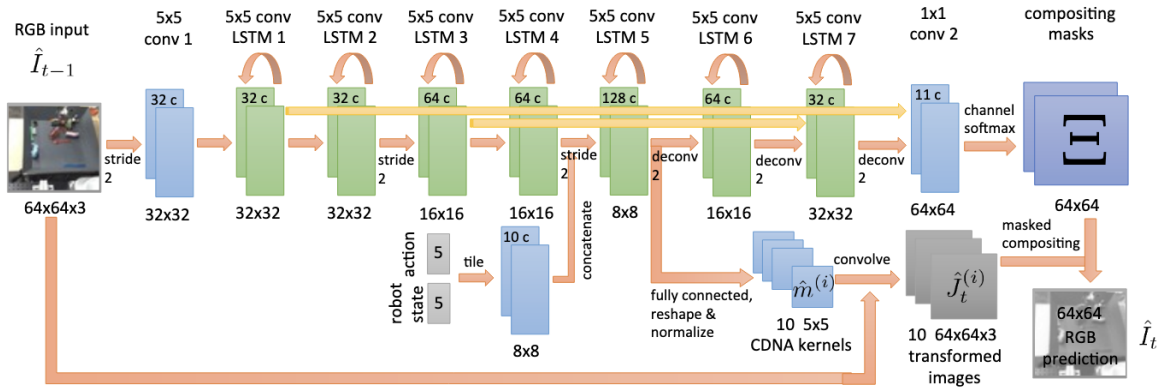


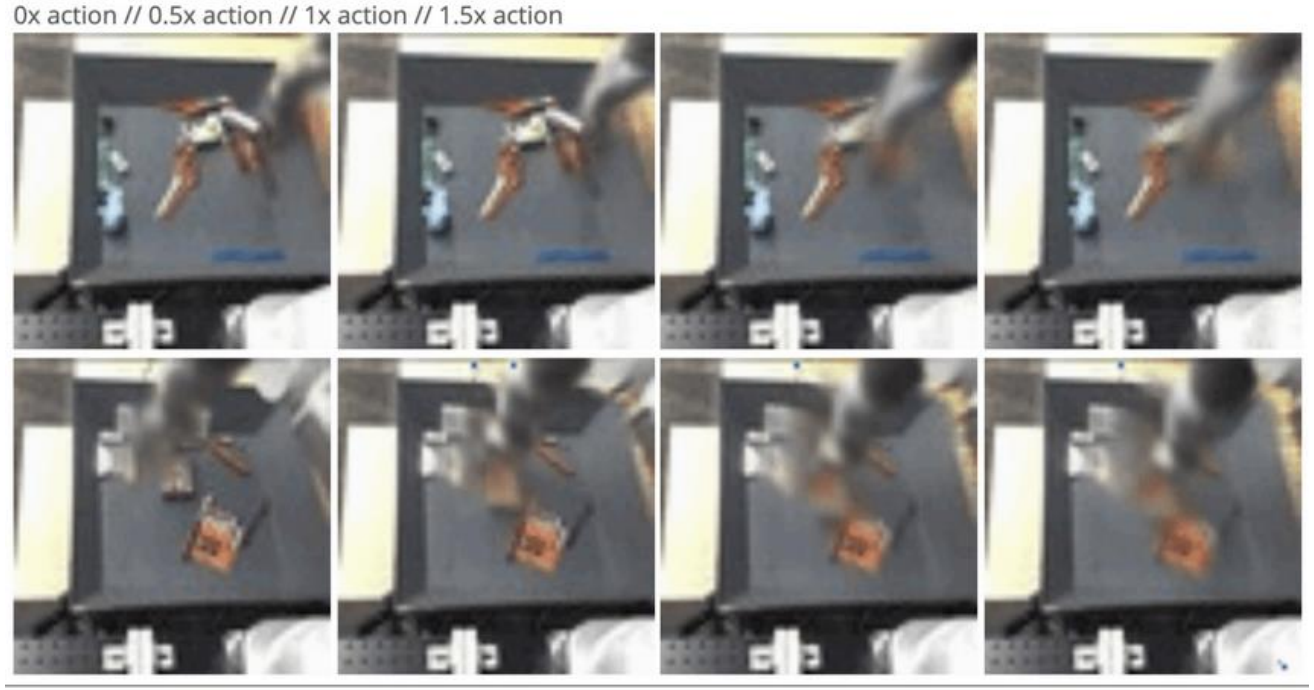
Unrolling in the latent space.



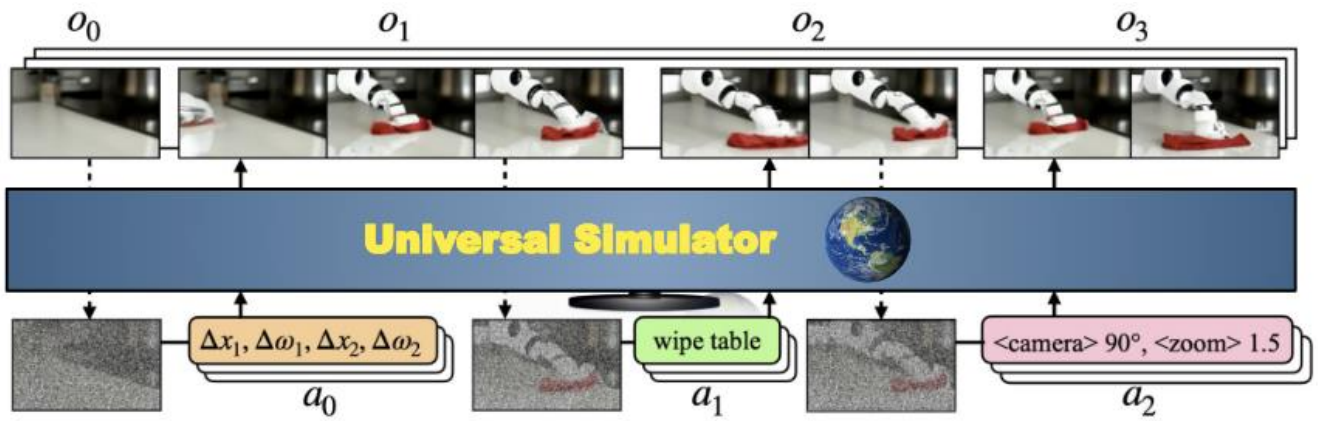
Slide credit K. Fragkiadaki

Unrolling in the Observation Space





Unrolling in the Observation Space



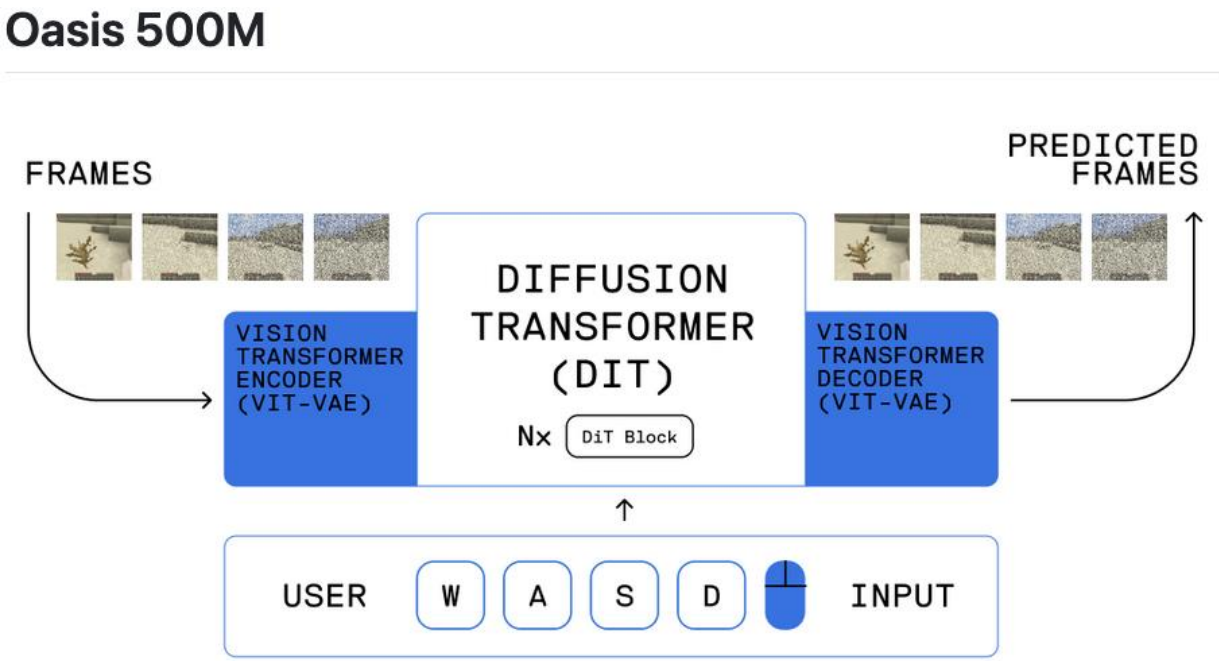
Simulated rollout from $\Delta x, \Delta y$ moving left, right, down, up

Simulated rollout from $\Delta x, \Delta y$ moving diagonally

Real-robot execution of "move blue cube to green circle"

Learning Interactive Real-world Simulator. Yang et al.

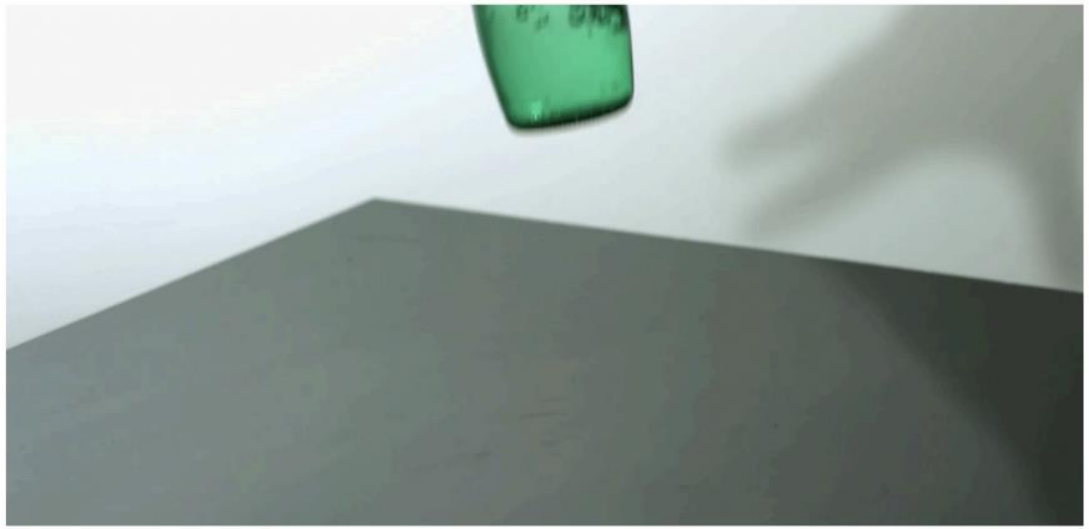
Unrolling in the Observation Space



https://oasis-model.github.io/

What are the Problems?

What will happen on dropping the bottle?

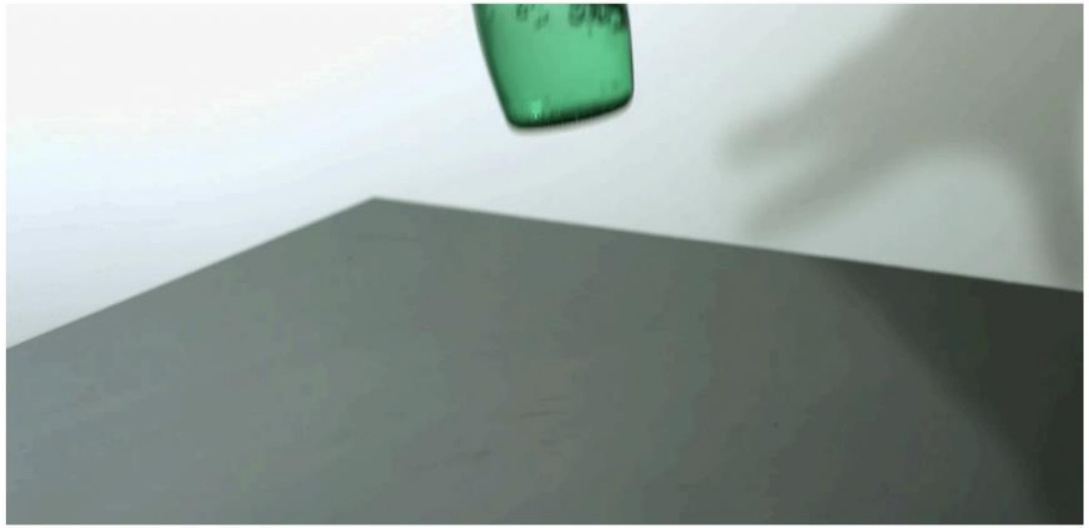


Simulating the water splash and breaking bottle is hard...
Do we need to model them?

Slide credit P. Agrawal

What are the Problems?

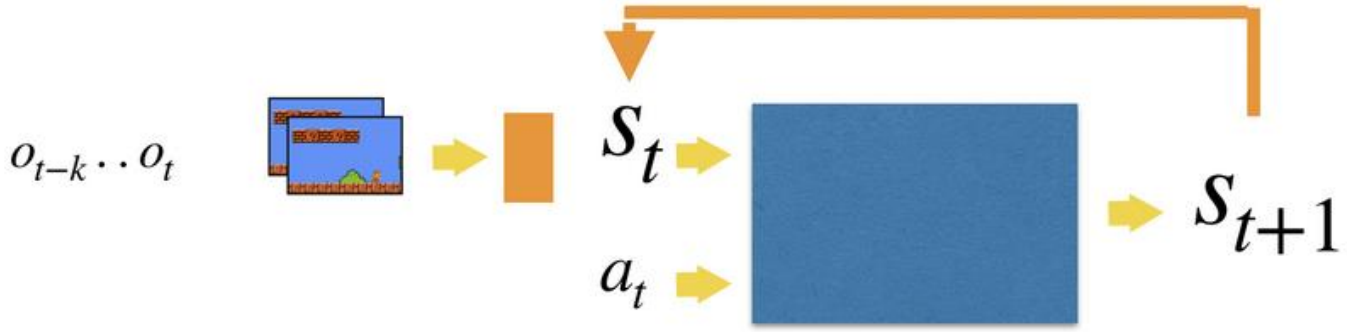
What will happen on dropping the bottle?

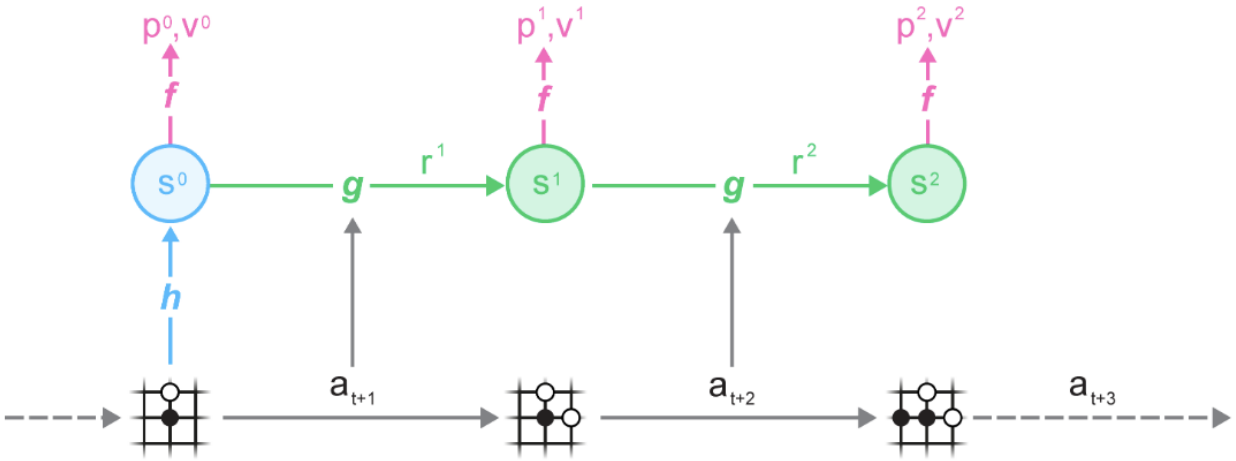


Simulating the water splash and breaking bottle is hard...
Do we need to model them?

Simulating every pixel takes time, can we afford to do test-time searching?

Unroll in the Latent Space





Remember in MuZero:

- The state representations s have no semantics of the environment, but encodes information of policies, values and rewards
- Plan in the latent space

Slide credit K. Fragkiadaki

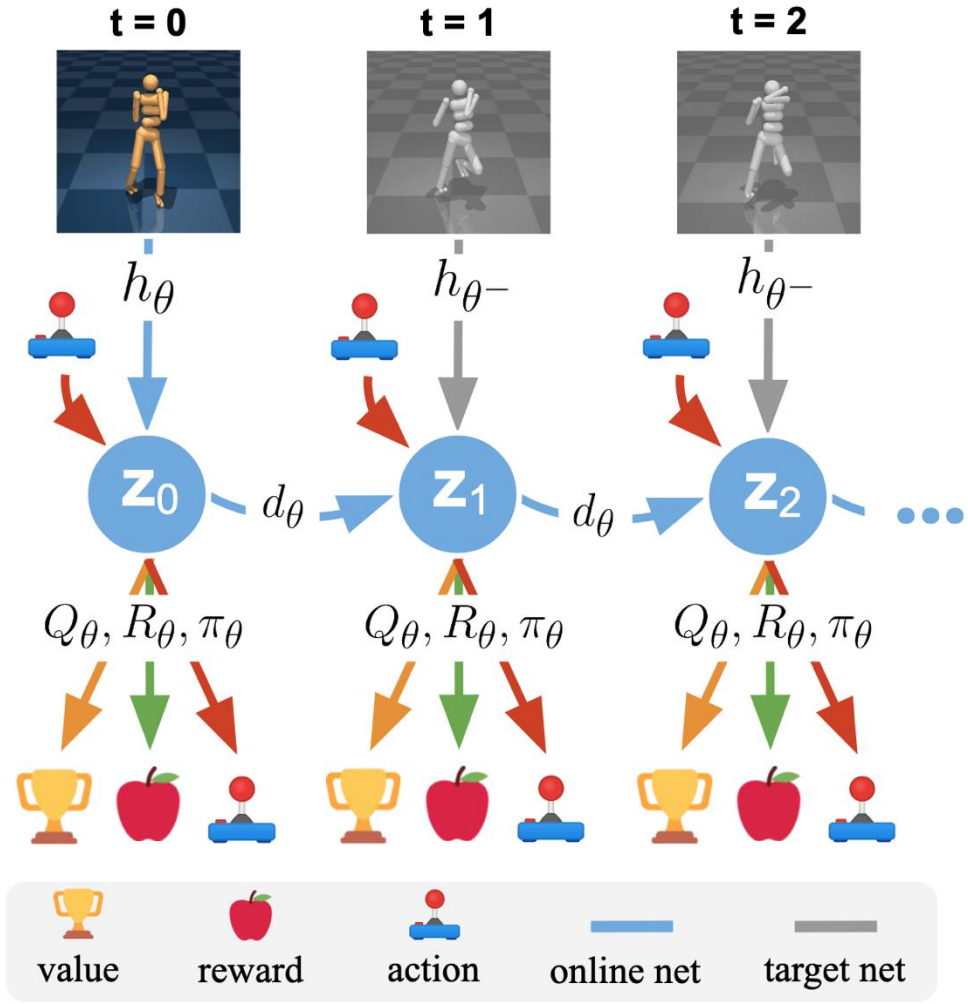
Task-Relevant Latent Dynamic Models

- Model-based RL typically models everything in the environment
- Model-free RL only retains information predictive of reward

- Idea:
  - Infer a latent space under all task-relevant losses: action prediction, reward prediction, Q value prediction
  - Learn jointly the dynamics model and the value estimation function

Slide credit K. Fragkiadaki

Task-Relevant Latent Dynamic Models



Representation:	$\mathbf{z}_t = h_\theta(\mathbf{s}_t)$
Latent dynamics:	$\mathbf{z}_{t+1} = d_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Reward:	$\hat{r}_t = R_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Value:	$\hat{q}_t = Q_\theta(\mathbf{z}_t, \mathbf{a}_t)$
Policy:	$\hat{\mathbf{a}}_t \sim \pi_\theta(\mathbf{z}_t)$

TD-Learning for MPC. Hansen et al.

In Vanilla MPC

Algorithm:

1. (Testing) Plan an optimal sequence of actions $a^*_{0:T}$:

$$a^*_{0:T} = \max_{a_{0:T}} r(s_0, a_0) + r(f(s_0, a_0), a_1) + r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

2. Execute the first planned action a^*_0 and observe resulting state s_1

- Major challenges of MPC:
  - MPC optimizes a trajectory over a shorter, finite horizon, which yields only temporally local optimal solutions
  - Long-horizon planning has compounding errors of dynamics models and is computationally expensive

In Vanilla MPC

Algorithm:

1. (Testing) Plan an optimal sequence of actions $a^*_{0:T}$:

$$a^*_{0:T} = \max_{a_{0:T}} r(s_0, a_0) + r(f(s_0, a_0), a_1) + r(f(f(s_0, a_0), a_1), a_2) + \cdots$$

2. Execute the first planned action a^*_0 and observe resulting state s_1

- Major challenges of MPC:
  - MPC optimizes a trajectory over a shorter, finite horizon, which yields only temporally local optimal solutions
  - Long-horizon planning has compounding errors of dynamics models and is computationally expensive

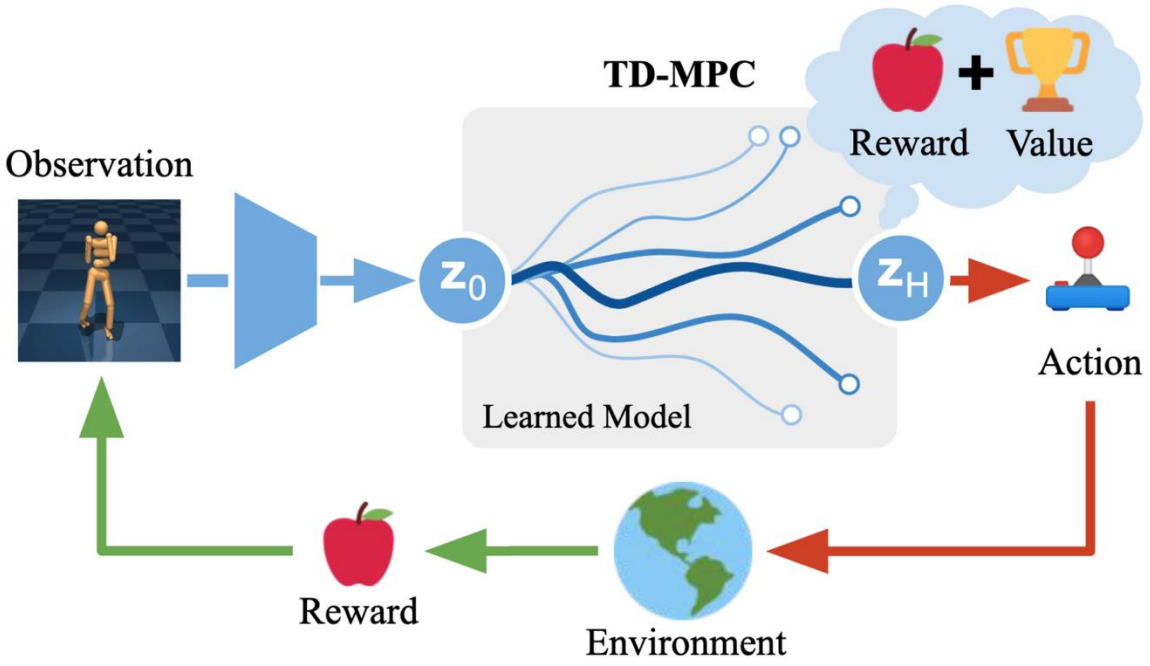
Instead of unrolling infinite-horizon trajectories, use a value function to approximate the globally optimal solution!

TD Learning for MPC

Inference *(planning)*

- Planning in latent space
- Return estimate:

$$\mathbb{E}_\Gamma \Big[\underbrace{\gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)}_{\text{Value}} + \underbrace{\sum_{t=0}^{H-1} \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)}_{\text{Rewards}} \Big]$$



Model Predictive Path Integral Control for Planning

Algorithm 1 TD-MPC *(inference)*

Require: θ : learned network parameters
μ^0, σ^0: initial parameters for $\mathcal{N}$
N, N_π: num sample/policy trajectories
$\mathbf{s}_t, H$: current state, rollout horizon

Encode state $\mathbf{z}_t \leftarrow h_\theta(\mathbf{s}_t)$ ◁ *Assuming TOLD model*
for each iteration $j = 1..J$ **do**
  Sample N traj. of len. H from $\mathcal{N}(\mu^{j-1}, (\sigma^{j-1})^2\mathrm{I})$
  Sample N_π traj. of length H using π_θ, d_θ
  // Estimate trajectory returns ϕ_Γ using $d_\theta, R_\theta, Q_\theta$, starting from $\mathbf{z}_t$ and initially letting $\phi_\Gamma = 0$:
  for all $N + N_\pi$ trajectories $(\mathbf{a}_t, \mathbf{a}_{t+1}, \ldots, \mathbf{a}_{t+H})$ **do**
    for step $t = 0..H-1$ **do**
      $\phi_\Gamma = \phi_\Gamma + \gamma^t R_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Reward*
      $\mathbf{z}_{t+1} \leftarrow d_\theta(\mathbf{z}_t, \mathbf{a}_t)$ ◁ *Latent transition*
    $\phi_\Gamma = \phi_\Gamma + \gamma^H Q_\theta(\mathbf{z}_H, \mathbf{a}_H)$ ◁ *Terminal value*
  // Update parameters μ, σ for next iteration:
  $\mu^j, \sigma^j =$ Equation 4 (and Equation 5)
return $\mathbf{a} \sim \mathcal{N}(\mu^J, (\sigma^J)^2\mathrm{I})$

TD-Learning for MPC. Hansen et al.

Slide credit K. Fragkiadaki

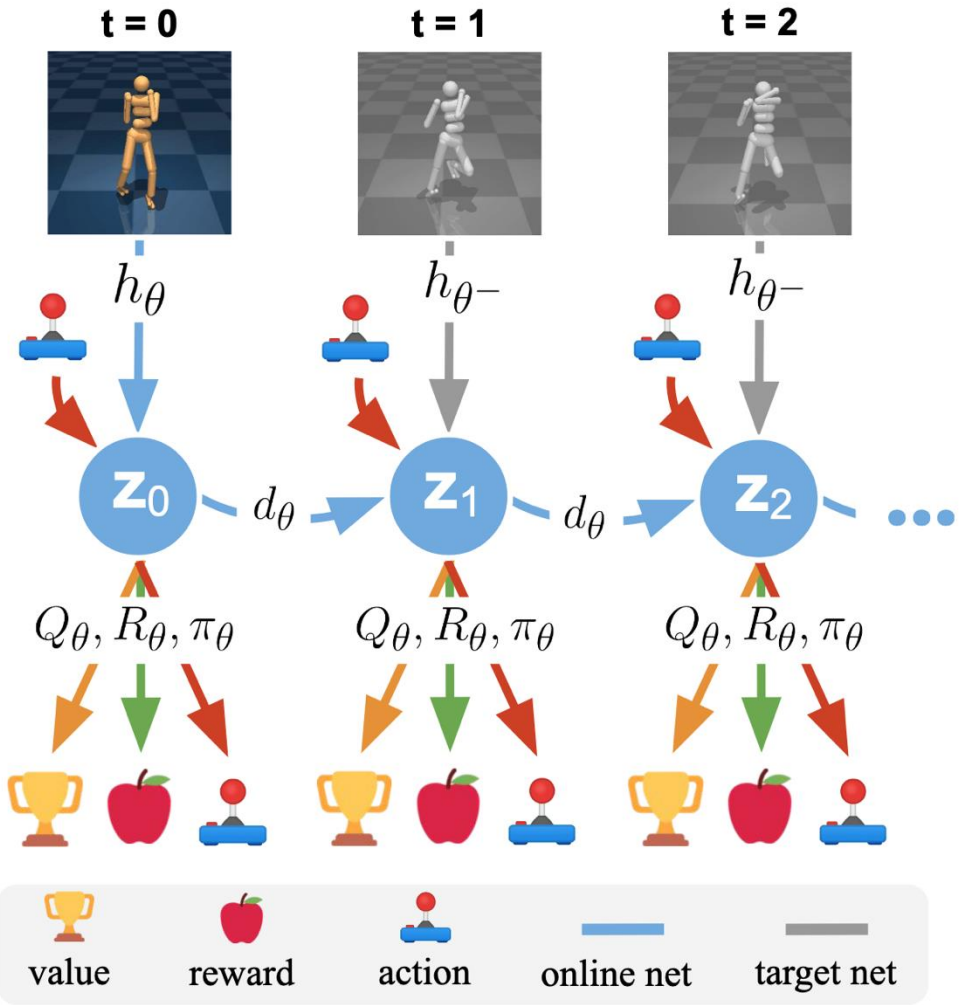
Model Predictive Path Integral Control for Planning

- Update the distribution of trajectories towards high-value samples:

$$\mu^j = \frac{\sum_{i=1}^k \Omega_i \Gamma_i^\star}{\sum_{i=1}^k \Omega_i}, \; \sigma^j = \sqrt{\frac{\sum_{i=1}^k \Omega_i (\Gamma_i^\star - \mu^j)^2}{\sum_{i=1}^k \Omega_i}},$$

$$\Omega_i = e^{\tau(\phi^\star_{\Gamma,i})}$$

Learning Objectives



TOLD minimizes the objective

$$\mathcal{J}(\theta;\Gamma) = \sum_{i=t}^{t+H} \lambda^{i-t} \mathcal{L}(\theta;\Gamma_i)\,, \tag{7}$$

where

$$\mathcal{L}(\theta;\Gamma_i) = c_1 \underbrace{\|R_\theta(\mathbf{z}_i,\mathbf{a}_i) - r_i\|_2^2}_{\text{reward}} \tag{8}$$

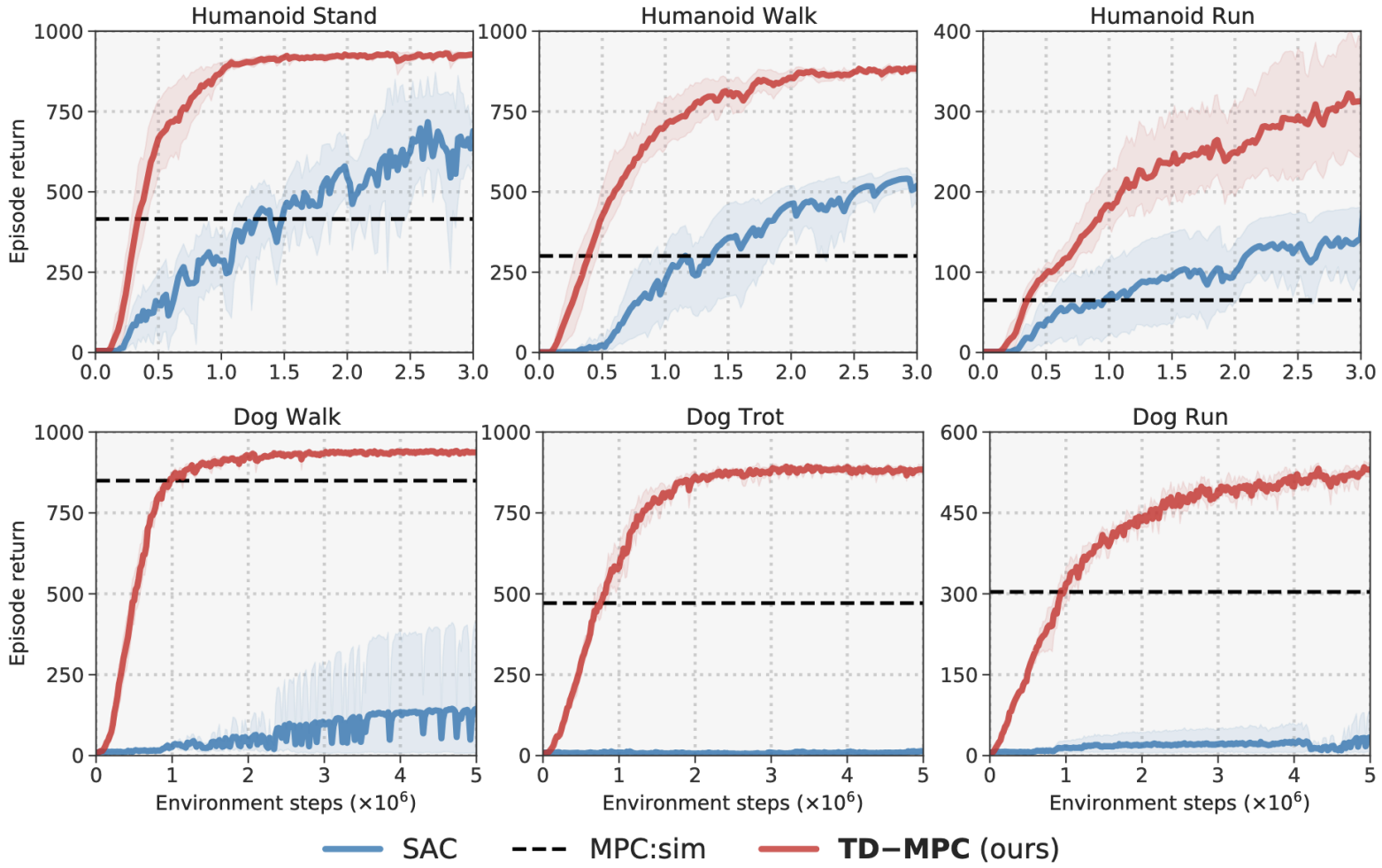
$$+ c_2 \underbrace{\|Q_\theta(\mathbf{z}_i,\mathbf{a}_i) - (r_i + \gamma Q_{\theta^-}(\mathbf{z}_{i+1}, \pi_\theta(\mathbf{z}_{i+1})))\,\|_2^2}_{\text{value}} \tag{9}$$

$$+ c_3 \underbrace{\|d_\theta(\mathbf{z}_i,\mathbf{a}_i) - h_{\theta^-}(\mathbf{s}_{i+1})\|_2^2}_{\text{latent state consistency}} \tag{10}$$

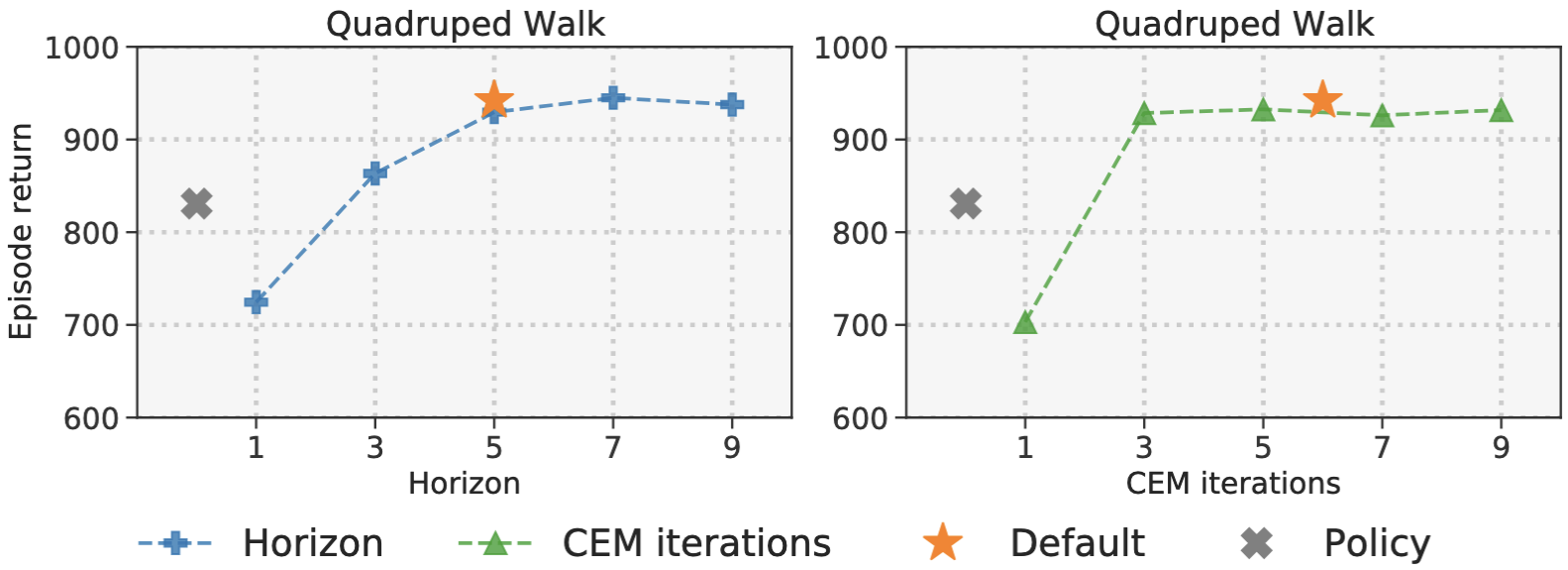
and the policy minimizes

$$\mathcal{J}_\pi(\theta;\Gamma) = -\sum_{i=t}^{t+H} \lambda^{i-t} Q_\theta(\mathbf{z}_i, \pi_\theta(\text{sg}(\mathbf{z}_i)))\,, \tag{11}$$

TD-MPC is More Sample Efficient



Likewise, Planning Helps



A Quick Summary of Model-Based RL

- Goodness of MBRL:
  - MBRL is more sample efficient
  - MBRL enables planning in simulation
  - MBRL allows evaluation of an action
  - MBRL re-uses interaction episodes, even those with low rewards
  - With careful architectural design, MBRL can adapt to non-stationary dynamics

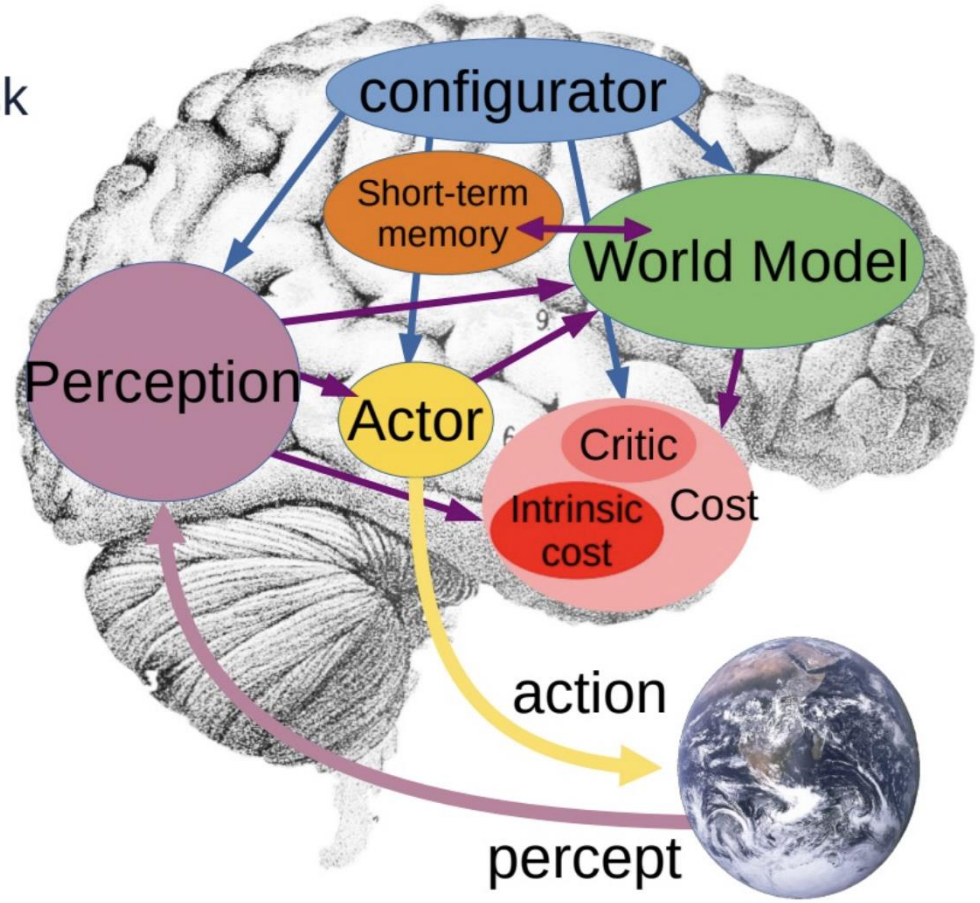
- Challenges of MBRL:
  - Compounding errors of long-horizon rollouts
  - Inaccurate modeling of dynamics due to domain shift, overfitting, underfitting…
  - What is the right state representations for the dynamics models?

Learning the Dynamics Models Enable Us to Predict the Outcome of Our Actions

Modular Architecture for Autonomous AI

Y. LeCun

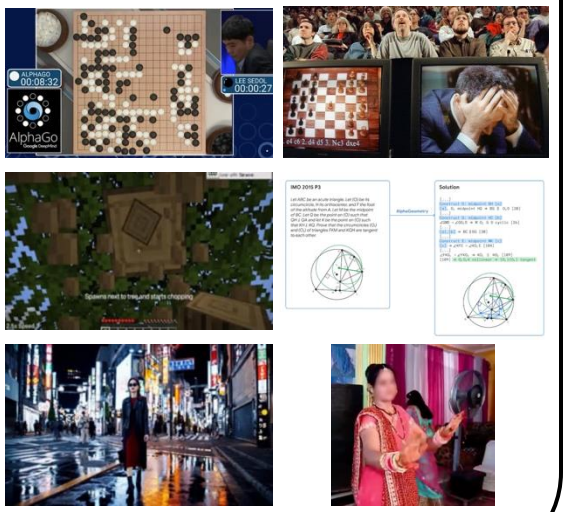
- **Configurator**
  - Configures other modules for task
- **Perception**
  - Estimates state of the world
- **World Model**
  - Predicts future world states
- **Cost**
  - Compute “discomfort”
- **Actor**
  - Find optimal action sequences
- **Short-Term Memory**
  - Stores state-cost episodes



Remember Moravec's Paradox?

What makes the difference?
The key is that we are in a physical world

Easy universe



Hard universe

"*We are all prodigious Olympians in perceptual and motor areas, so good that we make the difficult look easy. Abstract thought, though, is a new trick, perhaps less than 100 thousand years old. We have not yet mastered it. It is not all that intrinsically difficult; it just seems so when we do it*"

Hans Moravec

"*The main lessons of thirty-five years of AI research is that the hard problems are easy and the easy problems are hard. The mental abilities of a four-year-old that we take for granted-recognizing a face, lifting a pencil, walking across a room, answering a question-in fact solve some of the hardest engineering problems ever conceived.*"

Steven Pinker

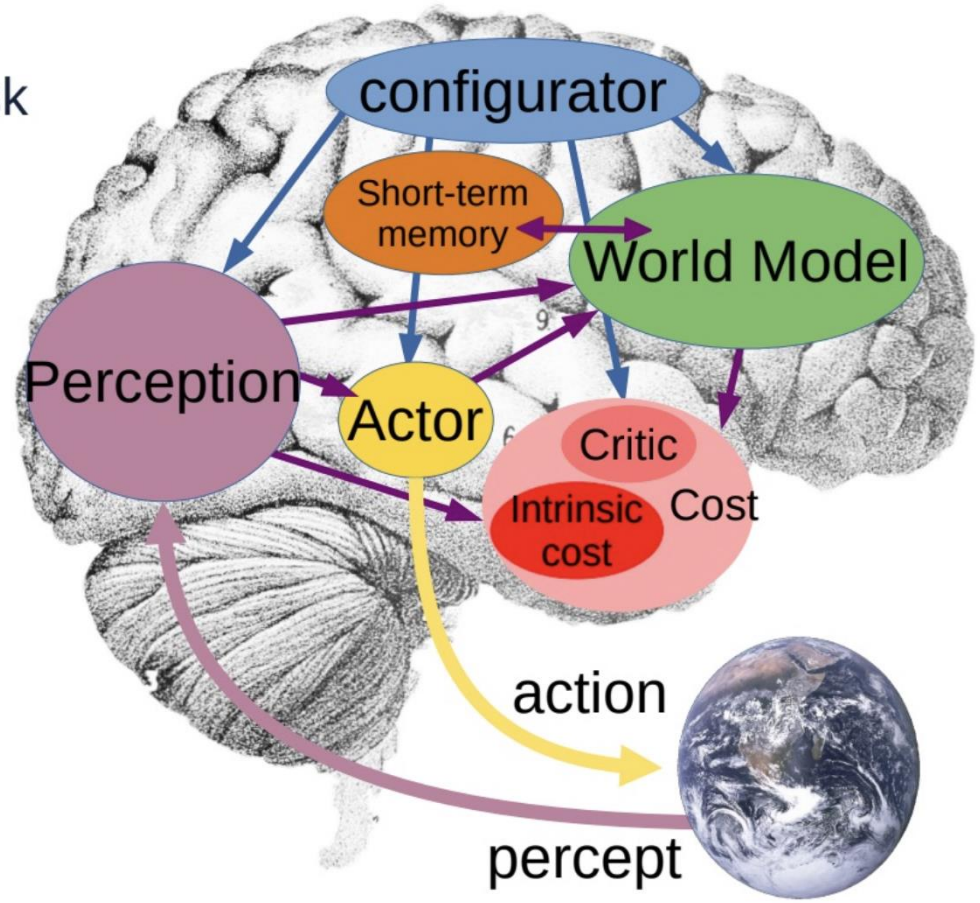
Slide adapted from S. Levine

Is world model the next frontier in our path to AGI?

Modular Architecture for Autonomous AI

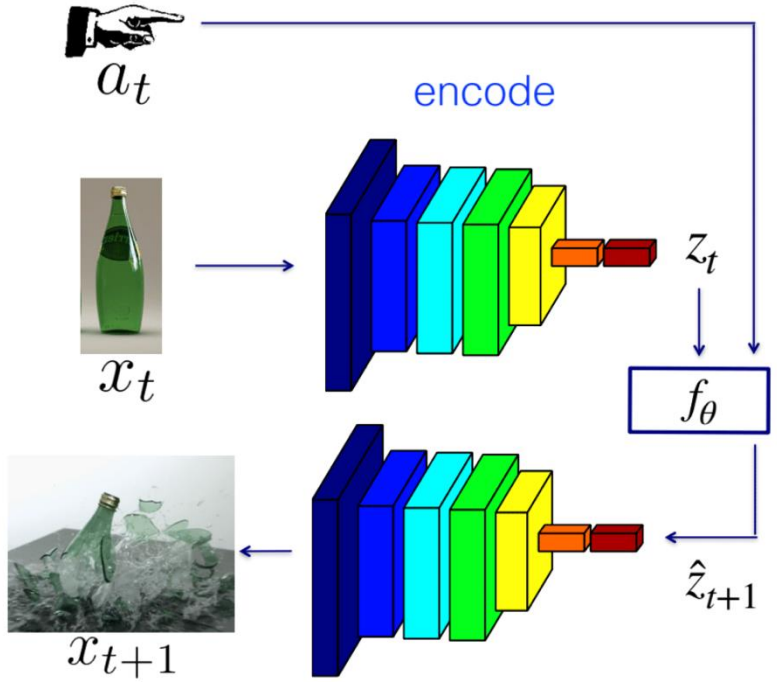
Y. LeCun

- **Configurator**
  - Configures other modules for task
- **Perception**
  - Estimates state of the world
- **World Model**
  - Predicts future world states
- **Cost**
  - Compute “discomfort”
- **Actor**
  - Find optimal action sequences
- **Short-Term Memory**
  - Stores state-cost episodes

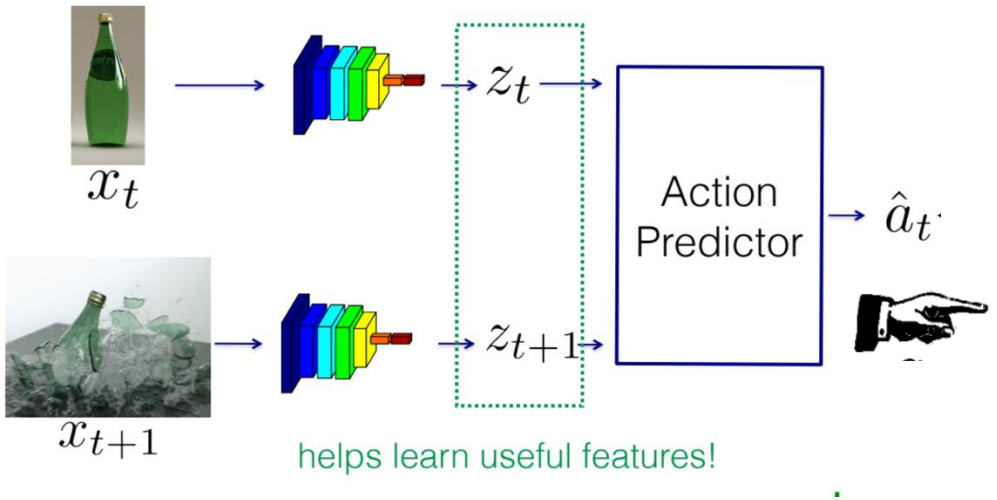


An Alternative to Model-Based RL: Sub-goal Prediction + Inverse Models

Forward Dynamic Models

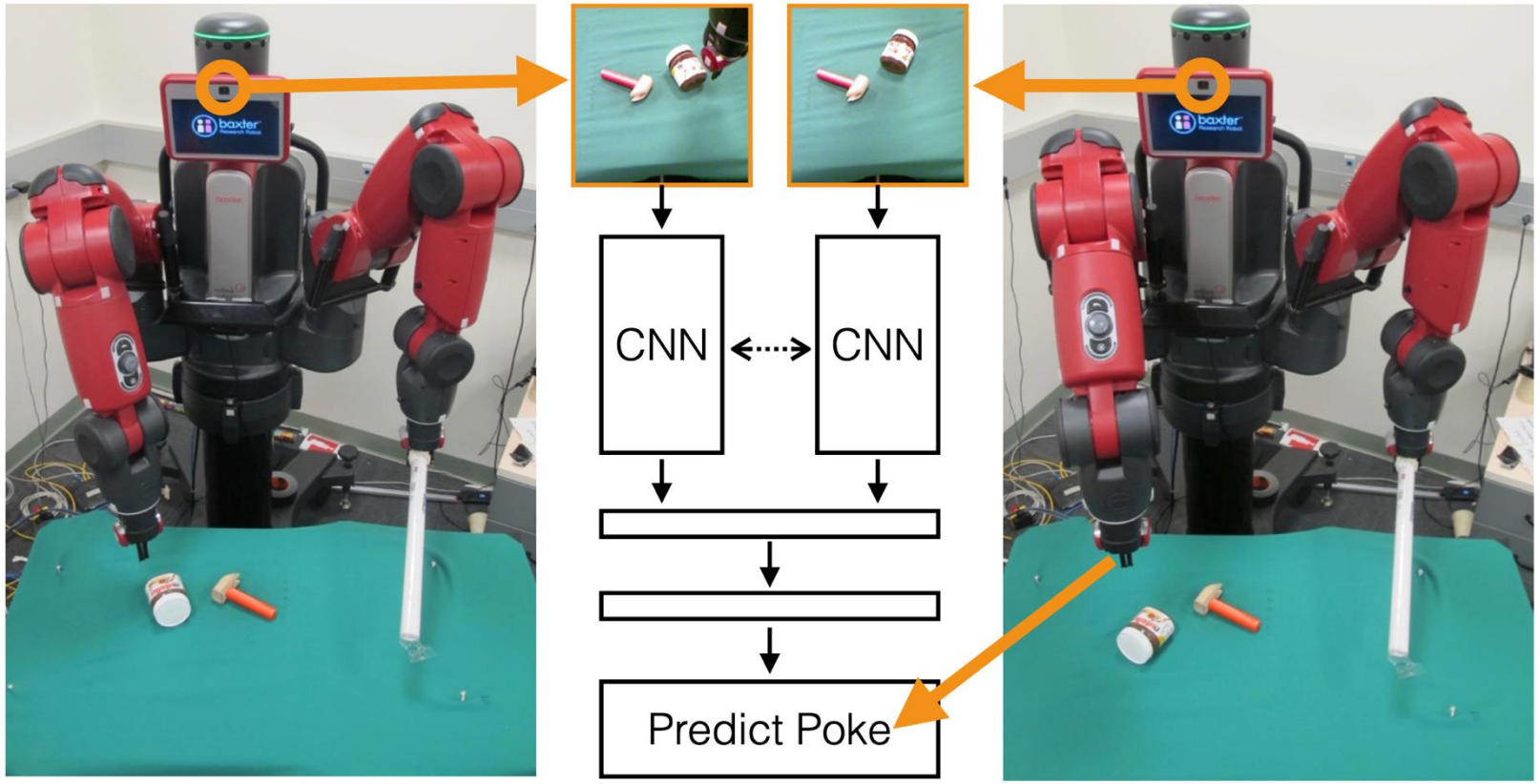


Inverse Models with Subgoals



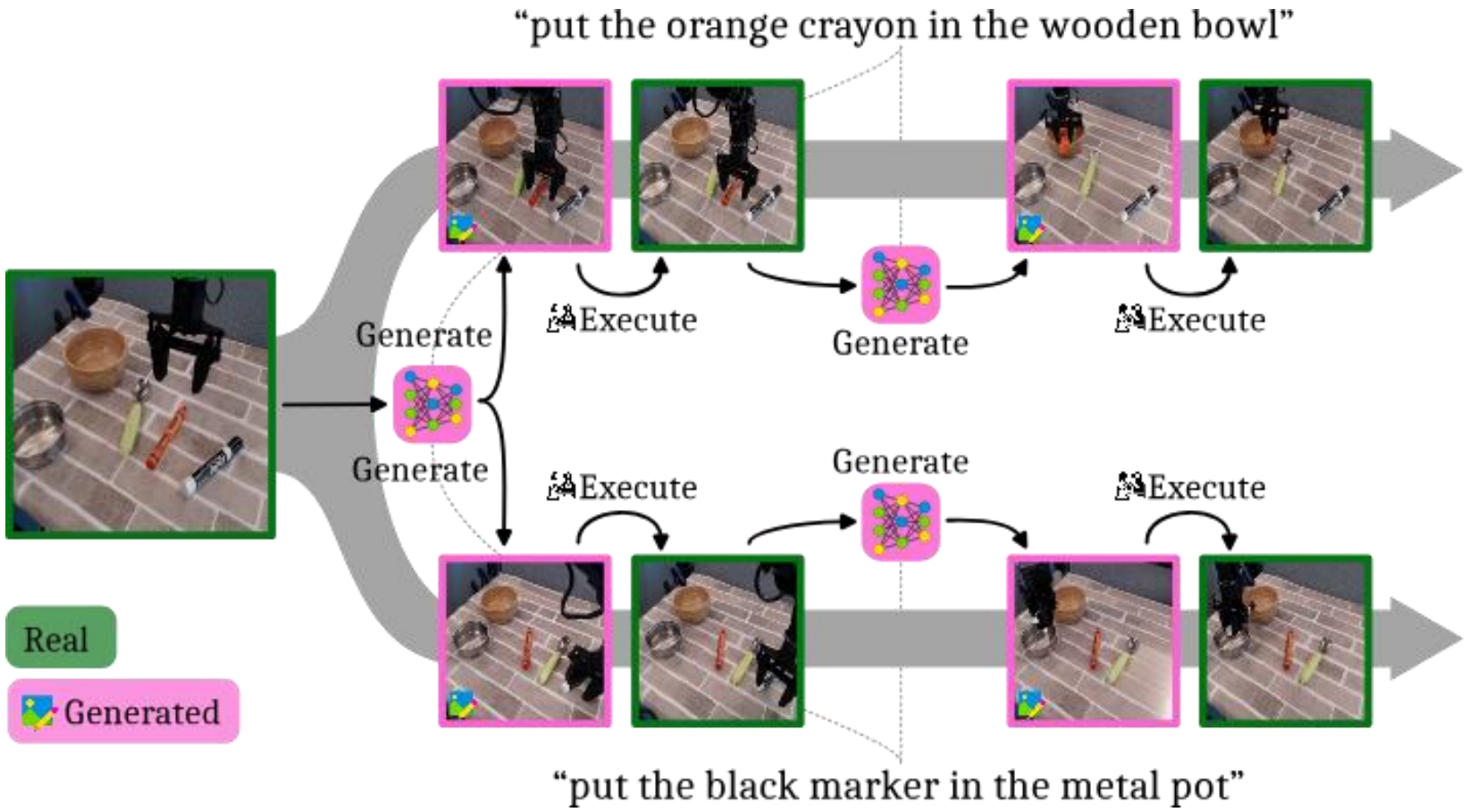
Slide credit P. Agrawal

An Alternative to Model-Based RL: Sub-goal Prediction + Inverse Models



Learning to Poke by Poking: Experiential Learning of Intuitive Physics. Agrawal et al.

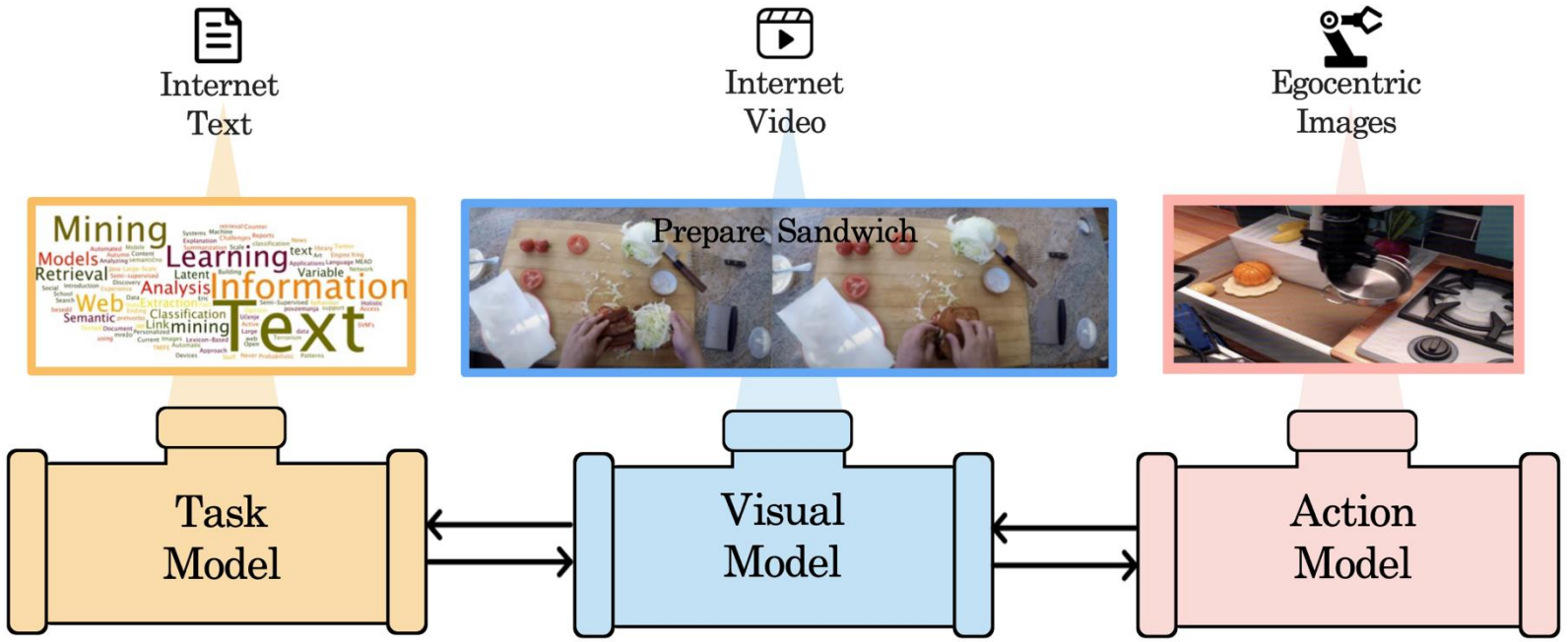
SuSIE: Predict Subgoals with an Image Generative Model, and Actions with a Goal-conditioned Policy



SuSIE: Predict Subgoals with an Image Generative Model, and Actions with a Goal-conditioned Policy

SuSIE: Subgoal Synthesis via Image Editing. Black et al.

An Alternative to Model-Based RL: LLM + Sub-goal Prediction + Inverse Models



An Alternative to Model-Based RL: LLM + Sub-goal Prediction + Inverse Models