

Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

Model Based Reinforcement Learning IV

Fall 2020, CMU 10-703

Katerina Fragkiadaki



MBRL

- **Sample experience** from the *learnt* model. Then use a model-free method to learn policies at training time. At test time use only the learnt policy. (*MBPO[1]*, *Model-ensemble TRPO[2]*, *MBPOforAtari[10]*).
- **Sample experience** from the known environment model. Then use a model-free method to learn policies at training time. At test time use only the learnt policy. (All *model-free RL*).
- **Plan through the model:** Unroll the model forward in time and optimize for action sequences to maximize rewards at training time. Then, learn policies that predict the actions found by model unrolling. At test time use only the learnt policy.
 - Differentiable optimization: back propagate the error through the dynamics function over time. (*Linear quadratic regulator(LQR)[3]*: linear models, *dream-to-control[4]*: non-linear models)
 - Evolutionary search for actions, e.g., CMA-ES. (*PETS[5]*, *PlaNet[11]*)
 - Monte Carlo tree search: combination of learnt policies and model unrolling. (*Playing Atari with offline MCTS[6]*)
- **Plan through the model:** Unroll the model forward in time and optimize for action sequences to maximize rewards at training time. Then, learn policies that predict the actions found by model unrolling. At test time, use both model unrolling and the learnt policy. (*AlphaGoZero[7]*, *MuZero[8]*)
- **Plan through the model:** Use model rollouts to supply better targets to Q functions (*Stochastic Ensemble Value Expansion[9]*). Learn a policy that maximizes the Q function, and use it at test time.

[1]When to Trust Your Model-Model-Based Policy Optimization (MBPO)

[2]Model-ensemble trust region policy optimization

[4]Dream-to-control:learning behaviors via latent imagination

[5]Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models (PETS)

[6]Deep Learning for Real-Time Atari Game Play Using Offline MCTS

[7]Mastering the game of Go without human knowledge(AlphaGoZero)

[8]Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model (MuZero)

[9]Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion

[10] Model-based RL for Atari

[11] Learning Latent Dynamics for Planning from Pixels

MBRL

- **Sample experience** from the *learnt* model. Then use a model-free method to learn policies at training time. At test time use only the learnt policy. (*MBPO[1]*, *Model-ensemble TRPO[2]*, *MBPOforAtari[10]*).
- **Sample experience** from the known environment model. Then use a model-free method to learn policies at training time. At test time use only the learnt policy. (All *model-free RL*).
- **Plan through the model:** Unroll the model forward in time and optimize for action sequences to maximize rewards at training time. Then, learn policies that predict the actions found by model unrolling. At test time use only the learnt policy.
 - **Differentiable optimization:** back propagate the error through the dynamics function over time. (**Linear quadratic regulator(LQR)[3]**: linear models, **dream-to-control[4]**: non-linear models)
 - Evolutionary search for actions, e.g., CMA-ES. (*PETS[5]*, *PlaNet[11]*)
 - Monte Carlo tree search: combination of learnt policies and model unrolling. (*Playing Atari with offline MCTS[6]*)
- **Plan through the model:** Unroll the model forward in time and optimize for action sequences to maximize rewards at training time. Then, learn policies that predict the actions found by model unrolling. At test time, use both model unrolling and the learnt policy. (*AlphaGoZero[7]*, *MuZero[8]*)
- **Plan through the model:** Use model rollouts to supply better targets to Q functions (*Stochastic Ensemble Value Expansion[9]*). Learn a policy that maximizes the Q function, and use it at test time.

[1]When to Trust Your Model-Model-Based Policy Optimization (MBPO)

[2]Model-ensemble trust region policy optimization

[4]Dream-to-control:learning behaviors via latent imagination

[5]Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models (PETS)

[6]Deep Learning for Real-Time Atari Game Play Using Offline MCTS

[7]Mastering the game of Go without human knowledge(AlphaGoZero)

[8]Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model (MuZero)

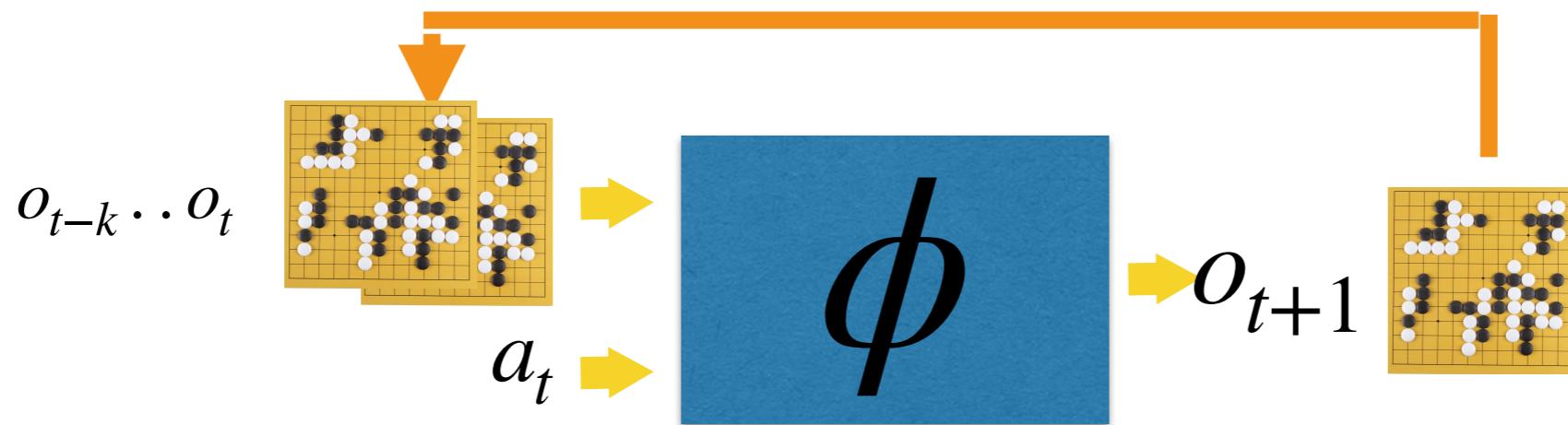
[9]Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion

[10] Model-based RL for Atari

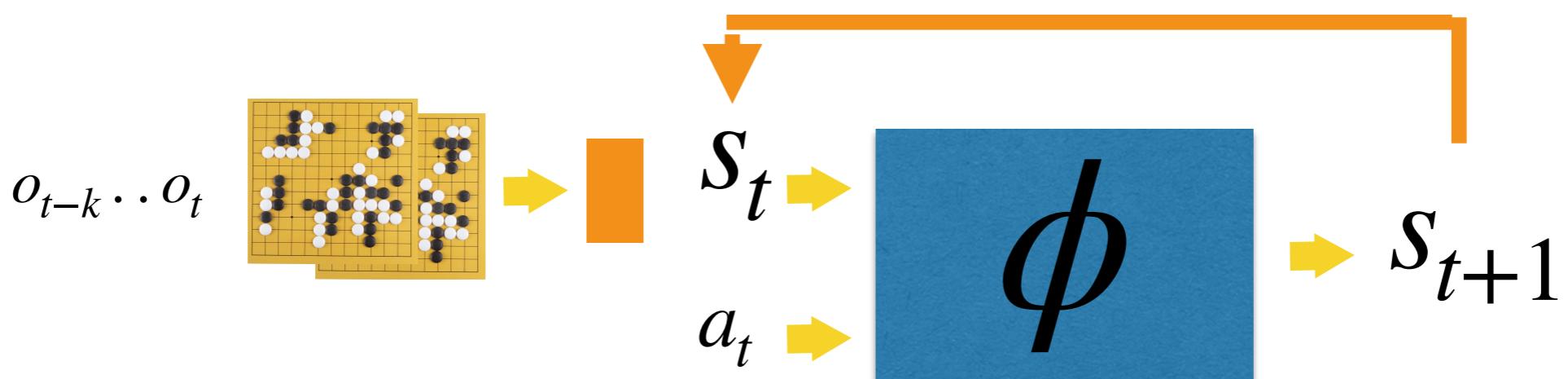
[11] Learning Latent Dynamics for Planning from Pixels

Model learning from sensory input

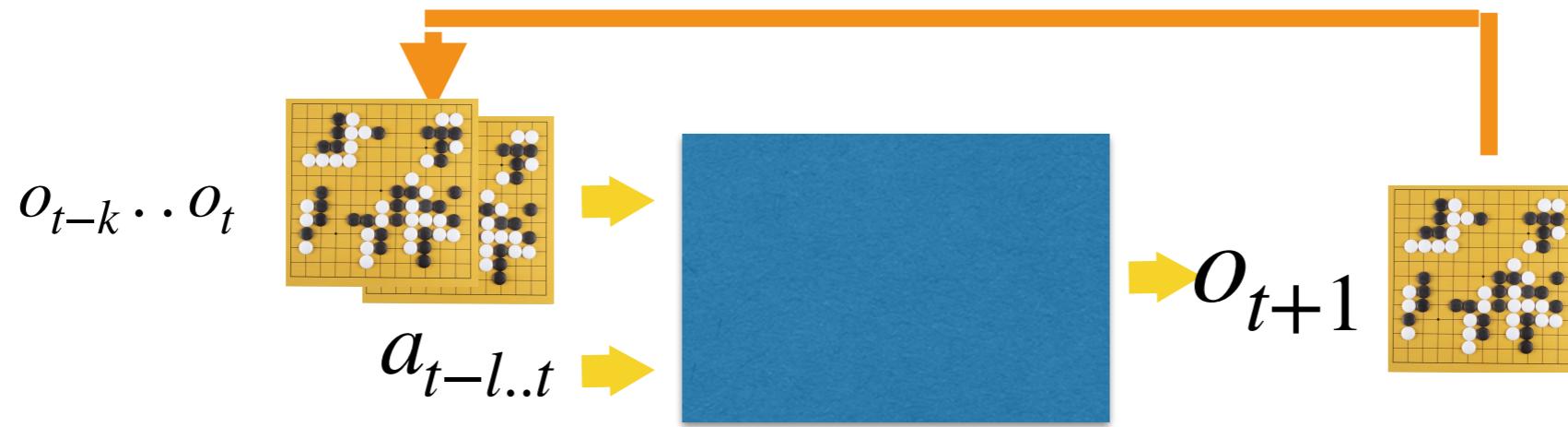
Unrolling in the observation space:



Unrolling in a latent space:



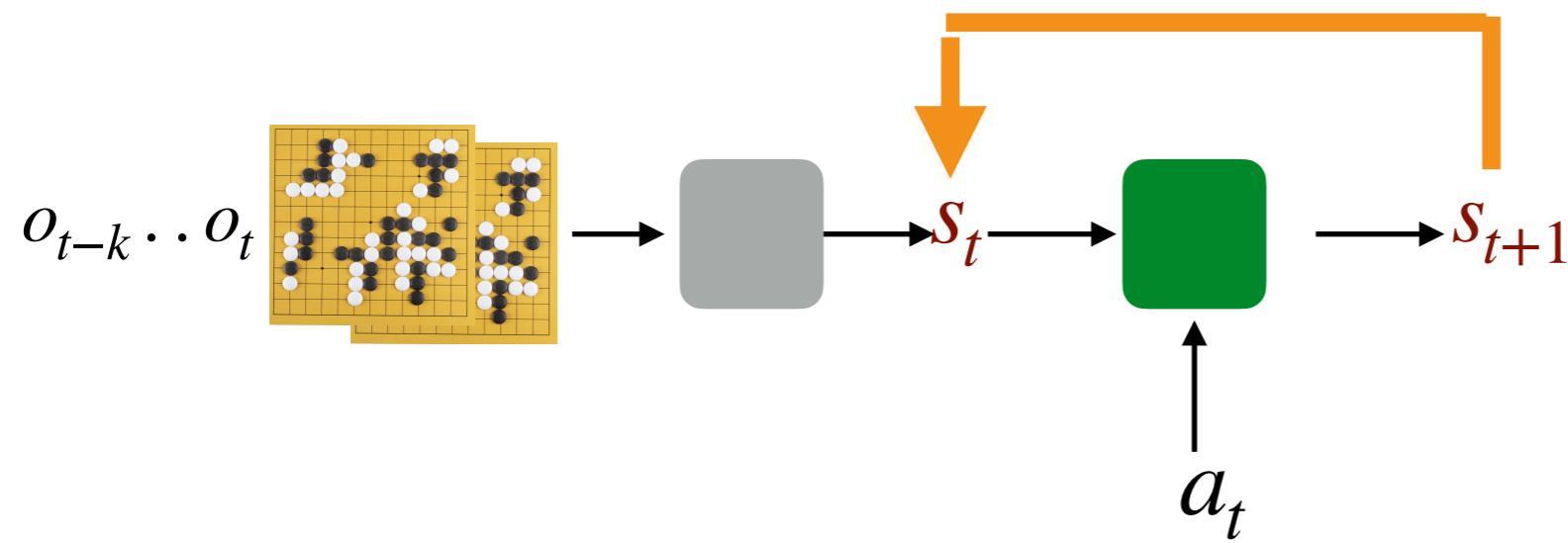
Unrolling in the observation space



$$p(o_{t+1:t+k} | o_{\leq t}, a_{\leq t}) = \prod_{r=t+1}^{t+k} p(o_r | f(o_{<r}, a_{<r}))$$

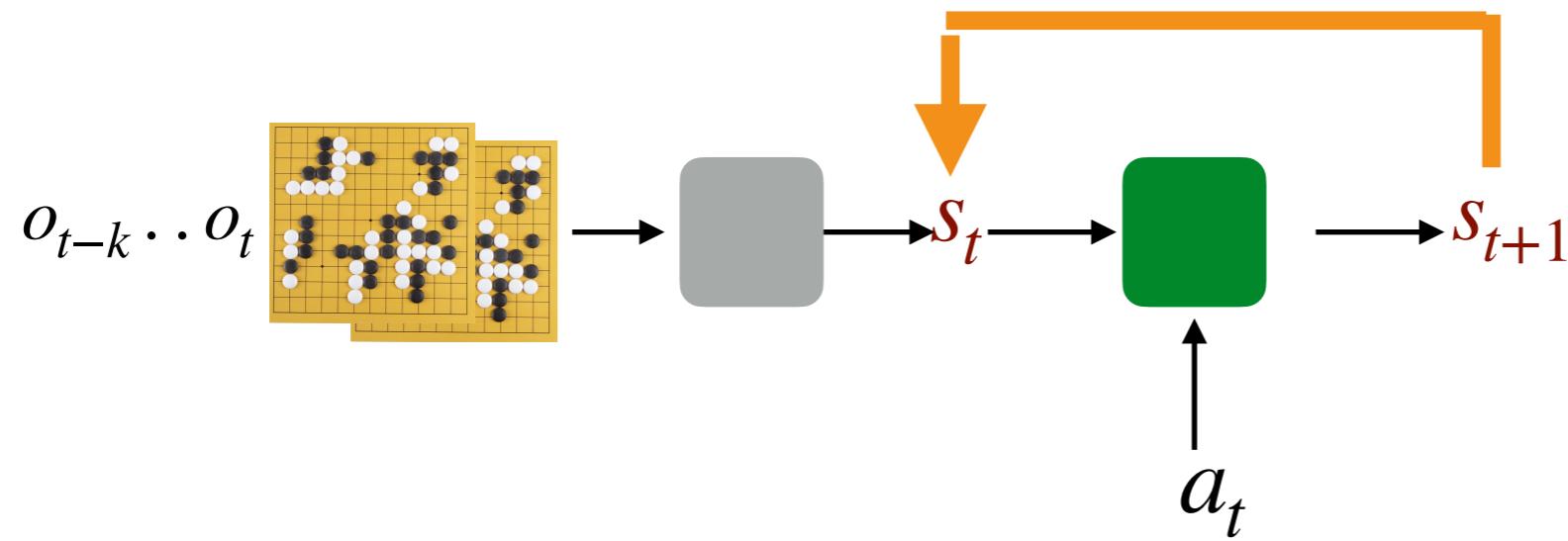
We saw this last time in the action-conditioned video prediction and the model based RL for Atari papers.

Unrolling in a latent state space



Deterministic transition model: $s_{t+1} = g(s_t, a_t)$

Unrolling in a latent state space

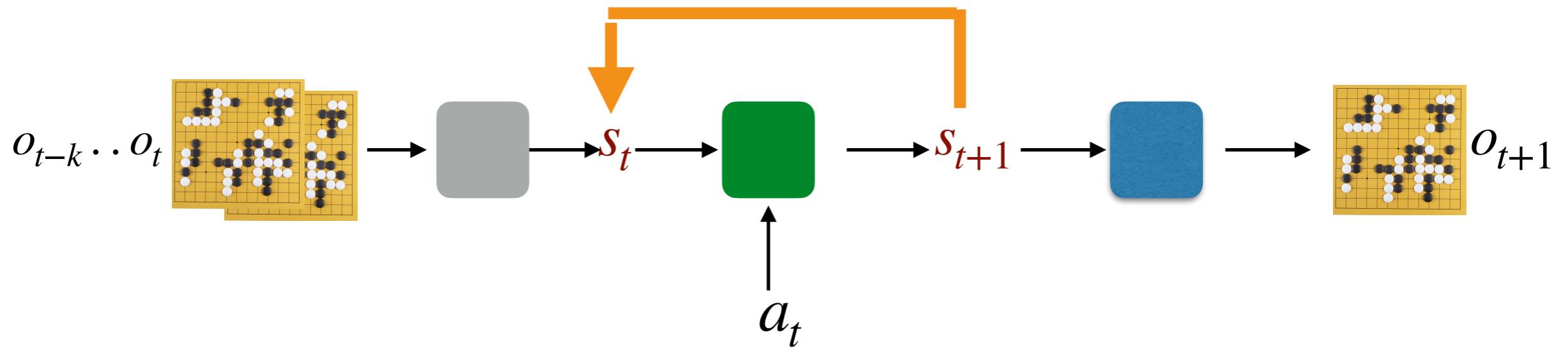


Deterministic transition model: $s_{t+1} = g(s_t, a_t)$

Stochastic transition model: $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$

Unrolling in a latent state space

We can predict rewards and observations from the state embeddings.

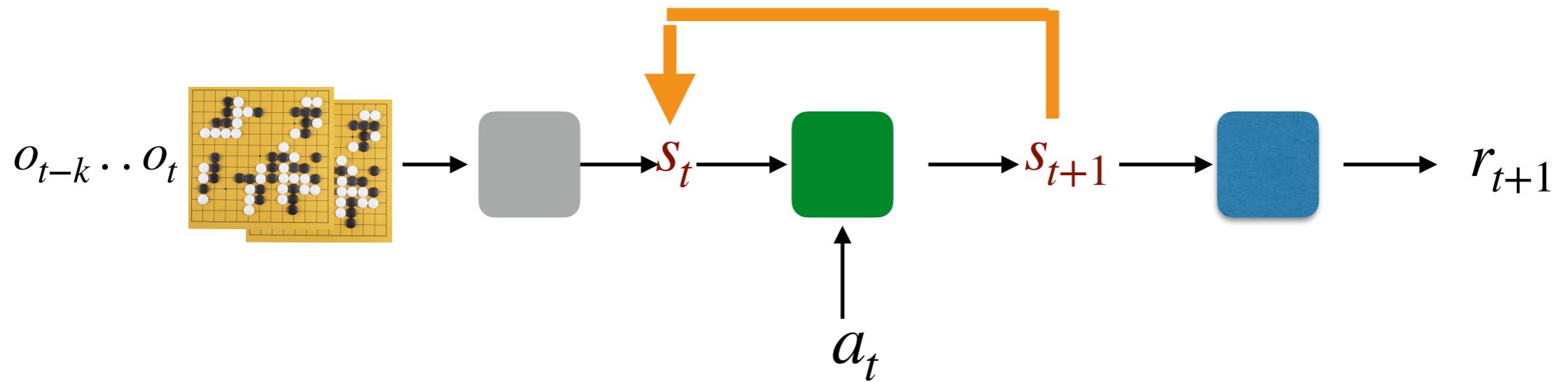


Deterministic transition model: $s_{t+1} = g(s_t, a_t)$

Stochastic transition model: $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$

Unrolling in a latent space

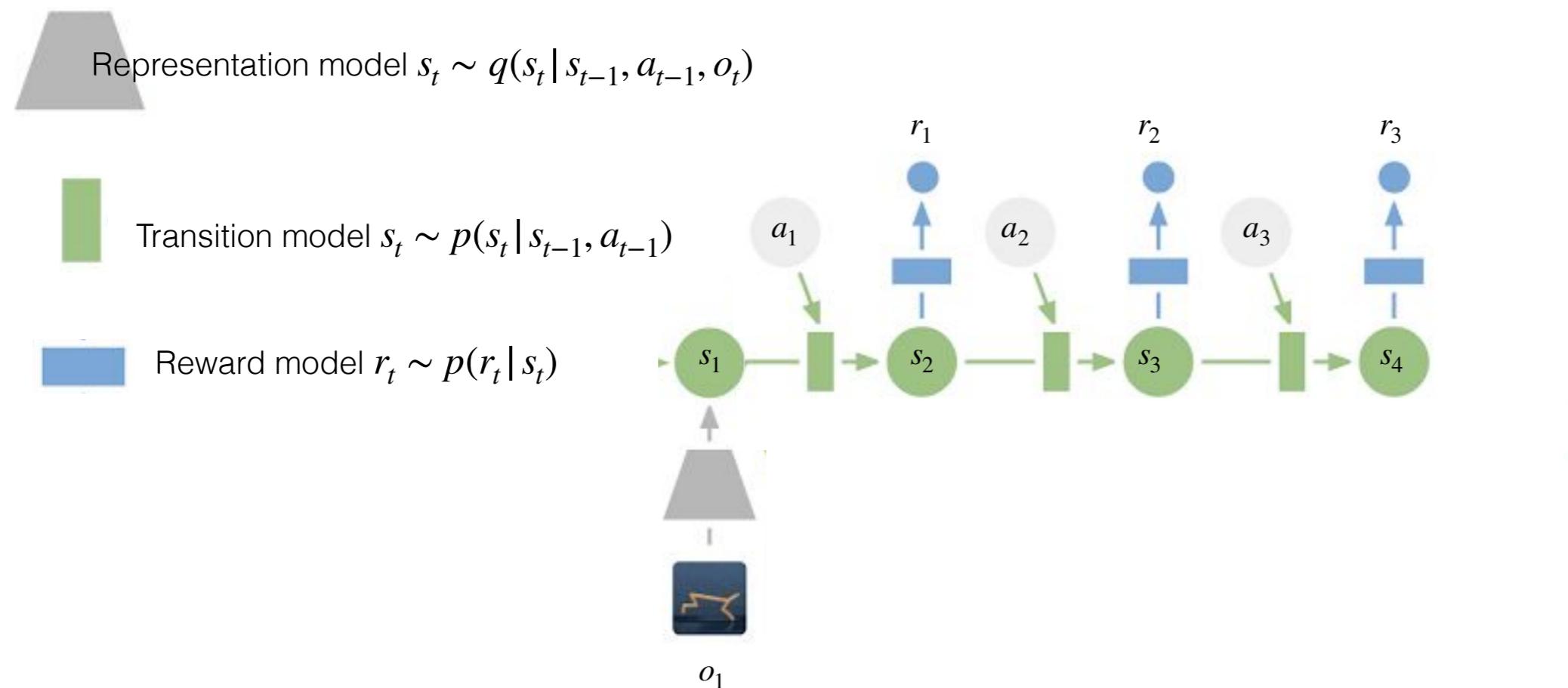
We can predict rewards and observations from the state embeddings.



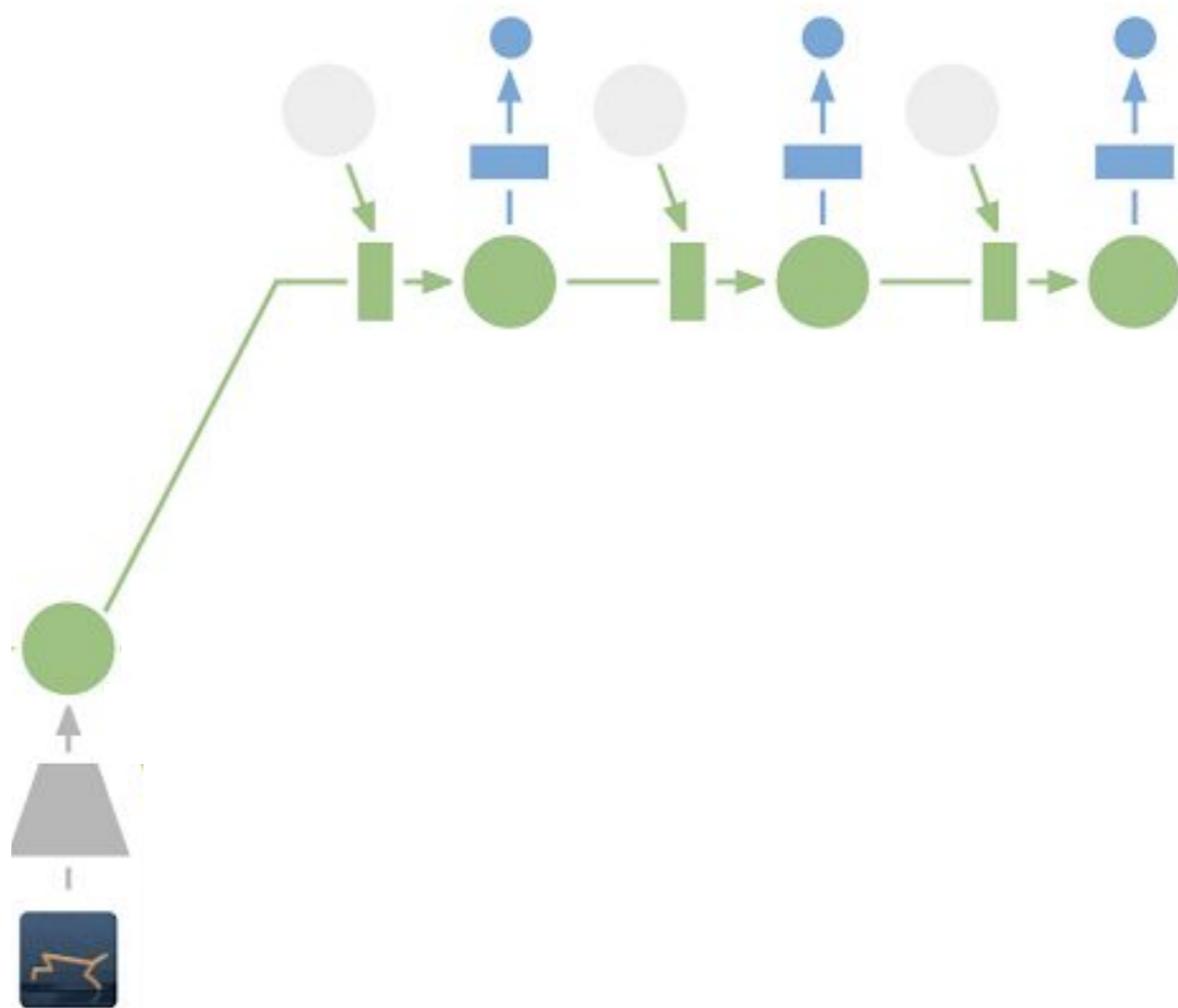
Deterministic transition model: $s_{t+1} = g(s_t, a_t)$

Stochastic transition model: $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$

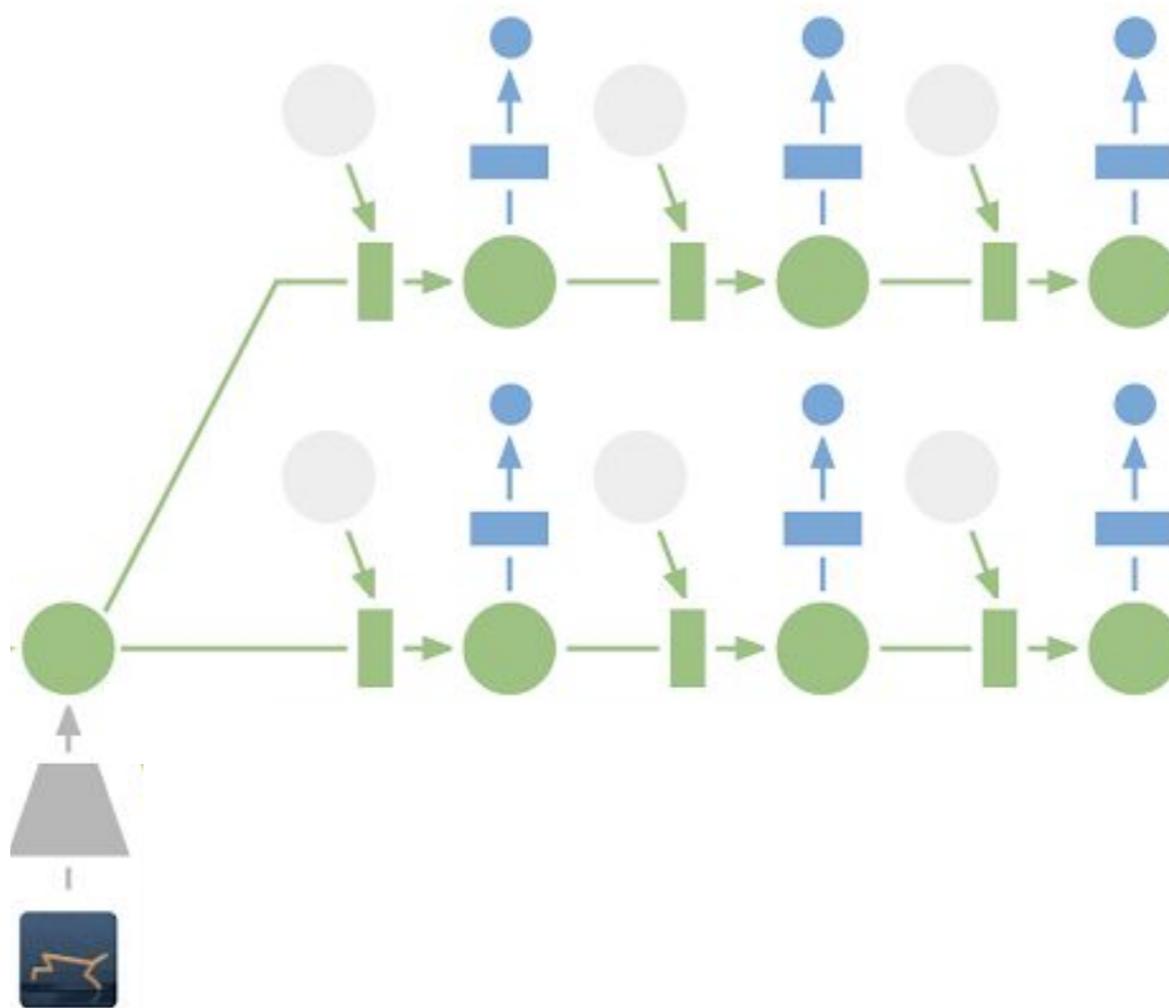
Planning in a latent space



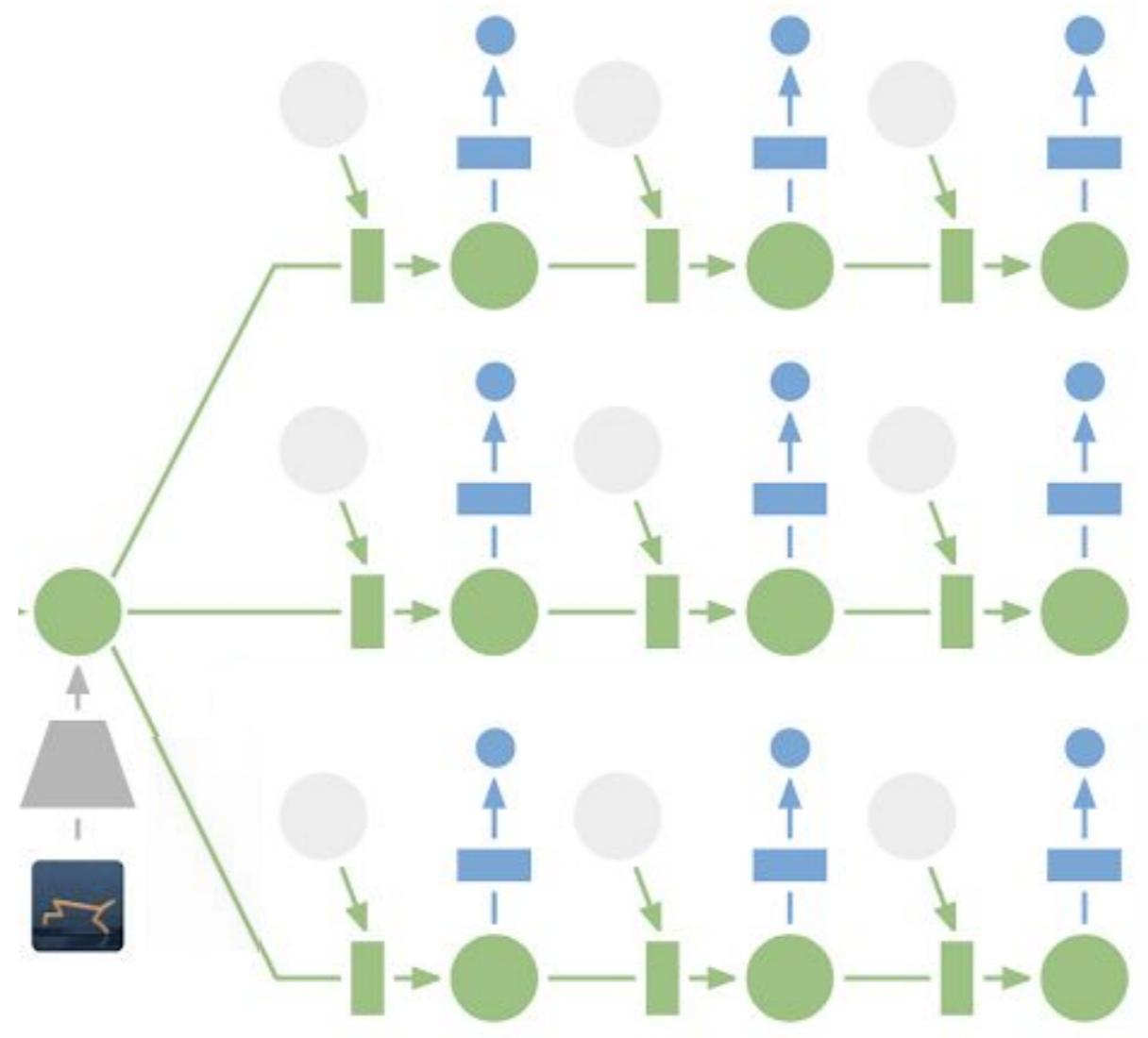
Planning in a latent space with ES



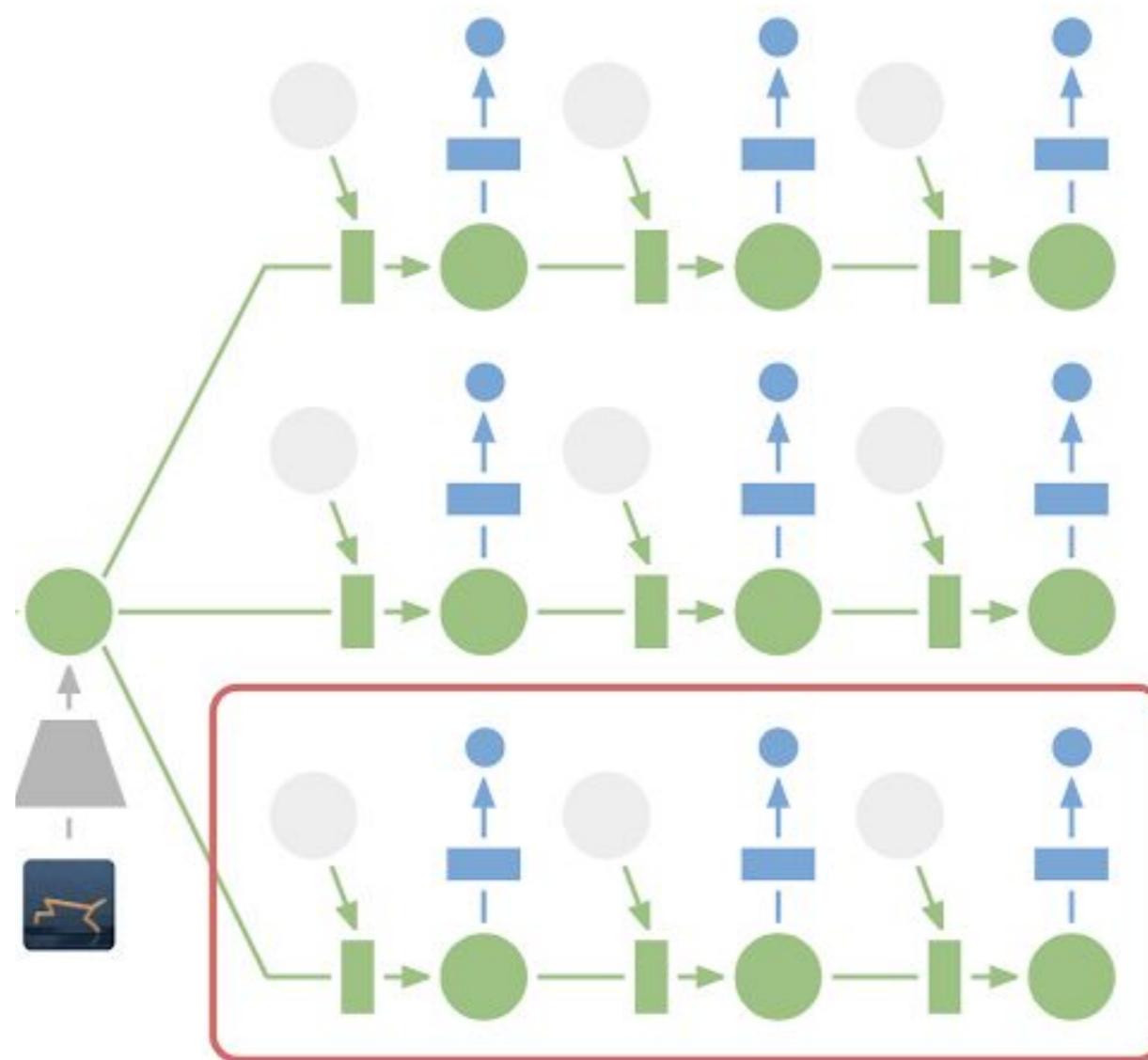
Planning in a latent space with ES



Planning in a latent space with ES

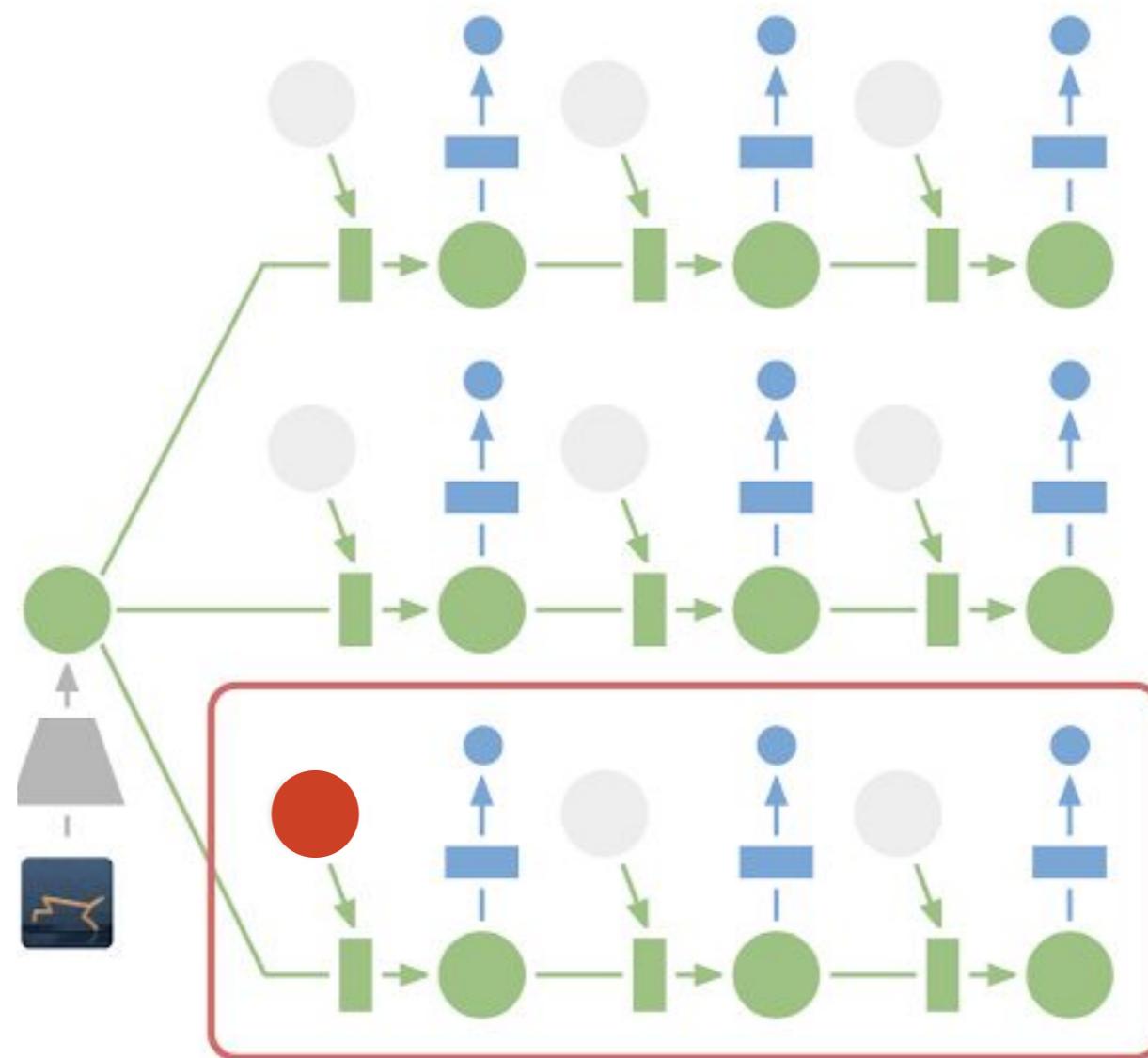


Planning in a latent space with ES



Select the action sequence with the highest *predicted* sum of rewards.

Planning in a latent space with ES



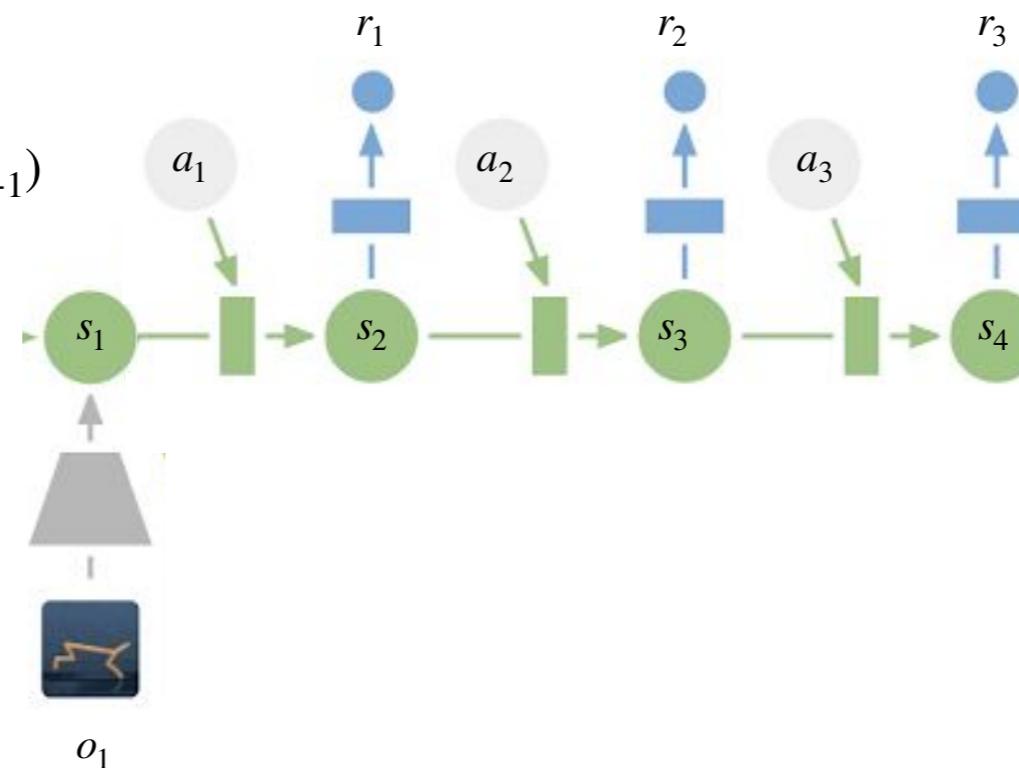
Execute the first inferred action, and repeat.

Planning in a latent space with backprop through time

Representation model $s_t \sim q(s_t | s_{t-1}, a_{t-1}, o_t)$

Transition model $s_t \sim p(s_t | s_{t-1}, a_{t-1})$

Reward model $r_t \sim p(r_t | s_t)$



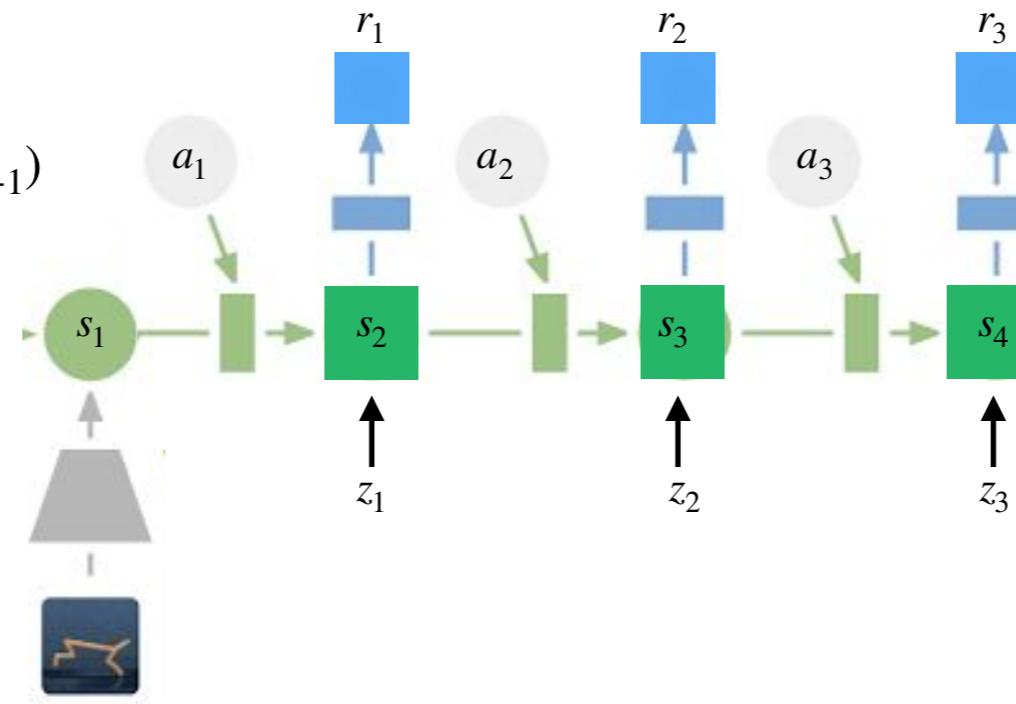
Since we know reward model and the transition model, we can back propagate all the way to actions.

Planning in a latent space with backprop through time

Representation model $s_t \sim q(s_t | s_{t-1}, a_{t-1}, o_t)$

Transition model $s_t \sim p(s_t | s_{t-1}, a_{t-1})$

Reward model $r_t \sim p(r_t | s_t)$



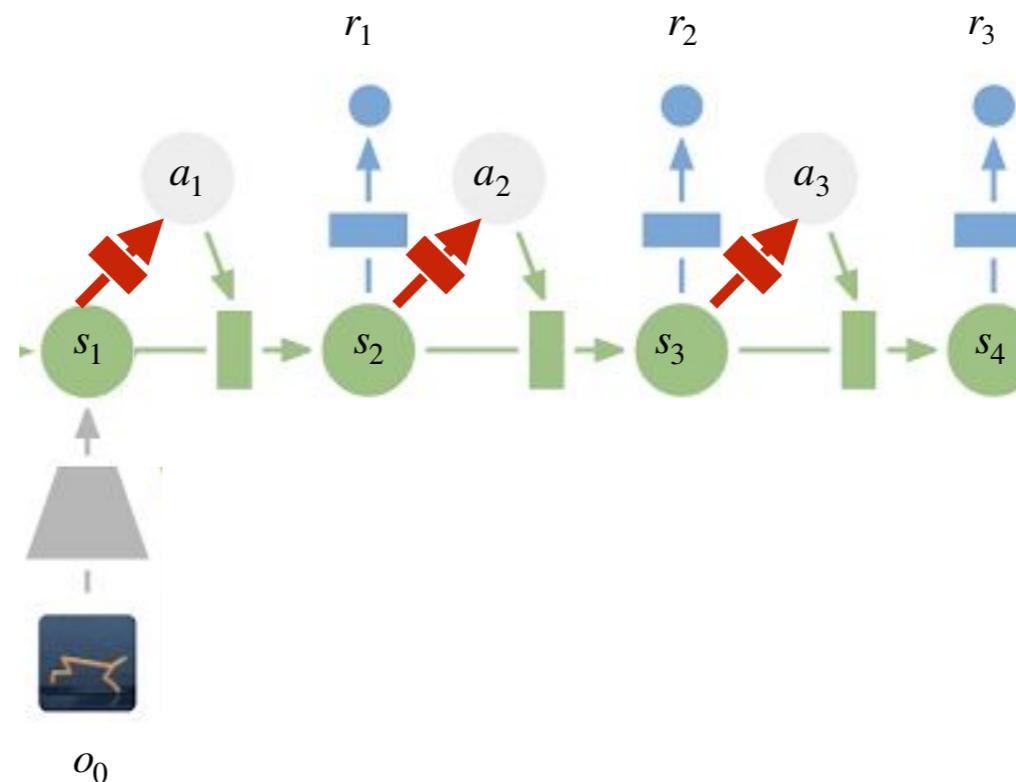
Since we know reward model and the transition model, we can back propagate all the way to actions.

When transitions and rewards are stochastic functions-> we use the re-parametrization trick.

Planning in a latent space for policy learning

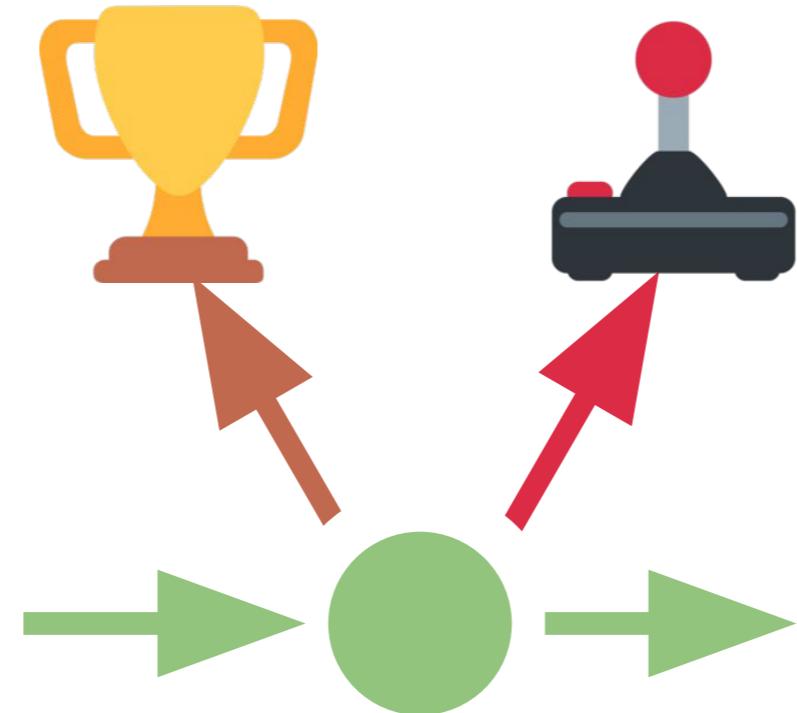
We make actions a functions of the state

We optimize for policy parameters with ES or backdrop.

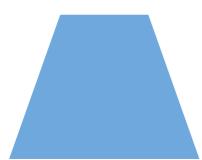


How can we make our model to consider a longer temporal horizon without enduring model errors?

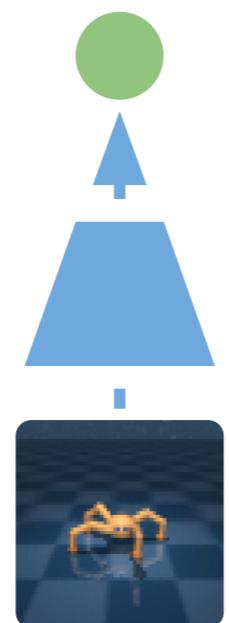
Learning Behaviors by Latent Imagination



Learning Behaviors by Latent Imagination

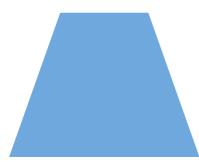


encode images



o_1

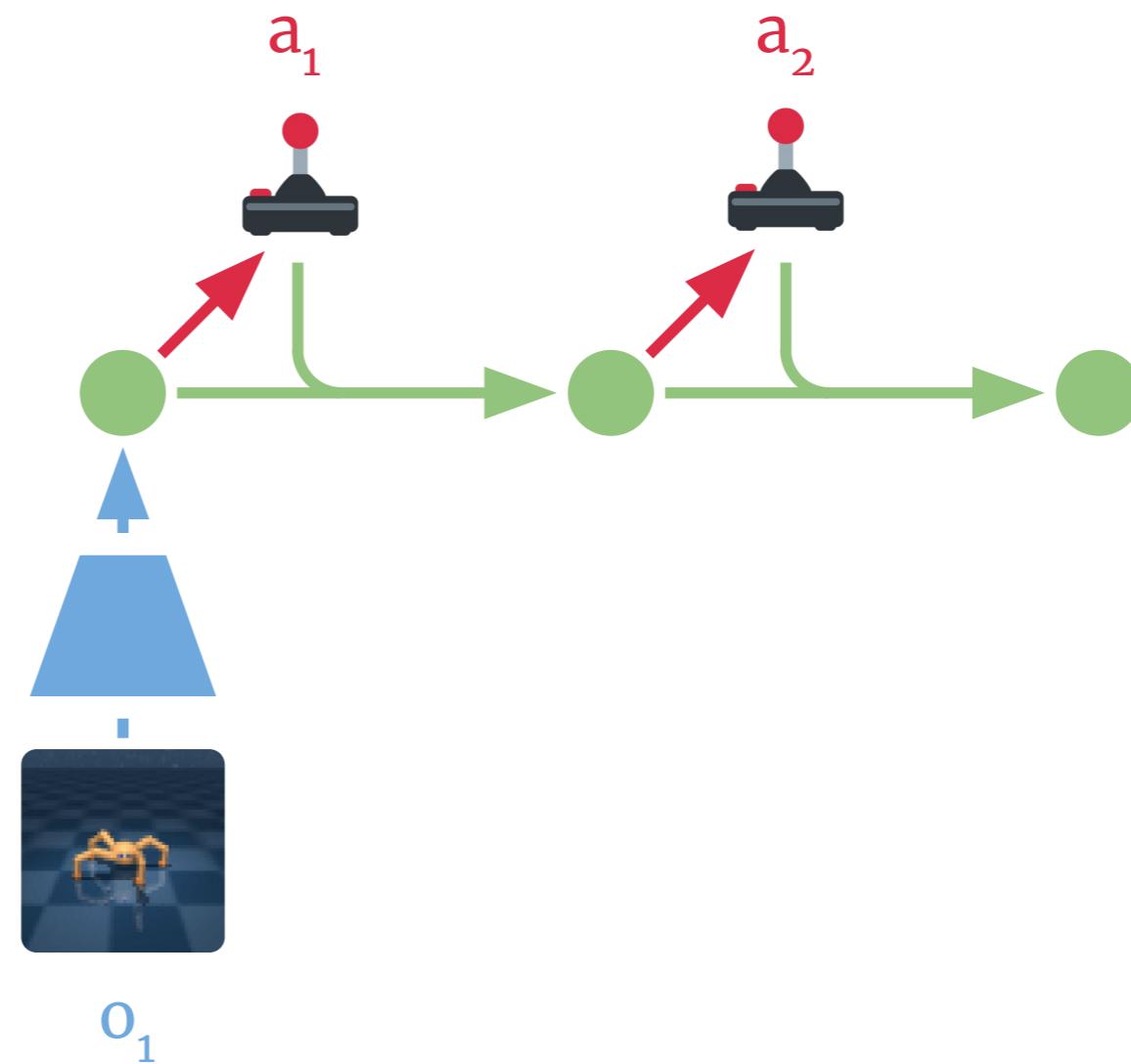
Learning Behaviors by Latent Imagination



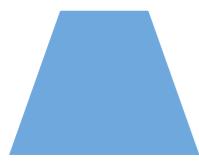
encode images



imagine ahead



Learning Behaviors by Latent Imagination



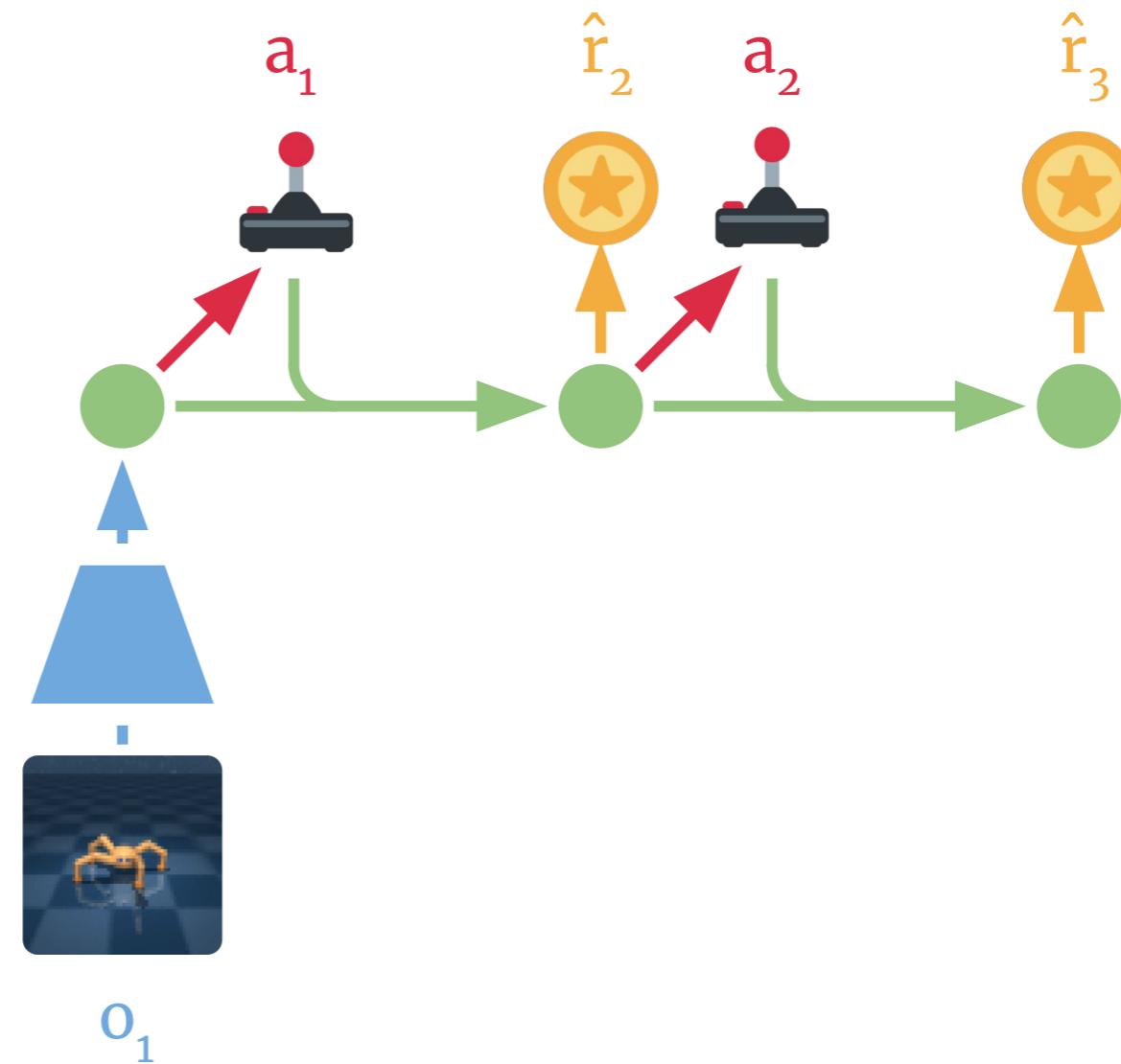
encode images



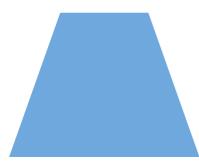
imagine ahead



predict rewards



Learning Behaviors by Latent Imagination



encode images



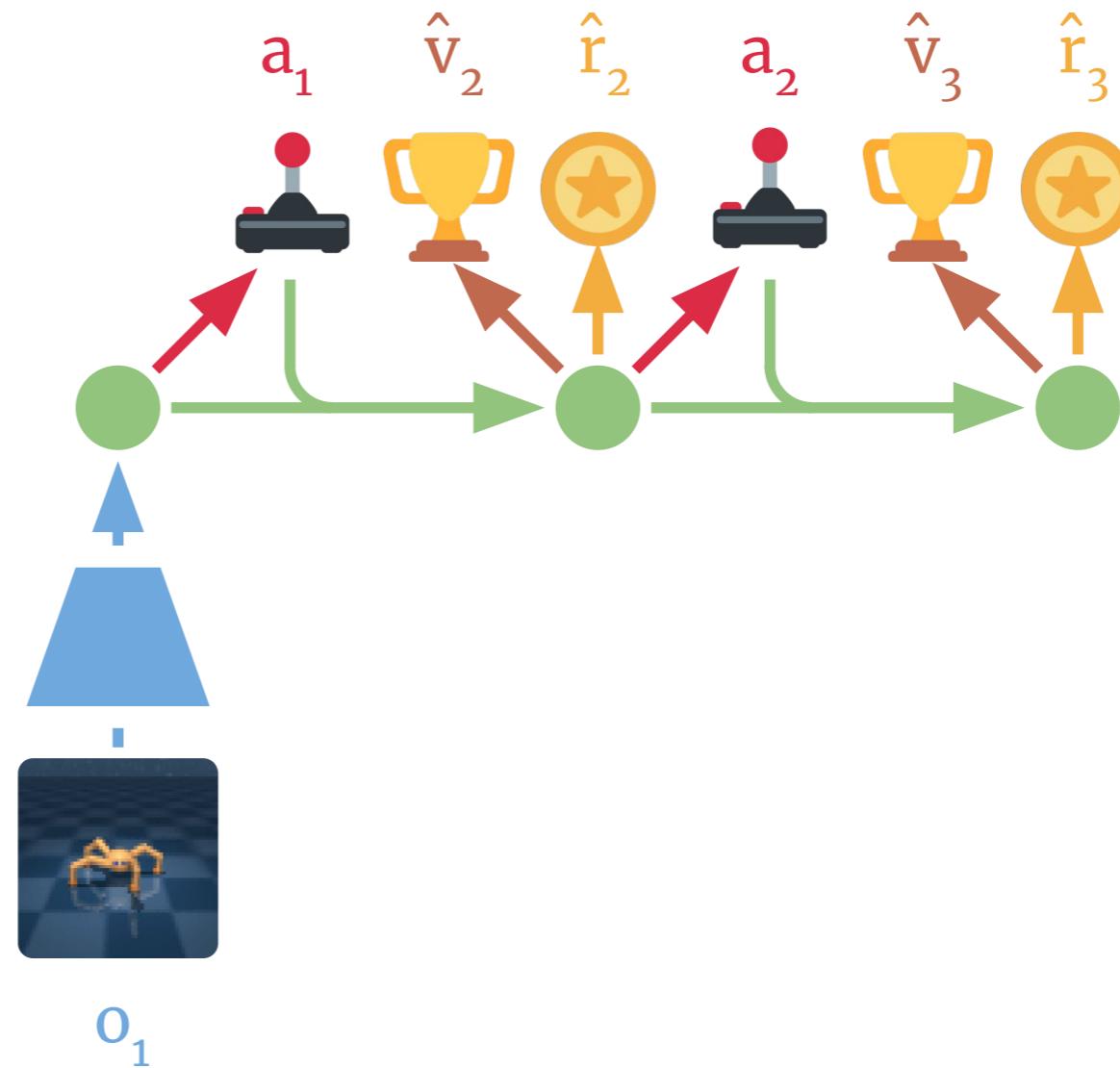
imagine ahead



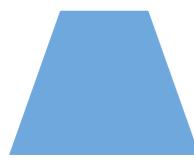
predict rewards



predict values



Learning Behaviors by Latent Imagination



encode images



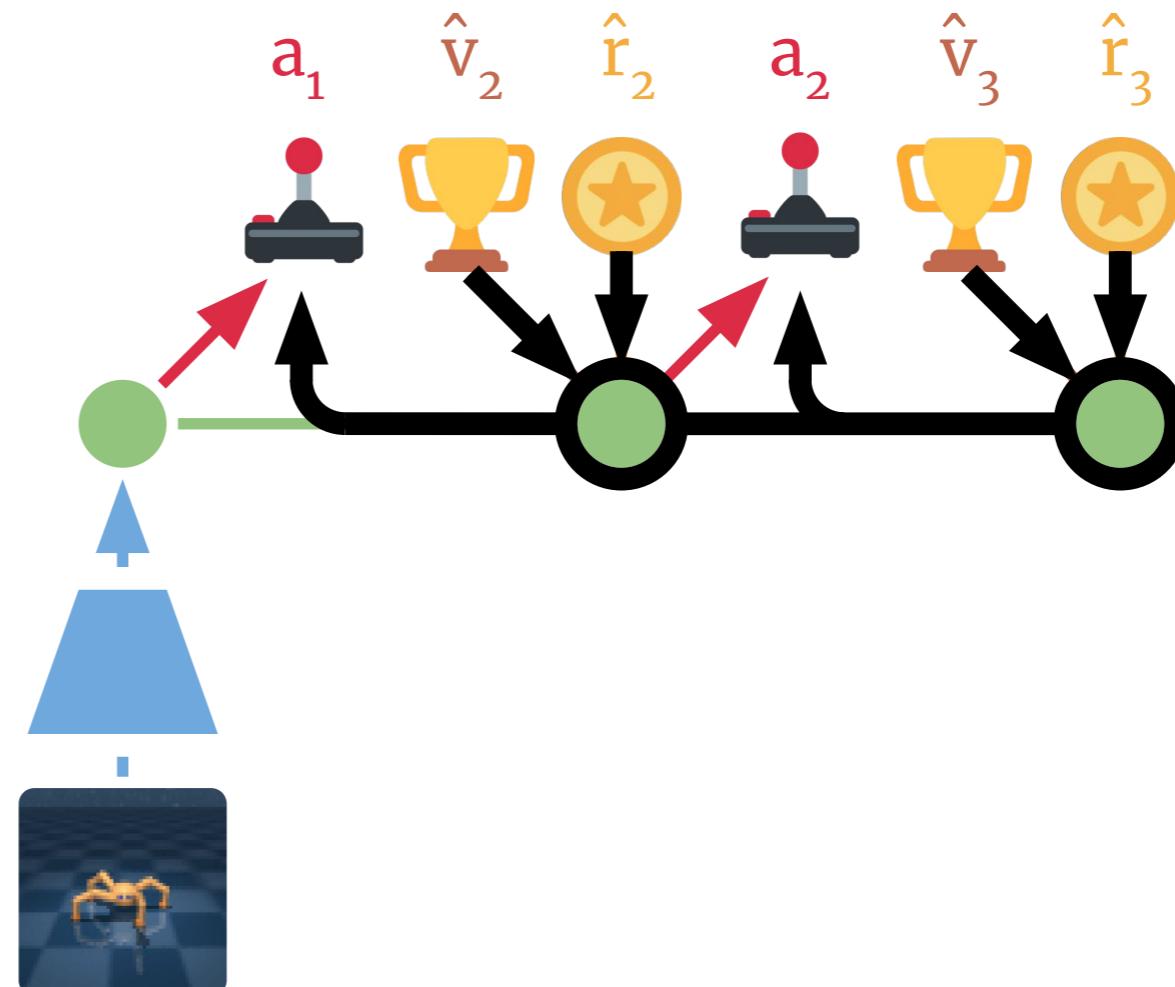
imagine ahead



predict rewards



predict values



DREAMER

Model components

- Representation model $p_\theta(s_t | s_{t-1}, a_{t-1}, o_t)$
- Transition model $p_\theta(s_t | s_{t-1}, a_{t-1})$
- Reward model $p_\theta(r_t | s_t)$
- Policy $\pi_\phi(a_t | s_t)$
- Value $v_\psi(s_t)$

Initialize policy θ, ϕ, ψ randomly and D_{env} with random trajectories $\{(o_t^{(i)}, a_t^{(i)}, r_t^{(i)})_{t=1}^T\}$.

1. Dynamics learning:

1. Sample trajectories from D_{env} and infer states from observations using the representation model: $s_t \sim p_\theta(s_t | s_{t-1}, a_{t-1}, o_t)$.
2. Train the dynamics model using variational inference and update θ .

2. Actor-critic learning from imagined rollouts:

1. Imagine trajectories seeded from s_t : $\{s_\tau, r_\tau, a_\tau, v_\tau\}_{\tau=t}^{t+H}$
2. Compute value targets $V_\lambda(s_\tau)$.
3. Update actor $\phi \rightarrow \phi + \alpha \nabla_\phi \sum_{\tau=t}^H V_\lambda(s_\tau)$
4. Update critic: $\psi \rightarrow \psi - \alpha \nabla_\psi \sum_{\tau} \|v_\psi - V_\lambda(s_\tau)\|$

3. Environment interaction

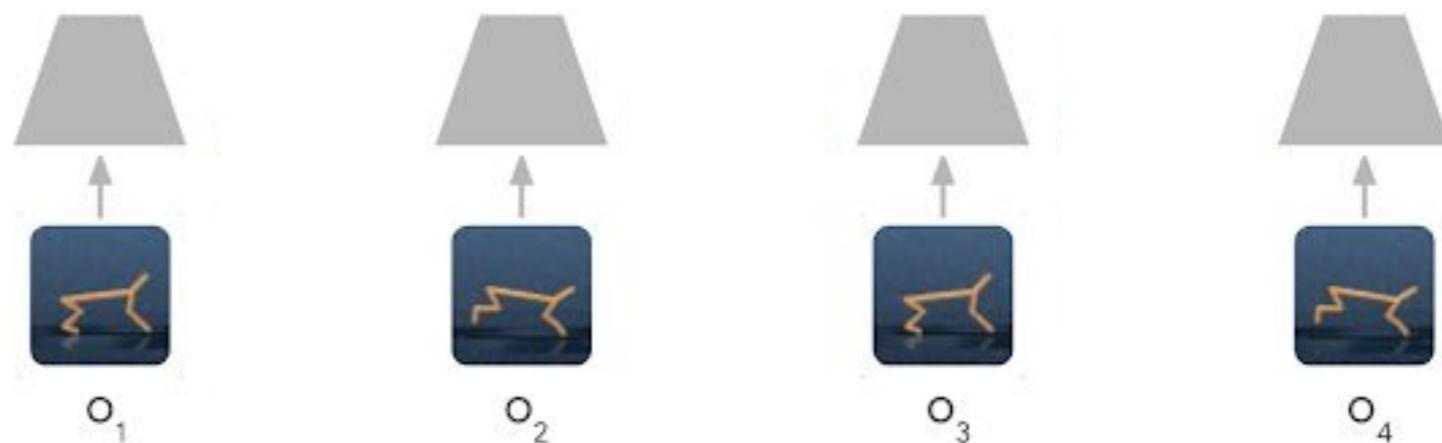
1. Deploy the actor in the environment adding exploration noise to the predicted actions and update D_{env}

How can we learn such model from experience?

Latent Dynamics Model



encode images



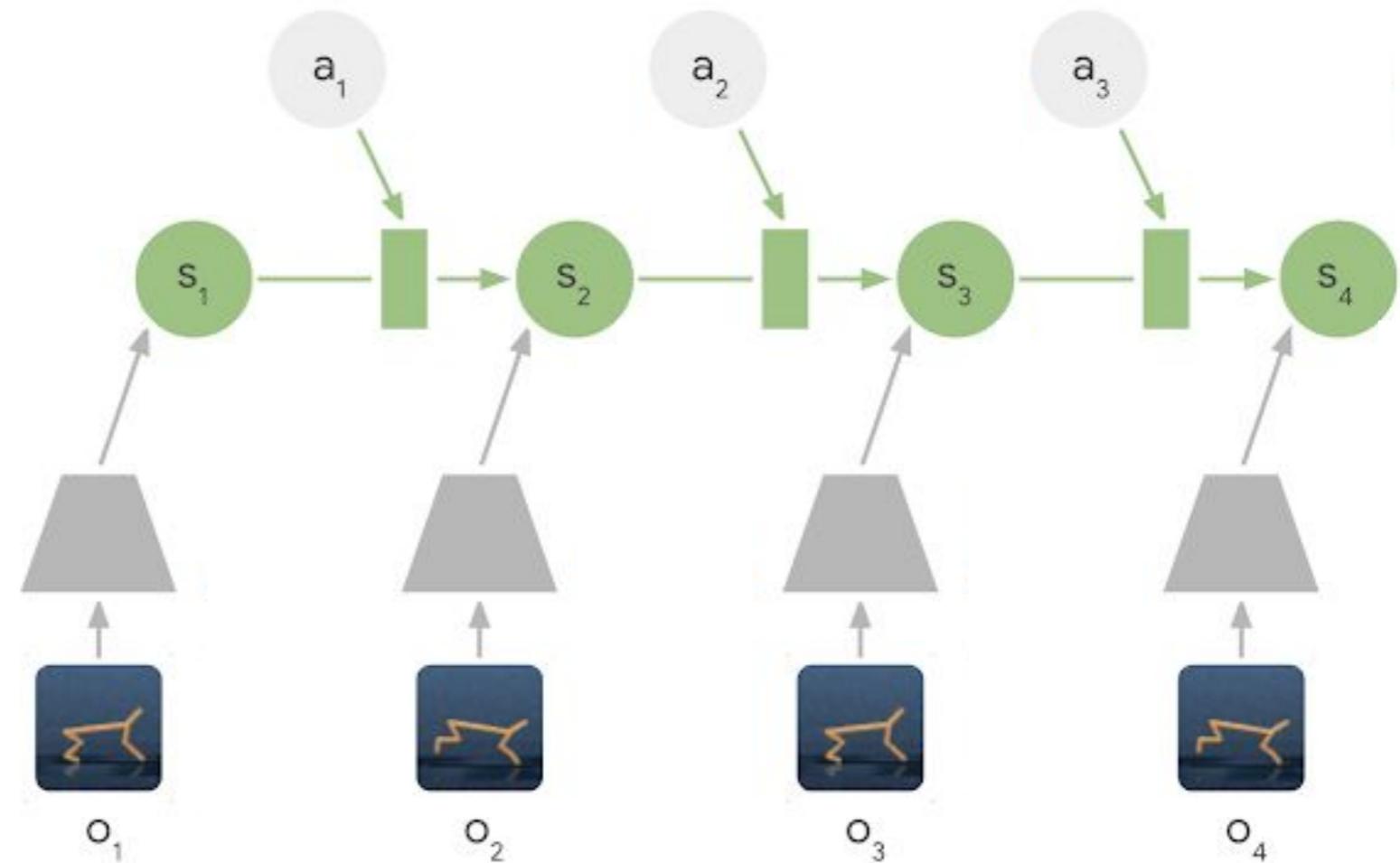
Latent Dynamics Model



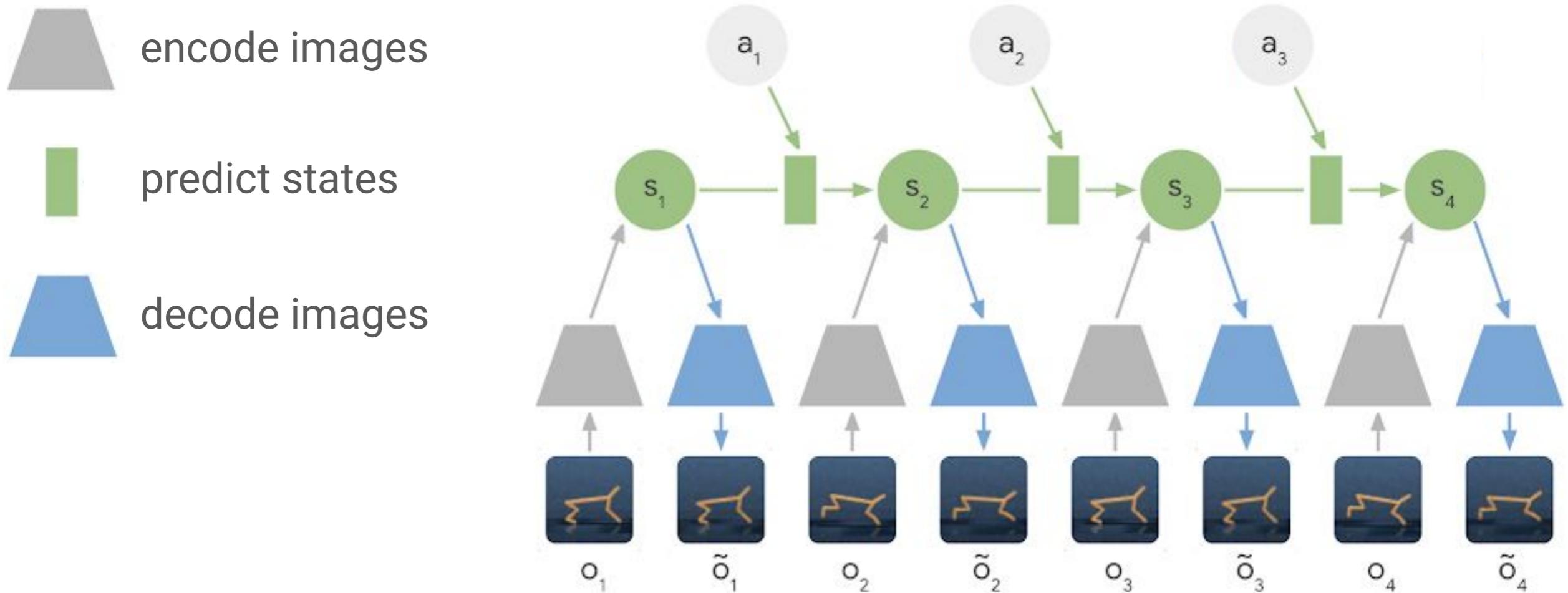
encode images



predict states

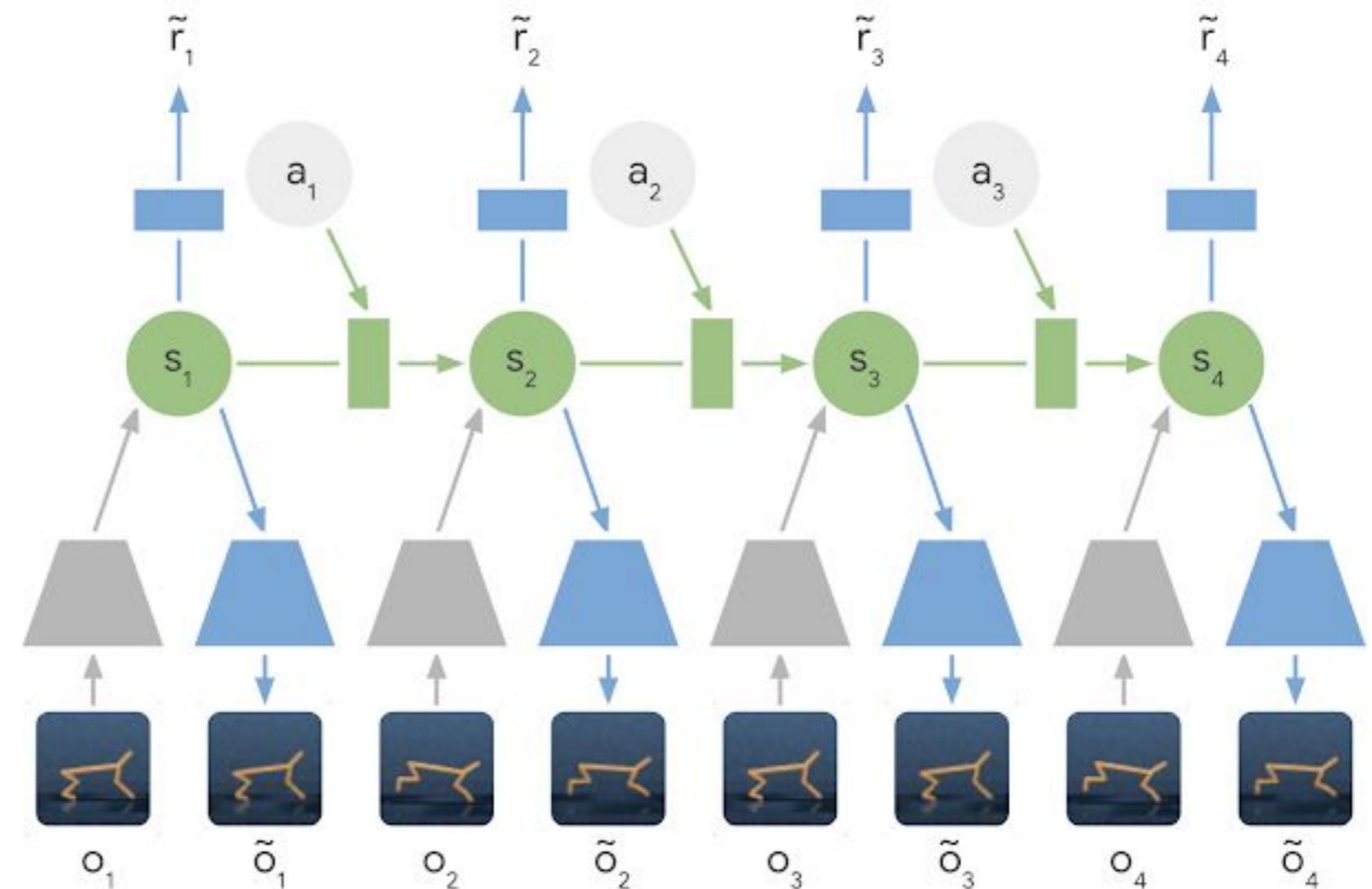


Latent Dynamics Model



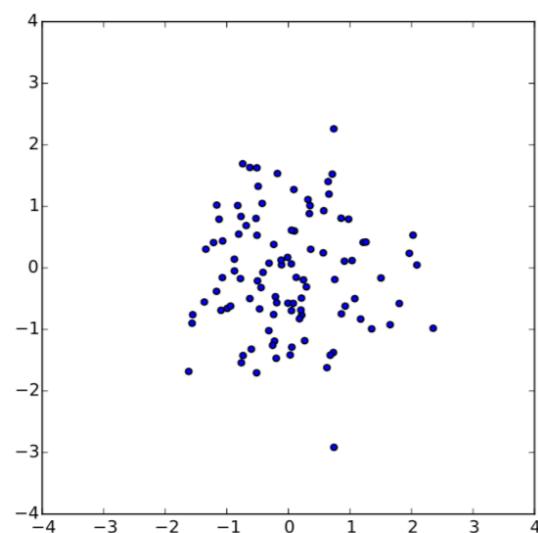
Latent Dynamics Model

- ▲ encode images
- predict states
- decode images
- decode rewards

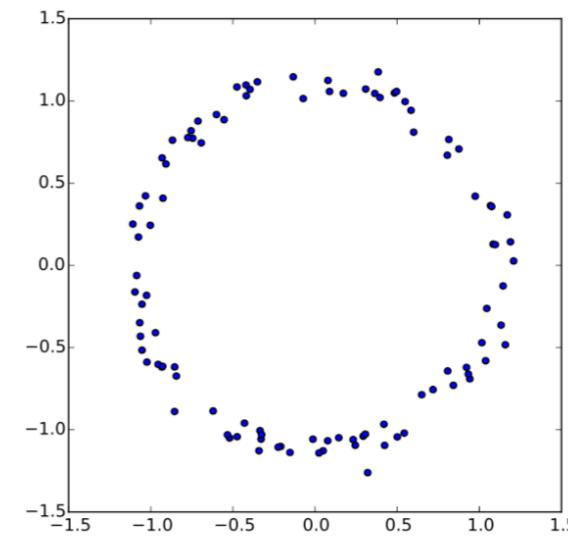


Learning stochastic generative models

- Our generative model will transforms the input Gaussian distributions into the desired action distribution.
- Why simple gaussian noise suffices to create complex outputs?
- The neural net will transform it to a complex distribution!

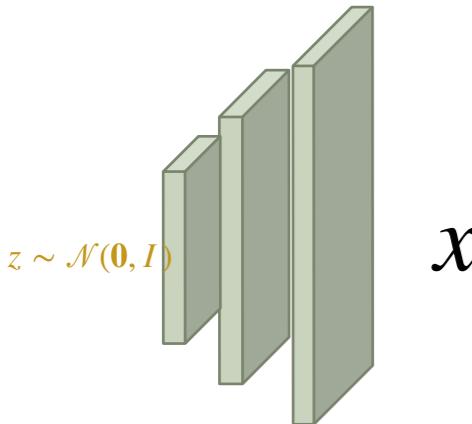


$$z \sim \mathcal{N}(\mathbf{0}, I)$$



$$f(z) = \frac{z}{10} + \frac{z}{\|z\|}$$

Unconditional generative models



Each sample z should give me a sample from the manifold I am trying to model once it passes through the neural network

We want to learn a mapping from z to the output x , usually we assume a Gaussian distribution to sample every coordinate of x from:

$$p(x | z; \theta) = \mathcal{N}(x | f(z; \theta), \sigma^2 \cdot I)$$

Let's maximize data likelihood:

$$\max_{\theta} . \quad p(x) = \int p(x | z; \theta) p(z) dz$$

Deep Variational Inference

$$p(x) = \int p(x|z)p(z)dz = \mathbb{E}_{z \sim q(z|x)} \frac{p(x|z)p(z)}{q(z|x)}$$

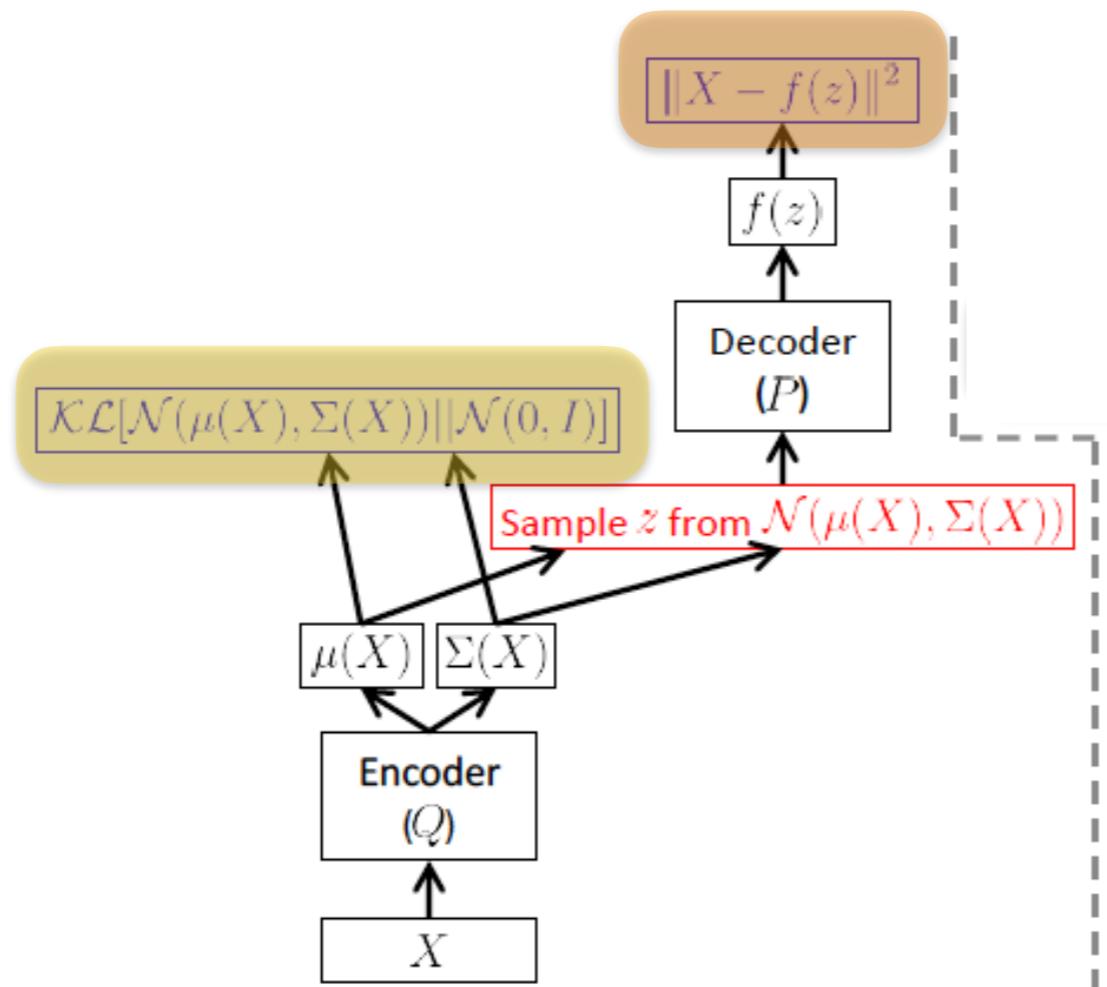
$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{z \sim q(z|x)} \log \frac{p(x|z)p(z)}{q(z|x)} \\ &= \mathbb{E}_{z \sim q(z|x)} \log p(x|z) + \mathbb{E}_{z \sim q(z|x)} \log \frac{p(z)}{q(z|x)} \\ &= \underbrace{\mathbb{E}_{z \sim q(z|x)} \log p(x|z)}_{\text{reconstruction}} - \underbrace{D_{KL}(q(z|x), p(z))}_{\text{complexity}} \end{aligned}$$

Deep Variational Inference

$$p(y) = \int p(x|z)p(z)dz = \mathbb{E}_{z \sim q(z|x)} \frac{p(x|z)p(z)}{q(z|x)}$$

$$\begin{aligned} \log p(x) &\geq \mathbb{E}_{z \sim q(z|x)} \log \frac{p(x|z)p(z)}{q(z|x)} \\ &= \mathbb{E}_{z \sim q(z|x)} \log p(x|z) + \mathbb{E}_{z \sim q(z|x)} \log \frac{p(z)}{q(z|x)} \\ &= \underbrace{\mathbb{E}_{z \sim q(z|x)} \log p(x|z)}_{\text{reconstruction}} - \underbrace{D_{KL}(q(z|x), p(z))}_{\text{complexity}} \end{aligned}$$

Variational Autoencoder

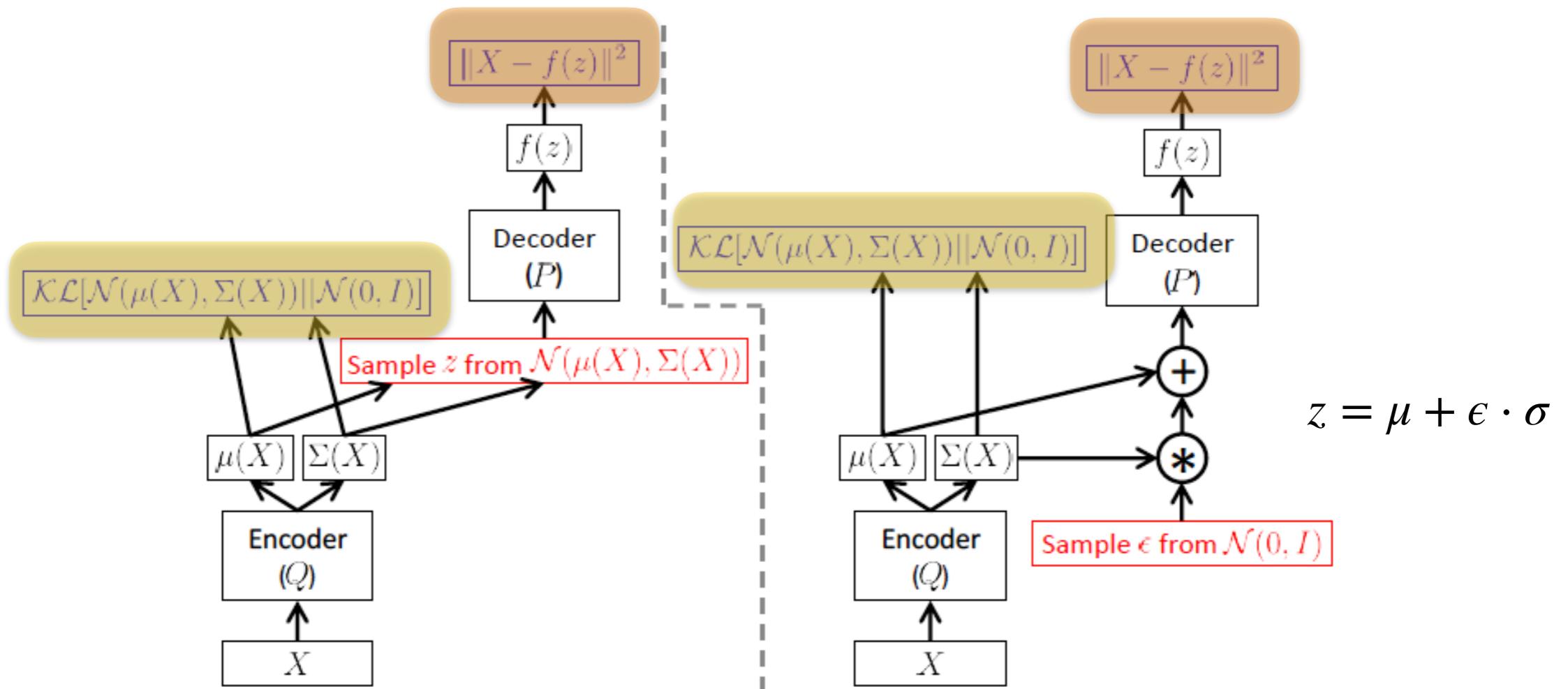


$$\min_{\phi, \theta} . \quad D_{KL}(Q(z|X; \phi) || P(z)) - \mathbb{E}_Q \log P(X|z; \theta)$$

encoder decoder

Variational Autoencoder

From left to right: re-parametrization trick!



$$\min_{\phi, \theta} . \quad D_{KL}(Q(z | X; \phi) || P(z)) - \mathbb{E}_Q \log P(X | z; \theta)$$

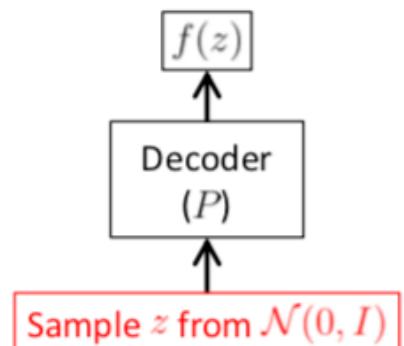
encoder decoder

$$D_{\text{KL}}(\mathcal{N}(\mu, \sigma) || \mathcal{N}(0, 1)) = -\frac{1}{2} \sum_{j=1}^J (1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

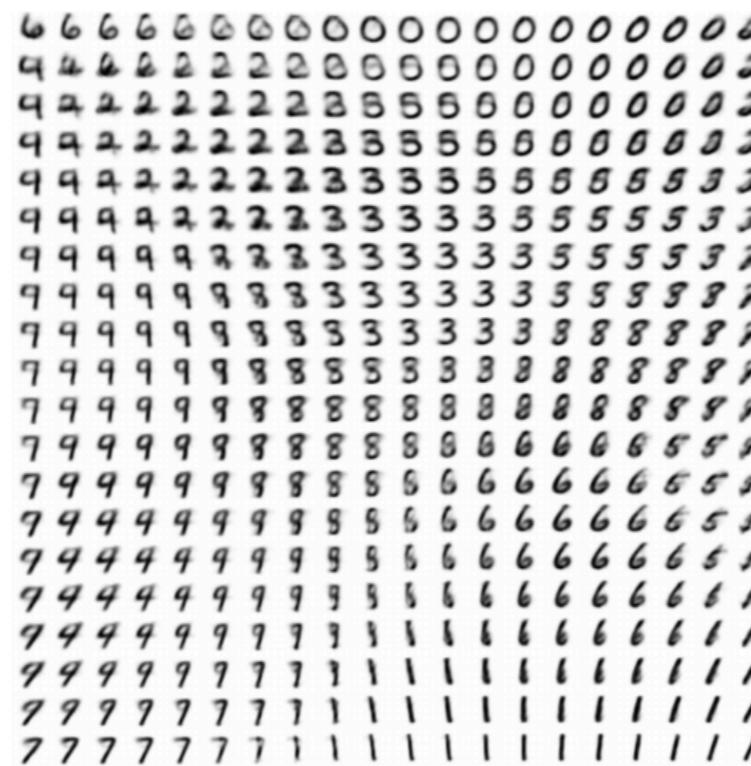
Tutorial on variational Autoencoders Doersch

Variational Autoencoder

At test time



(a) Learned Frey Face manifold



(b) Learned MNIST manifold

Conditional generative models



Given an input x and an output y , we define:

$$p(y|x) = \mathcal{N}(y|f(x, z), \sigma^2 \cdot I), z \sim \mathcal{N}(0, I)$$

Let's maximize data likelihood.

$$\max_{\theta} . \quad p(y|x) = \int p(y|z, x)p(z)dz$$

Deep Variational Inference

$$p(y|x) = \int p(y|z, x)p(z)dz = \mathbb{E}_{z \sim q(z|x,y)} \frac{p(y|x, z)p(z)}{q(z|x, y)}$$

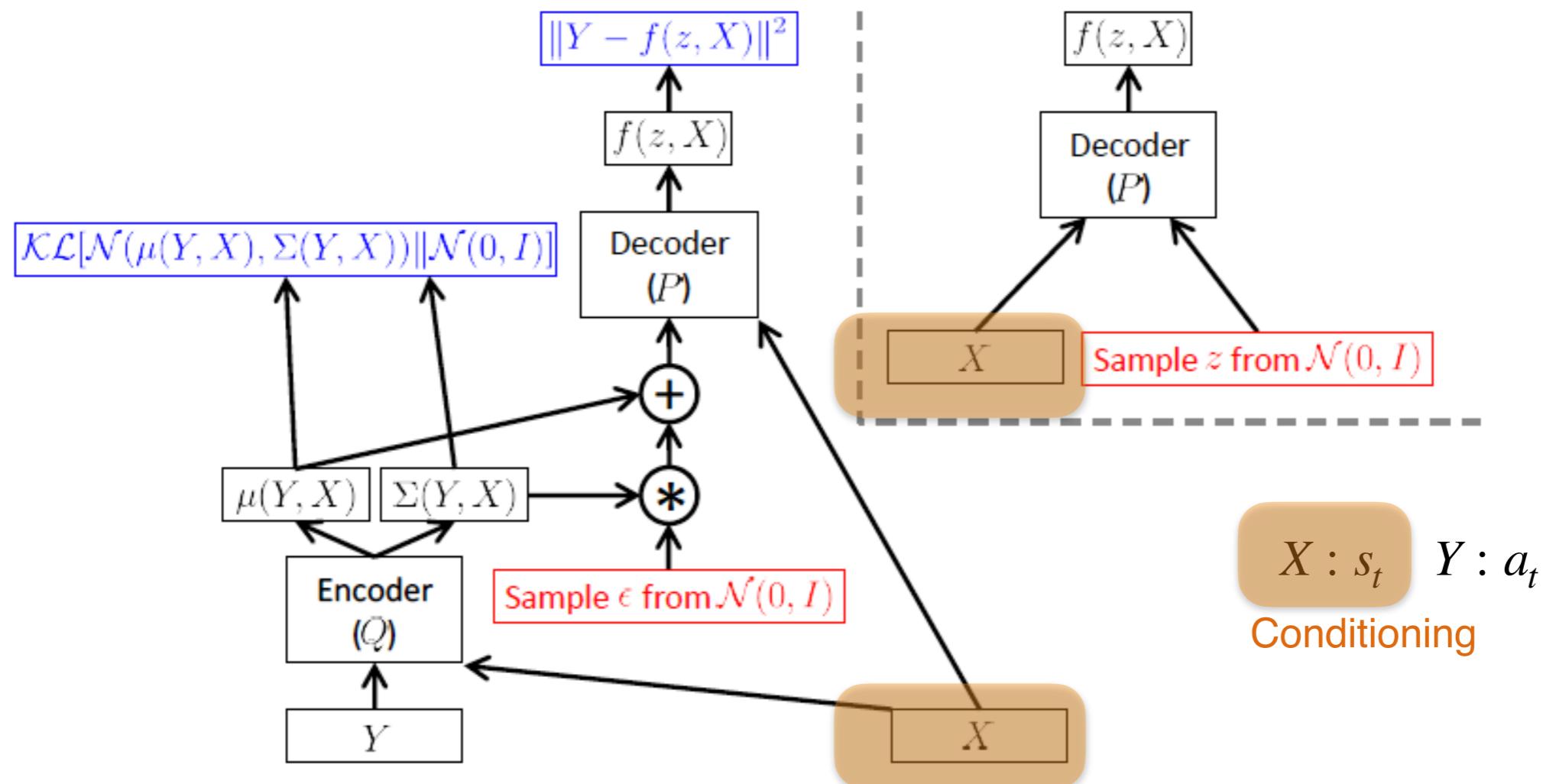
$$\begin{aligned} \log p(y|x) &\geq \mathbb{E}_{z \sim q(z|x,y)} \log \frac{p(y|x, z)p(z)}{q(z|x, y)} \\ &= \mathbb{E}_{z \sim q(z|x,y)} \log p(y|x, z) + \mathbb{E}_{z \sim q(z|x,y)} \log \frac{p(z)}{q(z|x, y)} \\ &= \underbrace{\mathbb{E}_{z \sim q(z|x,y)} \log p(y|x, z)}_{\text{reconstruction}} - \underbrace{D_{KL}(q(z|x, y), p(z))}_{\text{complexity}} \end{aligned}$$

Deep Variational Inference

$$p(y|x) = \int p(y|z, x)p(z)dz = \mathbb{E}_{z \sim q(z|x,y)} \frac{p(y|x, z)p(z)}{q(z|x, y)}$$

$$\begin{aligned} \log p(y|x) &\geq \mathbb{E}_{z \sim q(z|x,y)} \log \frac{p(y|x, z)p(z)}{q(z|x, y)} \\ &= \mathbb{E}_{z \sim q(z|x,y)} \log p(y|x, z) + \mathbb{E}_{z \sim q(z|x,y)} \log \frac{p(z)}{q(z|x, y)} \\ &= \underbrace{\mathbb{E}_{z \sim q(z|x,y)} \log p(y|x, z)}_{\text{decoder}} - \underbrace{D_{KL}(q(z|x, y), p(z))}_{\text{encode complexity}} \end{aligned}$$

Conditional VAE



$$\min_{\phi, \theta} . D_{KL}(Q(z|X, Y; \phi) || P(z)) - \mathbb{E}_Q \log P(Y|X, z; \theta)$$

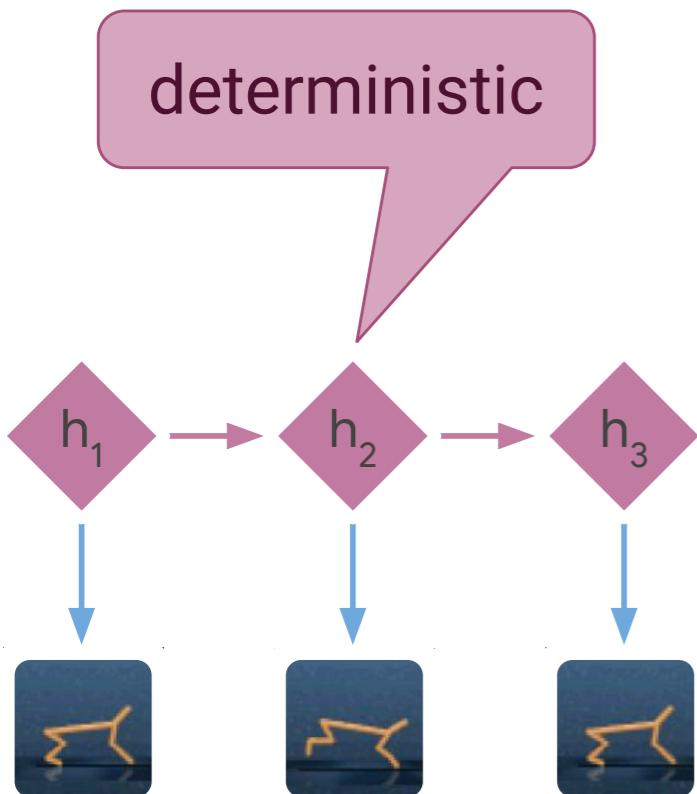
Deep Variational Inference for latent dynamics models

$$p(o_{1:T} | a_{0:T}, \hat{o}_0) = \int \prod_{t=1}^T p(o_t | s_t) p(s_t | s_{t-1}, a_{t-1}) p_{init}(s_0 | \hat{o}_0) ds_{0:T}$$

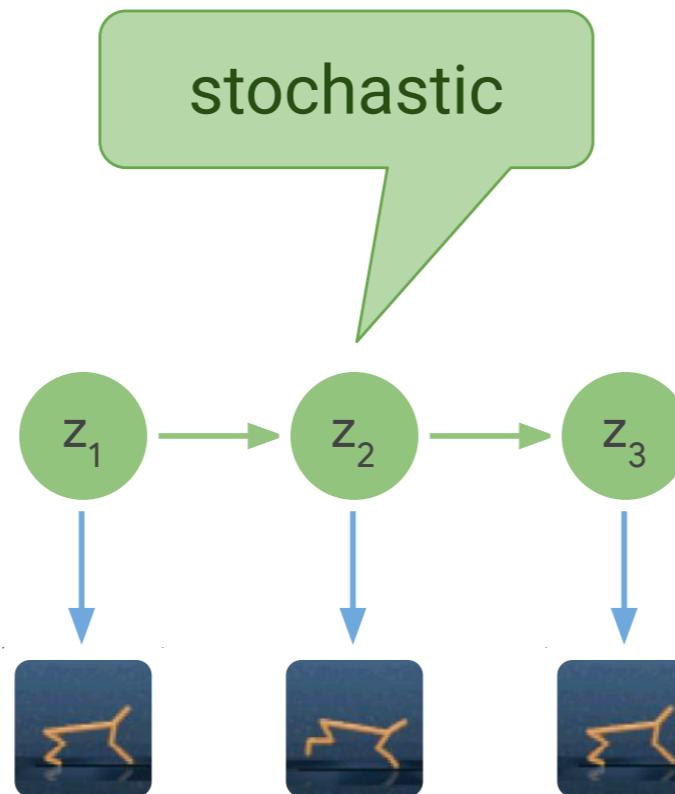
Imagine s_0 is fixed.

$$\begin{aligned} p(o_{1:T} | a_{0:T}, \hat{o}_0) &= \int \prod_{t=1}^T p(o_t | s_t) p(s_t | s_{t-1}, a_{t-1}) ds_{0:T} \\ \ln p(o_{1:T} | a_{1:T}) &\triangleq \ln \mathbb{E}_{p(s_{1:T} | a_{1:T})} \left[\prod_{t=1}^T p(o_t | s_t) \right] \\ &= \ln \mathbb{E}_{q(s_{1:T} | o_{1:T}, a_{1:T})} \left[\prod_{t=1}^T p(o_t | s_t) p(s_t | s_{t-1}, a_{t-1}) / q(s_t | o_{\leq t}, a_{<t}) \right] \\ &\geq \mathbb{E}_{q(s_{1:T} | o_{1:T}, a_{1:T})} \left[\sum_{t=1}^T \ln p(o_t | s_t) + \ln p(s_t | s_{t-1}, a_{t-1}) - \ln q(s_t | o_{\leq t}, a_{<t}) \right] \\ &= \sum_{t=1}^T \left(\frac{\mathbb{E}[\ln p(o_t | s_t)]}{\underset{\text{reconstruction}}{q(s_t | o_{\leq t}, a_{<t})}} - \frac{\mathbb{E}[\text{KL}[q(s_t | o_{\leq t}, a_{<t}) \| p(s_t | s_{t-1}, a_{t-1})]]}{\underset{\text{complexity}}{q(s_{t-1} | o_{\leq t-1}, a_{<t-1})}} \right). \end{aligned}$$

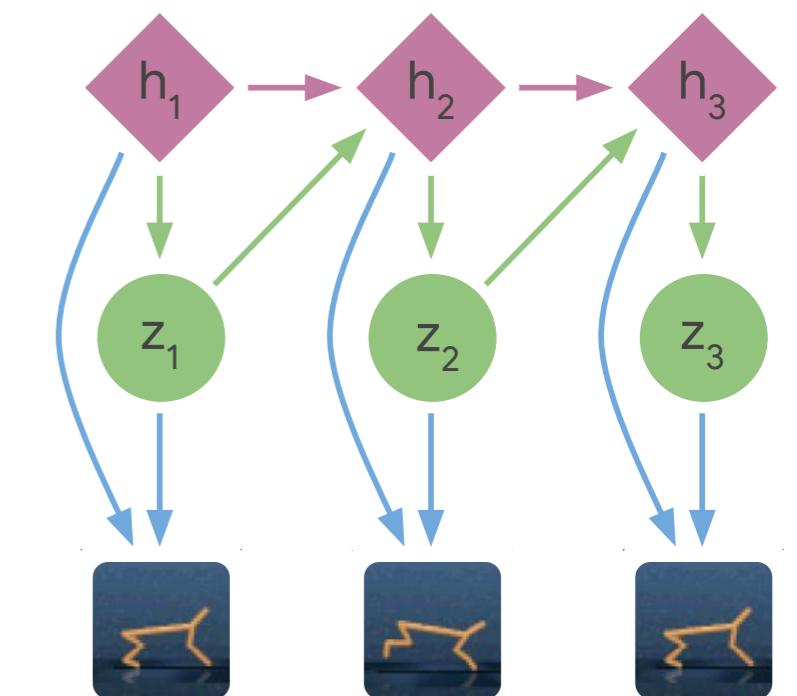
Recurrent State Space Model



Recurrent Neural Network

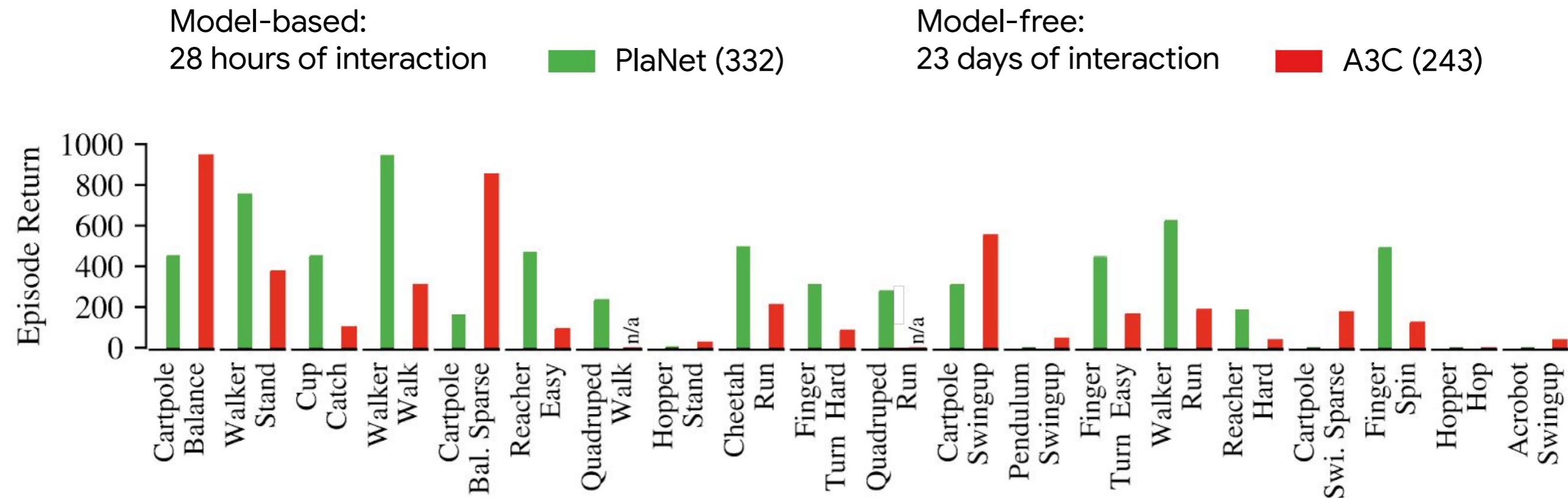


State Space Model

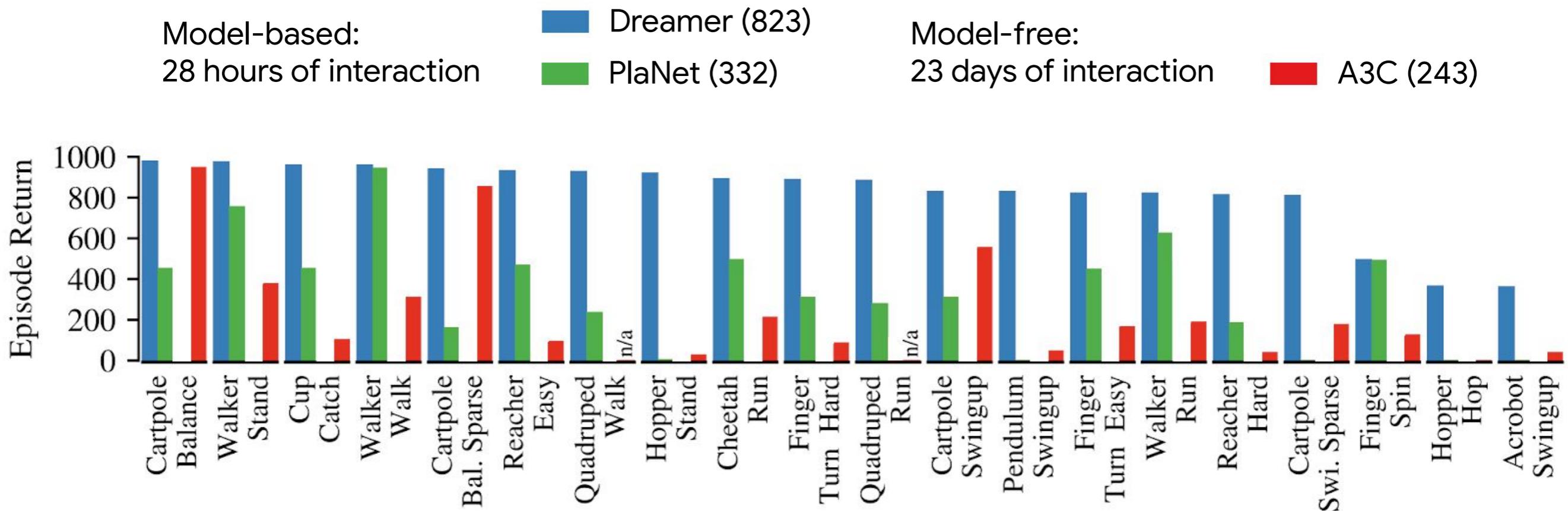


Recurrent State Space Model

Large-Scale Evaluation for Control from Pixels



Large-Scale Evaluation for Control from Pixels



Large-Scale Evaluation for Control from Pixels

