NeurIPS 2020

MLCF

Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model

Soft Actor Critic

Alex X. Lee[1,2] **Anusha Nagabandi**[1] **Pieter Abbeel**[1] **Sergey Levine**[1]
[1]University of California, Berkeley
[2]DeepMind
{alexlee_gk,nagaban2,pabbeel,svlevine}@cs.berkeley.edu

Reviewed by Susang Kim

Contents

1.Introduction - Motivation

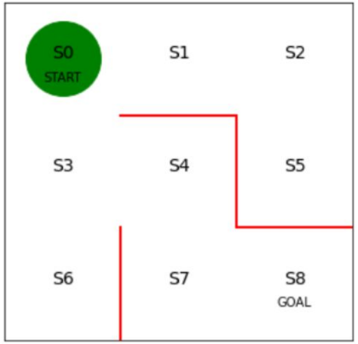
Challenges in **high-dimensional observation.** (e.g., image, unstructured, a large number of pixels)

The standard approach relies **on sensors to obtain information that would be helpful for learning.**

It's hard to use reinforcement learning algorithms to solve tasks using **only low-level observations**, such as learning **robotic control using only unstructured raw image data.** (Robot Vision)
⇒ **<u>tackle these two problems separately</u>**, by explicitly learning latent representations

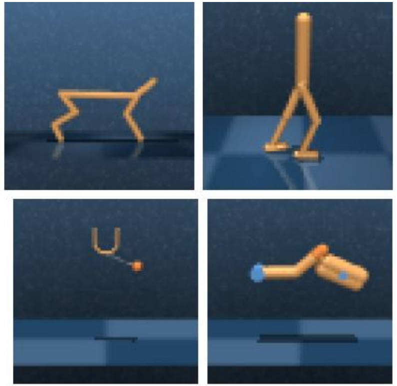
Learning from only image data is hard because the RL algorithm must learn **both a useful representation of the data** and **the task itself. (**+representation learning problem)

The agent (cheetah) didn't have any prior knowledge about movement.




```
state = np.array([[np.nan, 1, 1, np.nan],  # s0
                  [np.nan, 1, np.nan, 1],  # s1
                  [np.nan, np.nan, 1, 1],  # s2
                  [1, 1, 1, np.nan],  # s3
                  [np.nan, np.nan, 1, 1],  # s4
                  [1, np.nan, np.nan, np.nan],  # s5
                  [1, np.nan, np.nan, np.nan],  # s6
                  [1, 1, np.nan, np.nan], # s7, s8
                  ])
```

Sensor numerical data



raw image data

1.Introduction - Stochastic Latent Actor-Critic

MLCF

Standard model-free deep RL aims to unify these challenges of **representation learning and task learning** into a end-to-end training procedure. (policy or value function)
⇒ inefficient, practice be slow and sensitive to hyperparameters

SLAC: a sample-efficient and high-performing RL algorithm for learning policies for **complex continuous control tasks** directly from **high-dimensional image inputs**.

SLAC substantially outperforms both prior **model-free and model-based RL(agent in environment) algorithms on a range of image-based** continuous control benchmark tasks.

[SLAC]
1) High-dimensional observations as **a latent process, with a Gaussian prior and latent dynamics**.
2) **Partially Observed Markov Decision Process (POMDP)** : where the stochastic latent state enables the model to represent **uncertainty** about any of the state variables, given the past observations
3) Markovian **critic on latent state samples** and trains an **actor on a history of observations and actions**, resulting in our stochastic latent actor-critic (SLAC) algorithm.

The main contribution is a novel and principled approach that integrates learning **stochastic sequential models** and RL into a **single method** in the model's learned latent space.

1.Introduction - Contribution

MLCF

Representation learning in RL : “Representation learning bottleneck”: a considerable portion of the learning period must be spent **acquiring good representations of the observation space**
⇒ Jointly modeling between consecutive latent states, we make it feasible for our proposed algorithm to perform bellman backups directly in the latent space of the learned model

Partial observability in RL : Recent work has proposed end-to-end RL methods that use recurrent neural networks to process histories of observations and actions, but without constructing a model of the POMDP. Other works, **learn latent-space dynamical system models and then use them to solve the POMDP** with model-based RL
⇒ Our approach does not use the model for prediction, and performs infinite horizon policy optimization. benefits from the good asymptotic performance of model-free RL, while at the same time leveraging the improved latent space representation for sample efficiency. our method learns a critic directly on latent state samples, which enables scaling to more complex tasks.

Sequential latent variable models : Various modeling choices to learn **stochastic sequential models. factorization of the generative and inference models**, their network architectures, and the objectives used in their training procedures.
⇒ Our approach is compatible with any of these sequential latent variable models, with the only requirement being that they provide a mechanism to sample latent states from the belief of the learned Markovian latent space.

2.Preliminaries - VAE: Variational Autoencoder

MLCF

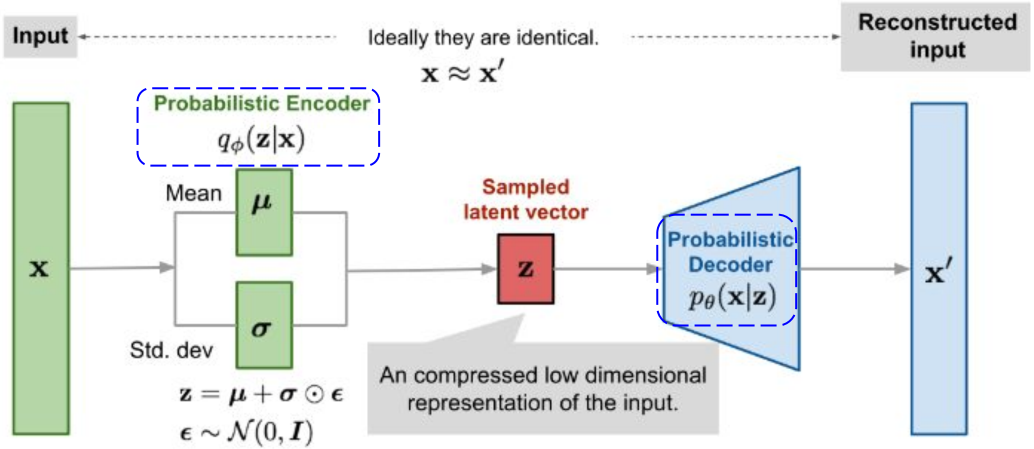
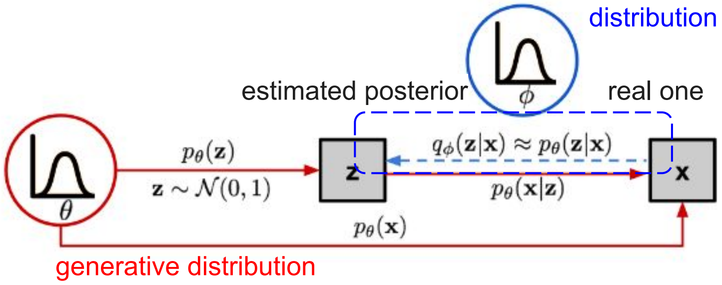


Fig. 9. Illustration of variational autoencoder model with the multivariate Gaussian assumption.



Prior $p_\theta(\mathbf{z})$

Likelihood $p_\theta(\mathbf{x}|\mathbf{z})$

Posterior $p_\theta(\mathbf{z}|\mathbf{x})$

$$
\begin{aligned}
& D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} \quad ; \text{Because } p(z|x) = p(z,x)/p(x) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \big(\log p_\theta(\mathbf{x}) + \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \big) d\mathbf{z} \quad ; \text{Because } \int q(z|x) dz = 1 \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})} d\mathbf{z} \quad ; \text{Because } p(z,x) = p(x|z)p(z) \\
&= \log p_\theta(\mathbf{x}) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \big[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} - \log p_\theta(\mathbf{x}|\mathbf{z}) \big] \\
&= \log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})
\end{aligned}
$$

$$\log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

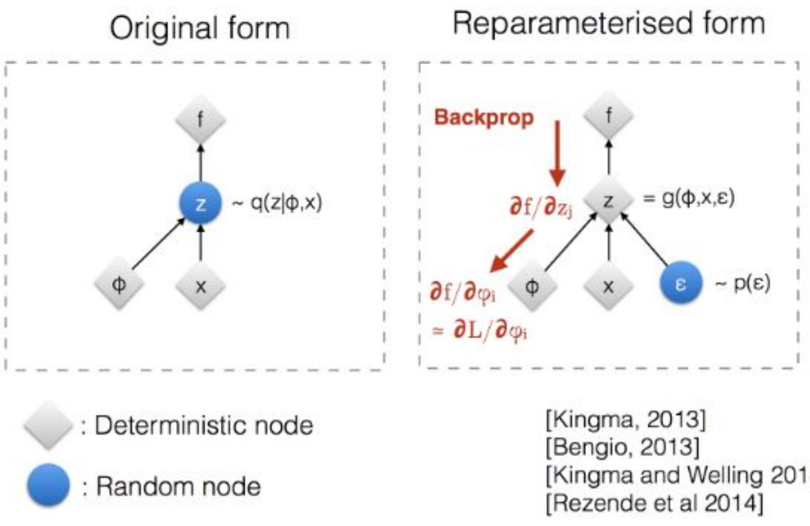
$$
\begin{aligned}
L_{\mathrm{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \\
&= -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) \\
\theta^*, \phi^* &= \arg\min_{\theta, \phi} L_{\mathrm{VAE}}
\end{aligned}
$$

non-negative

$$-L_{\mathrm{VAE}} = \log p_\theta(\mathbf{x}) - D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$

https://lilianweng.github.io/posts/2018-08-12-vae/

2.Preliminaries - Reparameterization Trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

The expectation term in the loss function invokes generating samples from $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$.
Sampling is a stochastic process and therefore we cannot backpropagate the gradient. To make it trainable.
(input -> model -> deterministic <-> stochastic)

$$\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)}\boldsymbol{I})$$
$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I})$$

(SAC - backpropagation : noise vector)

$$\mathrm{u}_i^j = \mathrm{f}_\theta(\mathrm{x}_i, \boxed{\eta_j})$$

$$\mathrm{u}_i^j \sim \mathcal{N}(\mu_\theta(\mathrm{x}_i),\ \sigma_\theta^2(\mathrm{x}_i))$$
$$= \mu_\theta(\mathrm{x}_i) + \sigma_\theta(\mathrm{x}_i)\boxed{\eta_j,\ \eta_j \sim} \mathcal{N}(0,\ I)$$

$$L_\pi(\theta) = \mathbb{E}_{\mathrm{x}_i \sim \mathcal{D}}[\mathbb{E}_{\boxed{\eta \sim}\mathcal{N}}[\alpha \log \pi_\theta(\mathrm{u}_i|\mathrm{x}_i) - Q_\phi(\mathrm{x}_i,\ \mathrm{u}_i)] \,|\, \mathrm{x}_i]$$

$$\nabla_\theta L_\pi(\theta) = \sum_i [\alpha \nabla_\theta \log \pi_\theta(\mathrm{u}_i|\mathrm{x}_i) + \nabla_\theta f_\theta(\mathrm{x}_i, \boxed{\eta_i})(\alpha \nabla_{\mathrm{u}_i} \log \pi_\theta(\mathrm{u}_i|\mathrm{x}_i) - \nabla_{\mathrm{u}_i} Q_\phi(\mathrm{x}_i,\ \mathrm{u}_i))]$$

https://lilianweng.github.io/posts/2018-08-12-vae/

2.Preliminaries - ELBO(Evidence Lower Bound)

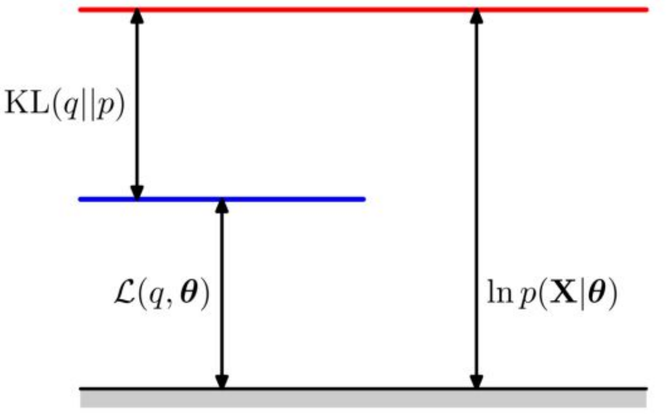
Variational Lower Bound

Variational Inference
gaussian distribution
posterior

$$\log p(\mathbf{x}) \geq E_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z})\right)$$

Reconstruction Error
Decoder
Cross Entropy
Observation model

Regularization parameter
Encoder
KL divergence
Variational approximate



Bishop – Pattern Recognition and Machine Learning

SLAC: RL objective for learning model, Actor and Critic.

$$\log p(\mathbf{x}_{1:\tau+1}|\mathbf{a}_{1:\tau}) \geq \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1}\sim q}\left[\sum_{t=0}^{\tau} \log p(\mathbf{x}_{t+1}|\mathbf{z}_{t+1}) - \mathrm{D}_{\mathrm{KL}}\left(q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1},\mathbf{z}_t,\mathbf{a}_t) \parallel p(\mathbf{z}_{t+1}|\mathbf{z}_t,\mathbf{a}_t)\right)\right]$$

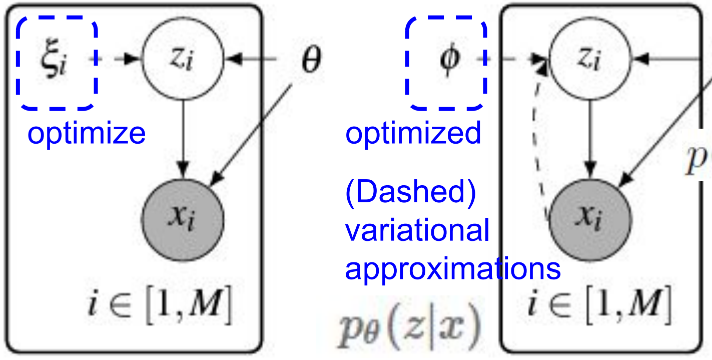
$$q(\mathbf{z}_1|\mathbf{x}_1,\mathbf{z}_0,\mathbf{a}_0) := q(\mathbf{z}_1|\mathbf{x}_1) \qquad p(\mathbf{z}_1|\mathbf{z}_0,\mathbf{a}_0) := p(\mathbf{z}_1)$$

encoder

decoder

2.Preliminaries - Amortized Variational Inference

Amortized variational inference don’t need latent fitting.



(a) SVI — Variational Inference

(b) VAE — Amortized Variational Inference

generative process

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}.$$

Amortized inference usually refers to inference over local variables. Instead of approximating separate variables for each data point, as shown in Figure (a) amortized VI assumes that the local variational parameters can <u>be predicted by a parameterized function of the data</u>. Thus, once this function is estimated, the latent variables can <u>be acquired by passing new data points through the function</u>, as shown in Figure (b). Deep neural networks used in this context are also called inference networks. <u>Amortized VI with inference networks thus combines probabilistic modeling with the representational power of deep learning.</u>

Zhang, Cheng, et al. "Advances in variational inference." *TPAMI 2018* https://arxiv.org/pdf/1711.05597.pdf

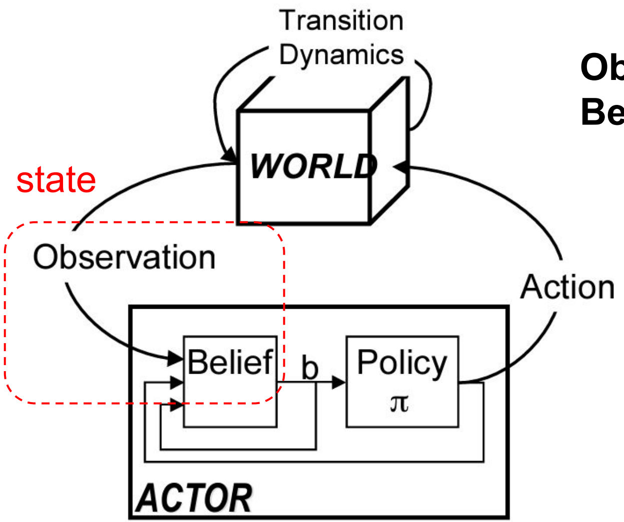
2.Preliminaries - Partially Observed Markov Decision Process (POMDP)

MLCF

POMDP : Partially Observable Markov Decision Process.
Uncertainty about the action outcome and the world state due to partial information

MDP: Tractable to solve, Relatively easy to specify, Assumes perfect knowledge of state
POMDP : Treats all sources of uncertainty uniformly, Allows for information gathering actions, intractable to solve optimally **(Add 2 Parameter : Observations, Belief State)**

$$< S, A, P, R, \gamma > \Rightarrow < S, A, P, R, \Omega, O, \gamma >$$

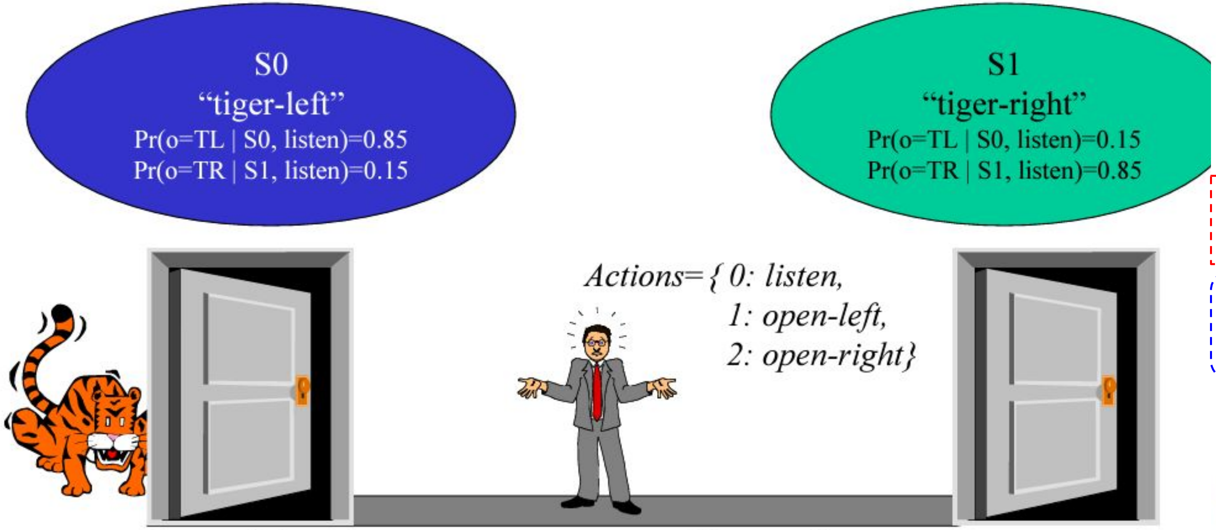


Observation- indirect measurements (sensations of state and reward)
Beliefs - Understanding of state with **uncertainty**

Markov Models		Do we have control over the state transitons?	
		NO	YES
Are the states completely observable?	YES	Markov Chain	MDP Markov Decision Process
	NO	HMM Hidden Markov Model	POMDP Partially Observable Markov Decision Process

http://www.pomdp.org/faq.html
https://www.techfak.uni-bielefeld.de/~skopp/Lehre/STdKI_SS10/POMDP_tutorial.pdf

2.Preliminaries - POMDP : Continuous Space Belief MDP



Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

SLAC Model

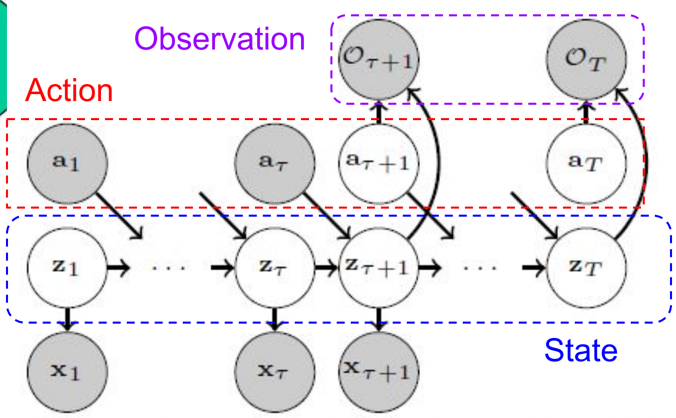


Figure 1: Graphical model of POMDP with optimality variables for $t \geq \tau + 1$.

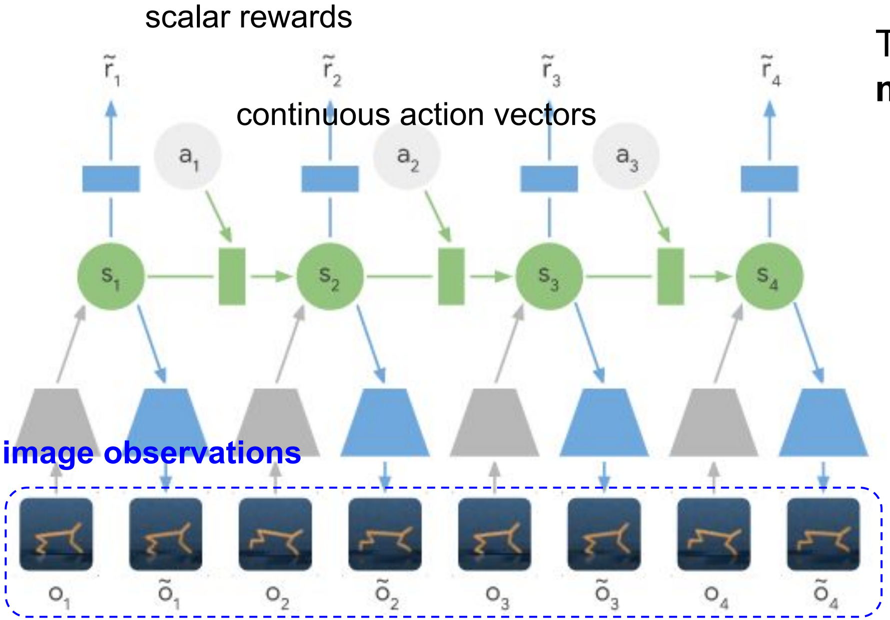
Prob. (LEFT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Observation

Prob. (LISTEN)	O: TL	O: TR
Tiger: left	0.85	0.15
Tiger: right	0.15	0.85

https://www.techfak.uni-bielefeld.de/~skopp/Lehre/STdKI_SS10/POMDP_tutorial.pdf

2.Preliminaries - Latent dynamics

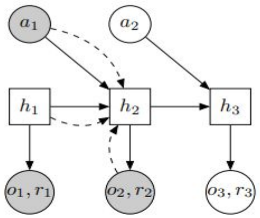


To solve unknown environments via planning, we need to **model the environment dynamics from experience**

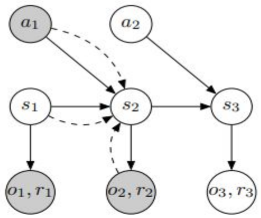
Sequences $\{o_t, a_t, r_t\}_{t=1}^{T}$

$$
\begin{aligned}
&\text{Transition function:} && s_t \sim \mathrm{p}(s_t \mid s_{t-1}, a_{t-1}) \\
&\text{Observation function:} && o_t \sim \mathrm{p}(o_t \mid s_t) \\
&\text{Reward function:} && r_t \sim \mathrm{p}(r_t \mid s_t) \\
&\text{Policy:} && a_t \sim \mathrm{p}(a_t \mid o_{\le t}, a_{<t}),
\end{aligned} \tag{1}
$$

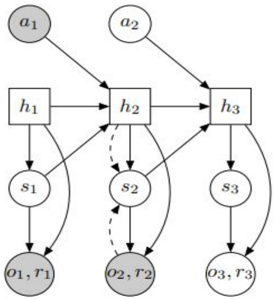
Circles : stochastic variables
Squares : deterministic variables
Solid lines : generative process
Dashed lines : inference model.



(a) Deterministic model (RNN)



(b) Stochastic model (SSM)



(c) Recurrent state-space model (RSSM)

Hafner, Danijar, et al. "Learning latent dynamics for planning from pixels." *ICML 2019*

3.Methods - Maximum Entropy RL in Fully Observable MDPs

MLCF

The probability distribution which best represents the current state of knowledge
⇒ minimize Worst-loss policy

Objective Function(maximize the expected entropy)
The entropy term **encourages exploration**

$$\mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t)).$$

Entropy Bonus

Maximum entropy objective function

$$\underbrace{\textstyle\sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\rho_\pi}\left[r(\mathbf{s}_t,\mathbf{a}_t)\right.}_{\text{RL}} + \boxed{\alpha}\left.\mathcal{H}(\pi_\phi(\cdot|\mathbf{s}_t))\right]$$

temperature parameter that trades off between maximizing for the reward and for the policy entropy

Minimize the soft Bellman residual

discount factor

$$J_Q(\theta) = \frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \left(r_t + \boxed{\gamma} \mathop{\mathbb{E}}_{\mathbf{a}_{t+1}\sim\pi_\phi}\left[Q_{\boxed{\bar{\theta}}}(\mathbf{s}_{t+1},\mathbf{a}_{t+1}) - \alpha\log\pi_\phi(\mathbf{a}_{t+1}|\mathbf{s}_{t+1})\right]\right)\right)^2$$

delayed parameters

The policy parameters φ are optimized to update the policy towards the exponential of the soft Q-function, resulting in the policy loss

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{a}_t\sim\pi_\phi}\left[\alpha\log(\pi_\phi(\mathbf{a}_t|\mathbf{s}_t)) - Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\right]$$

SLAC builds on this maximum entropy RL, by integrating **representation learning and partial observability**

3.Methods - POMDP & Factorize variational distribution

MLCF

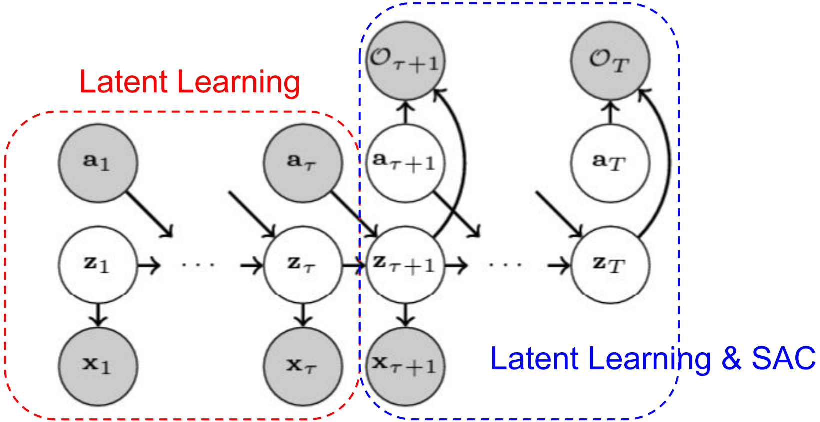


Figure 1: Graphical model of POMDP with optimality variables for $t \geq \tau + 1$.

binary random variable 0 is not optimal

$$p(\mathcal{O}_t = 1|\mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

maximize the marginal likelihood

$$p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}|\mathbf{a}_{1:\tau})$$

factorize variational distribution

$$q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) = \prod_{t=0}^{\tau} \underbrace{q(\mathbf{z}_{t+1}|\mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t)}_{\text{recognition terms}} \prod_{t=\tau+1}^{T-1} \underbrace{p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t)}_{\text{dynamics terms}} \prod_{t=\tau+1}^{T} \underbrace{\pi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})}_{\text{policy terms}}.$$

3.Methods - Derivation of the ELBO

$$
\begin{aligned}
&\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T} | \mathbf{a}_{1:\tau}) \\
&= \log \int_{\mathbf{z}_{1:T}} \int_{\mathbf{a}_{\tau+1:T}} p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{a}_{1:\tau}) \, d\mathbf{z}_{1:T} \, d\mathbf{a}_{\tau+1:T} && (13) \\
&\geq \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}) \sim q} \Big[\log p(\mathbf{x}_{1:\tau+1}, \mathcal{O}_{\tau+1:T}, \mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{a}_{1:\tau}) - \log q(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \Big] && (14) \\
&= \mathop{\mathbb{E}}_{(\mathbf{z}_{1:T}, \mathbf{a}_{\tau+1:T}) \sim q} \Bigg[\underbrace{\sum_{t=0}^{\tau} \Big(\log p(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}) - \mathrm{D}_{\mathrm{KL}}(q(\mathbf{z}_{t+1} | \mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \parallel p(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t)) \Big)}_{\text{model objective terms}} \\
&\qquad + \underbrace{\sum_{t=\tau+1}^{T} \Big(r(\mathbf{z}_t, \mathbf{a}_t) + \log p(\mathbf{a}_t) - \log \pi(\mathbf{a}_t | \mathbf{x}_{1:t}, \mathbf{a}_{1:t-1}) \Big)}_{\text{policy objective terms}} \Bigg], && (15)
\end{aligned}
$$

SLAC which maximizes the ELBO using function approximators to model the prior and posterior distributions.(Latent variable model & Actor and critic)

3.Methods - Latent Variable Factorization

MLCF

Solid arrows as part of the generative model and all dashed arrows as part of approximate posterior.

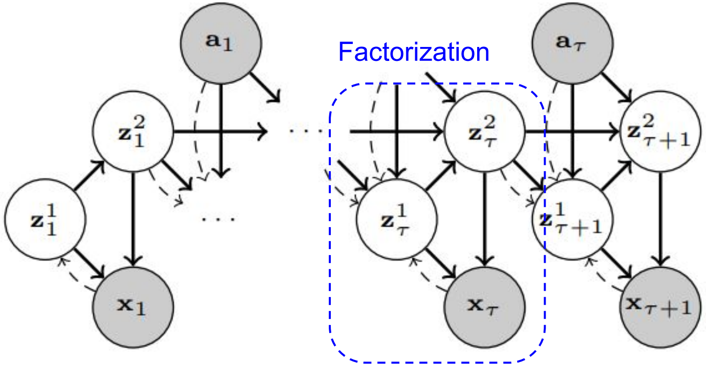


Figure 7: Diagram of our full model. Solid arrows show the generative model, dashed arrows show the inference model. Rewards are not shown for clarity.

The generative model consists of the probability distributions:

$$
\begin{aligned}
\mathbf{z}_1^1 &\sim p(\mathbf{z}_1^1) \\
\mathbf{z}_1^2 &\sim p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1) \\
\mathbf{z}_{t+1}^1 &\sim p_\psi(\mathbf{z}_{t+1}^1|\mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{z}_{t+1}^2 &\sim p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{x}_t &\sim p_\psi(\mathbf{x}_t|\mathbf{z}_t^1, \mathbf{z}_t^2) \\
r_t &\sim p_\psi(r_t|\mathbf{z}_t^1, \mathbf{z}_t^2, \mathbf{a}_t, \mathbf{z}_{t+1}^1, \mathbf{z}_{t+1}^2).
\end{aligned}
$$

multivariate standard normal distribution N(0,I).

two fully connected layers, each with 256 hidden units, and a Gaussian output layer

The variational distribution q, also referred to as the inference model or the posterior, is following factorization

$$
\begin{aligned}
\mathbf{z}_1^1 &\sim q_\psi(\mathbf{z}_1^1|\mathbf{x}_1) \\
\mathbf{z}_1^2 &\sim p_\psi(\mathbf{z}_1^2|\mathbf{z}_1^1) \\
\mathbf{z}_{t+1}^1 &\sim q_\psi(\mathbf{z}_{t+1}^1|\mathbf{x}_{t+1}, \mathbf{z}_t^2, \mathbf{a}_t) \\
\mathbf{z}_{t+1}^2 &\sim p_\psi(\mathbf{z}_{t+1}^2|\mathbf{z}_{t+1}^1, \mathbf{z}_t^2, \mathbf{a}_t).
\end{aligned}
$$

5 convolutional layers (32 5x5, 64 3x3, 128 3x3, 256 3x3, and 256 4x4 filters, respectively, stride 2 each, except for the last layer), 2 fully connected layers (256 units each), and a Gaussian output layer. The parameters of the convolution layers are shared among both distributions.

3.Methods - Compared with Soft Actor Critic(SAC)

MLCF

SAC learns two Q-networks, a V-network, and a policy network. Two Q-networks are used to mitigate overestimation bias.
A V-network is used to stabilize training. Taking gradients through the expectations is done using the reparameterization trick
Off-Policy(DDPG: ICLR 2016)+Soft Bellman(Soft Q-Learning: ICML 2017) + Stable Actor-Critic(TD3:ICML 2018)

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ.
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{s}_t)$
    $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Algorithm 1 Stochastic Latent Actor-Critic (SLAC)

Require: Environment E and initial parameters $\psi, \phi, \theta_1, \theta_2$ for the model, actor, and critics.
$\mathbf{x}_1 \sim E_{\text{reset}}()$
$\mathcal{D} \leftarrow (\mathbf{x}_1)$
for each iteration **do**
  for each environment step **do**
    $\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t|\mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$
    $r_t, \mathbf{x}_{t+1} \sim E_{\text{step}}(\mathbf{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{a}_t, r_t, \mathbf{x}_{t+1})$
  for each gradient step **do**
    $\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}, r_\tau \sim \mathcal{D}$
    $\mathbf{z}_{1:\tau+1} \sim q_\psi(\mathbf{z}_{1:\tau+1}|\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$
    $\psi \leftarrow \psi - \lambda_M \nabla_\psi J_M(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$
    $\bar{\theta}_i \leftarrow \nu\theta_i + (1 - \nu)\bar{\theta}_i$ for $i \in \{1, 2\}$

3.Methods - Algorithm

MLCF

Algorithm 1 Stochastic Latent Actor-Critic (SLAC)

Require: Environment E and initial parameters $\psi, \phi, \theta_1, \theta_2$ for the model, actor, and critics.

$\mathbf{x}_1 \sim E_{\text{reset}}()$ initial observation from the environment

$\mathcal{D} \leftarrow (\mathbf{x}_1)$ initialize replay buffer with initial obs.

for each iteration **do**

for each environment step **do**

$\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t | \mathbf{x}_{1:t}, \mathbf{a}_{1:t-1})$ sample action from the policy

$r_t, \mathbf{x}_{t+1} \sim E_{\text{step}}(\mathbf{a}_t)$ sample transition from the environment

$\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{a}_t, r_t, \mathbf{x}_{t+1})$ Store the transition in the reply buffer

for each gradient step **do**

$\mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}, r_\tau \sim \mathcal{D}$

$\mathbf{z}_{1:\tau+1} \sim q_\psi(\mathbf{z}_{1:\tau+1} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau})$

$\psi \leftarrow \psi - \lambda_M \nabla_\psi J_M(\psi)$

$\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$

$\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$

$\bar{\theta}_i \leftarrow \nu \theta_i + (1 - \nu)\bar{\theta}_i$ for $i \in \{1, 2\}$

Putting it all Together

Both the latent variable model and agent are trained together SLAC improves the SAC by learning the representation spaces with a latent variable model which is more stable and efficient for complex continuous control tasks. It can improve both the exploration and robustness of the learned model.

$$V_\theta(\mathbf{z}_{\tau+1}) = \mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi} \left[Q_\theta(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) - \alpha \log \pi_\phi(\mathbf{a}_{\tau+1} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) \right],$$

Model

$$J_M(\psi) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \left[\sum_{t=0}^{\tau} -\log p_\psi(\mathbf{x}_{t+1} | \mathbf{z}_{t+1}) + \mathrm{D_{KL}}(q_\psi(\mathbf{z}_{t+1} | \mathbf{x}_{t+1}, \mathbf{z}_t, \mathbf{a}_t) \| p_\psi(\mathbf{z}_{t+1} | \mathbf{z}_t, \mathbf{a}_t)) \right],$$

Critic

$$J_Q(\theta) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \left[\frac{1}{2} \left(Q_\theta(\mathbf{z}_\tau, \mathbf{a}_\tau) - (r_\tau + \gamma \boxed{V_{\bar{\theta}}(\mathbf{z}_{\tau+1})}) \right)^2 \right]$$

Actor

$$J_\pi(\phi) = \mathop{\mathbb{E}}_{\mathbf{z}_{1:\tau+1} \sim q_\psi} \left[\mathop{\mathbb{E}}_{\mathbf{a}_{\tau+1} \sim \pi_\phi} \left[\alpha \log \pi_\phi(\mathbf{a}_{\tau+1} | \mathbf{x}_{1:\tau+1}, \mathbf{a}_{1:\tau}) - Q_\theta(\mathbf{z}_{\tau+1}, \mathbf{a}_{\tau+1}) \right] \right]$$

3.Methods - Network Architectures

MLCF

Latent variables $\mathbf{z}_t^1 \in \mathbb{R}^{32}$ and $\mathbf{z}_t^2 \in \mathbb{R}^{256}$.

Image observations $\mathbf{x}_t \in [0, 1]^{64 \times 64 \times 3}$.

leaky ReLU non-linearities.

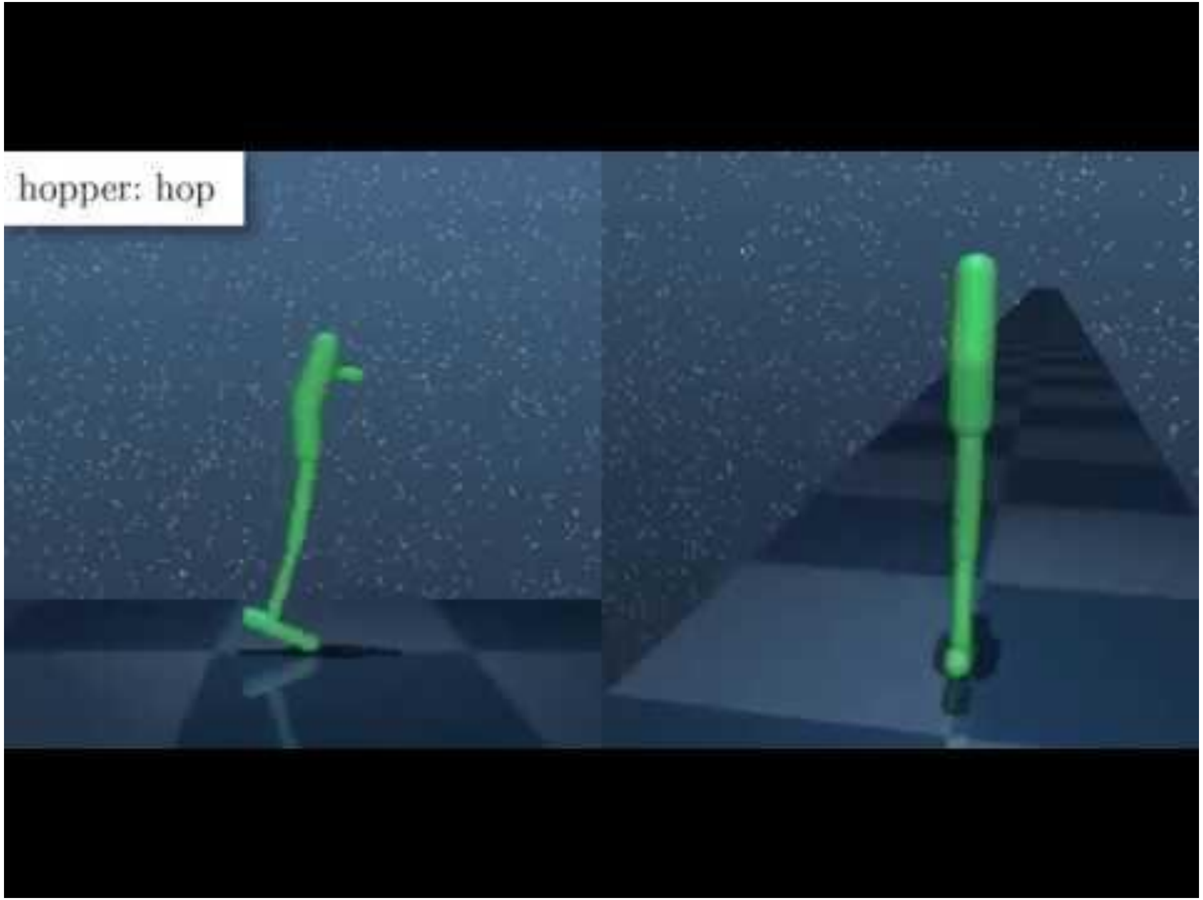
Critic network(Q) consisting of 2 fully connected layers (256 units each) and a linear output layer.

Actor network(P) consists of 5 convolutional layers, 2 fully connected layers (256 units each), a Gaussian layer, and a tanh bijector, which constrains the actions to be in the bounded action space of [-1, 1].

The convolutional layers are shared with the ones from the latent variable model, but the parameters of these layers are only updated by the model objective and not by the actor objective.

3.Methods - Experimental Evaluation

MLCF



The DeepMind Control Suite is a set of **continuous control tasks with a standardised structure** and interpretable rewards, intended to serve as performance benchmarks for reinforcement learning agent

Pixel observations by default, control suite environments return **low-dimensional feature observations**. The pixel.Wrapper adds or replaces these with images

Tassa, Yuval, et al. "Deepmind control suite." arXiv (2018). https://youtu.be/rAai4QzcYbs

3.Methods - Experimental Evaluation

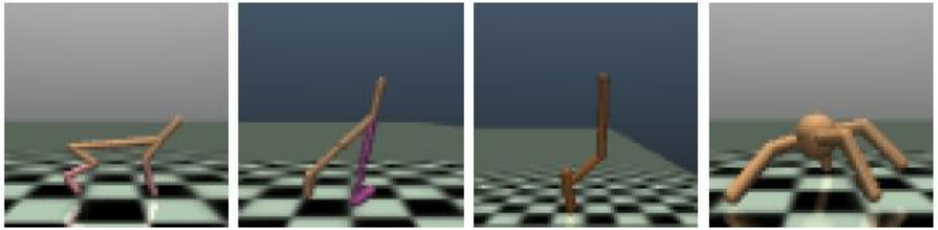
Image observations for continuous control benchmark tasks

64 x 64 pixels

DeepMind
Control Suite

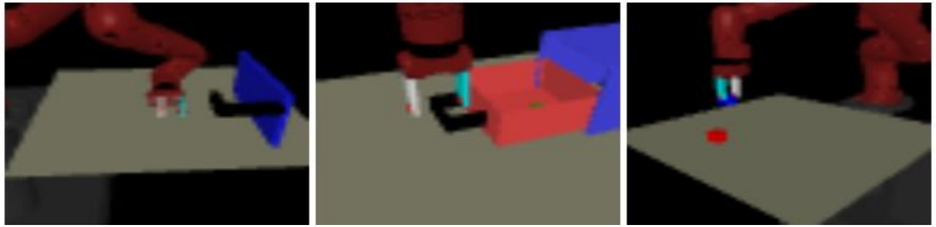
Cheetah run | Walker walk | Ball-in-cup catch | Finger spin

OpenAI
Gym's



Cheetah | Walker | Hopper | Ant

9-DoF
3-fingered
DClaw robot



Push a door | Close a drawer | Reach out / pick up an object

SLAC algorithm
achieves above action

3.Methods - Related Works

MLCF

Comparative Evaluation on Continuous Control Benchmark Tasks

SAC : Off-policy actor-critic algorithm, model-free learning. We include experiments showing the performance of SAC based on true state (upper bound on performance) from raw images.

D4PG : Off-policy actor-critic algorithm, learning directly from images

MPO : Off-policy actor-critic algorithm that performs an expectation maximization form of policy iteration, learning directly from raw images.

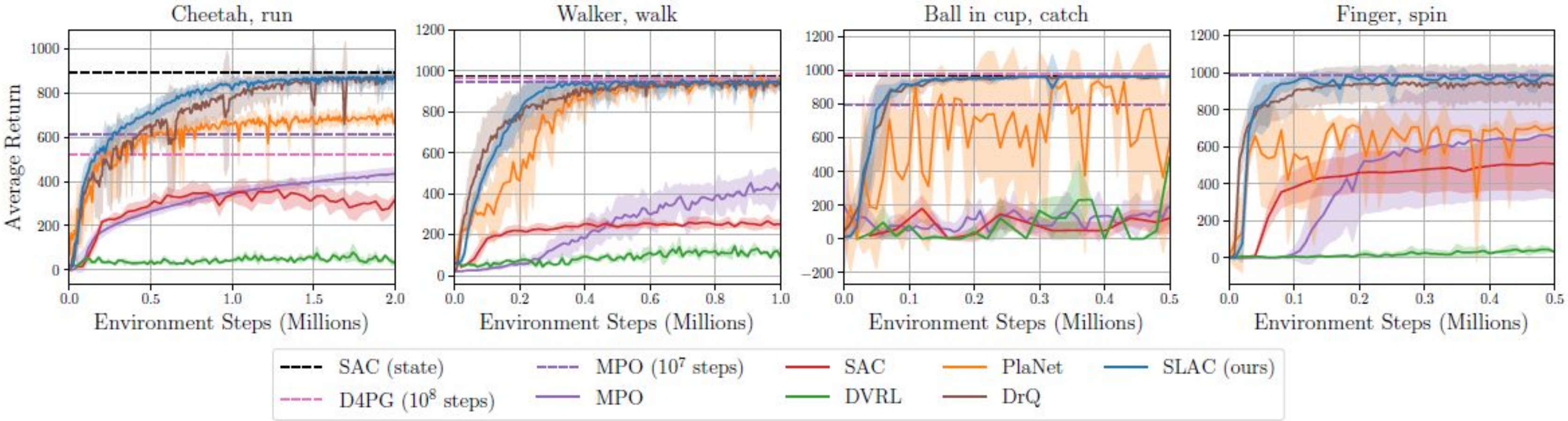
DVRL : On-policy model-free RL algorithm, Mixed deterministic/stochastic latent-variable POMDP modelas opposed to our method, which trains the critic with the latent state and the actor with a history of actions and observations

PlaNet : Model-based RL method for learning from images, which uses a partially stochastic sequential latent variable model, but without explicit policy learning.

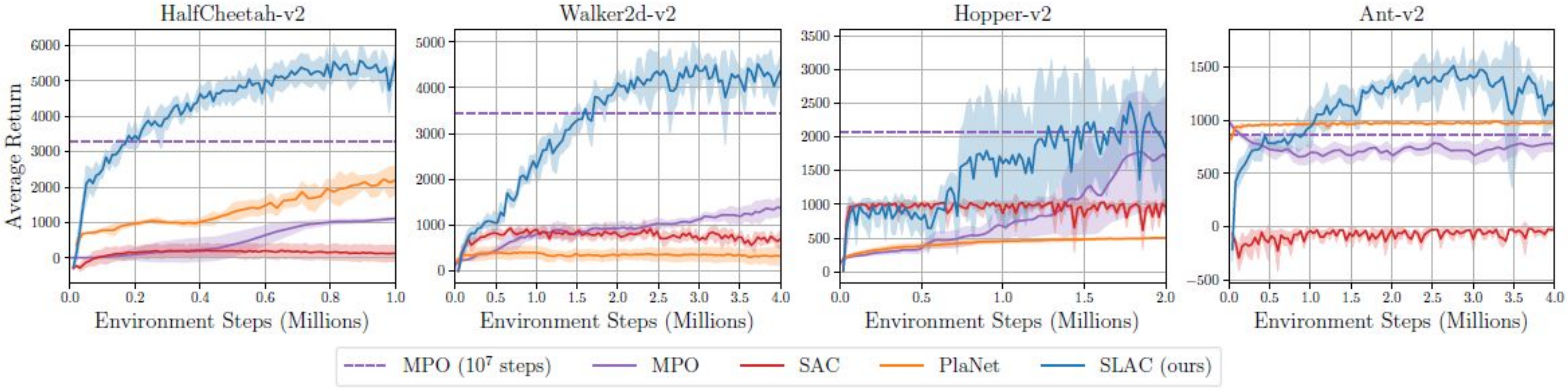
DrQ : the same as the SAC algorithm, but combined with data augmentation on the image inputs.

3.Methods - Experiments

DeepMind Control Suite from images

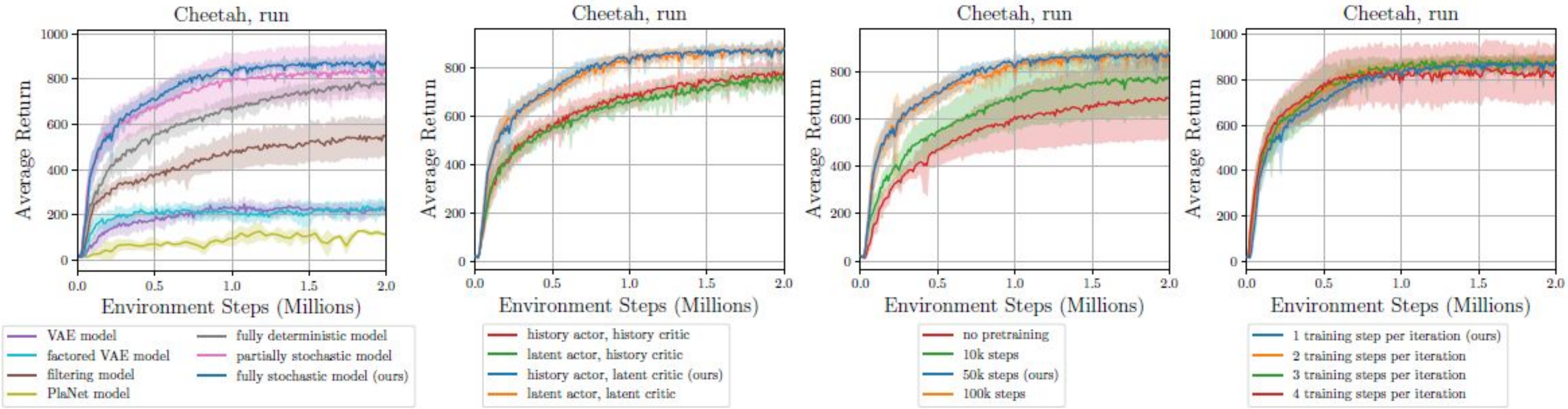


OpenAI Gym benchmark tasks from images



3.Methods - Ablation Experiments

MLCF



(a) Latent variable model **(b)** Actor and critic inputs **(c)** Model pretraining **(d)** Train steps per iteration

(a) **Latent variable model** : 1) fully stochastic model to a standard non-sequential VAE model, 2) two-variable factorization but without any temporal dependencies, 3) sequential filtering model that uses temporal dependencies but without the two-variable factorization, 4) without the two-variable factorization, the partially stochastic model used by PlaNet

(b) **Actor and critic inputs** : actor and critic inputs as either the observation-action history or the latent sample

(c) **Model pre-training** : the agent first collects a small amount of data by executing random actions, and then the model is pre-trained with that data

(d) **Training updates per iteration** : the effect of the number of training updates per iteration

4.Conclusion

[Contribution]

Combined off-policy model-free RL with representation learning via a sequential stochastic state model.

SLAC algorithm for learning from high-dimensional image inputs.(learn directly from raw image observations)

Previous work have used mixed deterministic-stochastic models, but SLAC's model is purely stochastic
SLAC's fully stochastic model outperforms other latent variable models.
- Achieved improved sample efficiency and final task performance
 (Four DeepMind Control Suite tasks and four OpenAI Gym tasks)

Simulation on robotic manipulation tasks (9-DoF 3-fingered DClaw robot on four tasks)

[Limitation]

In real-world setups such as robotics how to choice reward function?
It takes lots of time to train latent space.(Change environment and complexity)
For fairness, performance evaluations for other models seems necessary
(not just SLAC RL framework, compare on different latent variable models)
Performance on other image-based continuous control tasks
States benefits of using two layers of latent variables (Insufficient explanation on why it brings good balance)

Openreview (ICLR 2020)

Paper that are not properly backed up by evidence, it is not sufficiently clear to what degree the shown performance improvement is due to the stochastic nature of the model used.

Official Blind Review #4 (Rating: 3: Weak Reject)
- There are quite a few recent works about representation learning and yet the paper makes no references to previous works. I find it surprising that none of the past related works are mentioned in the paper. it does not explicitly require references to a POMDP. I do not understand why the critic evaluation in the latent space is even a good approach? It is more of an experimental design and engineering approach that combines previous known techniques.

Official Blind Review #2 (Rating: 8: Accept)
- Operates on a learned latent state, rather than an observed one, and therefore aims to jointly learn to represent high dimensional inputs and execute continuous control based on this representation.(Novel formulation, SOTA results, well written) the role of making the primary latent variable stochastic, is investigated, a deeper investigation of what makes the model more effective than existing techniques would be insightful.

Official Blind Review #3 (Rating: 3: Weak Reject)
- The real-benefit of the method may not be consistent. (My biggest concern)
Full stochasticity contributed to the practical gain but in the experiment Figure 6,
The simple filtering does not perform well. It seems that the benefit of the method is rather from such particular latent space design rather than the stochastic vs deterministic.
the experiments may be unfair, because, another partially stochastic method can easily utilize such design and further improve the performance.
Simple Filtering (without factoring model)

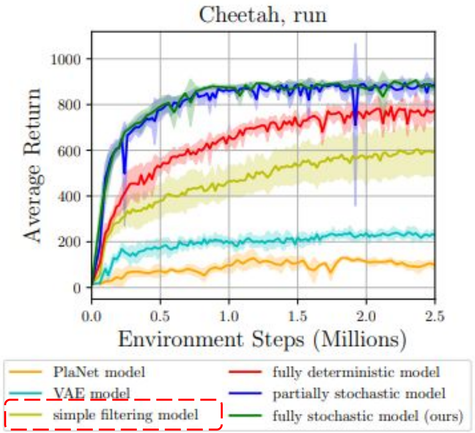


Figure 6: Comparison of different design choices for the latent variable model.

https://openreview.net/forum?id=HJxDugSFDB

References

MLCF

ICLR 2020 Conference homepage :https://openreview.net/forum?id=HJxDugSFDB

VAE : https://lilianweng.github.io/posts/2018-08-12-vae/

Zhang, Cheng, et al. "Advances in variational inference." *TPAMI 2018* https://arxiv.org/pdf/1711.05597.pdf

POMDP : http://www.pomdp.org/faq.html
https://www.techfak.uni-bielefeld.de/~skopp/Lehre/STdKI_SS10/POMDP_tutorial.pdf

CS 330: Deep Multi-Task and Meta Learning : http://cs330.stanford.edu/fall2019/index.html

Hafner, Danijar, et al. "Learning latent dynamics for planning from pixels." ICML 2019 https://planetrl.github.io/

Thanks
Any Questions?

You can send mail to
Susang Kim(healess1@gmail.com)