Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

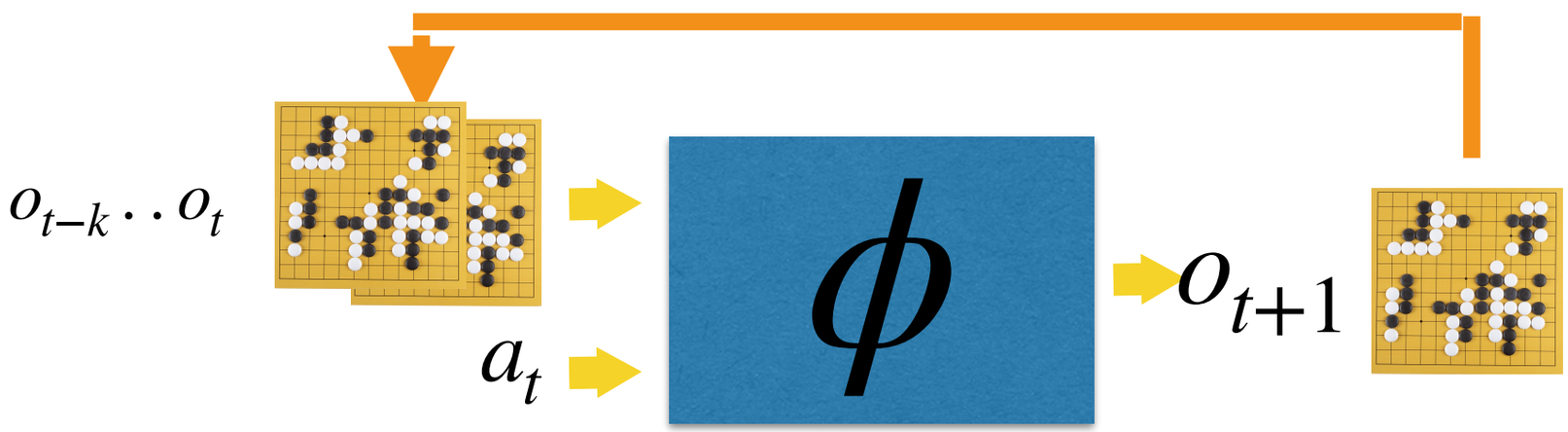
MBRL with latent dynamic models

Spring 2021, CMU 10-403

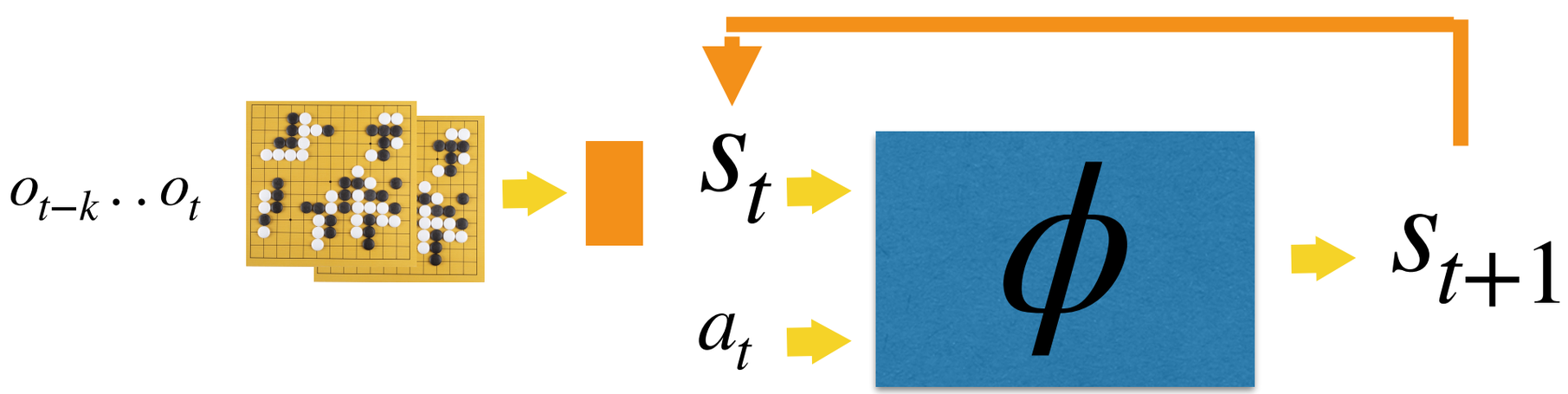
Katerina Fragkiadaki

Model learning from sensory input

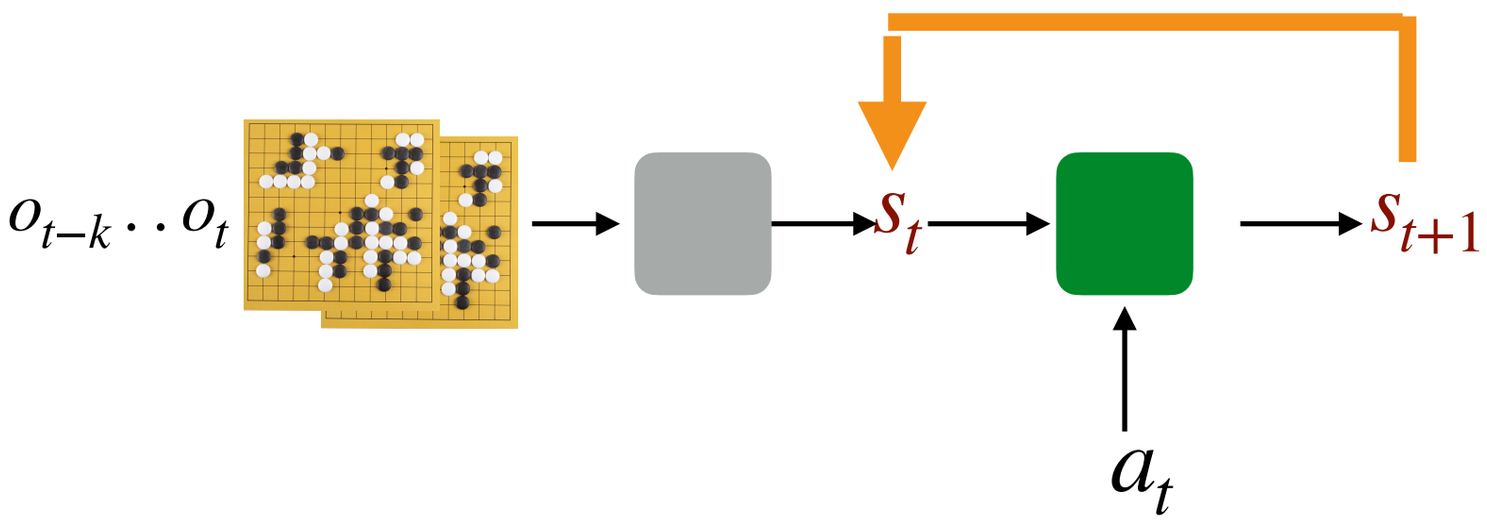
Unrolling in the observation space:



Unrolling in a latent space:

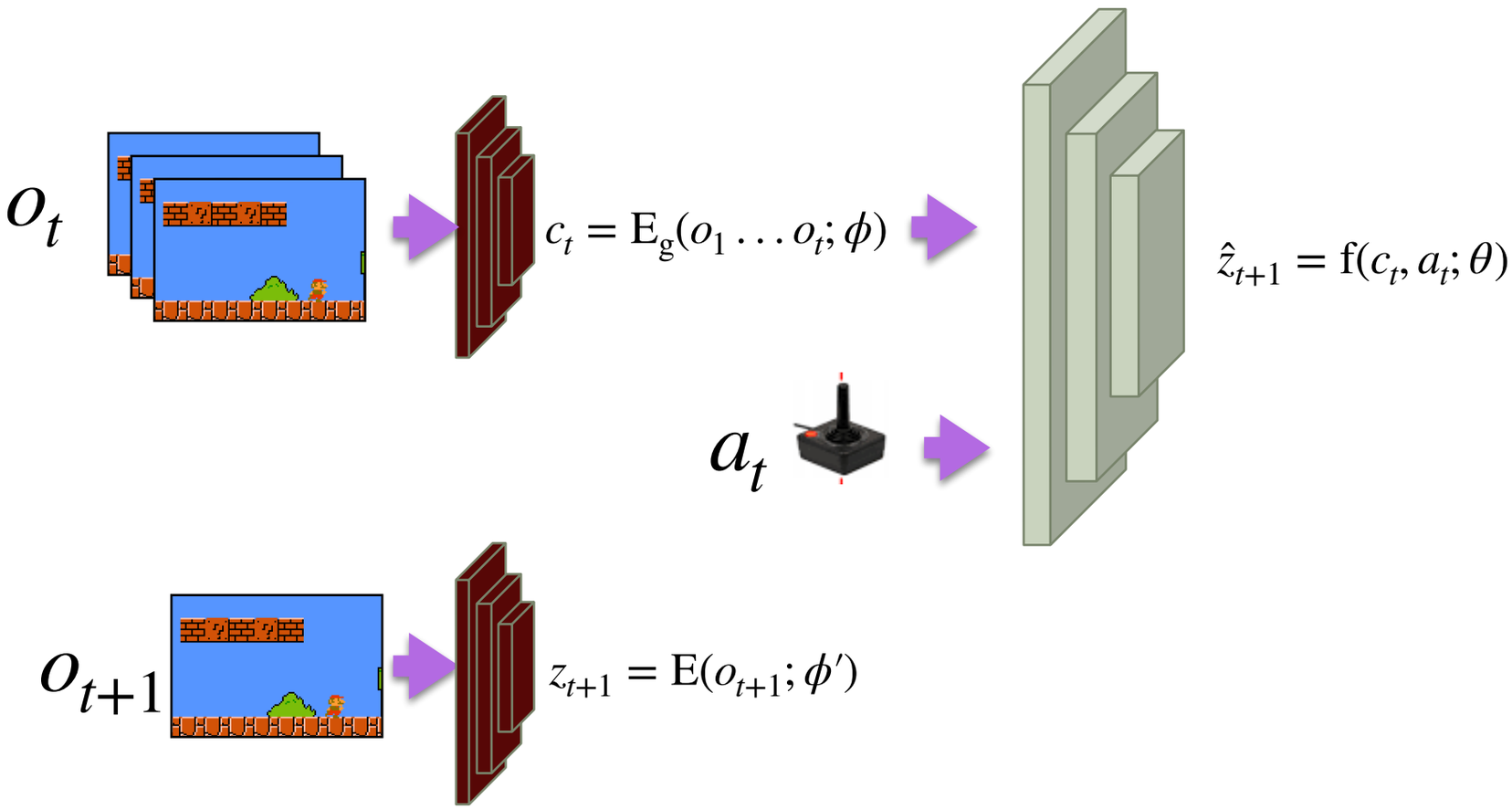


Unrolling in a latent state space



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

Prediction in a latent space



$$\min_{\theta,\phi,\phi'} . \quad \|\mathrm{f}(c_t, a_t; \theta) - z_{t+1}\|$$

Q:What is the problem with this optimization problem?

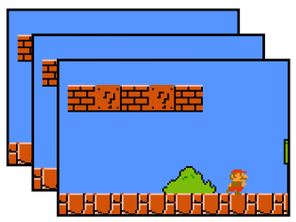
A:There is a trivial solution :-(

Q:Would the problem go away if instead we had: $\hat{z}_{t+1} = z_t + \mathrm{f}(c_t, a_t; \theta)$

A:No, it's exactly the same problem.

We need to predict additional information from the encodings to avoid the trivial solution

Prediction in a latent space - autoencoding

o_t 

$c_t = \mathrm{E_g}(o_1 \ldots o_t; \phi)$

a_t 

$\hat{z}_{t+1} = \mathrm{f}(c_t, a_t; \theta)$

$$\min_{\theta,\phi,\phi'} . \quad \|\mathrm{f}(c_t, a_t; \theta) - z_{t+1}\|$$

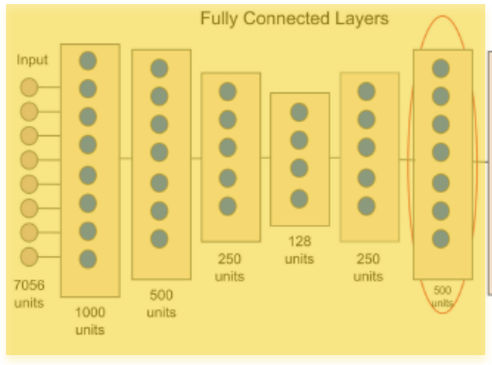
Autoencoding: $\min_{\phi',\omega} . \quad \|D(\mathrm{E}(o_t; \phi'), \omega) - o_t\|$

o_{t+1} 

$z_{t+1} = \mathrm{E}(o_{t+1}; \phi')$

$\mathrm{E}(o_t; \phi')$

o_t 



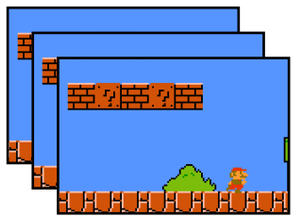
$\hat{o}_t = D(\mathrm{E}(o_t; \phi'), \omega)$

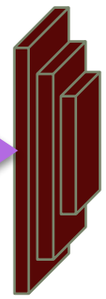
- Predict the image from the latent encoding
- …and suffer the problems of autoencoding reconstruction loss that has little to do with our task

Incentivizing exploration in RL with deep predictive models, Stadie et al.

Prediction in a latent space

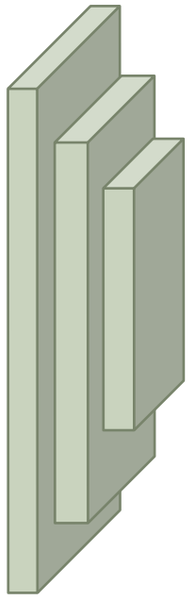
o_t





$c_t = \mathrm{E_g}(o_1 \ldots o_t; \phi)$

a_t



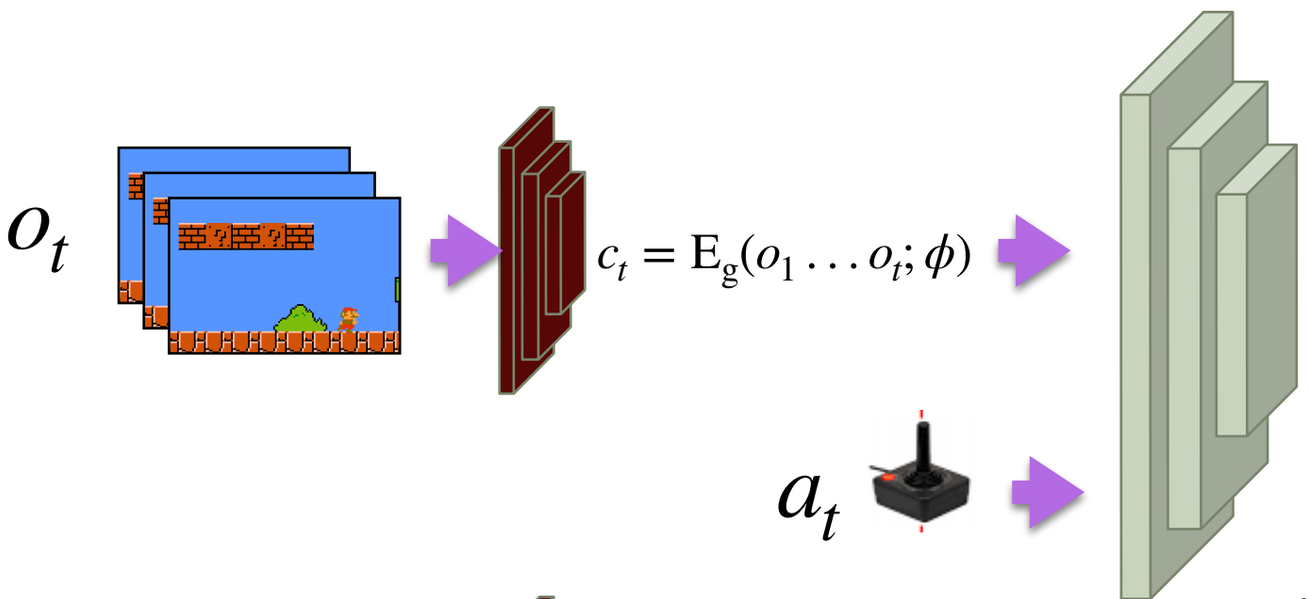
$\hat{z}_{t+1} = \mathrm{f}(c_t, a_t; \theta)$

$$\min_{\theta,\phi,\phi'} . \quad \|\mathrm{f}(c_t, a_t; \theta) - \hat{z}_{t+1}\|$$

o_{t+1}

$z_{t+1} = \mathrm{E}(o_{t+1}; \phi')$

Prediction in a latent space - inverse models



$$\min_{\theta,\phi,\phi',\psi} . \quad \|\mathrm{f}(z_t, a_t; \theta) - z_{t+1}\| + \|\mathrm{f}^{\mathrm{inv}}(z_t, z_{t+1}; \psi) - a_t\|$$

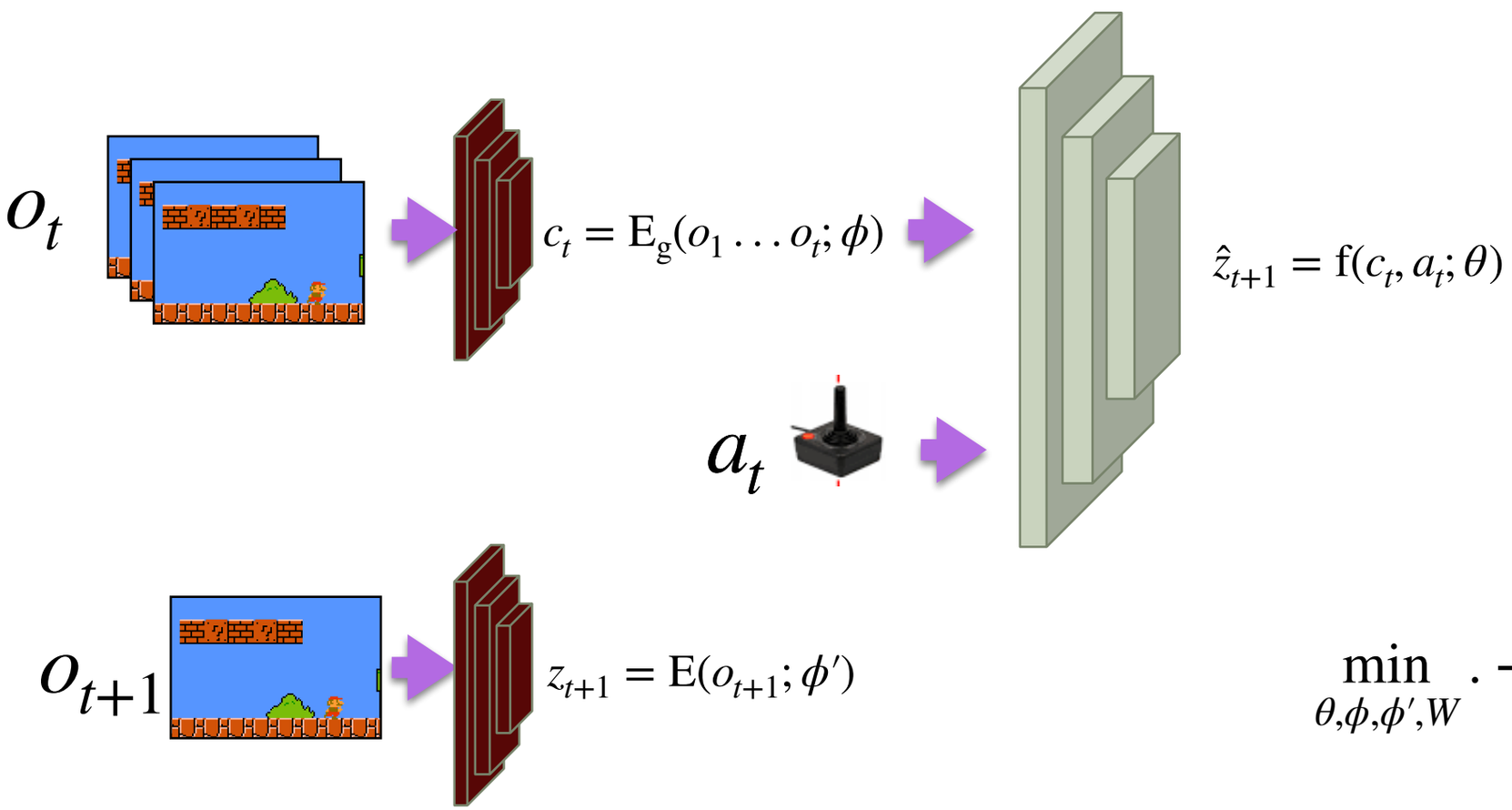
$$a_t = \mathrm{f}^{\mathrm{inv}}(z_t, z_{t+1}, \psi)$$

- Let's couple forward and inverse models (to avoid the trivial solution)
- ...then we will only predict things that the agent can control

Learning to poke by poking, *Agrawal et al. 2016*

Prediction in a latent space - contrastive prediction

$$P(d = i \mid c) = \frac{\exp(c^\top W z^i)}{\sum_j \exp(c^\top W z^j)}$$

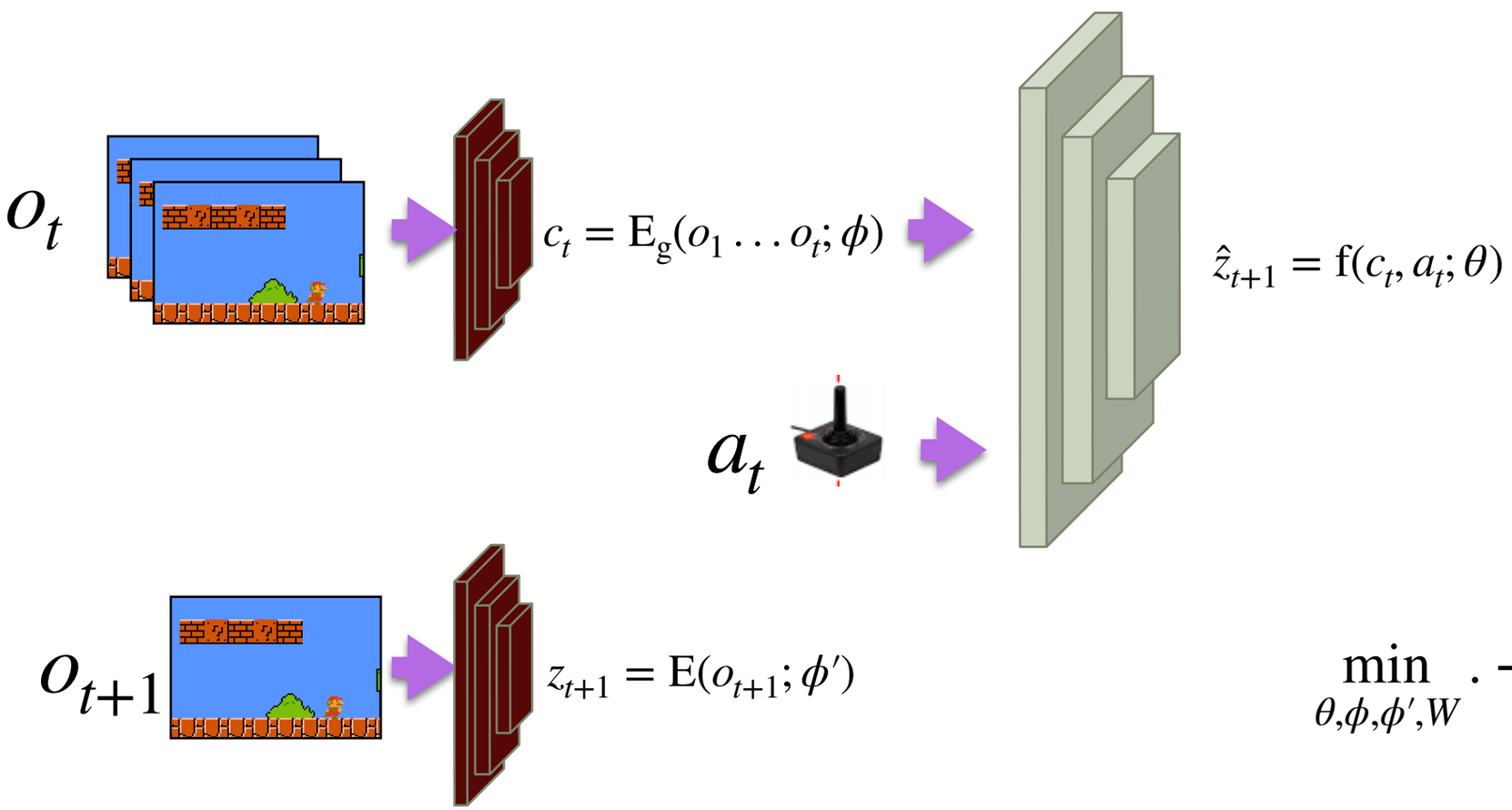


$$\min_{\theta,\phi,\phi',W} . - \log \frac{\exp(\hat{z}_{t+1}{}^\top W z_{t+1})}{\sum_{j \in Neg} \exp(\hat{z}_{t+1}{}^\top W z^j)}$$

- Generative: model the distribution of future observations/embeddings
- Discriminative: model how much closer you can match the future observations than other alternatives.
- Imagine we could discretize all the possible future: then we would just need to predict the right probability distribution over all (discrete set of) possibilities. Then, we want to maximize the probability of the correct outcome.

Prediction in a latent space - contrastive prediction

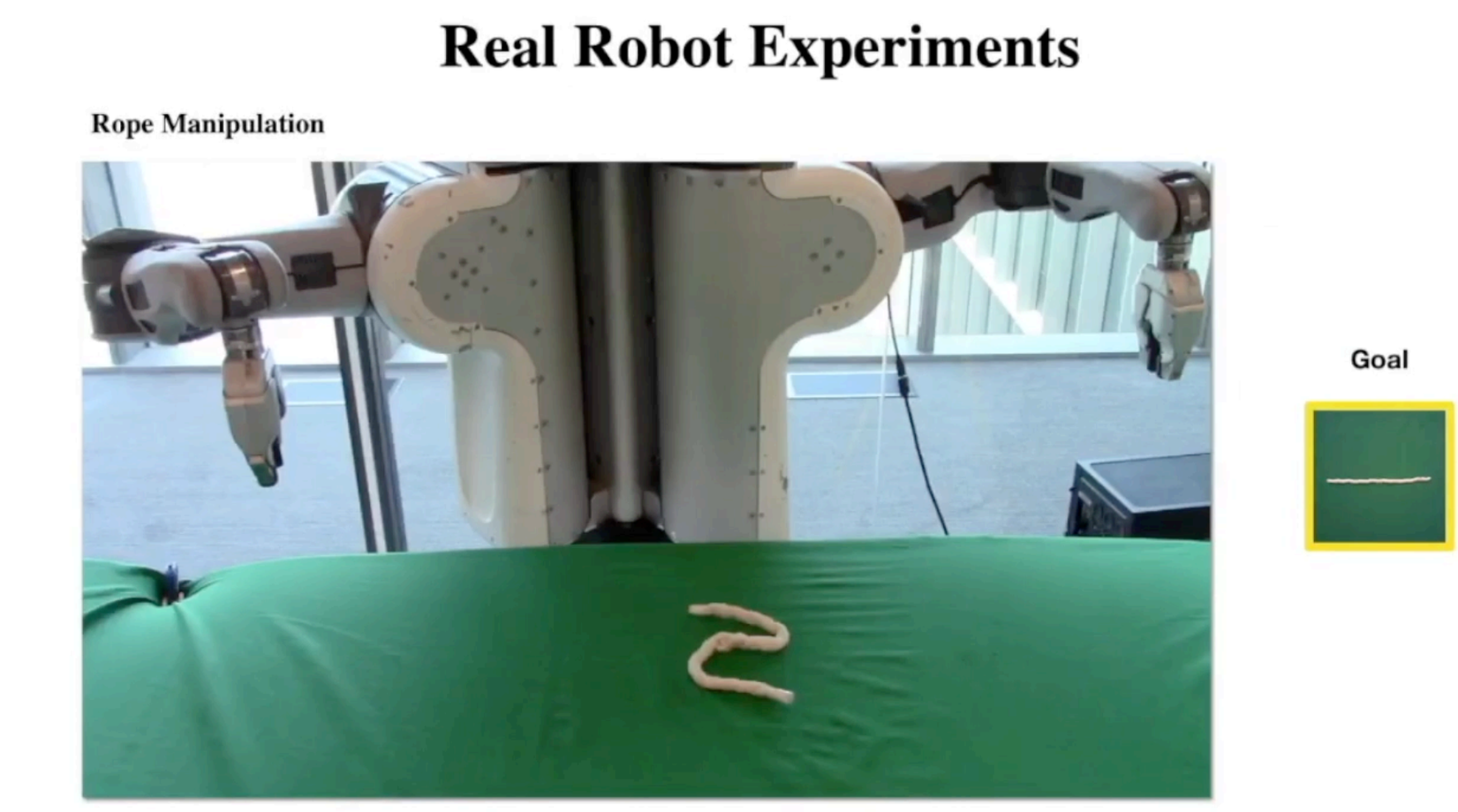
$$P(d = i \mid c) = \frac{\exp(c^\top W z^i)}{\sum_j \exp(c^\top W z^j)}$$



$$\min_{\theta,\phi,\phi',W} . - \log \frac{\exp(\hat{z}_{t+1}{}^\top W z_{t+1})}{\sum_{j \in Neg} \exp(\hat{z}_{t+1}{}^\top W z^j)}$$

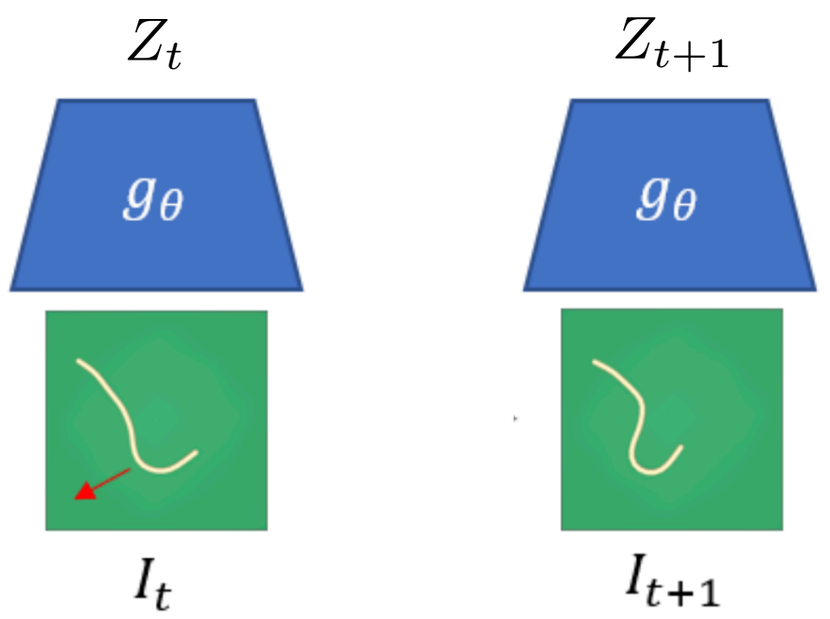
- Q:Since we do not directly predict the future z_{t+1}, how can we unroll this model forward in time?
- A: Through ranking. Consider a set of possibilities and rank them

Contrastive forward models



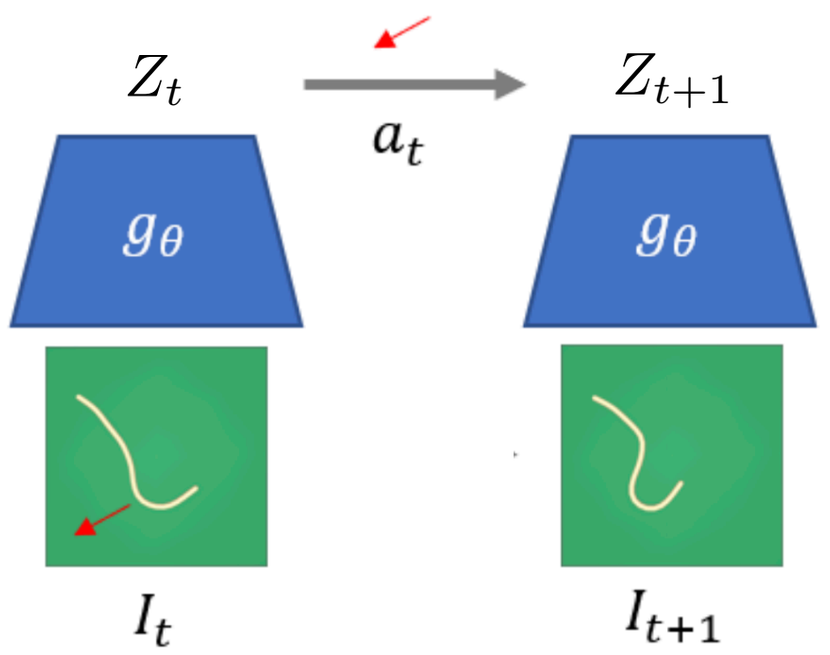
Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation



Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation



Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation

Contrastive loss:

$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}} \left[\log \frac{h(\hat{z}, z_{\text{pos}})}{\sum_{i=1}^{k} h(\hat{z}, z_{\text{neg}})} \right]$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation

Contrastive loss:

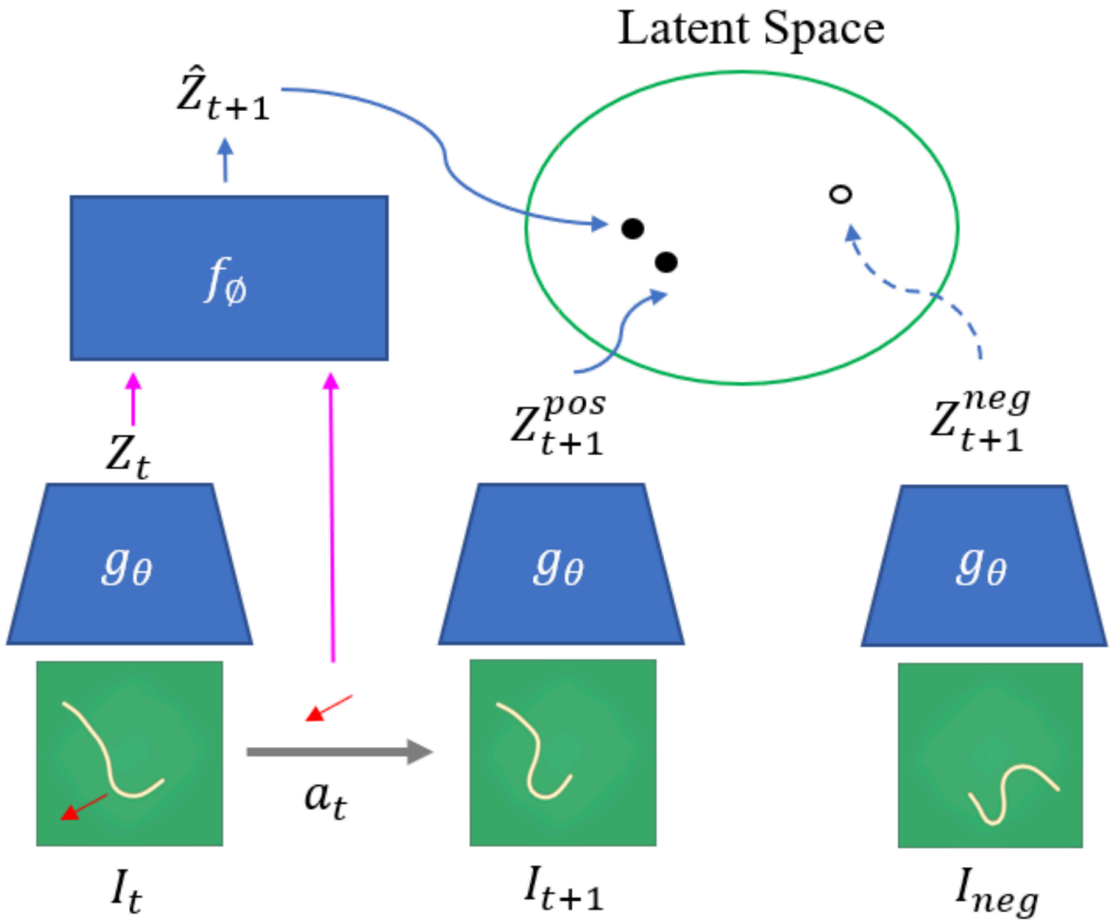
$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}} \left[\log \frac{h(\hat{z}, z_{\text{pos}})}{\sum_{i=1}^{k} h(\hat{z}, z_{\text{neg}})} \right]$$

Similarity function:

$$h(z_1, z_2) = \exp(z_1^T z_2)$$

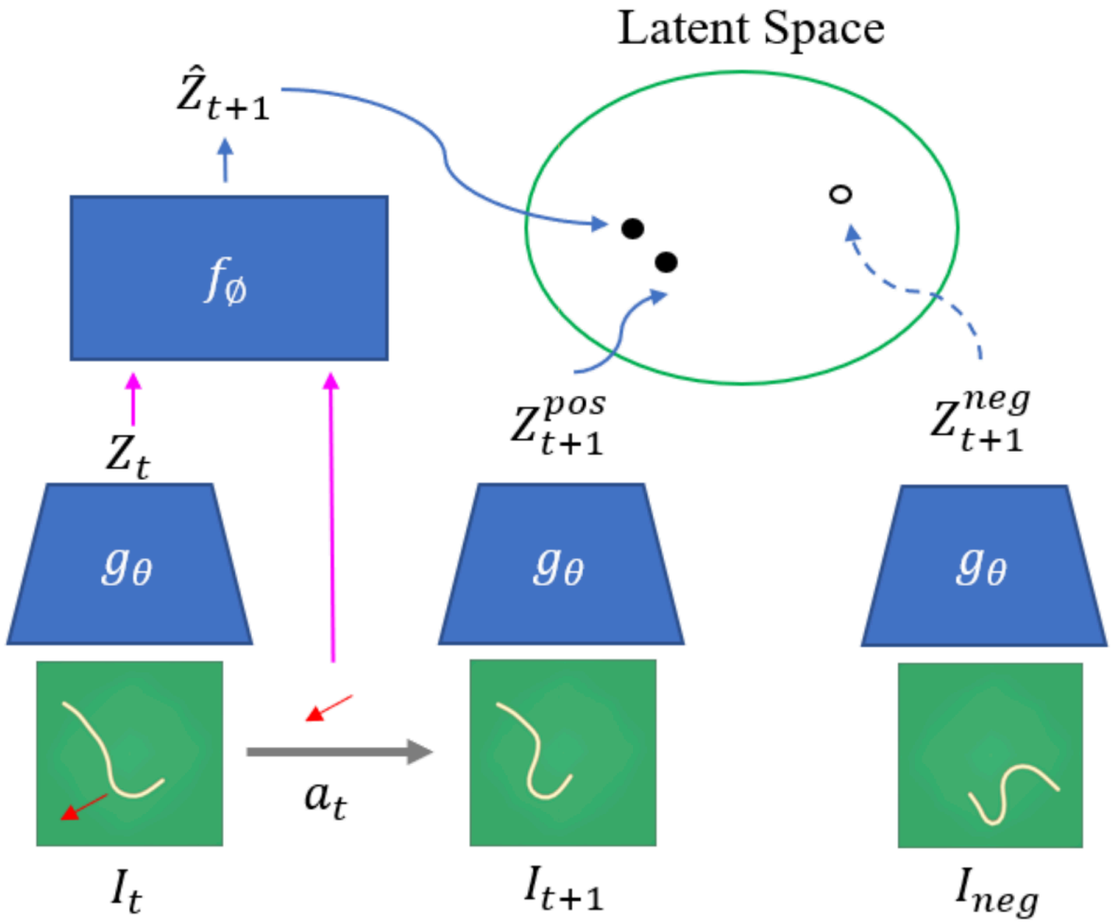
Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation



Learning Predictive Representations for Deformable Objects using Contrastive Estimation

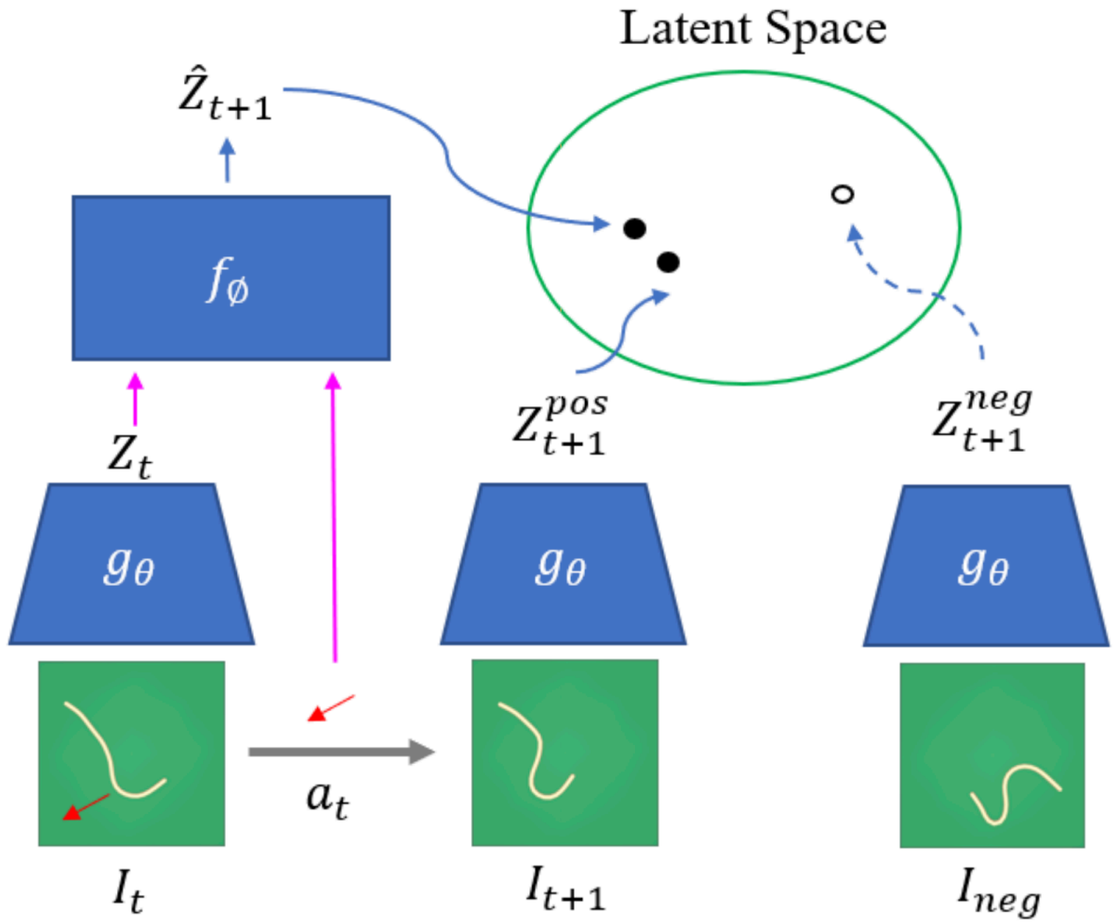
Contrastive forward models for deformable object manipulation



$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}_{t+1}, z_{t+1})}{\sum_{i=1}^{k} h(\hat{z}_{t+1}, \tilde{z}_i)}\right]$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation



$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}_{t+1}, z_{t+1})}{\sum_{i=1}^{k} h(\hat{z}_{t+1}, \tilde{z}_i)}\right]$$

$$h(z_1, z_2) = \exp(-\|z_1 - z_2\|^2)$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

One step Model-Predictive Control

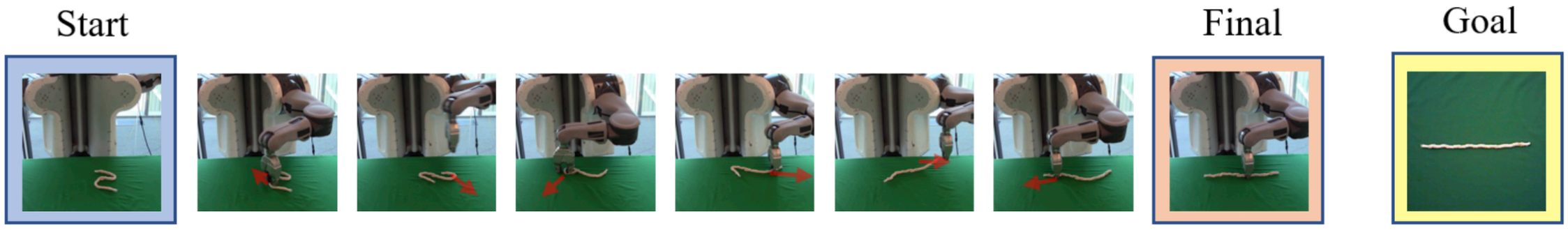
Start

Goal

$$a_t = \max\, h(f_\phi(z_t, a), z_g)$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

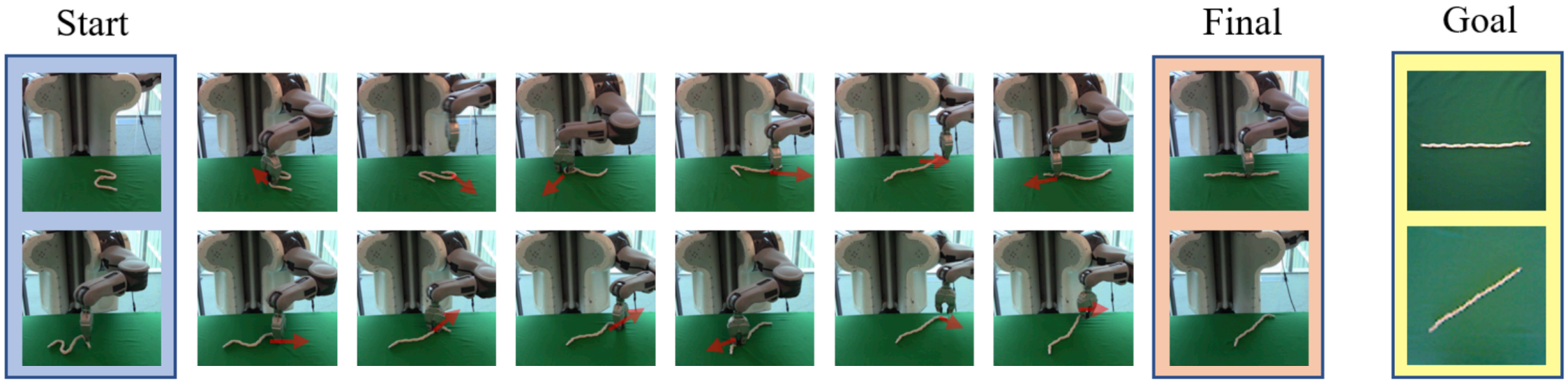
One step Model-Predictive Control



$$a_t = \max\, h(f_\phi(z_t, a), z_g)$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

One step Model-Predictive Control

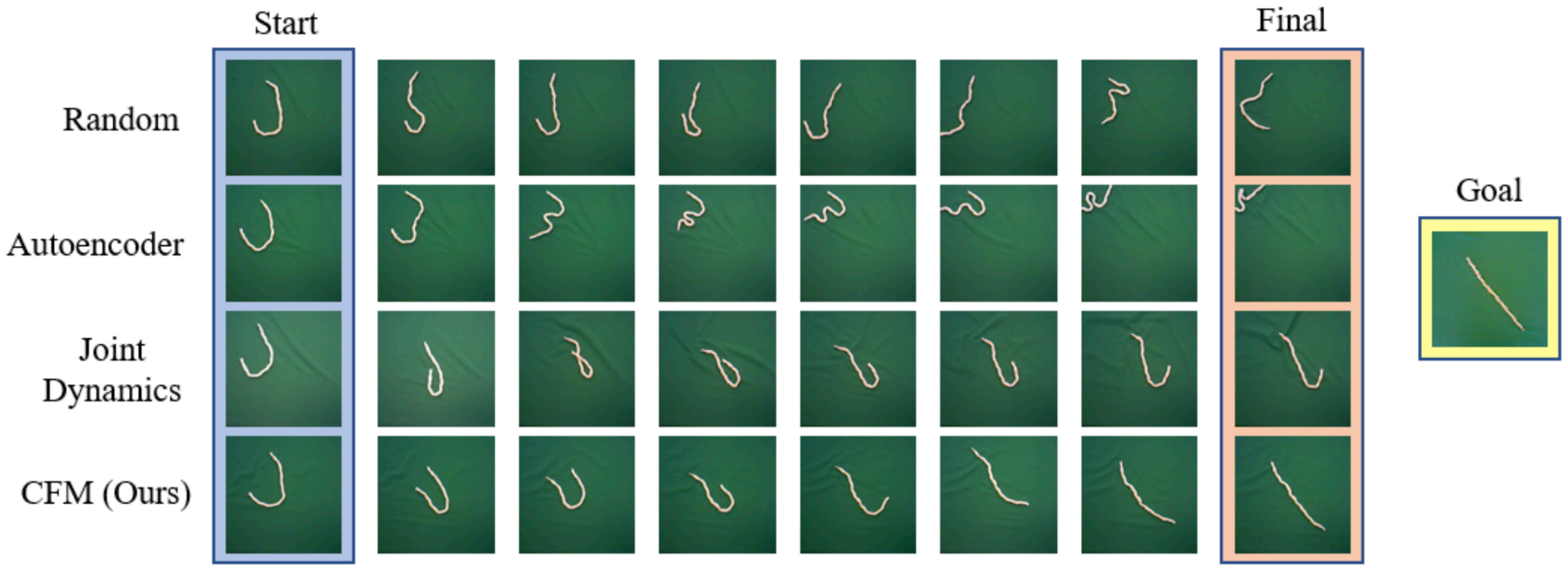


Learning Predictive Representations for Deformable Objects using Contrastive Estimation

One step Model-Predictive Control

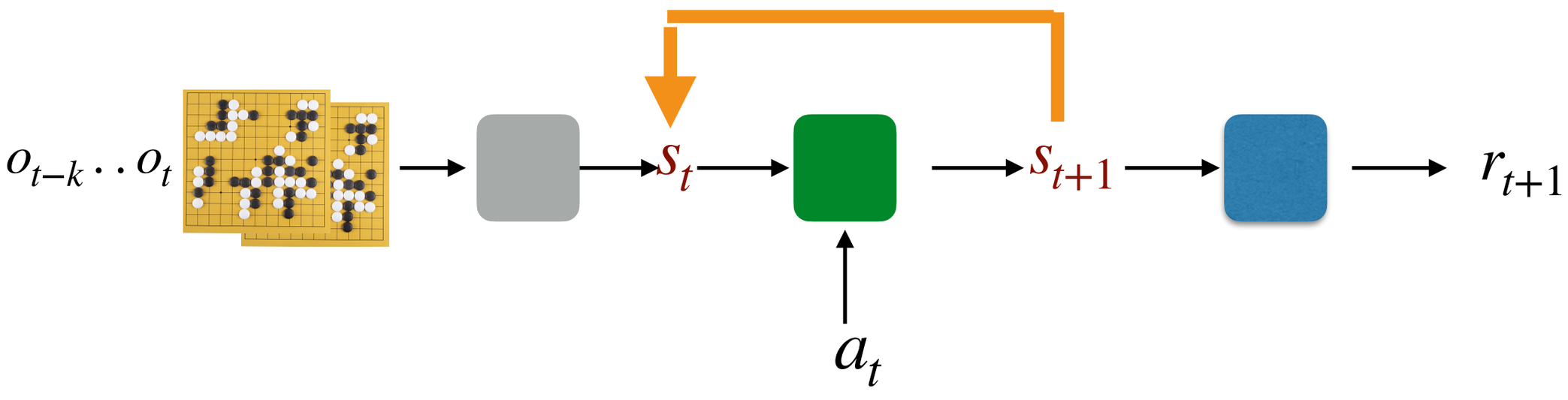
Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Qualitative evaluation



Unrolling in a latent state space

We can predict rewards and observations from the state embeddings.



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

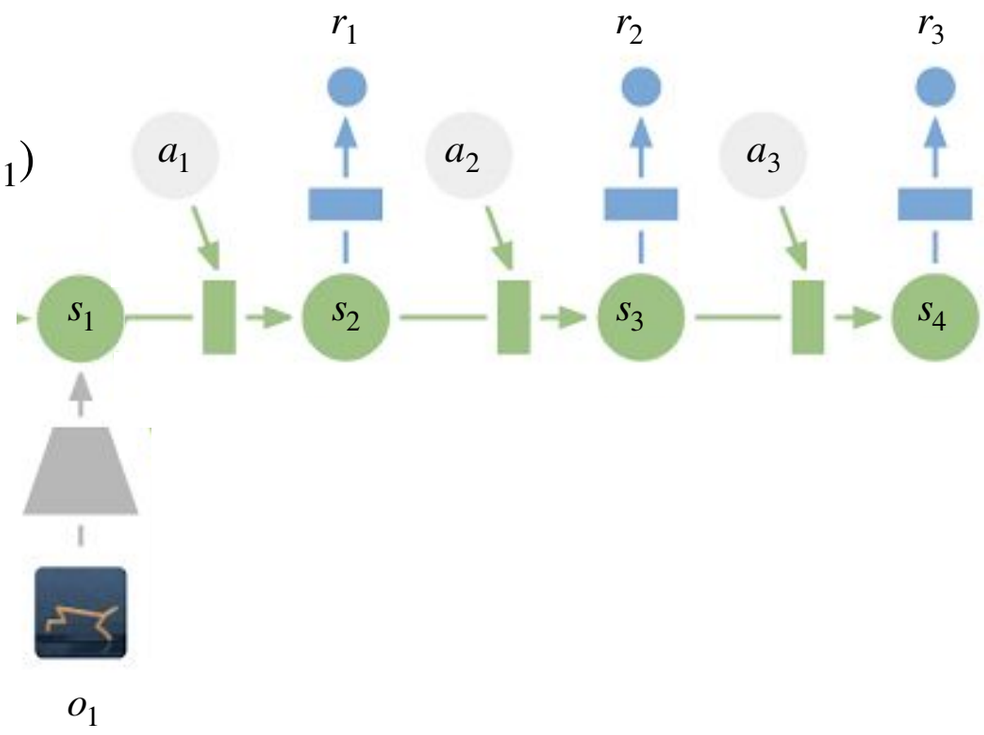
Stochastic transition model: $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$

Unrolling in a latent state space

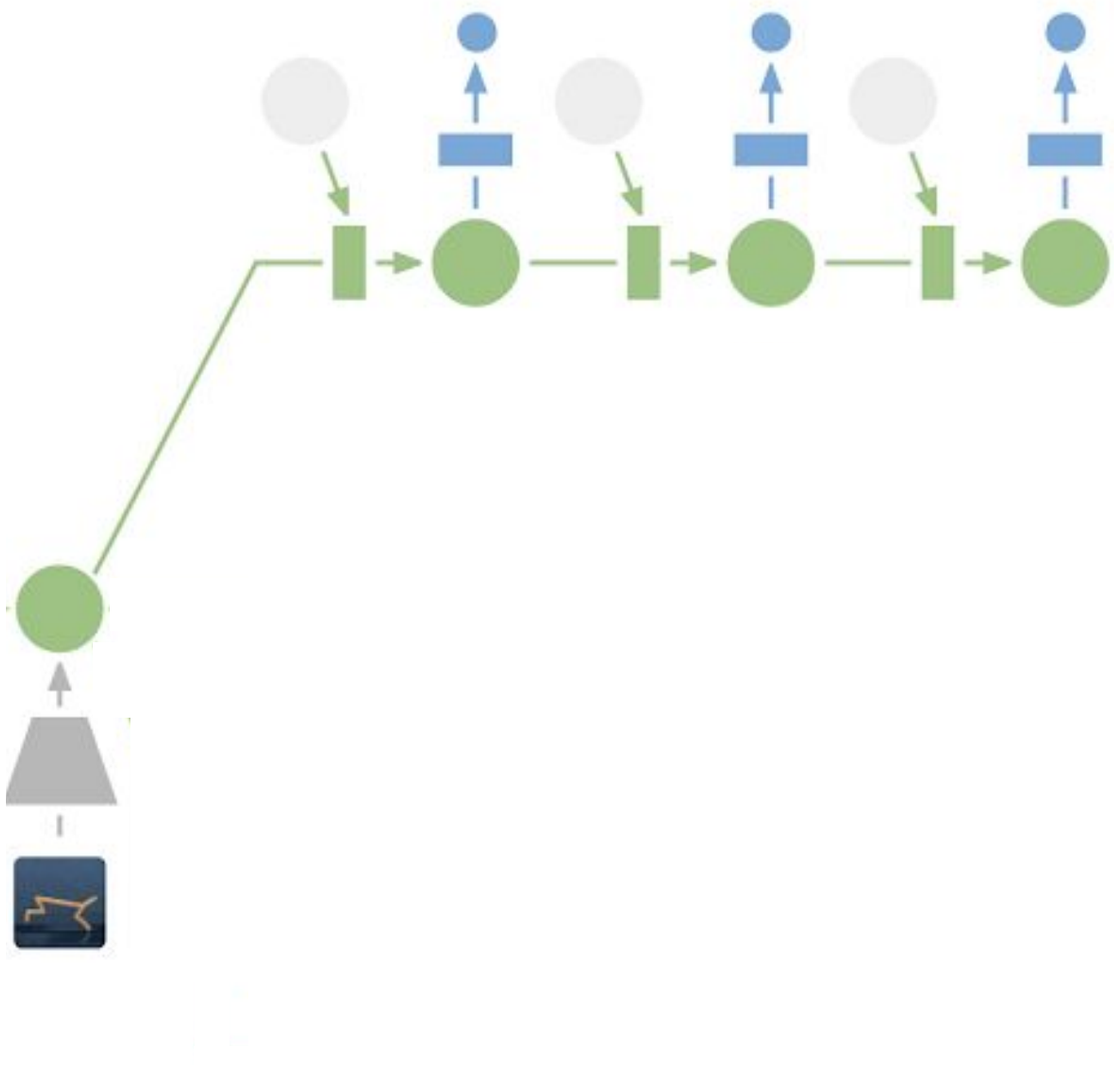
Representation model $s_t \sim q(s_t \mid s_{t-1}, a_{t-1}, o_t)$

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

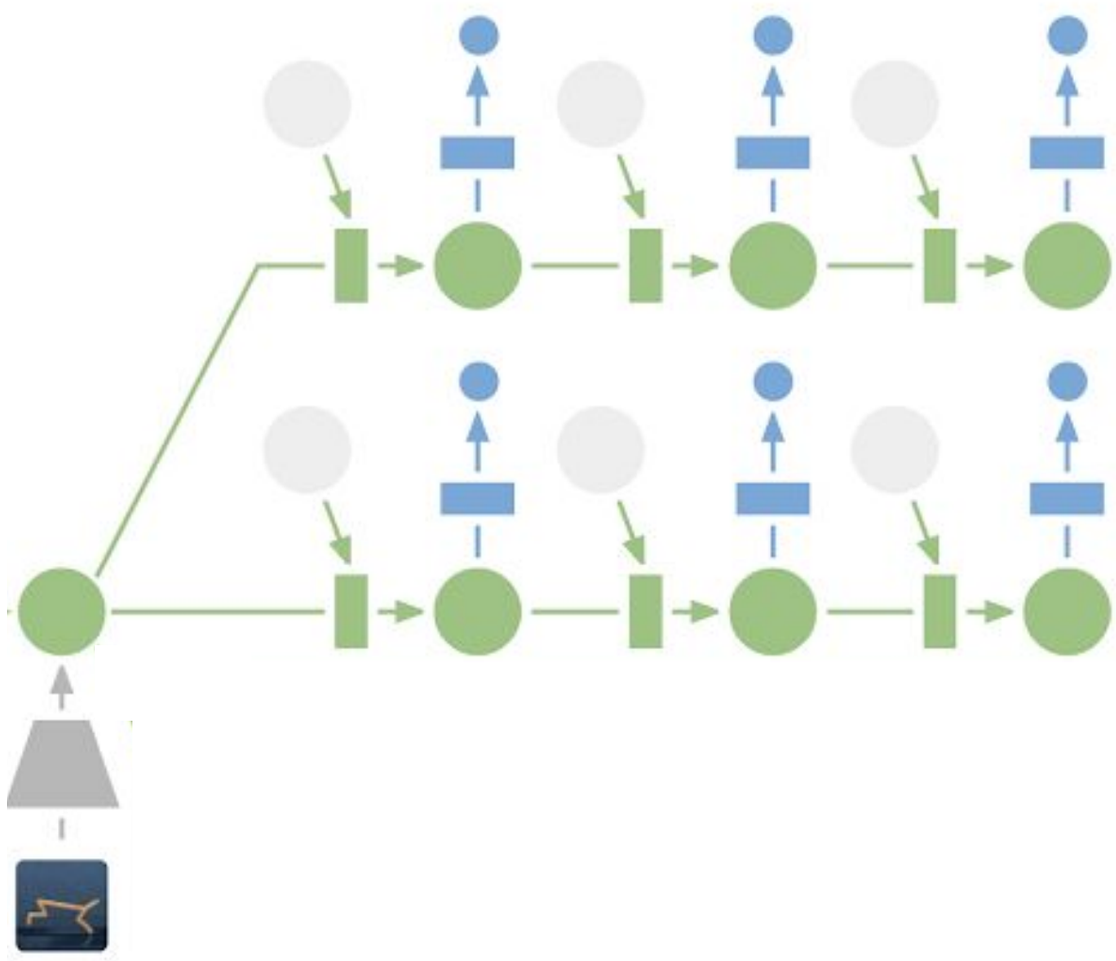
Reward model $r_t \sim p(r_t \mid s_t)$



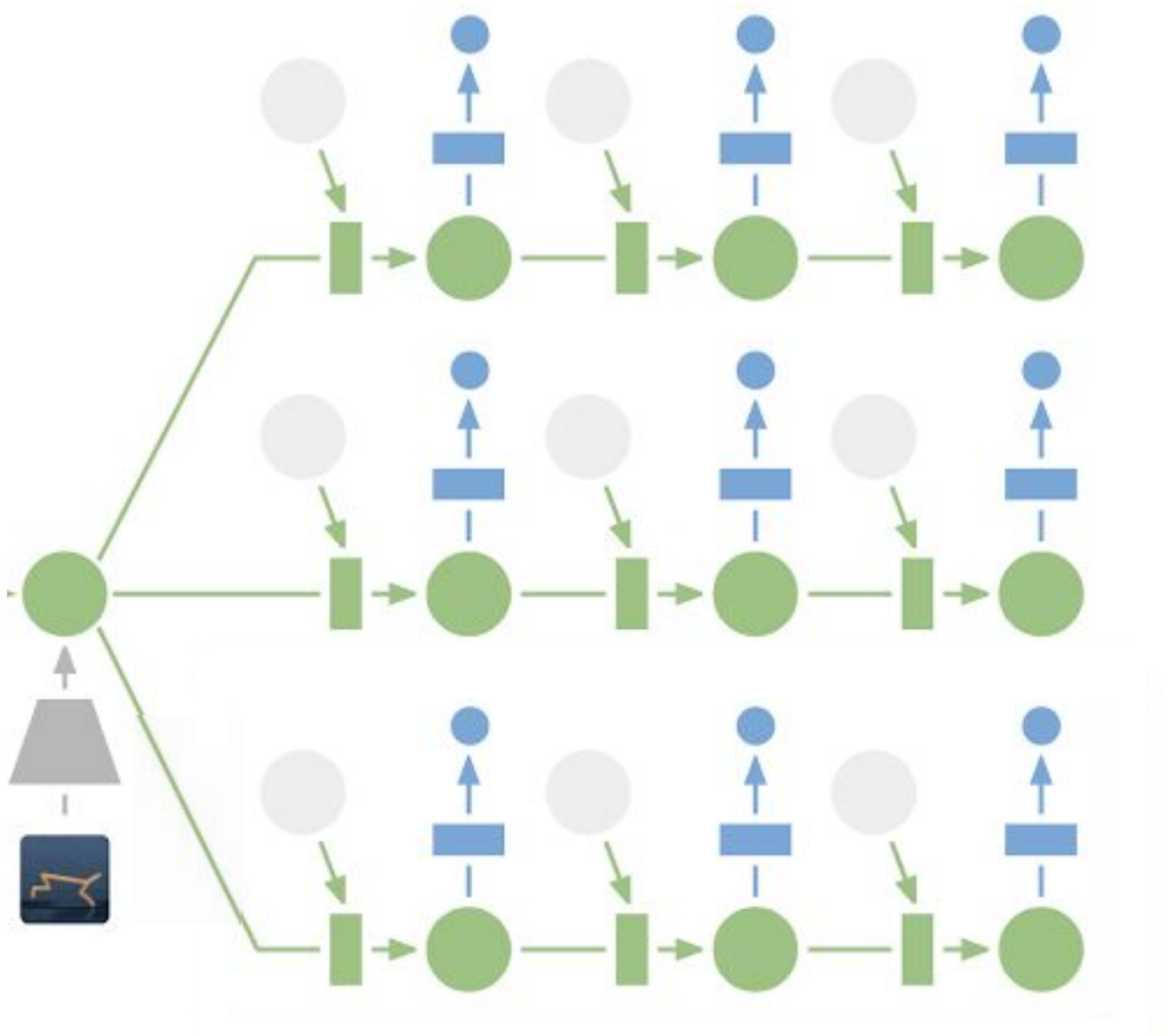
Planning in a latent space with ES



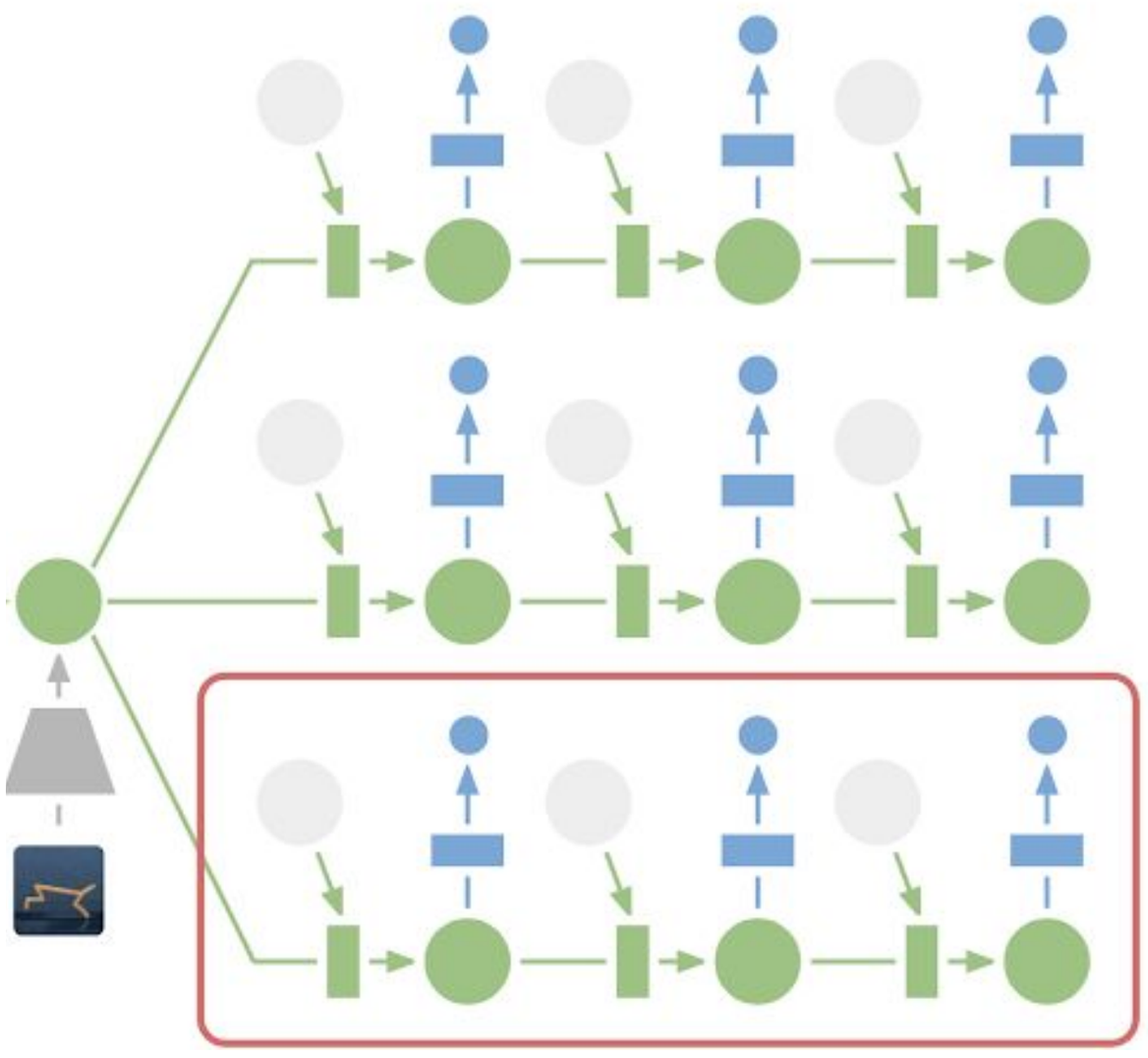
Planning in a latent space with ES



Planning in a latent space with ES

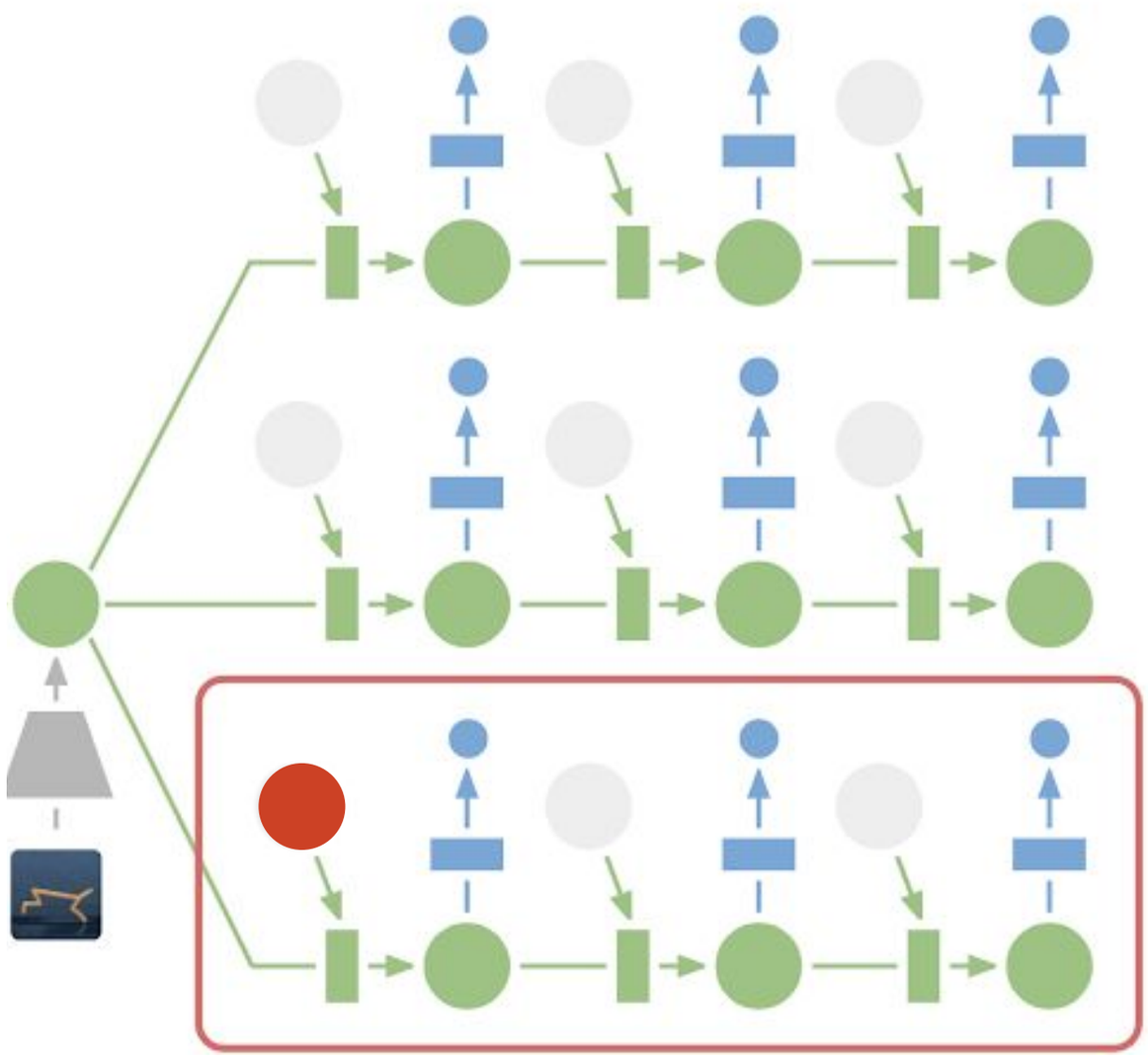


Planning in a latent space with ES



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

Planning in a latent space with ES



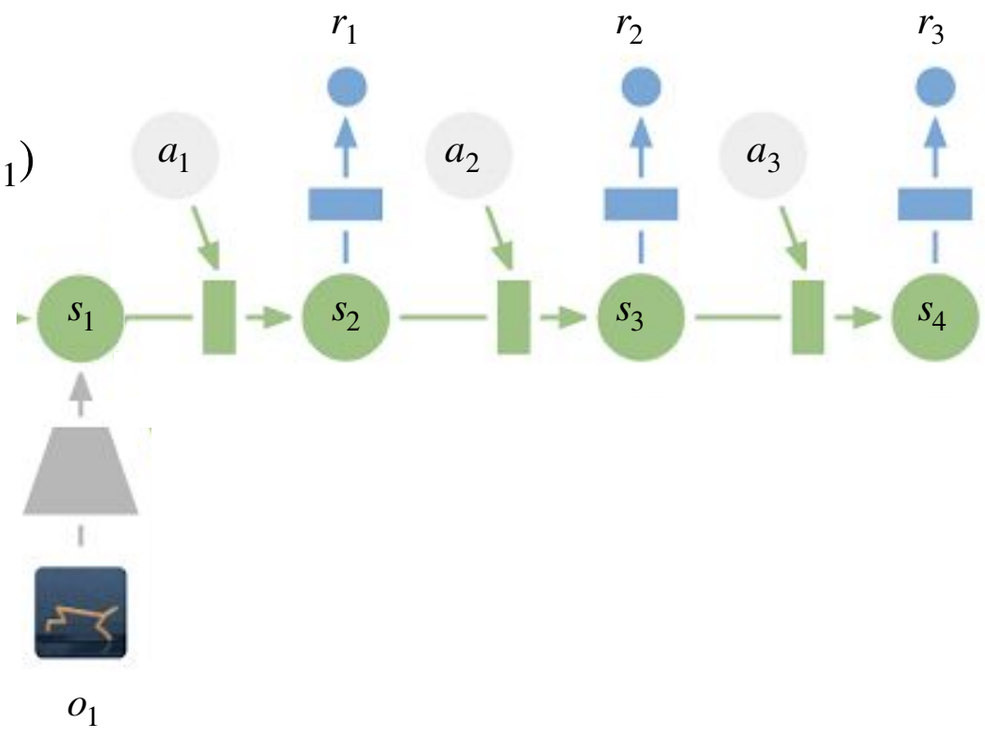
Execute the first inferred action, and repeat.

Planning in a latent space with backprop through time

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

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

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



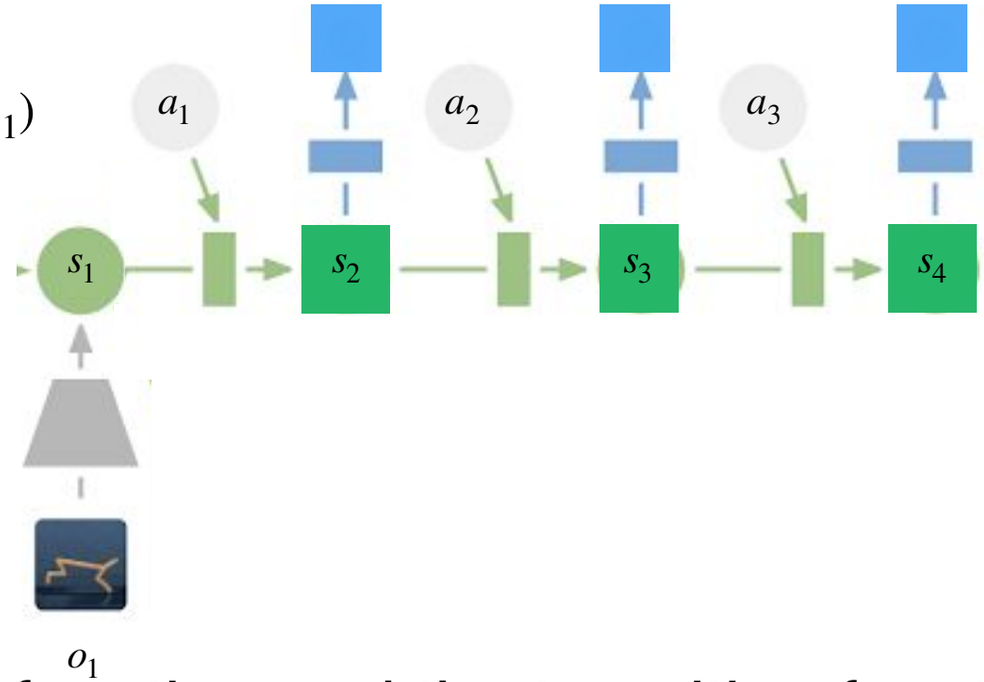
Since we know reward function and the transition function, we can back propagate all the way to actions.

Planning in a latent space with backprop through time

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

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

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



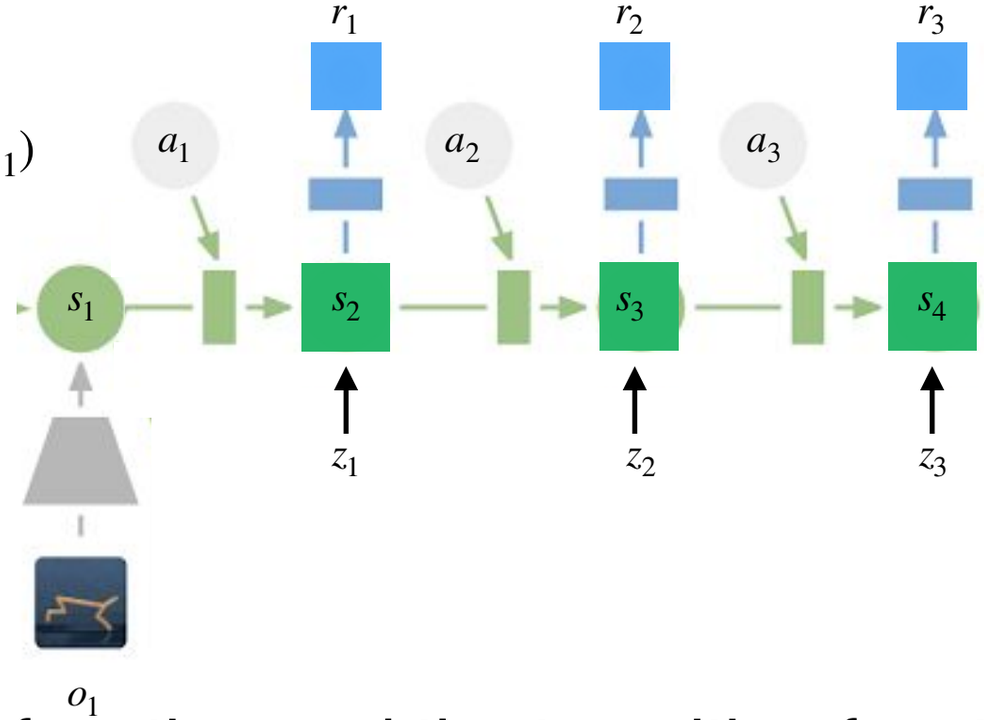
Since we know reward function and the transition function, we can back propagate all the way to actions.
When transitions and rewards are deterministic functions-> standard chain rule through time

Planning in a latent space with backprop through time

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

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

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

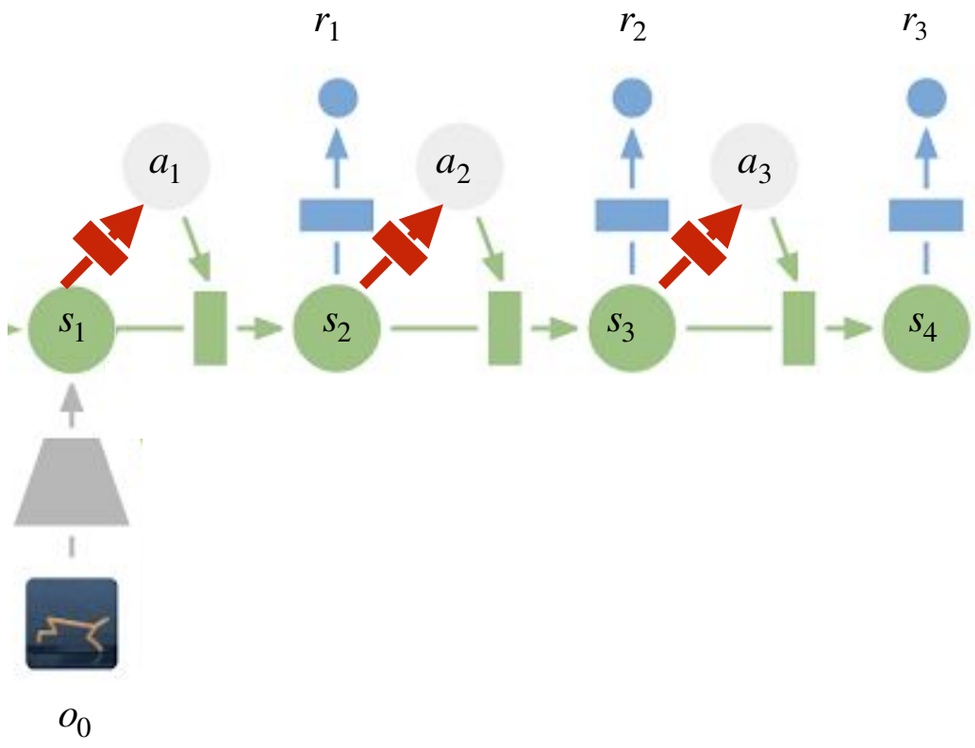


Since we know reward function and the transition function, we can back propagate all the way to actions.
When transitions and rewards are stochastic functions-> we use the re-parametrization trick.

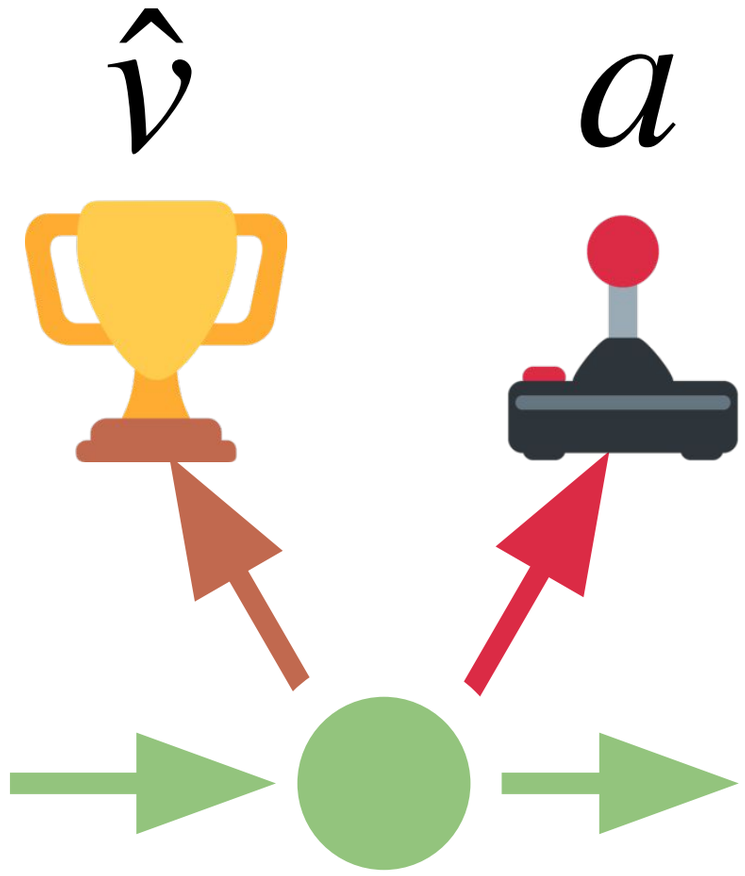
Planning in a latent space for policy learning

We make actions a functions of the state

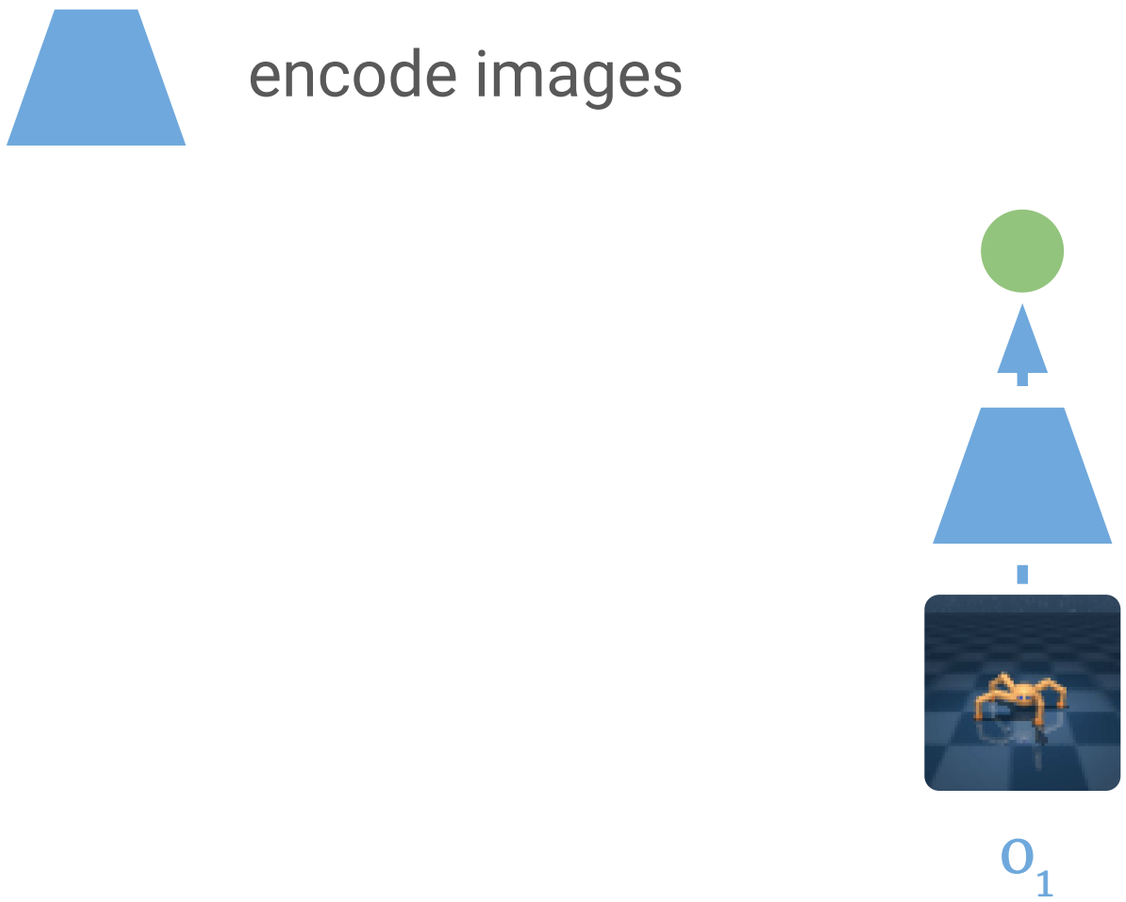
We optimize for policy parameters with ES or backpropagation.



Predict the state values from the state



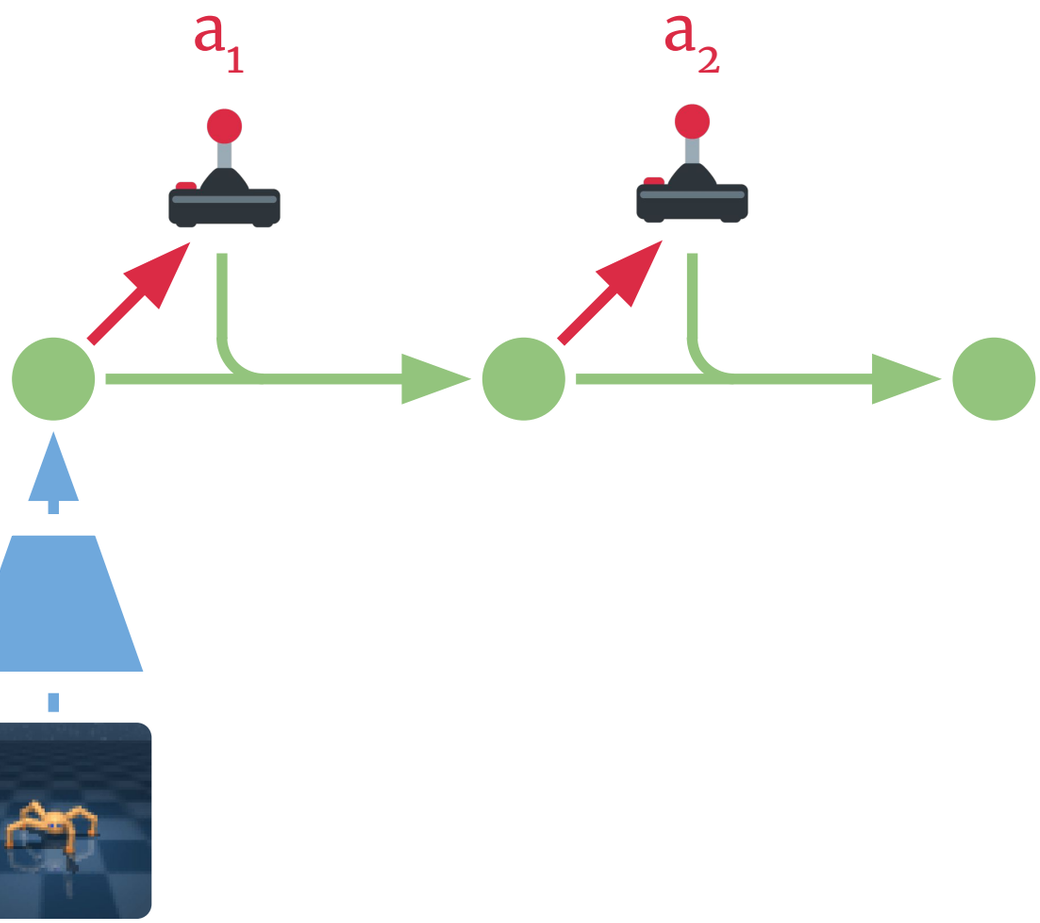
Planning in a latent space for policy learning



Planning in a latent space for policy learning

encode images

imagine ahead

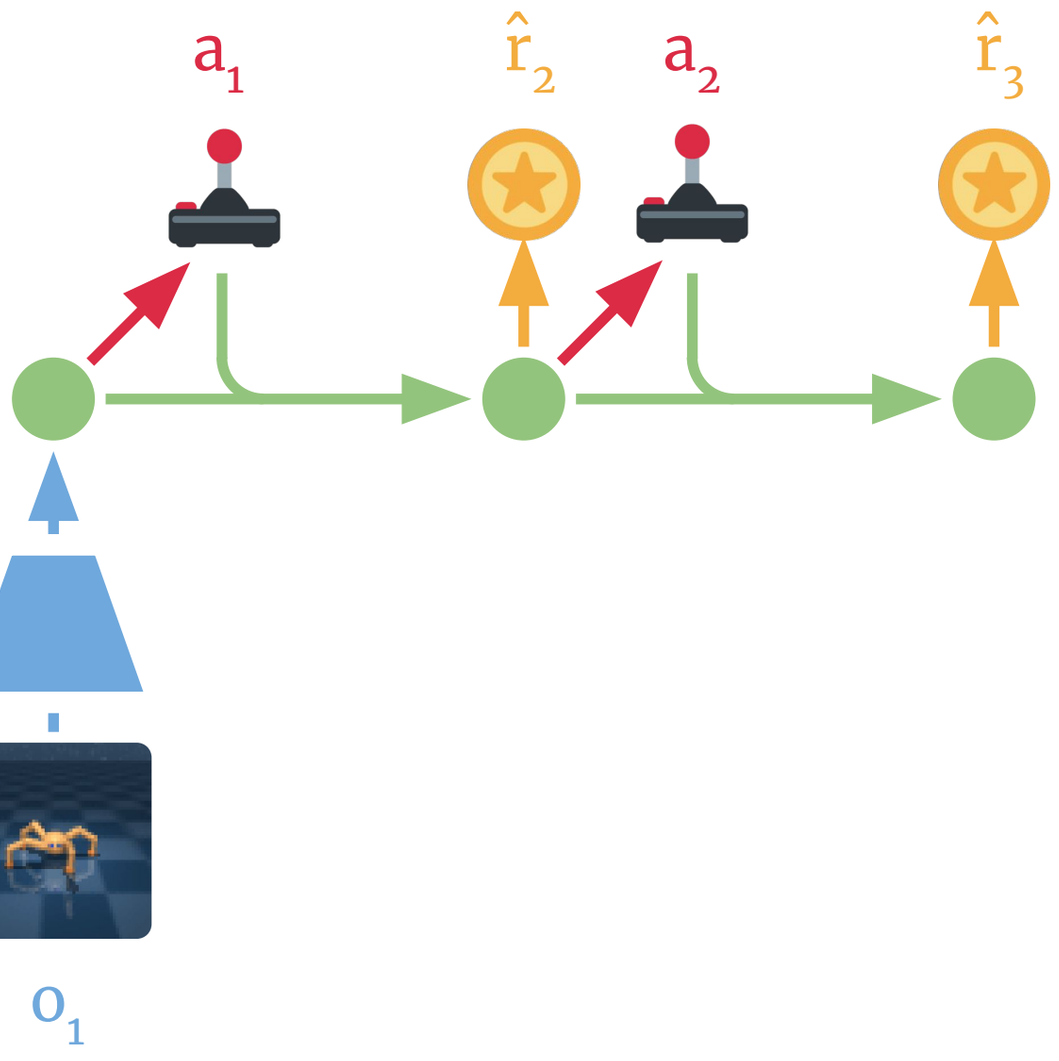


Planning in a latent space for policy learning

encode images

imagine ahead

predict rewards



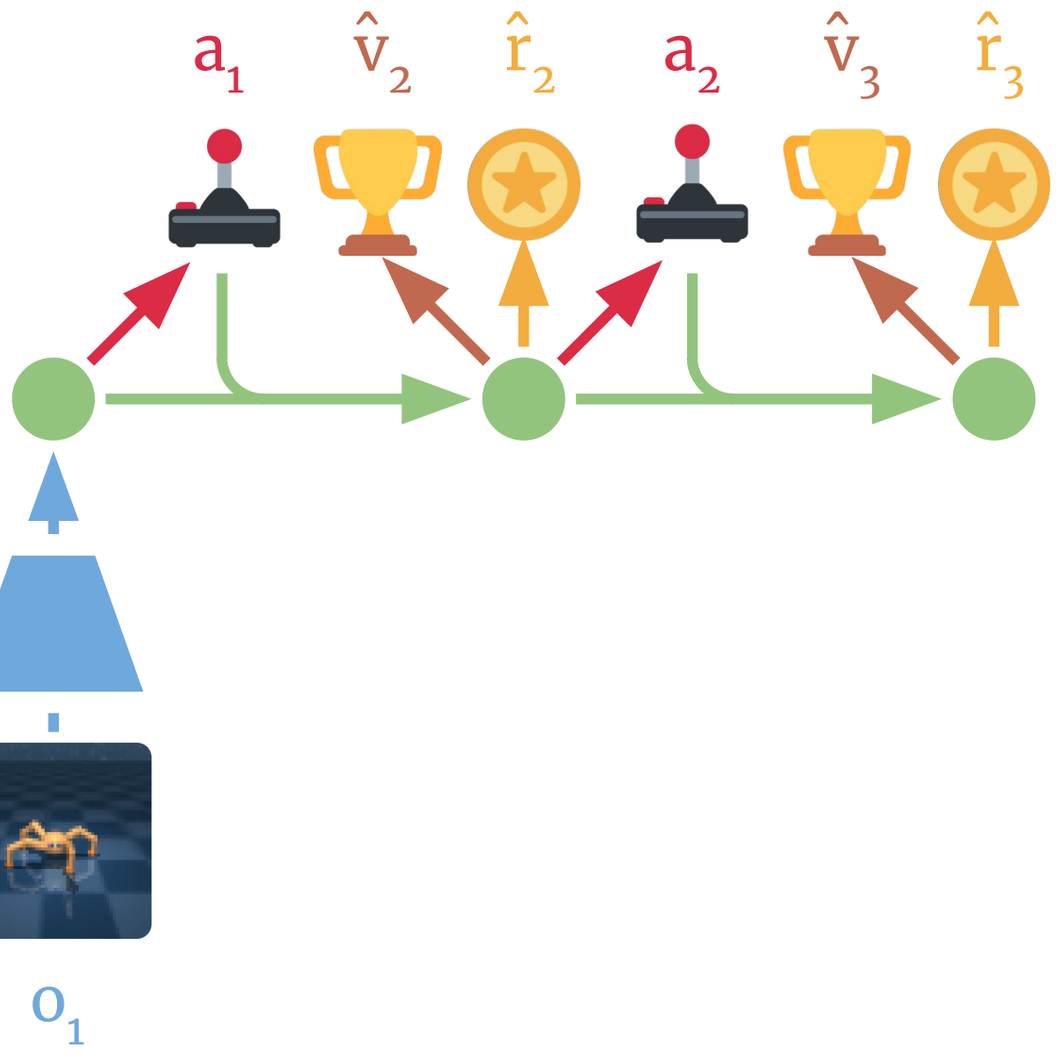
Planning in a latent space for policy learning

encode images

imagine ahead

predict rewards

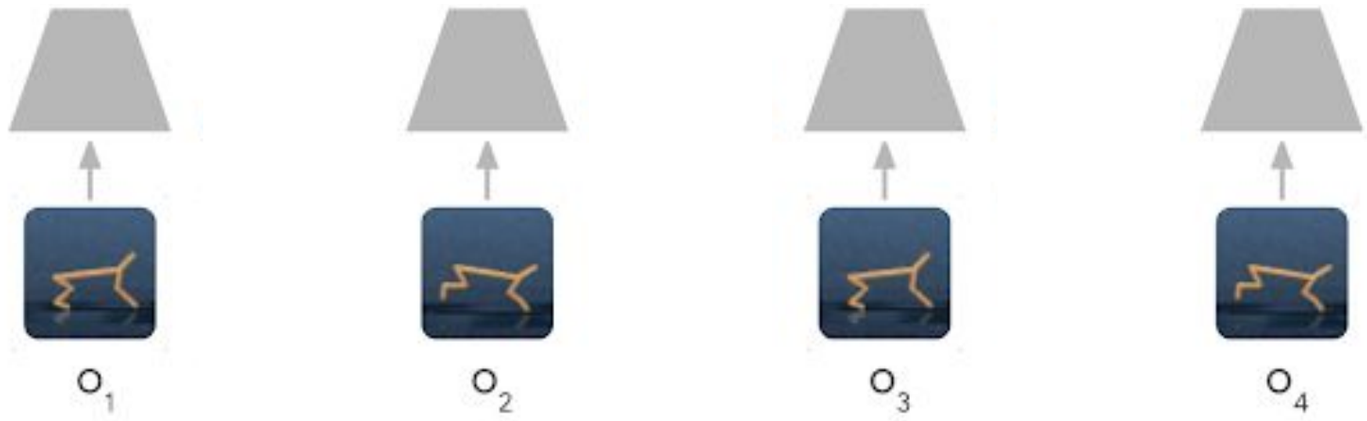
predict values



How can we learn such model from experience?

Latent Dynamics Model

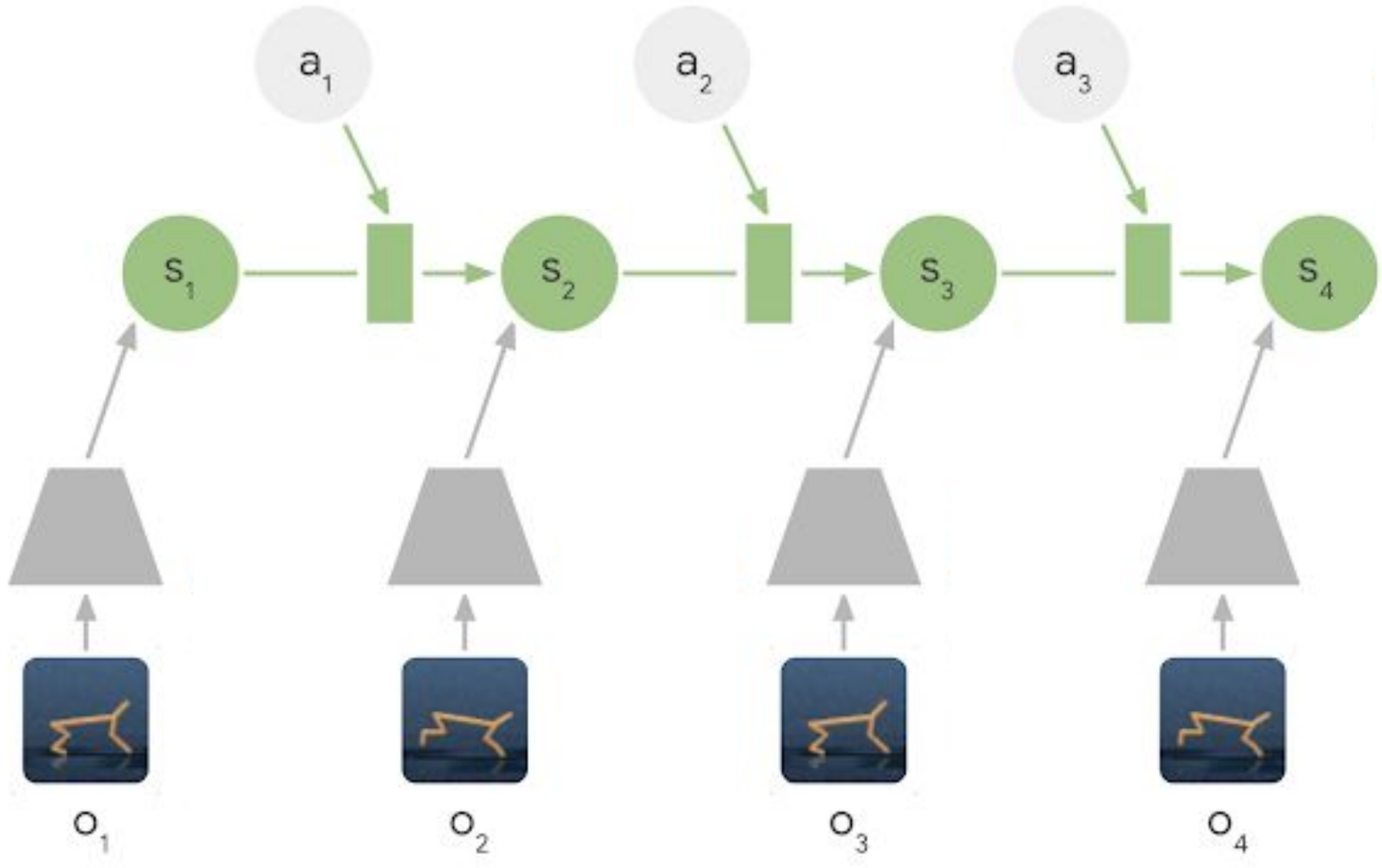
encode images



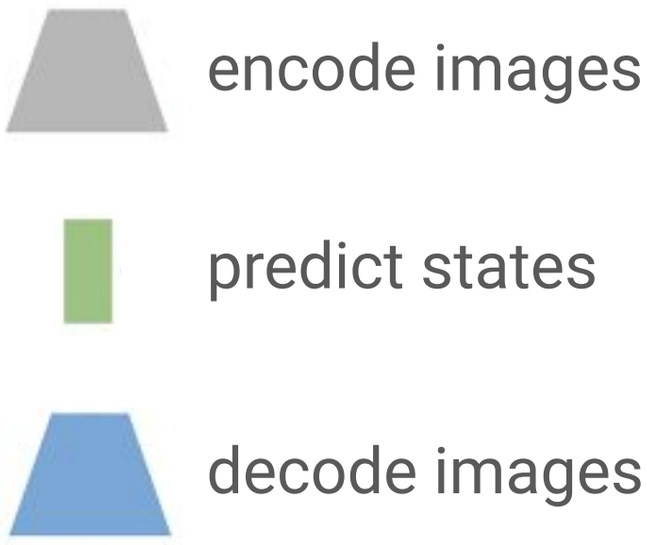
Latent Dynamics Model

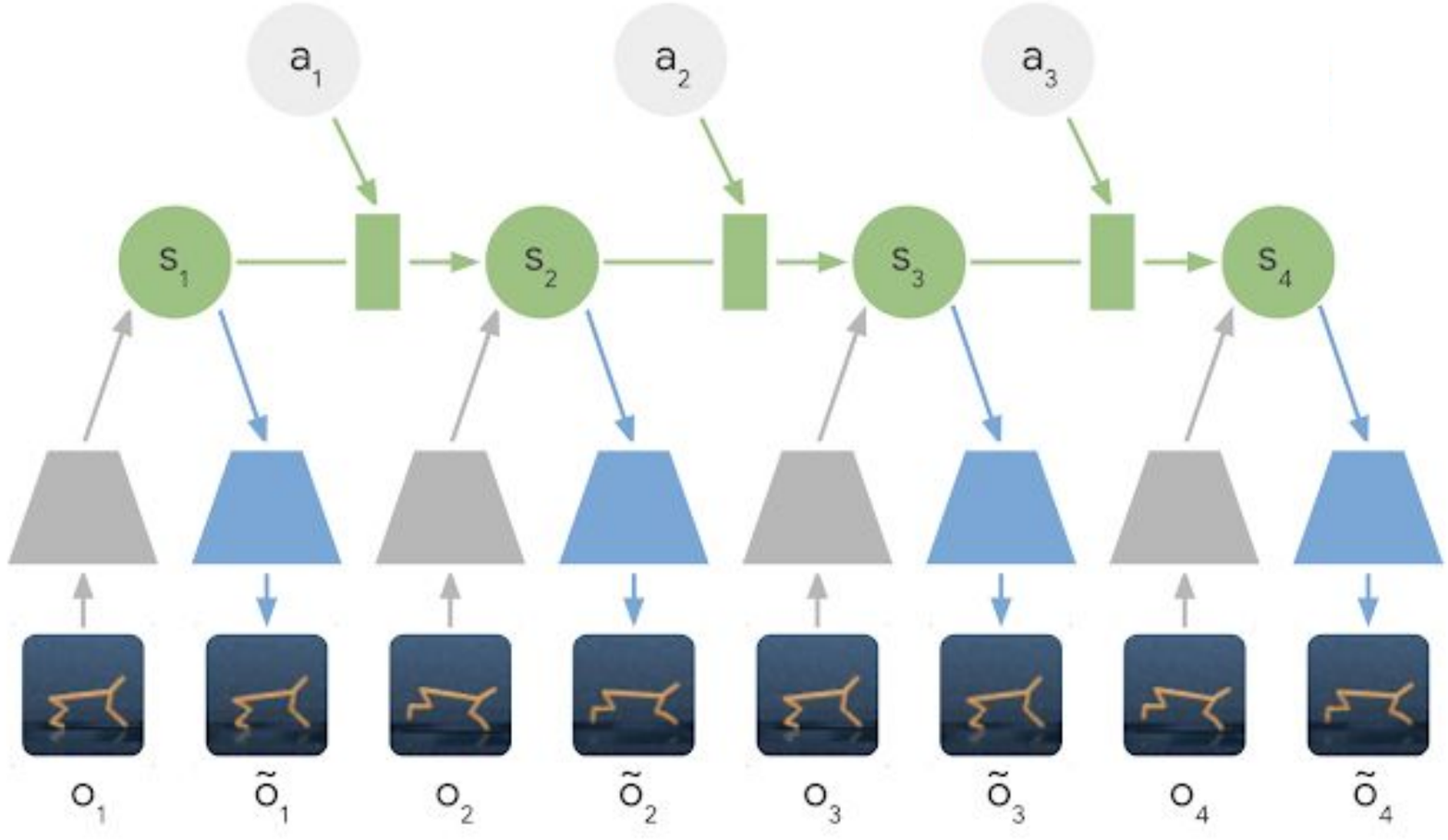
encode images

predict states



Latent Dynamics Model





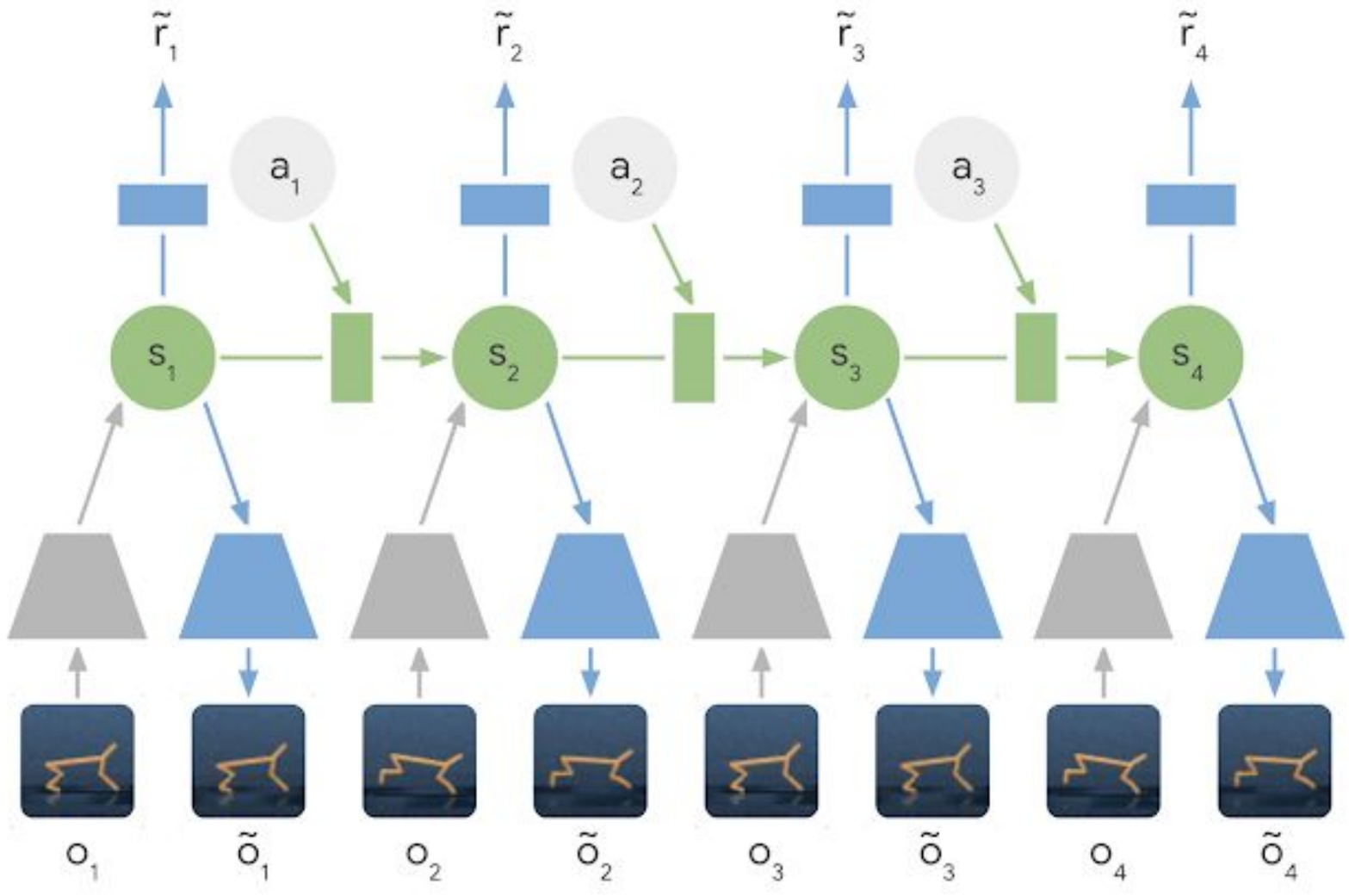
Latent Dynamics Model

encode images

predict states

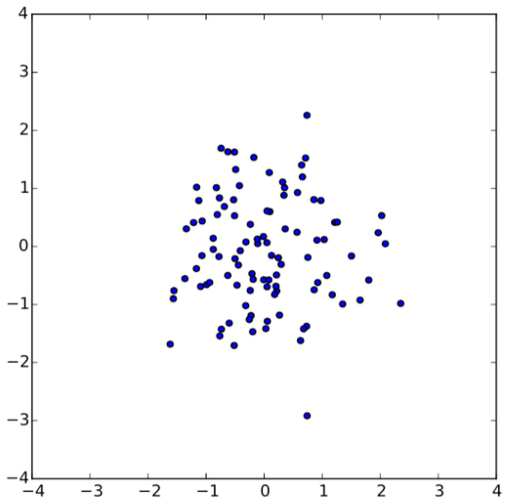
decode images

decode rewards

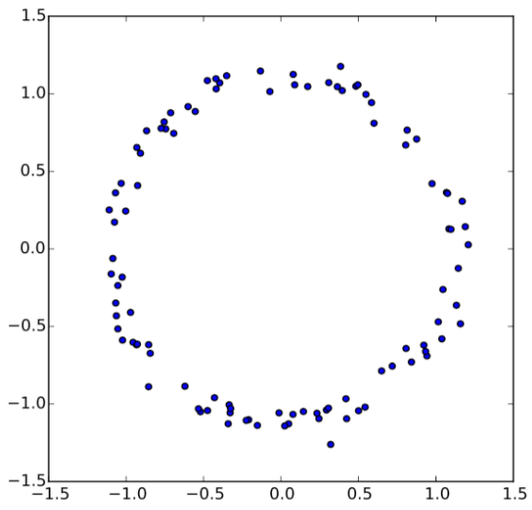


Learning stochastic generative models

- 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!

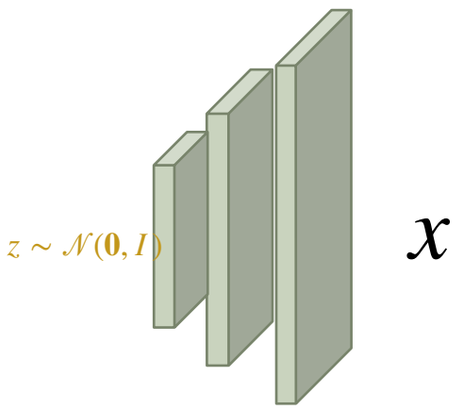


$$z \sim \mathcal{N}(\mathbf{0}, I)$$



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

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:

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

Let's maximize data likelihood:

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

Deep Variational Inference

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

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} \log \frac{p(x \mid z) p(z)}{q(z \mid x)}$$

$$= \mathbb{E}_{z \sim q(z|x)} \log p(x \mid z) + \mathbb{E}_{z \sim q(z|x)} \log \frac{p(z)}{q(z \mid x)}$$

$$= \underbrace{\mathbb{E}_{z \sim q(z|x)} \log p(x \mid z)}_{\text{reconstuction}} - \underbrace{D_{KL}(q(z \mid x), p(z))}_{\text{complexity}}$$

Deep Variational Inference

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

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} \log \frac{p(x \mid z) p(z)}{q(z \mid x)}$$

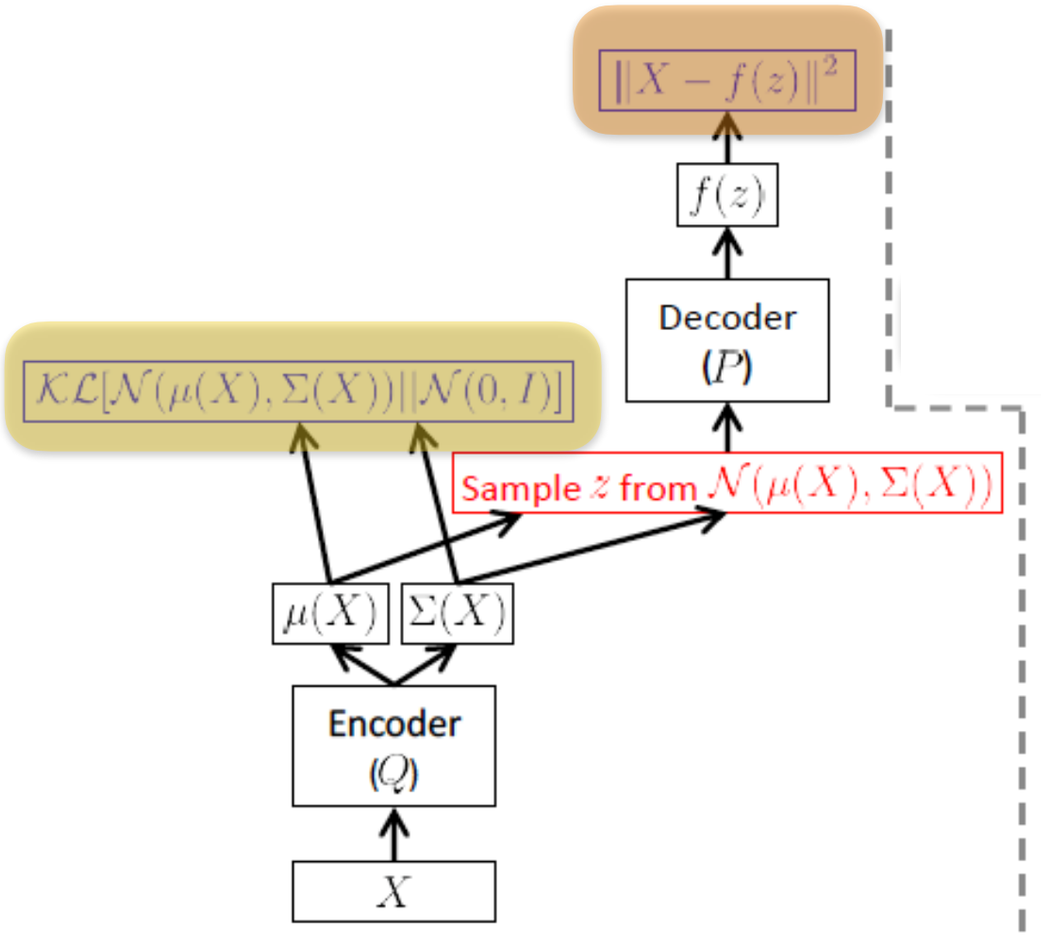
$$= \mathbb{E}_{z \sim q(z|x)} \log p(x \mid z) + \mathbb{E}_{z \sim q(z|x)} \log \frac{p(z)}{q(z \mid x)}$$

$$= \underbrace{\mathbb{E}_{z \sim q(z|x)} \log p(x \mid z)}_{\text{reconstuction}} - \underbrace{D_{KL}(q(z \mid x), p(z))}_{\text{complexity}}$$

decoder

encoder

Variational Autoencoder



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

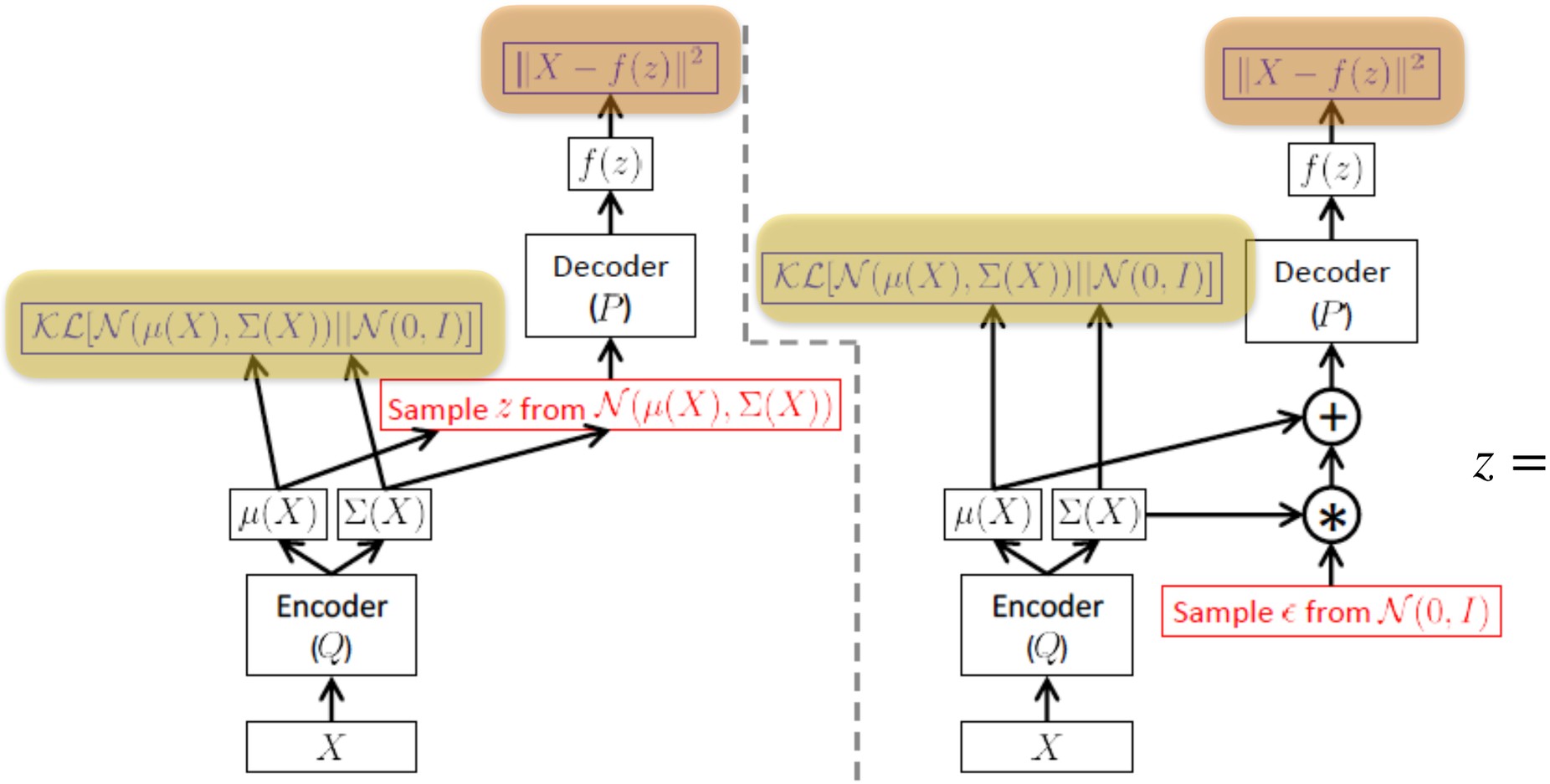
encoder

decoder

T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

From left to right: re-parametrization trick!



$$z = \mu + \epsilon \cdot \sigma$$

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

encoder

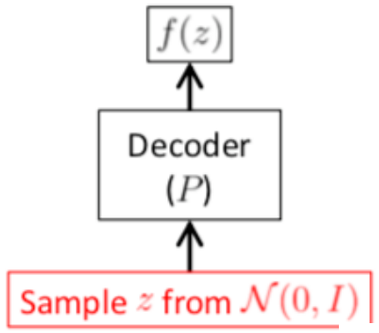
decoder

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

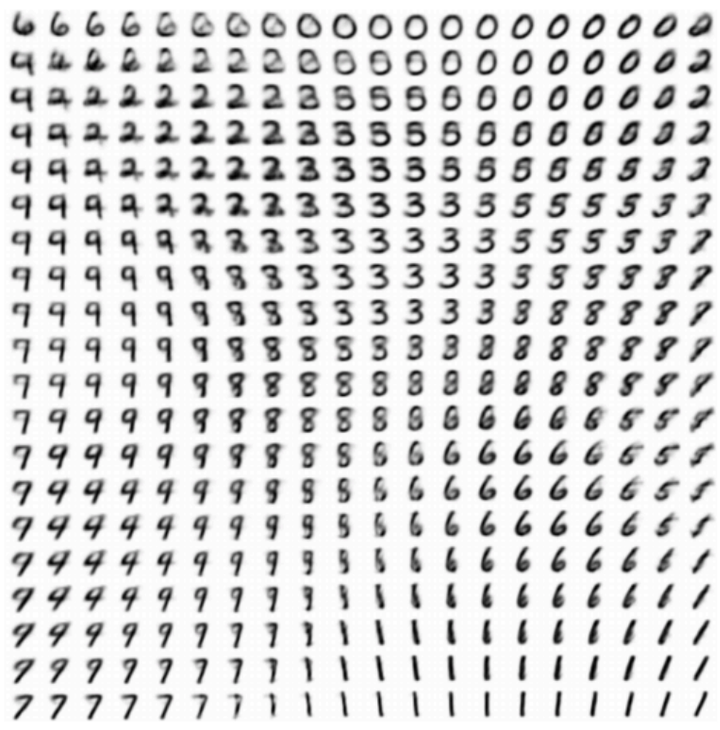
T*utotial on variational Autoencoders,* Doersch

Variational Autoencoder

At test time



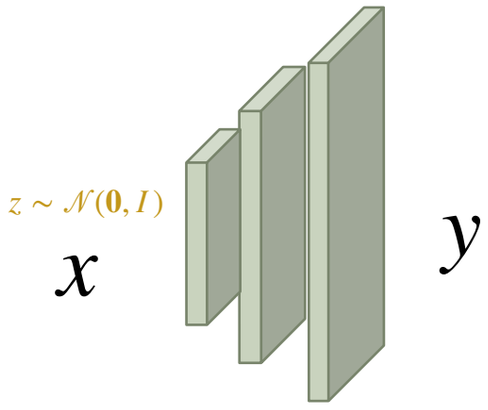
(a) Learned Frey Face manifold

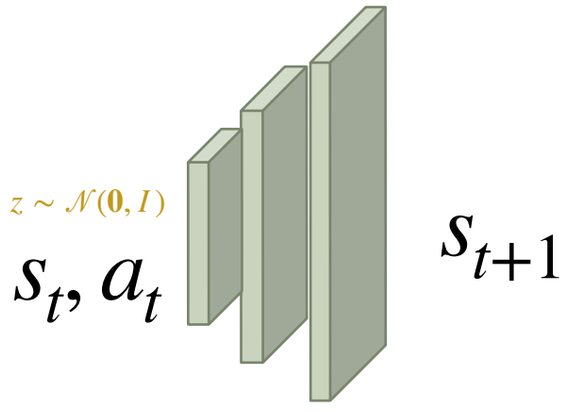


(b) Learned MNIST manifold

Auto-Encoding Variational Bayes, *Kingma and Welling*

Conditional generative models





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

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

Let's maximize data likelihood.

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

Deep Variational Inference

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

$$\log p(y \mid x) \geq \mathbb{E}_{z \sim q(z \mid x, y)} \log \frac{p(y \mid x, z) p(z)}{q(z \mid x, y)}$$

$$= \mathbb{E}_{z \sim q(z \mid x, y)} \log p(y \mid x, z) + \mathbb{E}_{z \sim q(z \mid x, y)} \log \frac{p(z)}{q(z \mid x, y)}$$

$$= \underbrace{\mathbb{E}_{z \sim q(z \mid x, y)} \log p(y \mid x, z)}_{\text{reconstuction}} - \underbrace{D_{KL}(q(z \mid x, y), p(z))}_{\text{complexity}}$$

Deep Variational Inference

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

$$\log p(y\,|\,x) \geq \mathbb{E}_{z\sim q(z|x,y)} \log \frac{p(y\,|\,x,z)p(z)}{q(z\,|\,x,y)}$$

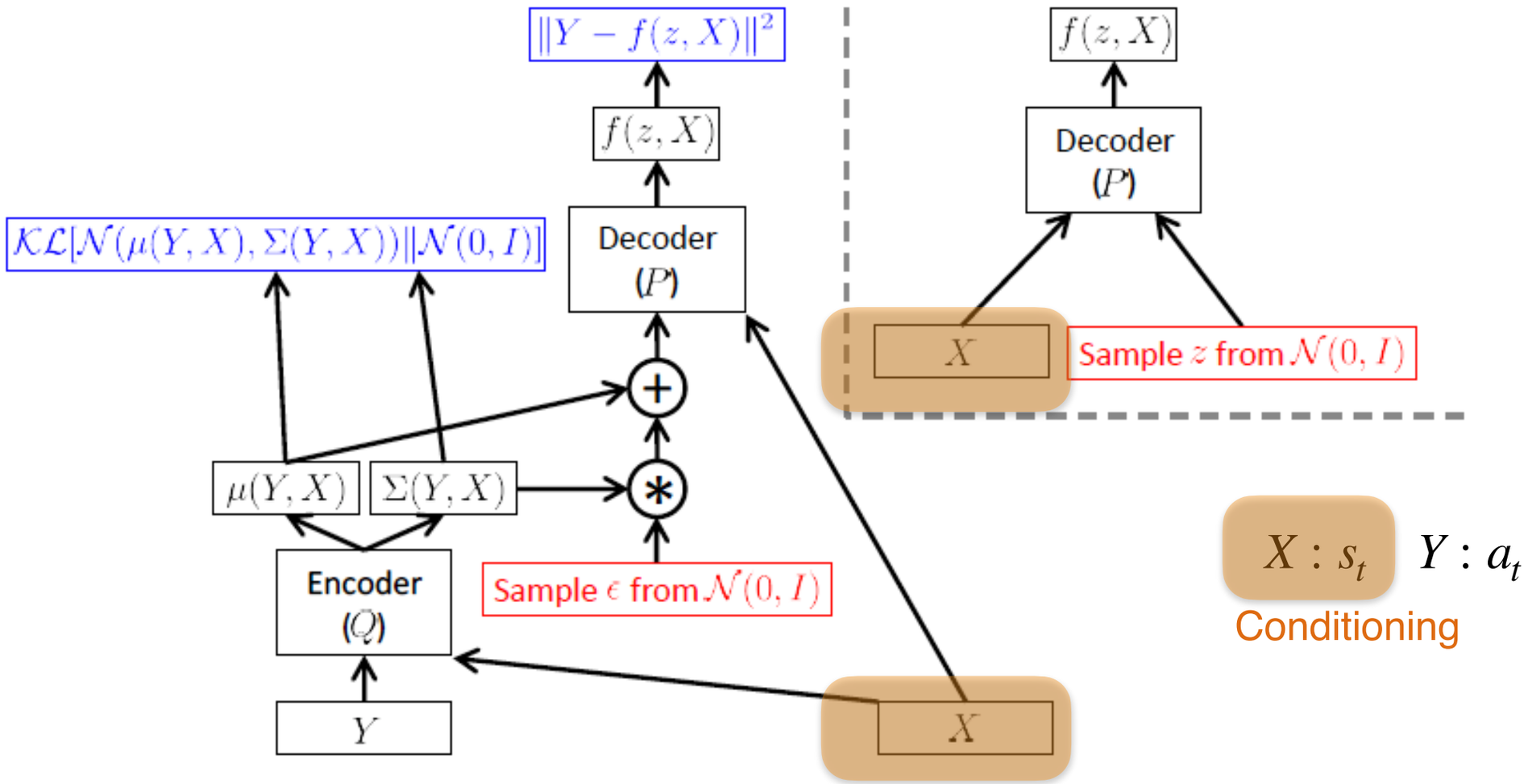
$$= \mathbb{E}_{z\sim q(z|x,y)} \log p(y\,|\,x,z) + \mathbb{E}_{z\sim q(z|x,y)} \log \frac{p(z)}{q(z\,|\,x,y)}$$

$$= \underbrace{\mathbb{E}_{z\sim q(z|x,y)} \log p(y\,|\,x,z)}_{\text{reconstuction}} - \underbrace{D_{KL}(q(z\,|\,x,y), p(z))}_{\text{complexity}}$$

decoder

encode

Conditional VAE



$X : s_t$ $Y : a_t$

Conditioning

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

Tutotial on variational Autoencoders, Doersch

Deep Variational Inference for latent dynamics models

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

Deep Variational Inference for latent dynamics models

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

Imagine s_0 is fixed.

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

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.

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

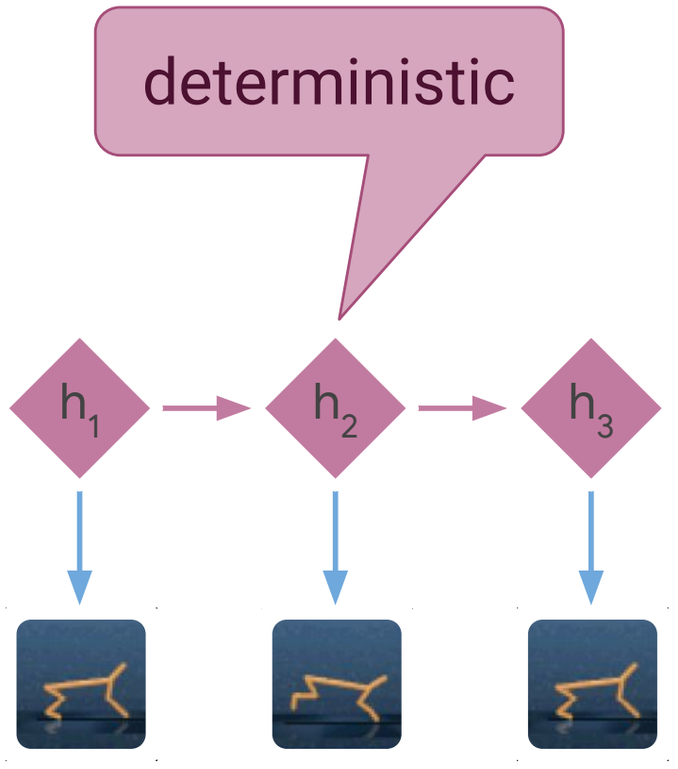
$$\ln p(o_{1:T} \mid a_{1:T}) \triangleq \ln \mathrm{E}_{p(s_{1:T}|a_{1:T})}\left[\prod_{t=1}^{T} p(o_t \mid s_t)\right]$$

$$= \ln \mathrm{E}_{q(s_{1:T}|o_{1:T},a_{1:T})}\left[\prod_{t=1}^{T} p(o_t \mid s_t)p(s_t \mid s_{t-1},a_{t-1})/q(s_t \mid o_{\leq t},a_{<t})\right]$$

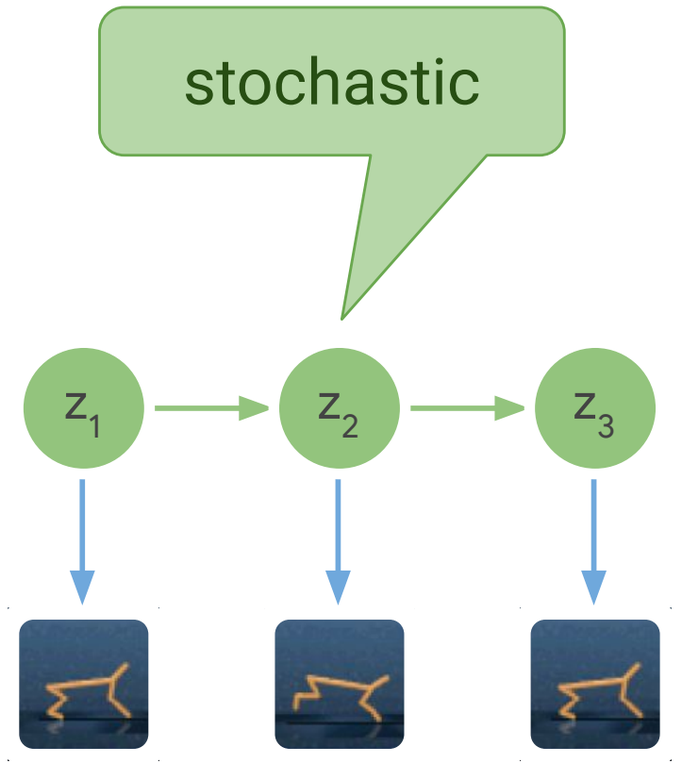
$$\geq \mathrm{E}_{q(s_{1:T}|o_{1:T},a_{1:T})}\left[\sum_{t=1}^{T} \ln p(o_t \mid s_t) + \ln p(s_t \mid s_{t-1},a_{t-1}) - \ln q(s_t \mid o_{\leq t},a_{<t})\right]$$

$$= \sum_{t=1}^{T} \Big(\underbrace{\mathrm{E}_{q(s_t|o_{\leq t},a_{<t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} - \underbrace{\mathrm{E}_{q(s_{t-1|o_{\leq t-1},a_{<a-1}})}\left[\mathrm{KL}[q(s_t \mid o_{\leq t},a_{<t}) \parallel p(s_t \mid s_{t-1},a_{t-1})]\right]}_{\text{complexity}} \Big).$$

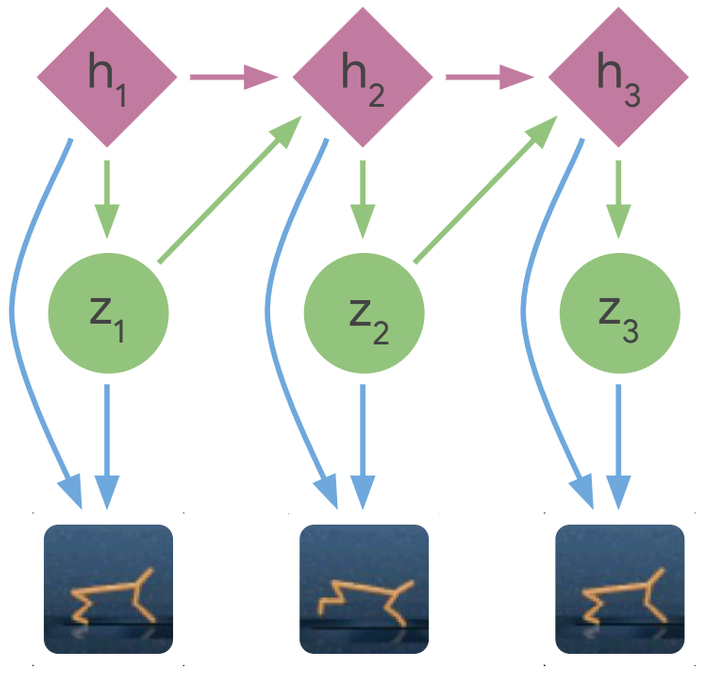
Recurrent State Space Model



Recurrent Neural Network



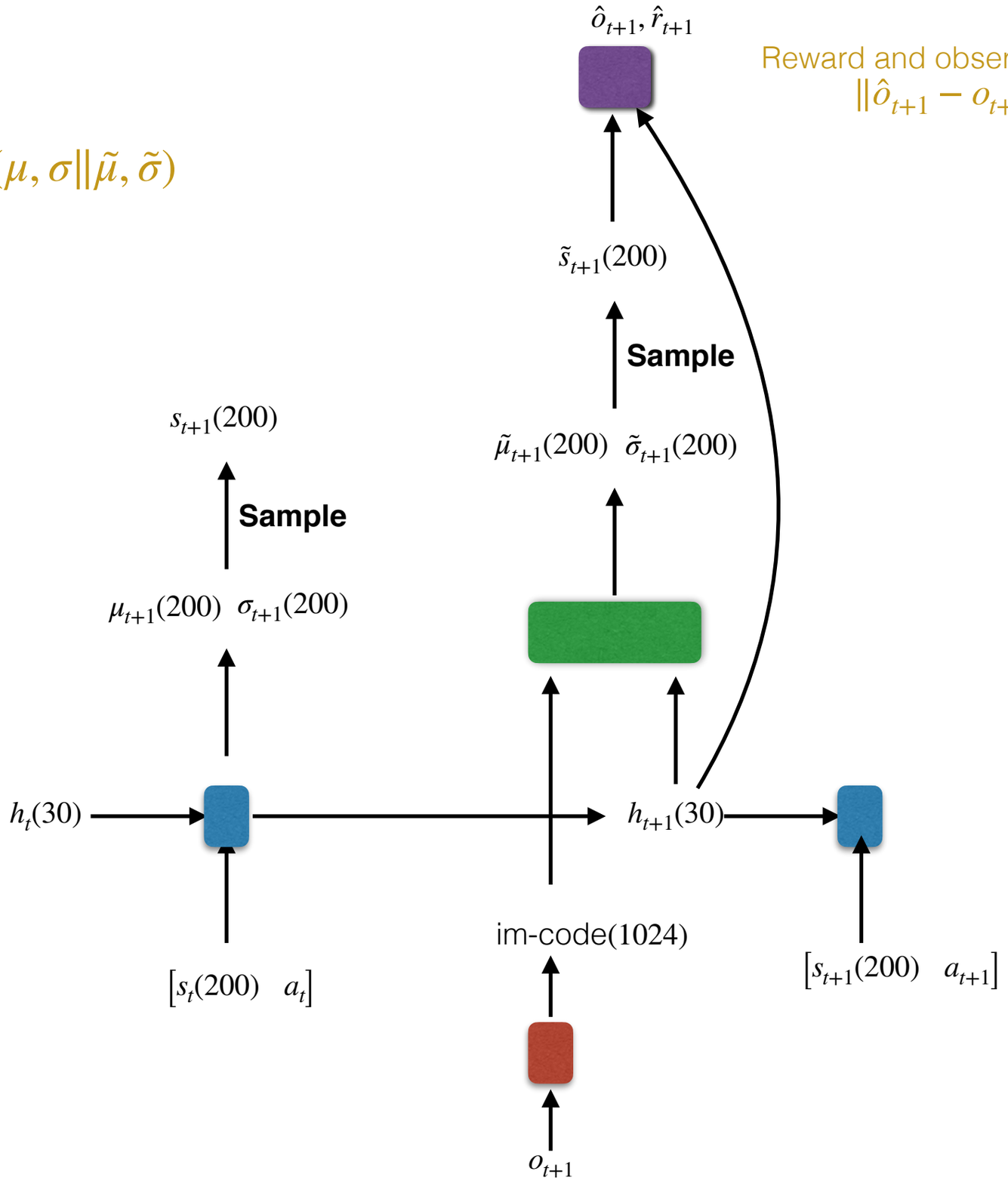
State Space Model



Recurrent State Space Model

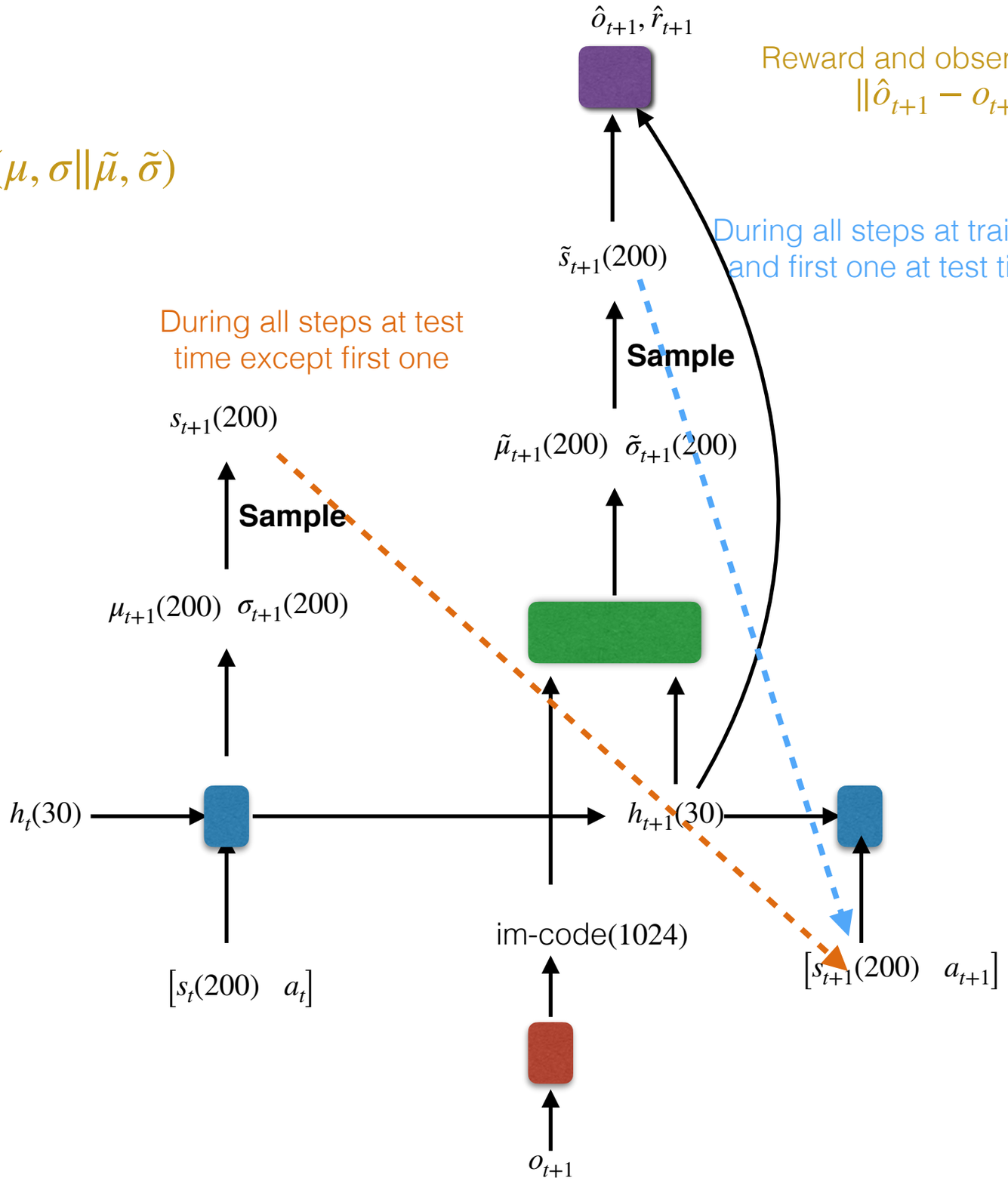
$D_{KL}(\mu, \sigma \| \tilde{\mu}, \tilde{\sigma})$

Reward and observation reconstruction loss:
$\|\hat{o}_{t+1} - o_{t+1}\| + \|\hat{r}_{t+1} - r_{t+1}\|$



$D_{KL}(\mu, \sigma \| \tilde{\mu}, \tilde{\sigma})$

Reward and observation reconstruction loss:
$\|\hat{o}_{t+1} - o_{t+1}\| + \|\hat{r}_{t+1} - r_{t+1}\|$



DREAMER

Model components
Representation model $p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$
Transition model $p_\theta(s_t \mid s_{t-1}, a_{t-1})$
Reward model $p_\theta(r_t \mid s_t)$
Policy $\pi_\phi(a_t \mid s_t)$
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: $s_t \sim p_\theta(s_t \mid s_{t-1}, a_{t-1}, o_t)$.
    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 $\phi \rightarrow \phi + \alpha \nabla_\phi \sum_{\tau}^{H} V_\lambda(s_\tau)$
    4. Update critic: $\psi \rightarrow \psi - \alpha \nabla_\psi \sum_{\tau}^{H} \| v_\psi - V_\lambda(s_\tau) \|$
3. **Environment interaction**
    1. Deploy the actor in the environment adding exploration noise to the predicted actions and update D_{env}

From Gaussian to Categorical latent distributions

Large-Scale Evaluation for Control from Pixels

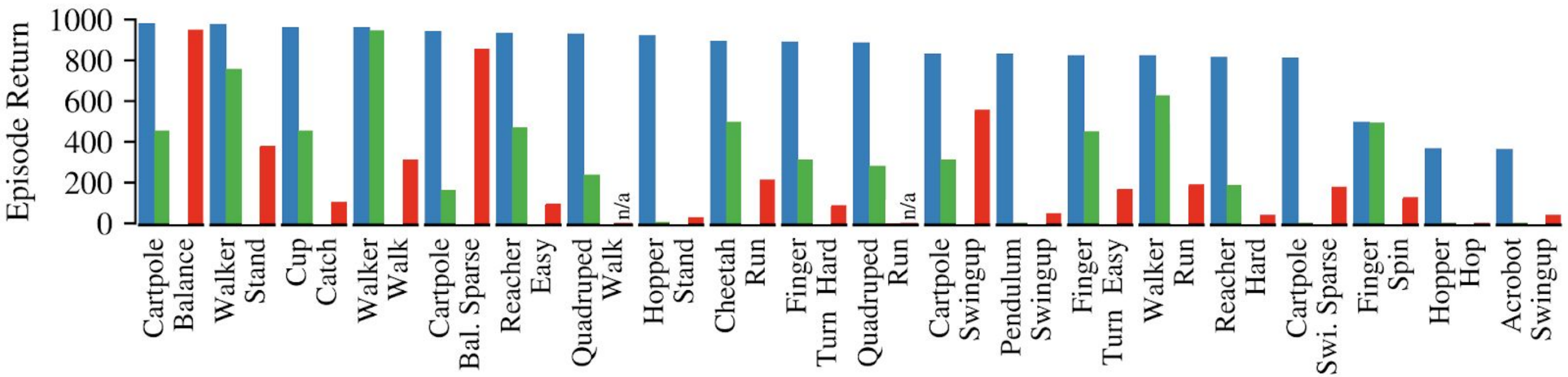
Model-based:
28 hours of interaction

Dreamer (823)

PlaNet (332)

Model-free:
23 days of interaction

A3C (243)



Large-Scale Evaluation for Control from Pixels

Model-based:
28 hours of interaction

Dreamer (823)

PlaNet (332)

Model-free:
23 days of interaction

D4PG (786)

A3C (243)