Partial Observability in DRL

Part 2

POMDP Models so far

Short overview

Model free

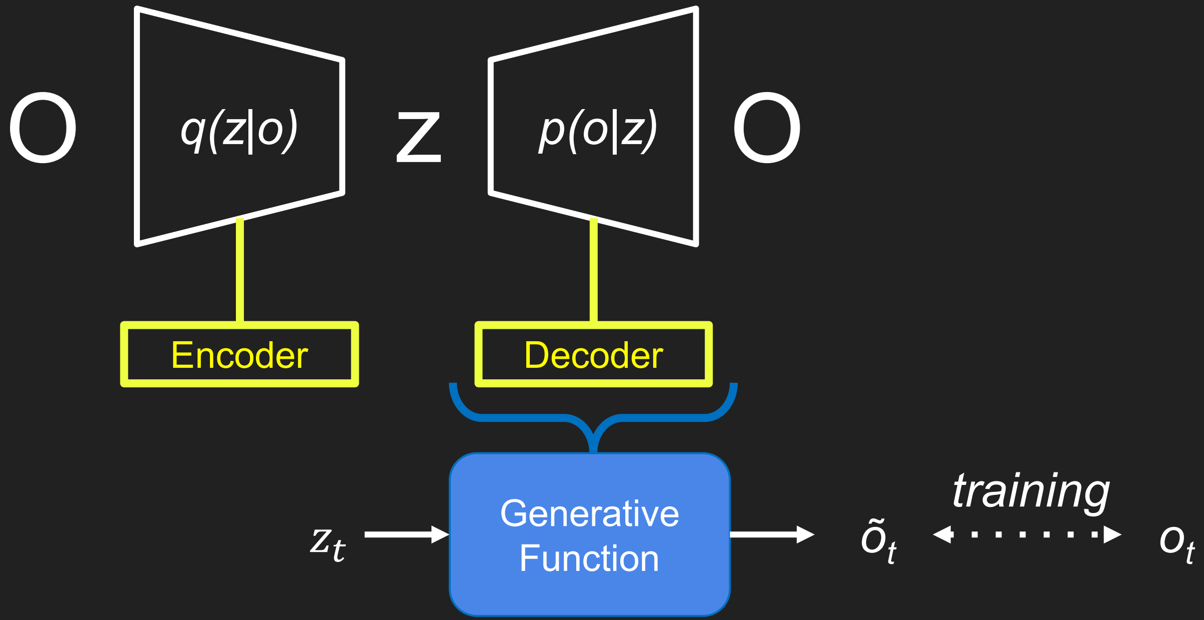
RNN
DRQN
ADRQN

Explicit Belief tracking

DVRL

Implicit Belief tracking

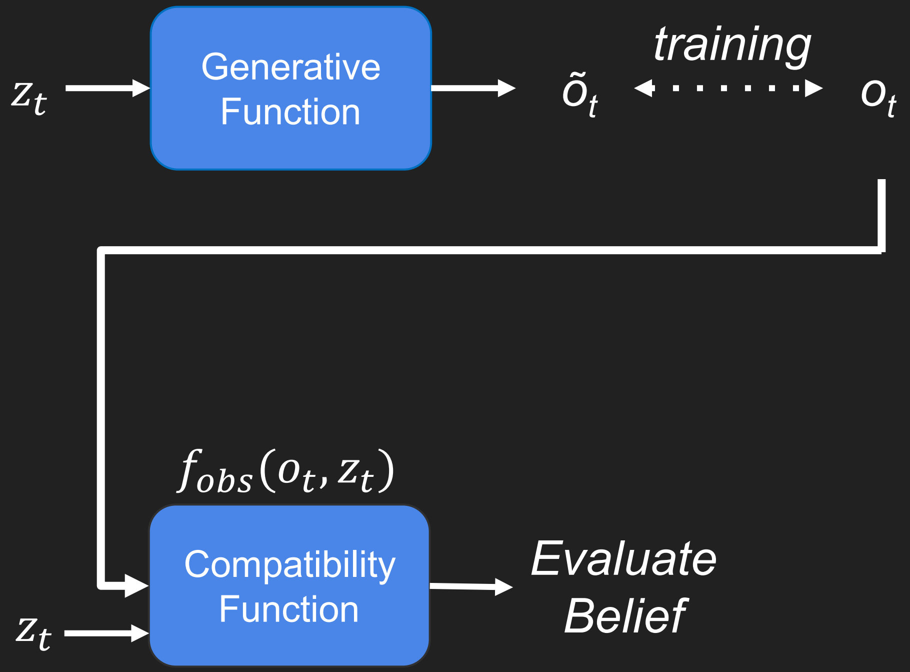
DVRL



(Igl et al. 2018)

DVRL

- Requires modeling all observations that define *p(o | h_t)*
  - *Needs to learn features irrelevant for RL*

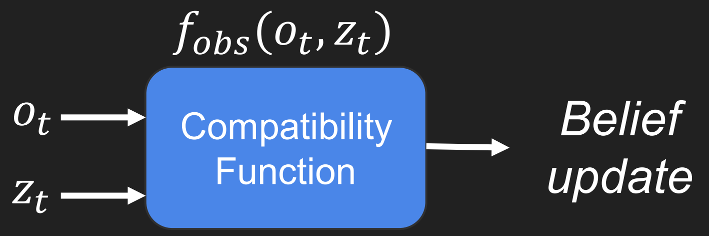


+ Directly updates belief based on observation

(Igl et al. 2018)

DPFRL

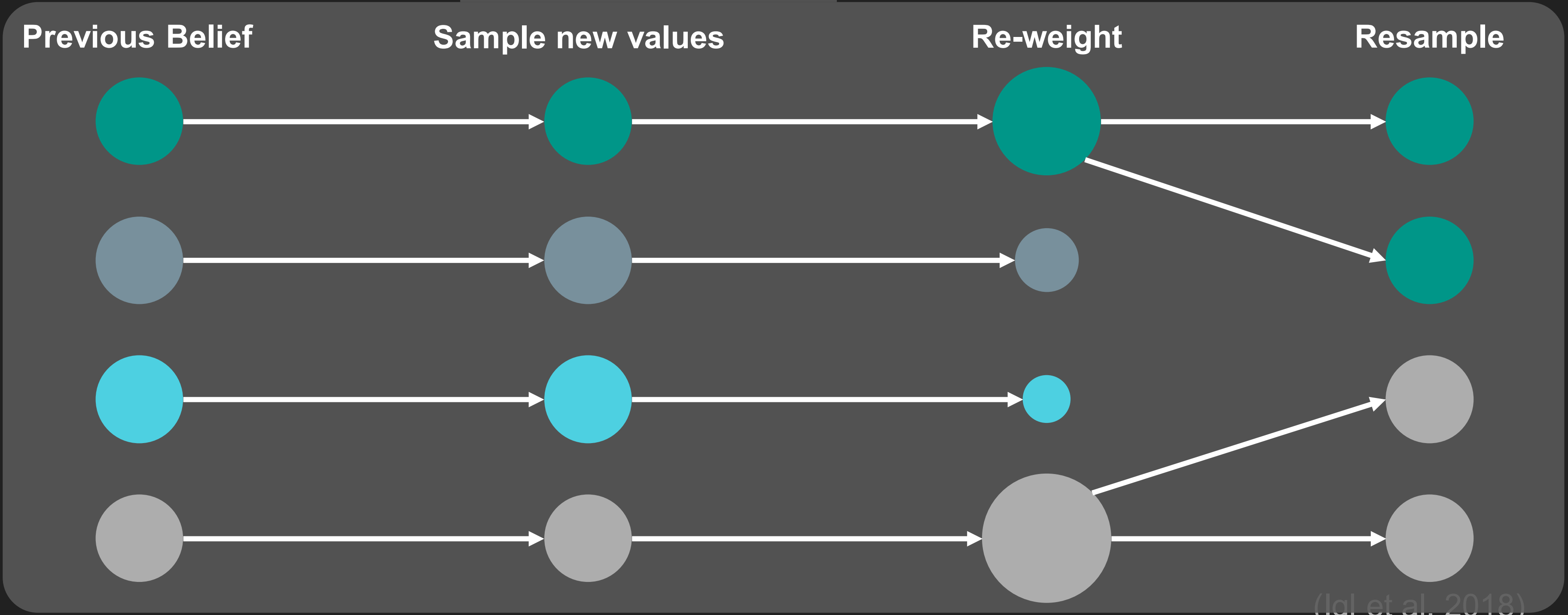
Discriminative Particle Filter Reinforcement Learning



(Igl et al. 2018)

DPFRL

Discriminative **Particle Filter** Reinforcement Learning



(Igl et al. 2018)

DPFRL

Discriminative **Particle Filter** Reinforcement Learning

<u>Previous Belief</u> **Sample new values** **Re-weight** **Resample**

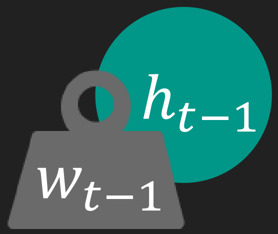
$$b_{t-1} \approx \{h_{t-1}, w_{t-1}\}_{k=1}^{K}$$

(Igl et al. 2018)

DPFRL

Discriminative **Particle Filter** Reinforcement Learning

<u>Previous Belief</u> | **Sample new values** | **Re-weight** | **Resample**



$\times\ K$

(Igl et al. 2018)

DPFRL

Discriminative **Particle Filter** Reinforcement Learning

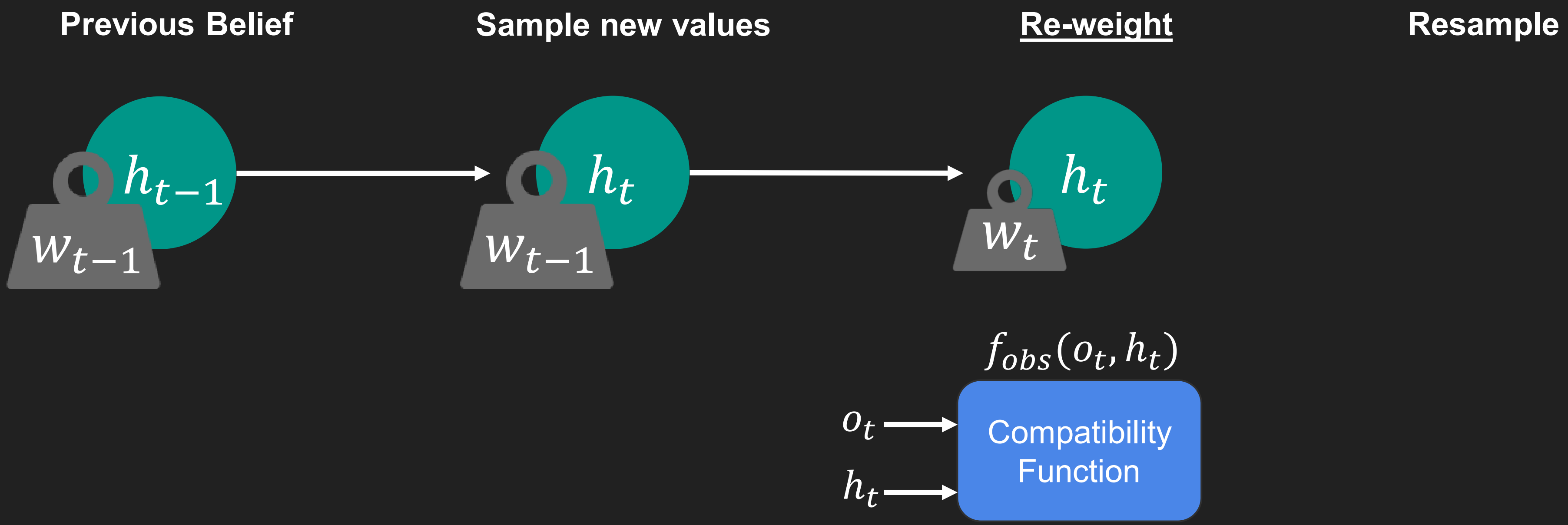
Previous Belief **Sample new values** **Re-weight** **Resample**

$$h_t \sim f_{trans}(h_{t-1}, a_t, o_t)$$

(Igl et al. 2018)

DPFRL

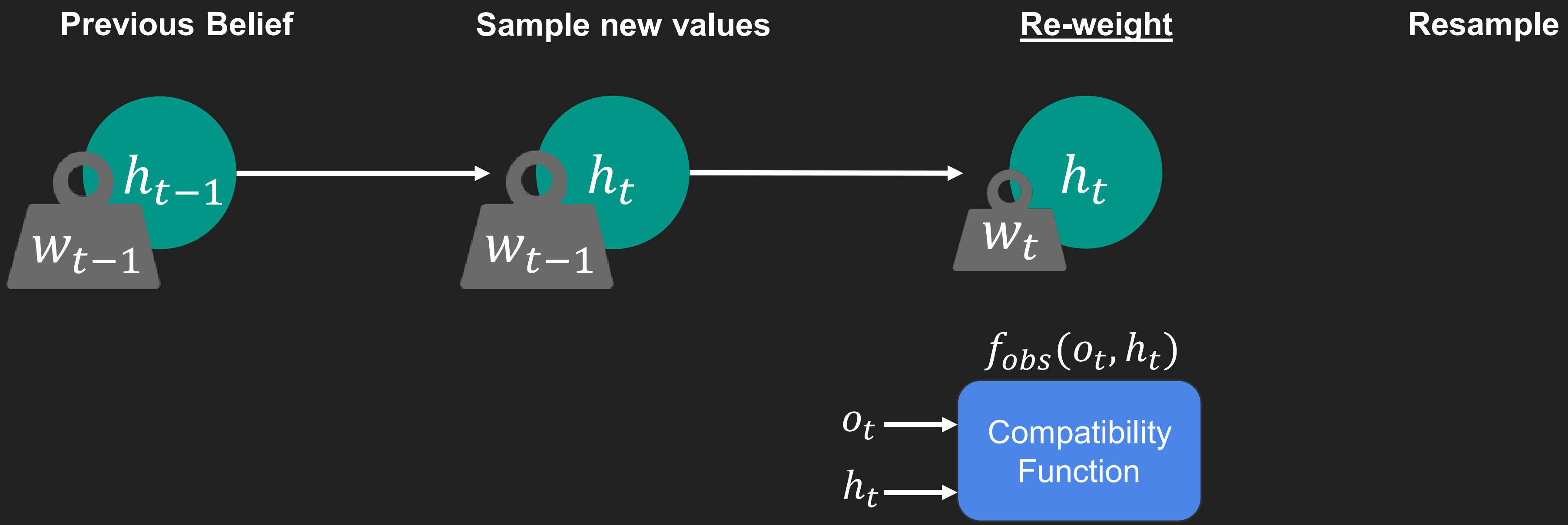
Discriminative **Particle Filter** Reinforcement Learning



(Igl et al. 2018)

DPFRL

Discriminative Particle Filter Reinforcement Learning



(Igl et al. 2018)

DVRL

DPFRL

Discriminative Particle Filter Reinforcement Learning

Decoder

Prior

Previous Belief

Sample new values

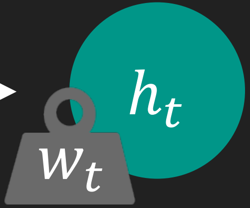
$$w_t = \frac{p_\theta\left(z_t | h_{t-1}, a_{t-1}\right) p_\theta\left(o_t | h_{t-1}, z_t, a_{t-1}\right)}{q_\phi\left(z_t | h_{t-1}, a_{t-1}, o_t\right)}$$

Encoder

$$ELBO(\theta, \phi) \approx \sum_{t=1}^{T} \log \left(\frac{1}{K} \sum_{k=1}^{K} w_t^k \right)$$

Re-weight

Resample



$f_{obs}(o_t, h_t)$

$o_t \rightarrow$ Compatibility Function

$h_t \rightarrow$

(Igl et al. 2018)

DVRL

Decoder

Prior

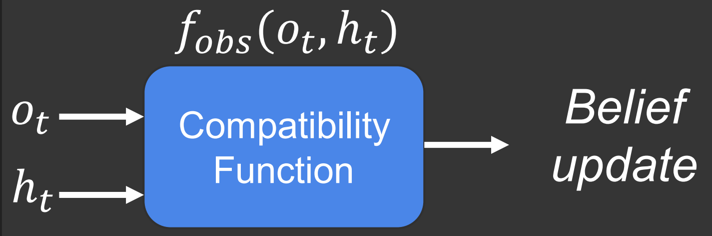
$$w_t = \frac{p_\theta\left(z_t | h_{t-1}, a_{t-1}\right) p_\theta\left(o_t | h_{t-1}, z_t, a_{t-1}\right)}{q_\phi\left(z_t | h_{t-1}, a_{t-1}, o_t\right)}$$

Encoder

$$ELBO(\theta, \phi) \approx \sum_{t=1}^{T} \log\left(\frac{1}{K}\sum_{k=1}^{K} w_t^k\right)$$

DPFRL

$$w_t \propto f_{obs}(h_t, o_t) w_{t-1}$$



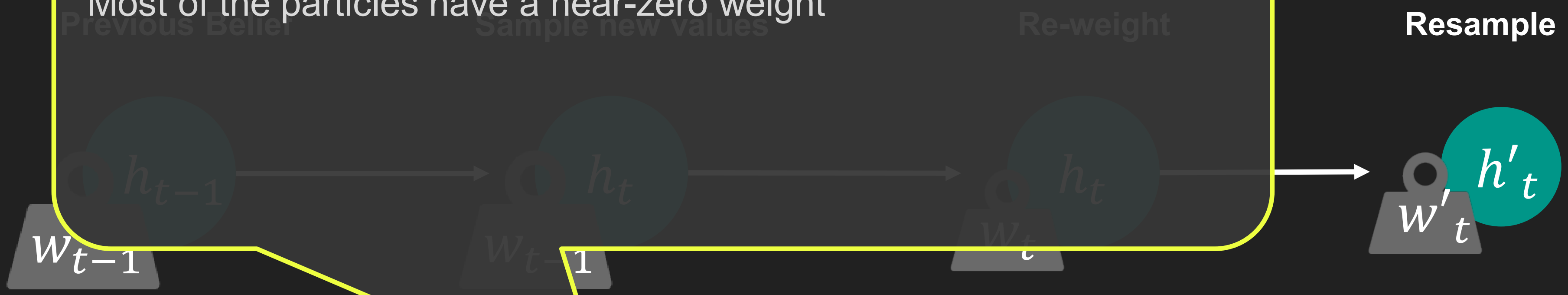
(Igl et al. 2018)

DPFRL

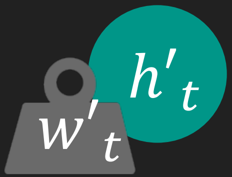
Learning

Problem: Particle degeneracy

Most of the particles have a near-zero weight



Resampling:



$\sim w_t$

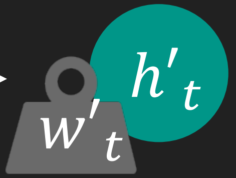
(Igl et al. 2018)

DPFRL

Discriminative Particle Filter Reinforcement Learning

Previous Belief | Sample new values | Re-weight | **Resample**

w_{t-1} h_{t-1} → w_{t-1} h_t → w_t h_t →



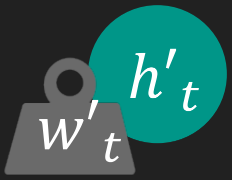
Problem: Particle degeneracy

Most of the particles have a near-zero weight

Solution: Soft-resampling

Provides approximate gradients for the non-differentiable resampling step

Soft-Resampling:

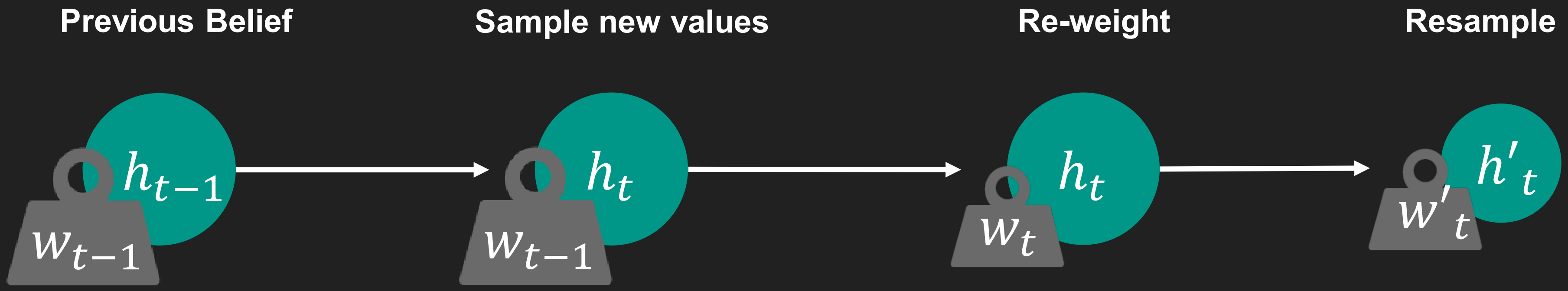


$$\sim \alpha w_t + (1-\alpha)\frac{1}{K}$$

(Igl et al. 2018)

DPFRL

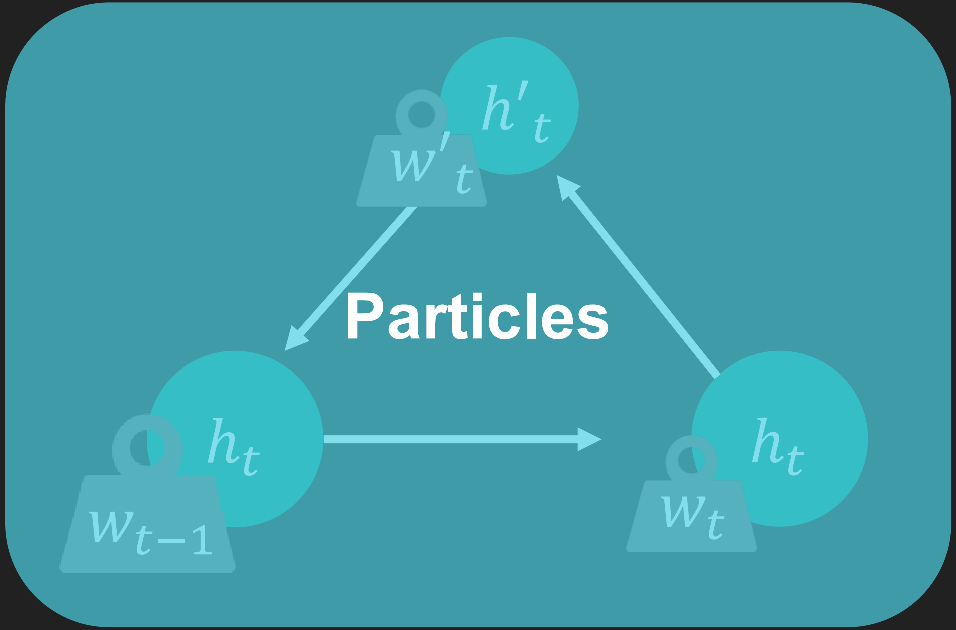
Discriminative Particle Filter Reinforcement Learning



(Igl et al. 2018)

DPFRL

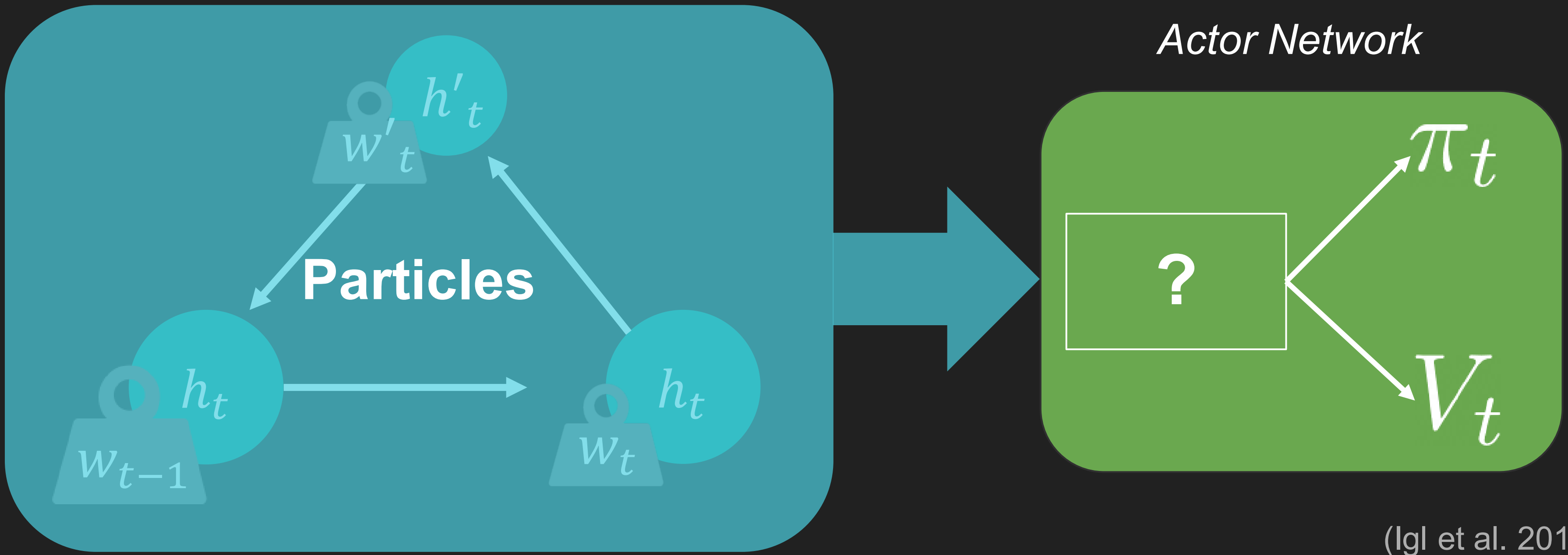
Discriminative Particle Filter Reinforcement Learning



(Igl et al. 2018)

DPFRL

Discriminative Particle Filter **Reinforcement Learning**



(Igl et al. 2018)

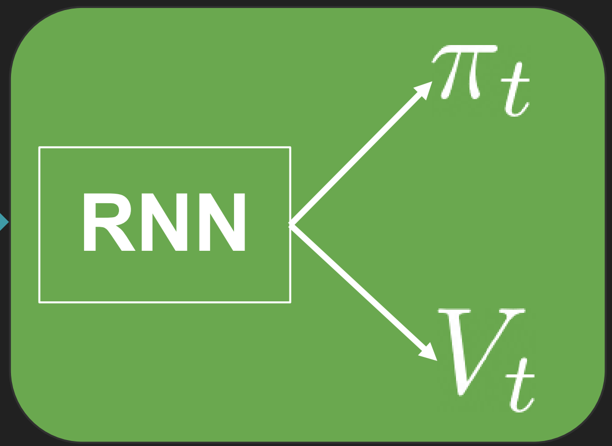
Recall **Reinforcement Learning** in DVRL

Particles

Properties

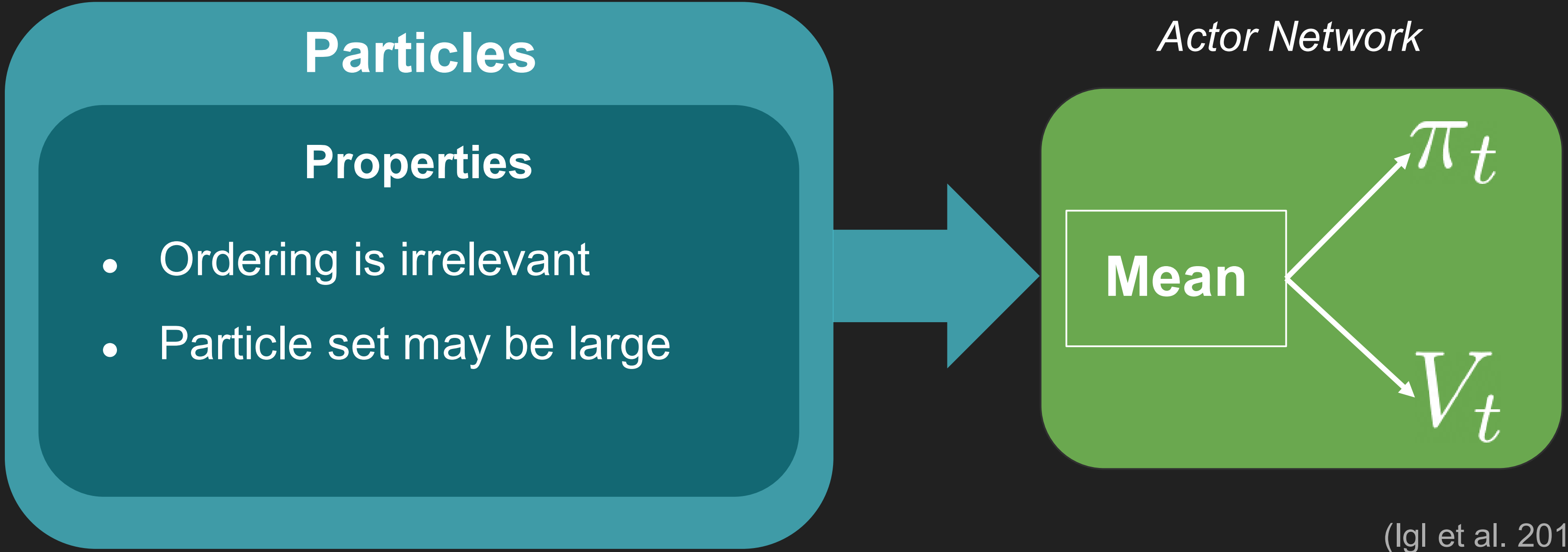
- Ordering is irrelevant
- Particle set may be large

Actor Network



(Igl et al. 2018)

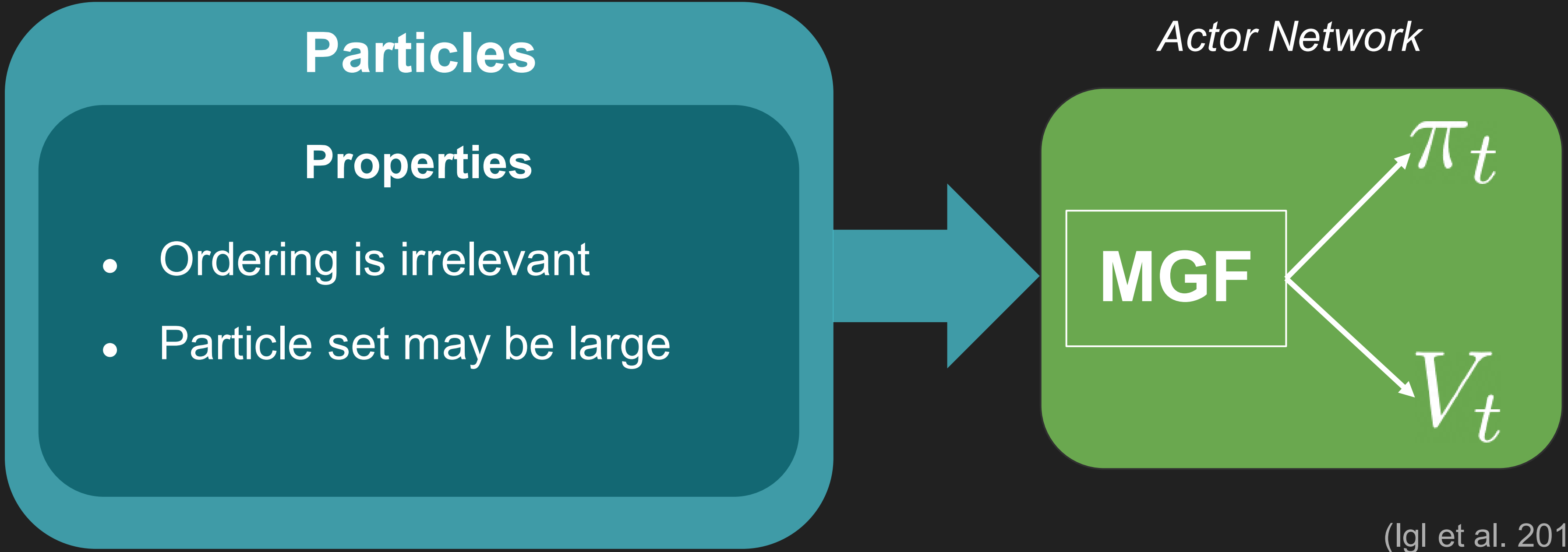
Simple but powerful solution (used in PF-RNNs)



(Igl et al. 2018)

DPFRL

Reinforcement Learning in DPFRL



(Igl et al. 2018)

Short excursion
Moment-Generating Functions (MGFs)

0th	Total probability
1st	Expectation
2nd	Variance
3rd	Skewness
…	…

Short excursion
Moment-Generating Functions (MGFs)

Centralized Moment

Total probability $\mathbb{P}[X] = 1$

Expectation $\mathbb{E}[X]$

Variance $\mathbb{E}\left[(X \boxed{- \mathbb{E}(X)})^2\right]$

Skewness $\mathbb{E}\left[(X - \mathbb{E}(X))^3\right]$

… $\mathbb{E}\left[(X - \mathbb{E}(X))^n\right]$

Short excursion
Moment-Generating Functions (MGFs)

Total probability	$\mathbb{E}(X^0)$
Expectation	$\mathbb{E}(X^1)$
Variance	$\mathbb{E}(X^2)$
Skewness	$\mathbb{E}(X^3)$
…	$\mathbb{E}(X^n)$

Raw Moment

Short excursion
Moment-**Generating Functions** (MGFs)

$$\mathbb{E}(X^0)$$

$$\mathbb{E}(X^1) \quad E[e^{t\,X}] = 1 + t\,E[X] + \frac{t^2\,E[X]^2}{2!} + \frac{t^3\,E[X]^3}{3!} + \cdots]$$

$$\mathbb{E}(X^2)$$

$$\mathbb{E}(X^3)$$

$$\mathbb{E}(X^n)$$

Raw Moment

Short excursion
Moment-**Generating Functions** (MGFs)

$\mathbb{E}(X^0)$

$$E[e^{tX}] = \boxed{1} + t\boxed{E[X]} + \frac{t^2 \boxed{E[X]^2}}{2!} + \frac{t^3 \boxed{E[X]^3}}{3!} + \cdots$$

$\mathbb{E}(X^1)$

$\mathbb{E}(X^2)$

$\mathbb{E}(X^3)$

$\mathbb{E}(X^n)$

Raw Moment

Short excursion

Moment-**Generating Functions** (MGFs)

$$\mathbb{E}(X^0)$$

$$\mathbb{E}(X^1)$$

$$E[e^{t\,X}] = 1 + t\,E[X] + \frac{t^2\,E[X]^2}{2!} + \frac{t^3\,E[X]^3}{3!} + \cdots$$

$$\mathbb{E}(X^2)$$

$$\mathbb{E}(X^3)$$

$$\boxed{E\left(X^n\right)} = \left.\frac{d^n\,E[e^{tX}]}{dt^n}\right|_{t=0}$$

$$\mathbb{E}(X^n)$$

Short excursion

Moment-Generating Function

Take home message:

A Moment-generating function $M_X(t)$ is an alternative specification of the probability distribution

$$M_X(t) := E[e^{tX}]$$

Properties

- \+ Permutation invariant
- \+ Computationally efficient
- \+ Easy to optimize
- \+ Also works well with a large number of particles

Trainable vector **v**

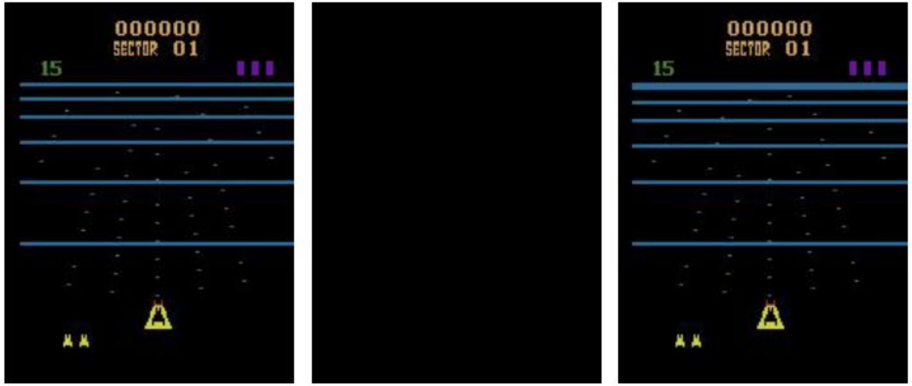
Allows model to extract useful moment features for decision making

Implementation in DPFRL

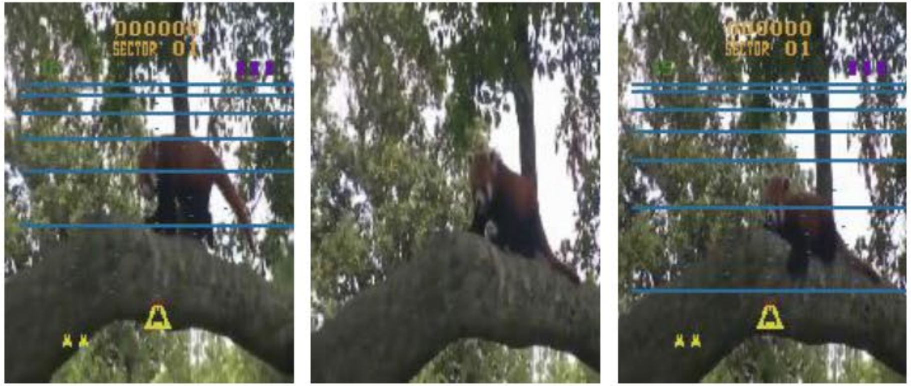
$$M_b = E[e^{\mathbf{v}^T h}] = \sum_i^K w_i e^{\mathbf{v}^T h_i}$$

DPFRL: Ablation study

Discriminative Particle Filter Reinforcement Learning



Flickering Atari Games



Natural Flickering Atari Games

DPFRL: Ablation study

Discriminative Particle Filter Reinforcement Learning

Environments	RNN	Mean	MGF
Envs	DPFRL-GRUmerge	DPFRL-mean	DPFRL
Pong	13.14±4.01	-5.53±14.35	**15.65±1.99**
ChopperCommand	1,530±29.31	1,091±109.9	**1,566±67.03**
MsPacman	1,930±48.54	1,878±63.86	2,106±123.9
Centipede	4,093±76.4	3,599±439.8	**4,164±23.0**
BeamRider	603.8±40.25	645.5±227.4	**682.9±37.42**
Frostbite	252.1±0.48	178.4±81.70	**260.2±4.60**
Bowling	**29.50±0.33**	26.0±0.81	**29.45±0.13**
IceHockey	-5.85±0.30	-6.25±1.96	-6.08±0.18
DDunk	-14.39±0.24	-14.42±0.18	-15.36±0.96
Asteroids	1,397±11.44	1,433±40.73	1,406±132.3

Results on the Natural Flickering Atari Games dataset

DPFRL: Ablation study

Discriminative Particle Filter Reinforcement Learning

Generative | **Discriminative**

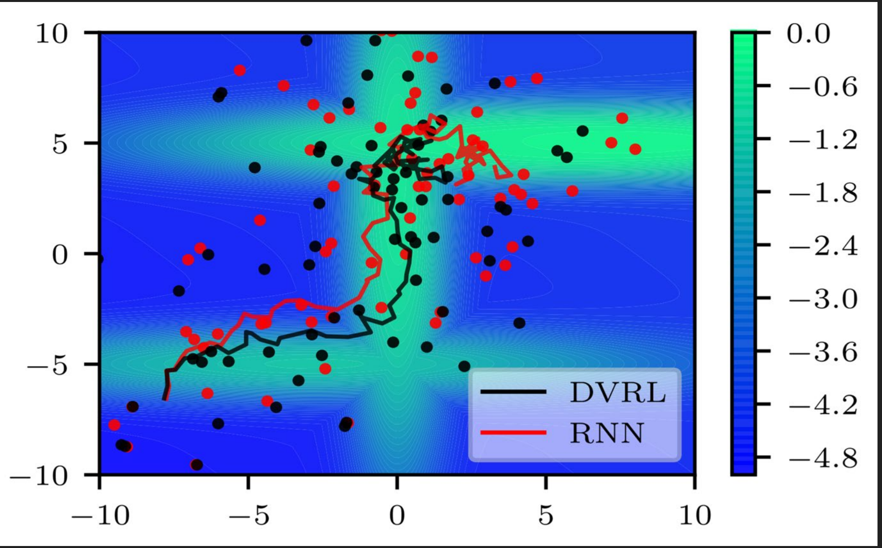
Environments

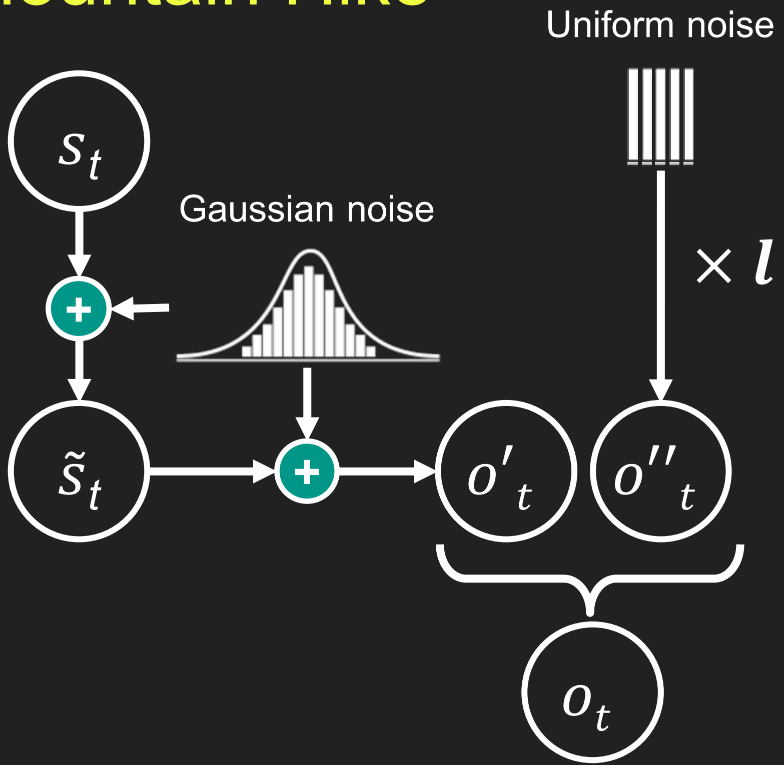
Envs	DPFRL-generative	DPFRL
Pong	-20.21±0.02	**15.65±1.99**
ChopperCommand	1,027± 12.94	**1,566±67.03**
MsPacman	2,130±182.3	2,106±123.9
Centipede	3,194±339.4	**4,164±23.0**
BeamRider	498.1±8.38	**682.9±37.42**
Frostbite	255.9±2.78	**260.2±4.60**
Bowling	24.68±0.13	**29.45±0.13**
IceHockey	-7.88±0.30	-6.08±0.18
DDunk	-15.59±0.06	-15.36±0.96
Asteroids	1,415±5.33	1,406±132.3

Results on the Natural Flickering Atari Games dataset

DPFRL: Noise robustness - Mountain Hike

Discriminative Particle Filter Reinforcement Learning



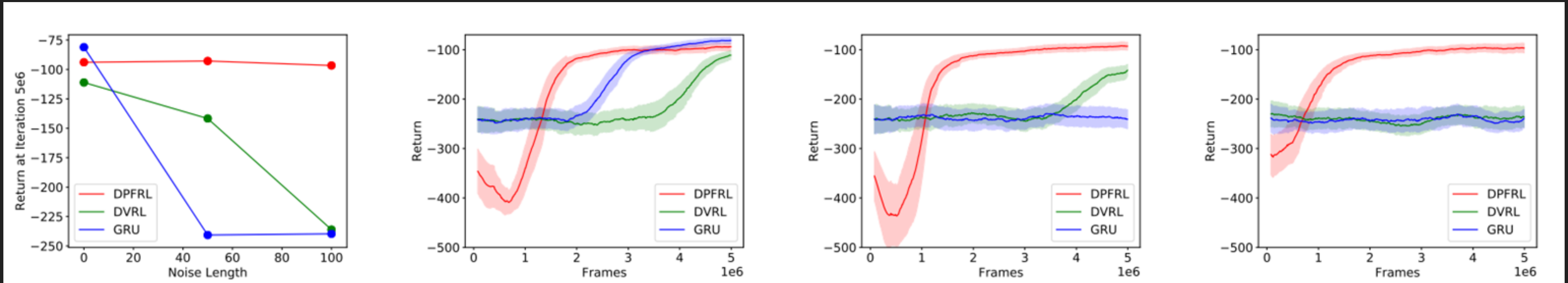


DPFRL: Noise robustness - Mountain Hike

Discriminative Particle Filter Reinforcement Learning

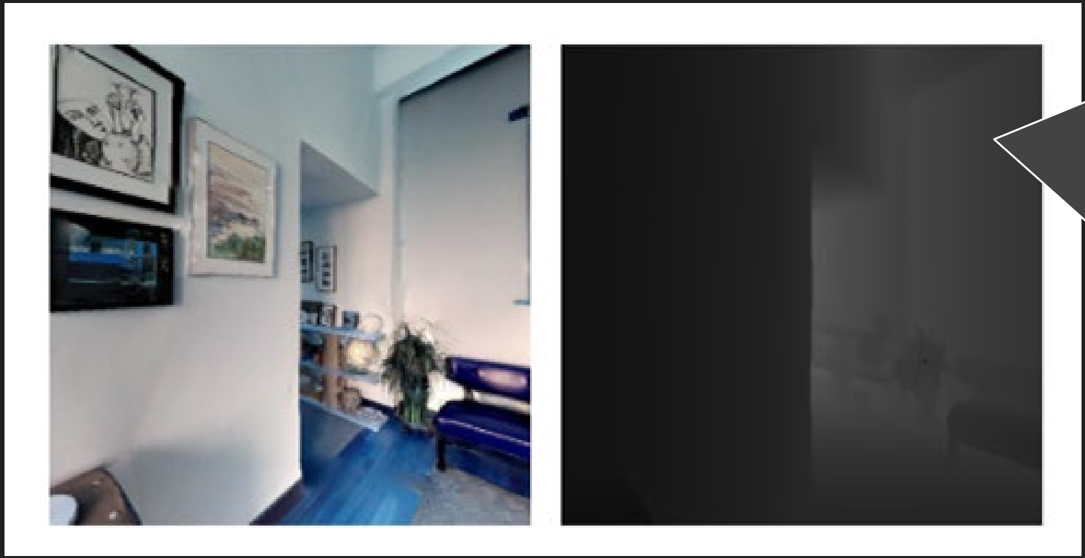
RNN **DVRL** **DPFRL**

Effect of varying l	$l = 0$	$l = 50$	$l = 100$



DPFRL: Real world environment

Discriminative Particle Filter Reinforcement Learning



Natural habitat dataset

Receiving a first-person RGB-D image in each step.

A robot needs to navigate in previously unseen environments.

DPFRL: Real world environment

Discriminative Particle Filter Reinforcement Learning

Results	SPL	Success Rate	Reward
DPFRL	**0.79**	**0.88**	**12.82±5.82**
DVRL	0.09	0.11	5.22±2.24
RNN	0.63	0.74	10.14±2.82

Natural habitat dataset

Receiving a first-person RGB-D image in each step.

A robot needs to navigate in previously unseen environments.

DPFRL: Summary

Advantages

+ No need to infer an observation model
+ Better robustness to noise in comparison to generative models
+ Simpler than DVRL

Weaknesses

- No reconstruction loss on observation restricts the learning signal
  - ➢ Limits sample efficiency and accuracy

POMDP

Approaches

Model free

RNN
DRQN
ADRQN

Explicit Belief tracking

DVRL
DPFRL

Implicit Belief tracking

VRN

Variational Recurrent Models

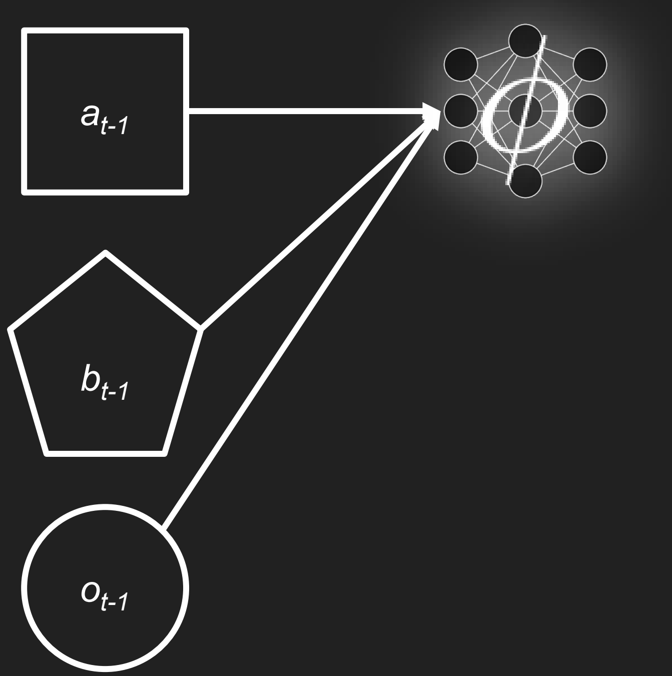
Objective

Piggyback on existing Algorithms for fully observable tasks

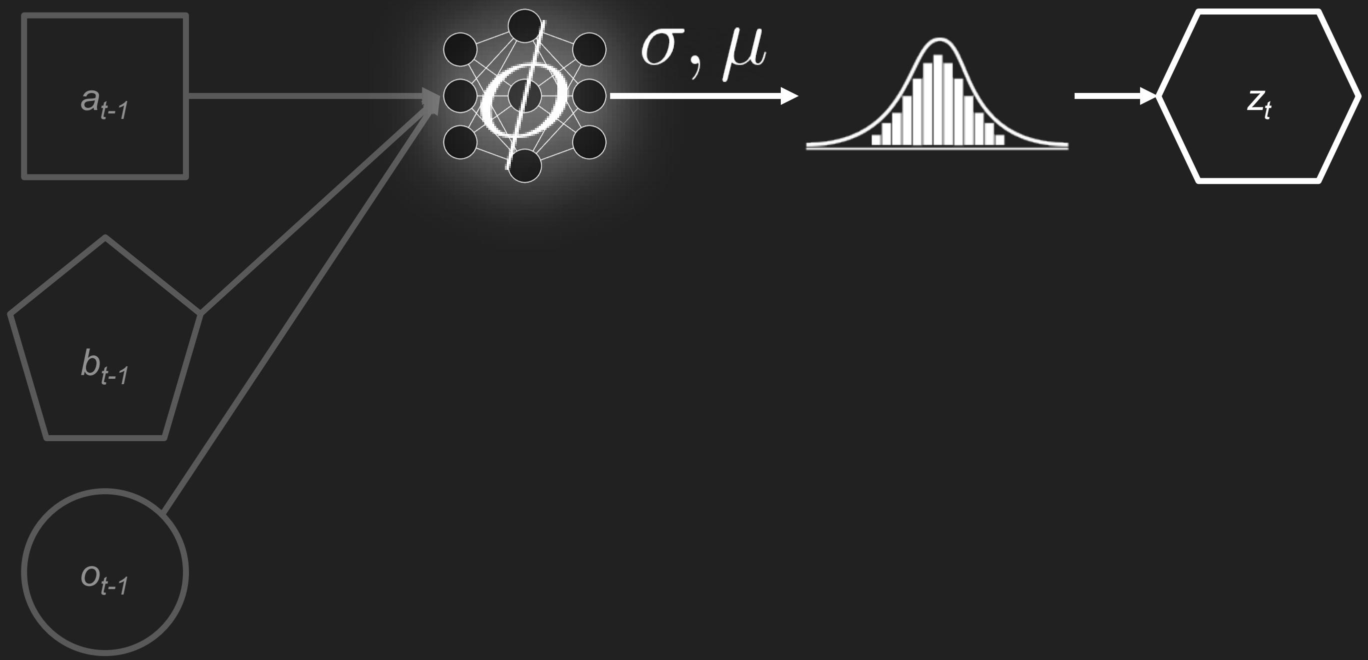
Idea

1. Conjecture a state *(Believe state)*
2. Solve the problem as if it was a Fully Observable MDP

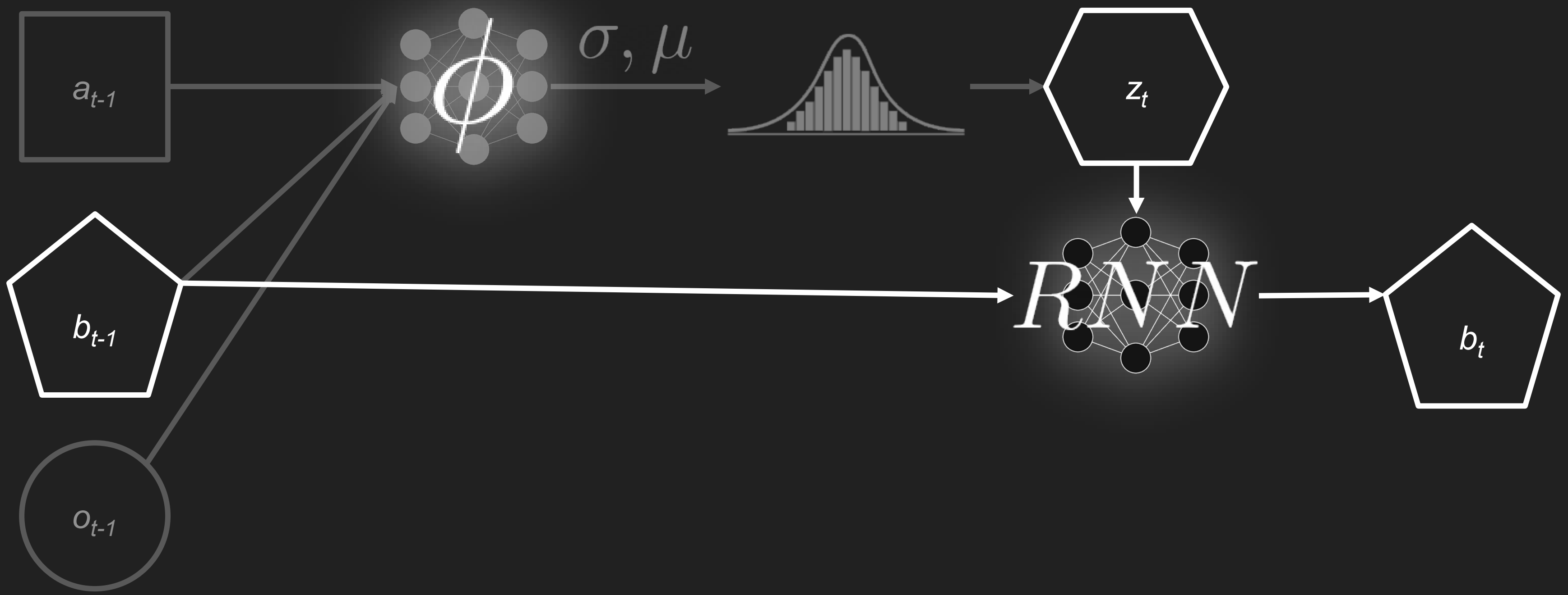
Variational Recurrent Models



Variational Recurrent Models



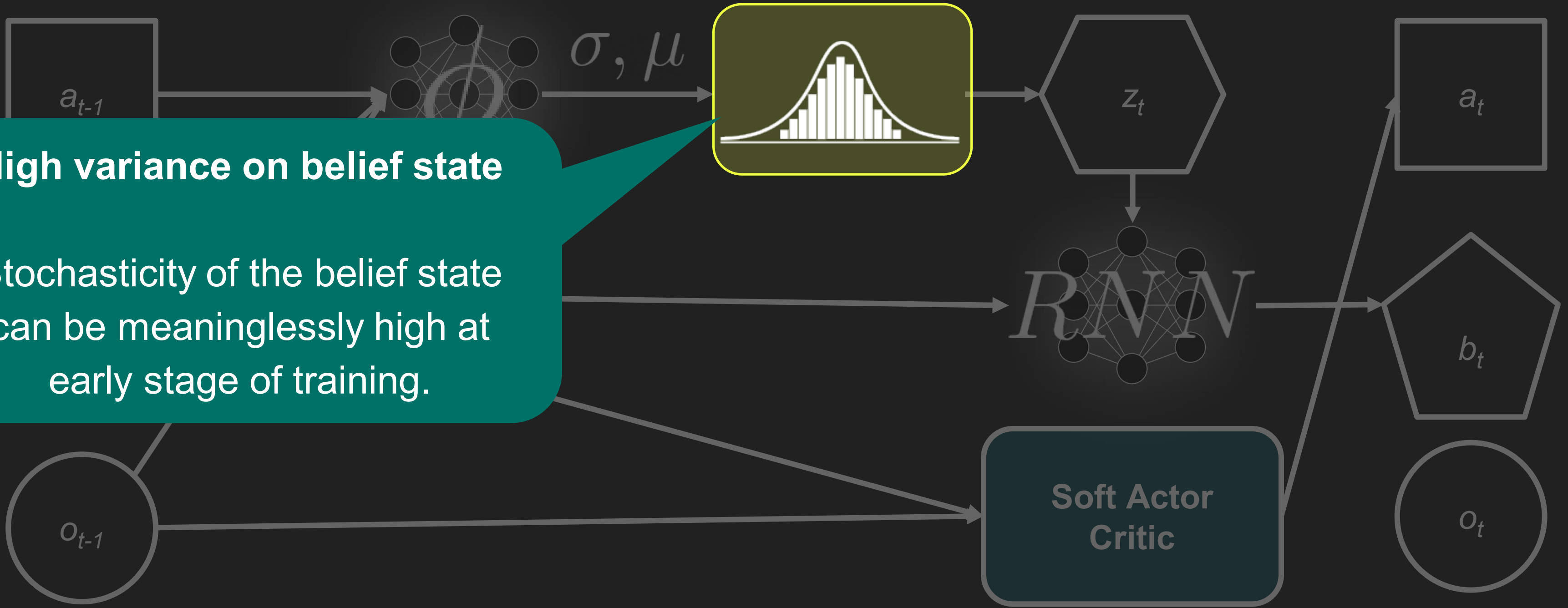
Variational Recurrent Models



Variational Recurrent Models

Variational Recurrent Models

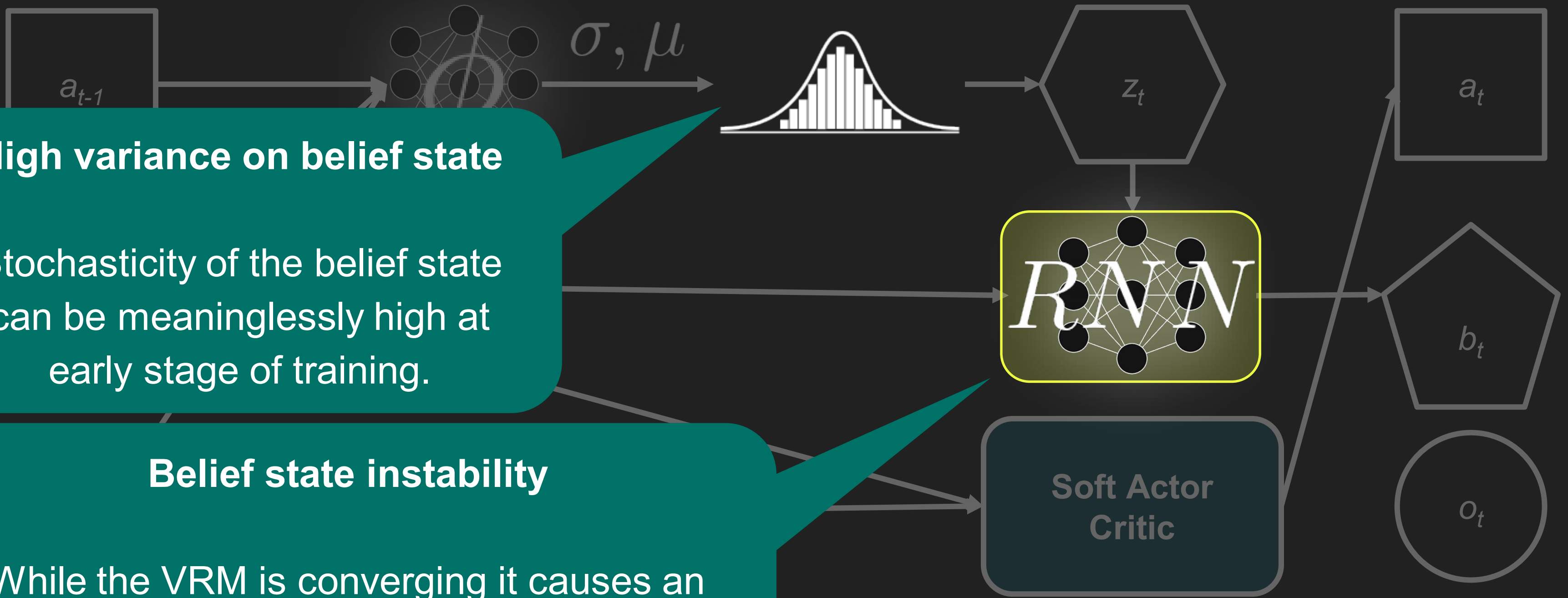
Variational Recurrent Models: Challenges



High variance on belief state

Stochasticity of the belief state can be meaninglessly high at early stage of training.

Variational Recurrent Models: Challenges



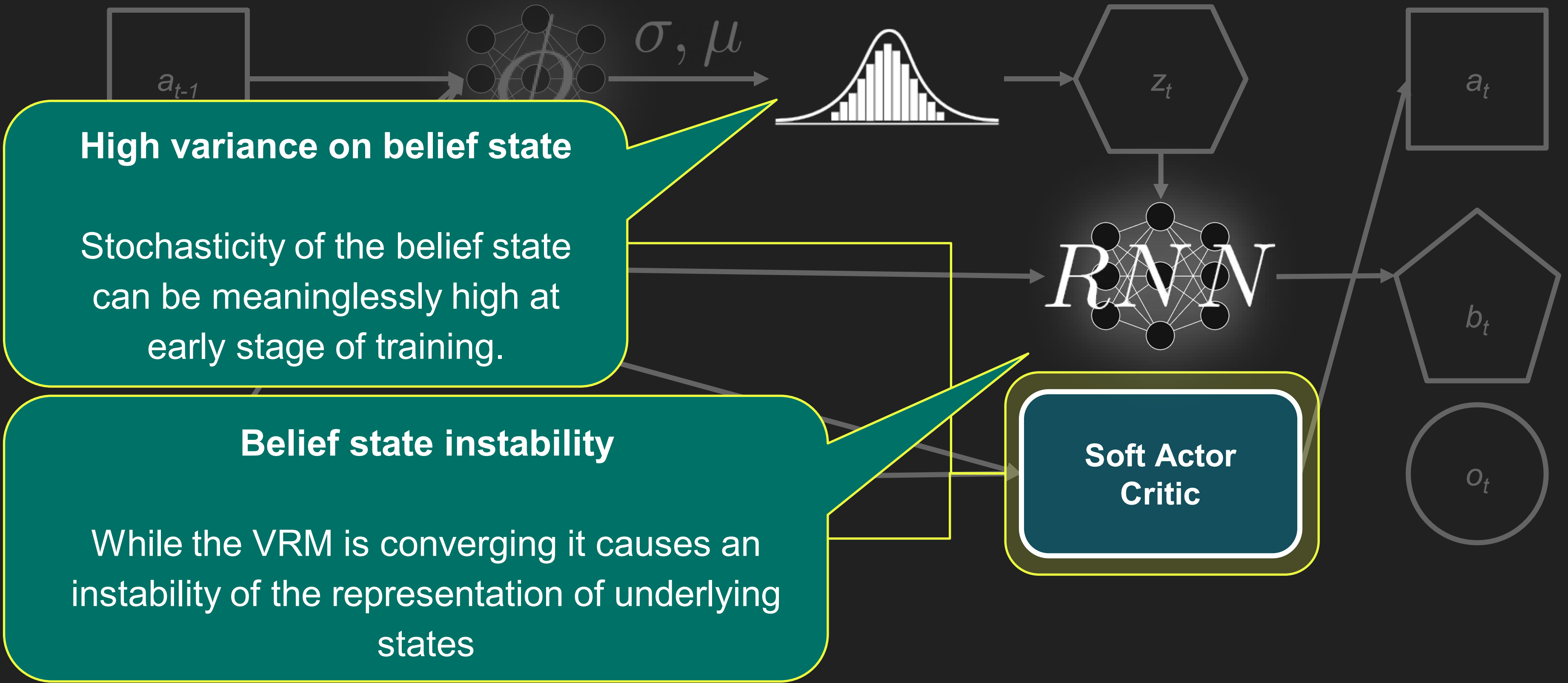
High variance on belief state

Stochasticity of the belief state can be meaninglessly high at early stage of training.

Belief state instability

While the VRM is converging it causes an instability of the representation of underlying states

Variational Recurrent Models: Challenges



High variance on belief state

Stochasticity of the belief state can be meaninglessly high at early stage of training.

Belief state instability

While the VRM is converging it causes an instability of the representation of underlying states

Variational Recurrent Models: Solution

High variance on belief state

VRM

First-impression model

- Keeps the representation stable for the RL during training
- Pre-trained and not updated during exploration

Keep-learning model

- Learn from new observations obtained after the policy update by the RL
- Trained during exploration

states

Variational Recurrent Models

Algorithm

Algorithm 1 **Variational Recurrent Models with Soft Actor Critic**

Initialize the first-impression VRM $\mathcal{M}_f$ and the keep-learning VRM $\mathcal{M}_k$, the RL controller $\mathcal{C}$, and the replay buffer $\mathcal{D}$, global step $t \leftarrow 0$.

repeat

  Initialize an episode, assign $\mathcal{M}$ with zero initial states.

  while episode not terminated **do**

    Sample an action $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{d}_t, \boldsymbol{x}_t)$ and execute a_t, $t \leftarrow t + 1$.

    Record $(\boldsymbol{x}_t, \boldsymbol{a}_t, done_t)$ into $\mathcal{B}$.

    Compute 1-step forward of both VRMs using inference models.

    if $t == step\_start\_RL$ **then**

      For N epochs, sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_f$ (Eq. 11).

    end if

    if $t > step\_start\_RL$ and $mod(t, train\_interval\_KLVRM) == 0$ **then**

      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_k$ (Eq. 5, 6, 7, 8) .

    end if

    if $t > step\_start\_RL$ and $mod(t, train\_interval\_RL) == 0$ **then**

      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{R}$ (Eq. 11) .

    end if

  end while

until training stopped

Initialization

Variational Recurrent Models

Algorithm

Algorithm 1 **Variational Recurrent Models with Soft Actor Critic**

Initialize the first-impression VRM $\mathcal{M}_f$ and the keep-learning VRM $\mathcal{M}_k$, the RL controller $\mathcal{C}$, and the replay buffer $\mathcal{D}$, global step $t \leftarrow 0$.
repeat
  Initialize an episode, assign $\mathcal{M}$ with zero initial states.
  while episode not terminated **do**
    Sample an action $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{d}_t, \boldsymbol{x}_t)$ and execute a_t, $t \leftarrow t + 1$.
    Record $(\boldsymbol{x}_t, \boldsymbol{a}_t, done_t)$ into $\mathcal{B}$.
    Compute 1-step forward of both VRMs using inference models.
    if $t == step\_start\_RL$ **then**
      For N epochs, sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_f$ (Eq. 11).
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_KLVRM) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_k$ (Eq. 5, 6, 7, 8) .
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_RL) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{R}$ (Eq. 11) .
    end if
  end while
until training stopped

Fill the replay buffer

Variational Recurrent Models

Algorithm

Algorithm 1 **Variational Recurrent Models with Soft Actor Critic**

Initialize the first-impression VRM $\mathcal{M}_f$ and the keep-learning VRM $\mathcal{M}_k$, the RL controller $\mathcal{C}$, and the replay buffer $\mathcal{D}$, global step $t \leftarrow 0$.
repeat
  Initialize an episode, assign $\mathcal{M}$ with zero initial states.
  while episode not terminated **do**
    Sample an action $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{d}_t, \boldsymbol{x}_t)$ and execute a_t, $t \leftarrow t+1$.
    Record $(\boldsymbol{x}_t, \boldsymbol{a}_t, done_t)$ into $\mathcal{B}$.
    Compute 1-step forward of both VRMs using inference models.
    if $t == step\_start\_RL$ **then**
      For N epochs, sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_f$ (Eq. 11).
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_KLVRM) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_k$ (Eq. 5, 6, 7, 8) .
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_RL) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{R}$ (Eq. 11) .
    end if
  end while
until training stopped

Train first-impression Model

Variational Recurrent Models

Algorithm

Algorithm 1 **Variational Recurrent Models with Soft Actor Critic**

Initialize the first-impression VRM $\mathcal{M}_f$ and the keep-learning VRM $\mathcal{M}_k$, the RL controller $\mathcal{C}$, and the replay buffer $\mathcal{D}$, global step $t \leftarrow 0$.

repeat

  Initialize an episode, assign $\mathcal{M}$ with zero initial states.

  while episode not terminated **do**

    Sample an action $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{d}_t, \boldsymbol{x}_t)$ and execute a_t, $t \leftarrow t+1$.

    Record $(\boldsymbol{x}_t, \boldsymbol{a}_t, done_t)$ into $\mathcal{B}$.

    Compute 1-step forward of both VRMs using inference models.

    if $t == step\_start\_RL$ **then**

      For N epochs, sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_f$ (Eq. 11).

    end if

    if $t > step\_start\_RL$ and $mod(t, train\_interval\_KLVRM) == 0$ **then**

      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_k$ (Eq. 5, 6, 7, 8) .

    end if

    if $t > step\_start\_RL$ and $mod(t, train\_interval\_RL) == 0$ **then**

      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{R}$ (Eq. 11) .

    end if

  end while

until training stopped

Train keep-learning Model

Variational Recurrent Models

Algorithm

Algorithm 1 Variational Recurrent Models with Soft Actor Critic

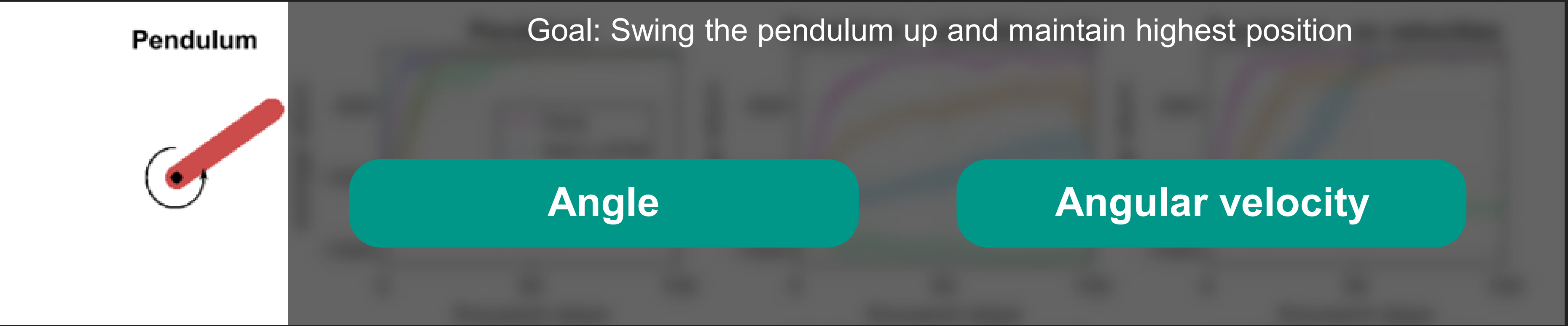
Initialize the first-impression VRM $\mathcal{M}_f$ and the keep-learning VRM $\mathcal{M}_k$, the RL controller $\mathcal{C}$, and the replay buffer $\mathcal{D}$, global step $t \leftarrow 0$.
repeat
  Initialize an episode, assign $\mathcal{M}$ with zero initial states.
  while episode not terminated **do**
    Sample an action $\boldsymbol{a}_t$ from $\pi(\boldsymbol{a}_t|\boldsymbol{d}_t, \boldsymbol{x}_t)$ and execute a_t, $t \leftarrow t+1$.
    Record $(\boldsymbol{x}_t, \boldsymbol{a}_t, done_t)$ into $\mathcal{B}$.
    Compute 1-step forward of both VRMs using inference models.
    if $t == step\_start\_RL$ **then**
      For N epochs, sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_f$ (Eq. 11).
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_KLVRM) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{M}_k$ (Eq. 5, 6, 7, 8) .
    end if
    if $t > step\_start\_RL$ and $mod(t, train\_interval\_RL) == 0$ **then**
      Sample a minibatch of samples from $\mathcal{B}$ to update $\mathcal{R}$ (Eq. 11) .
    end if
  end while
until training stopped

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

Variational Recurrent Models

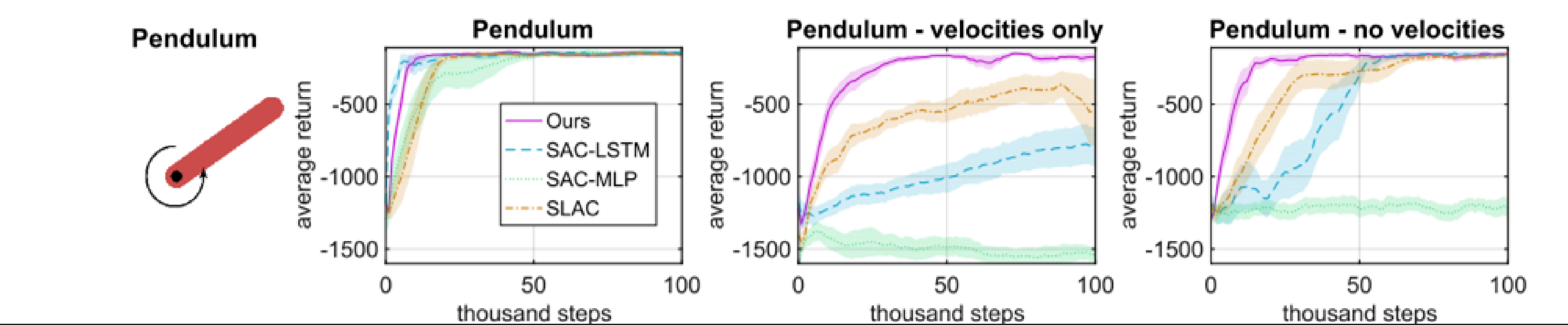
Results



Variational Recurrent Models

Results

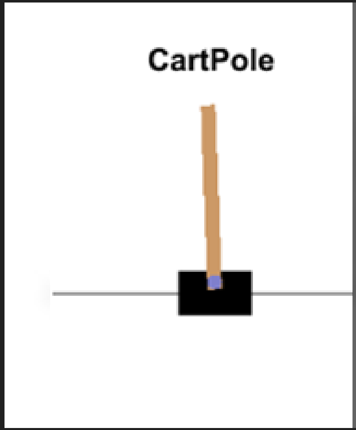
VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**



Variational Recurrent Models

Results

VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**



Goal: Balance pole upright without running away

Position

Velocity

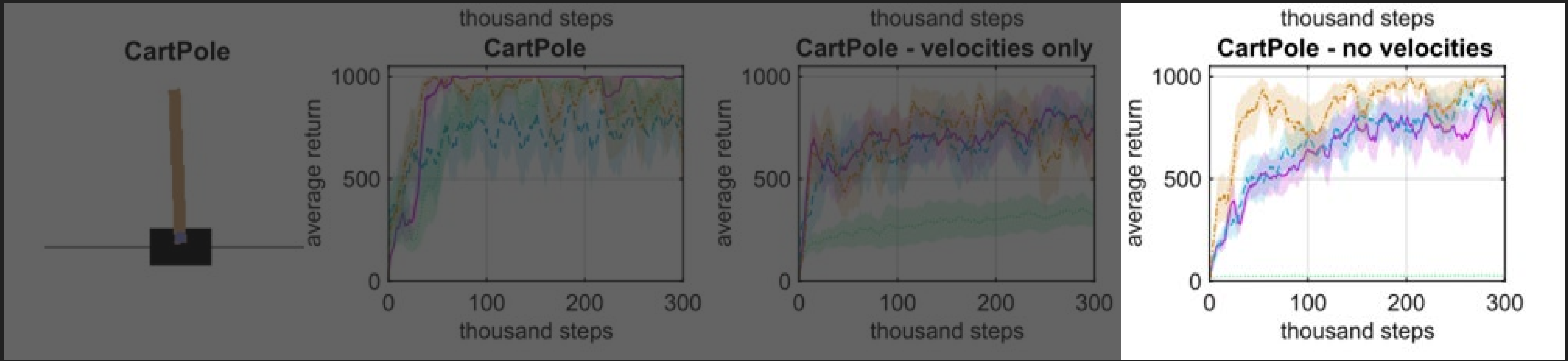
Angle

Angular velocity

Variational Recurrent Models

Results

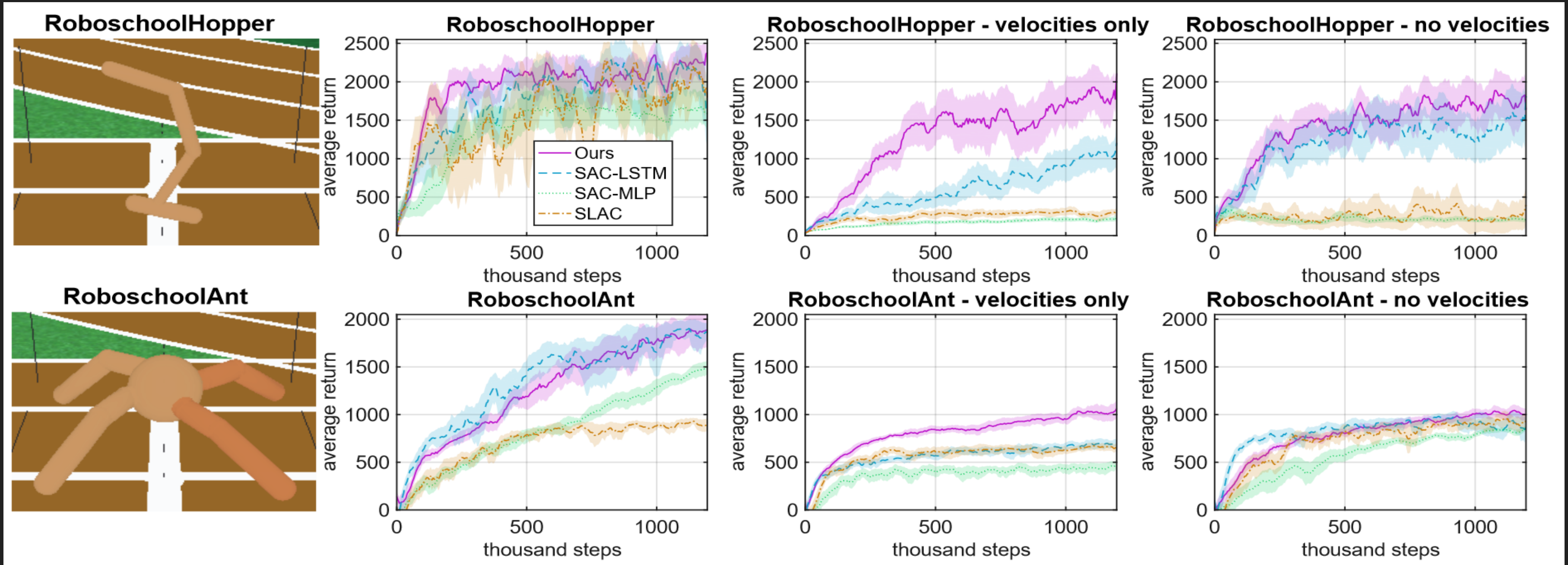
VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**



Variational Recurrent Models

Results

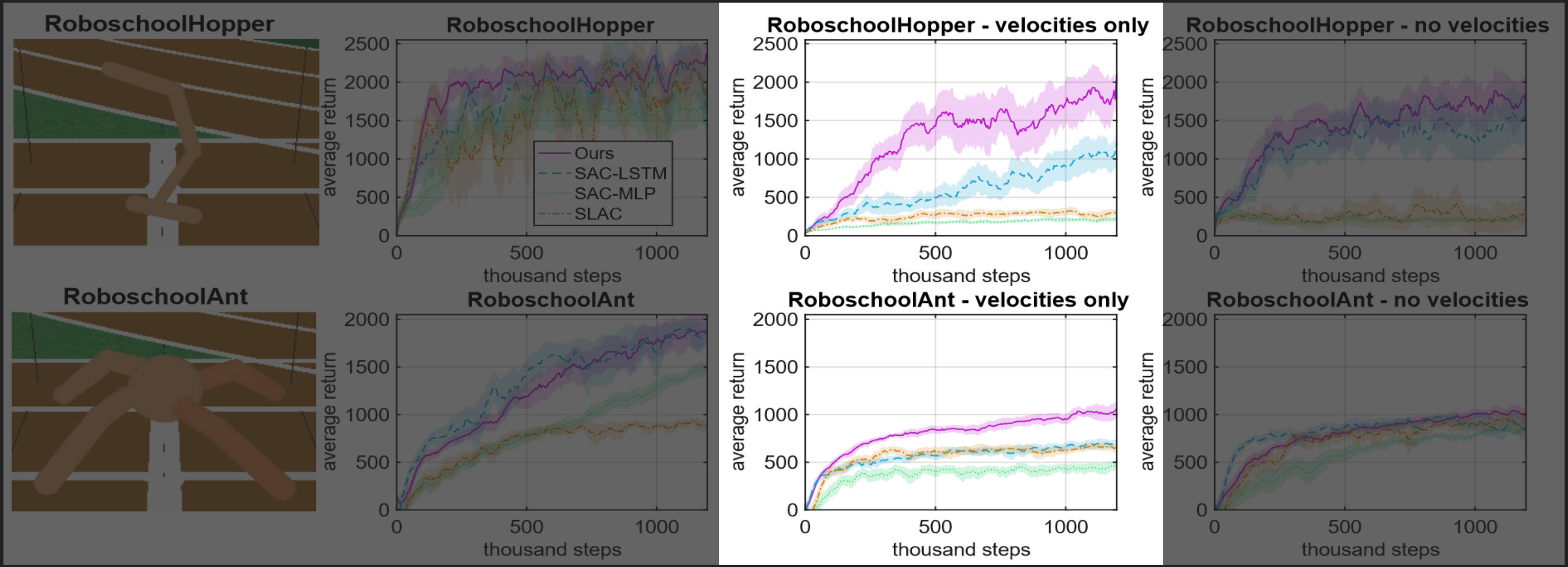
VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**



Variational Recurrent Models

Results

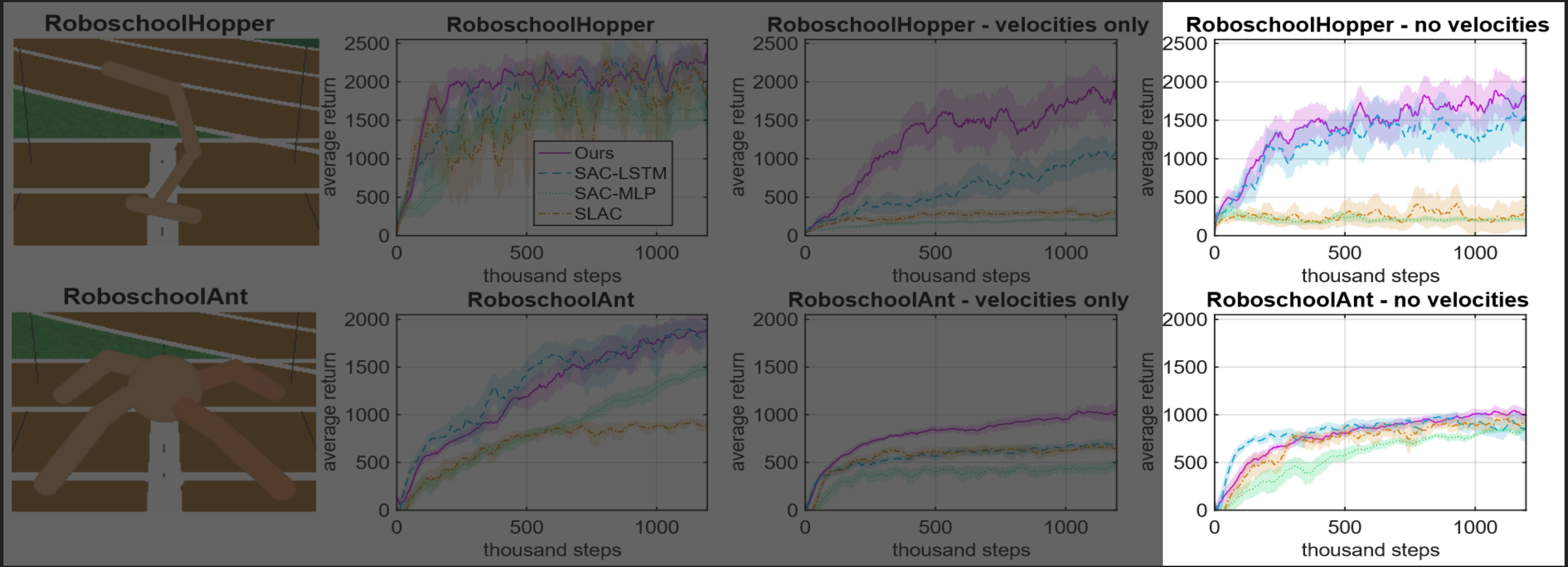
VRM **SAC-MLP** **SAC-RNN** **SLAC**



Variational Recurrent Models

Results

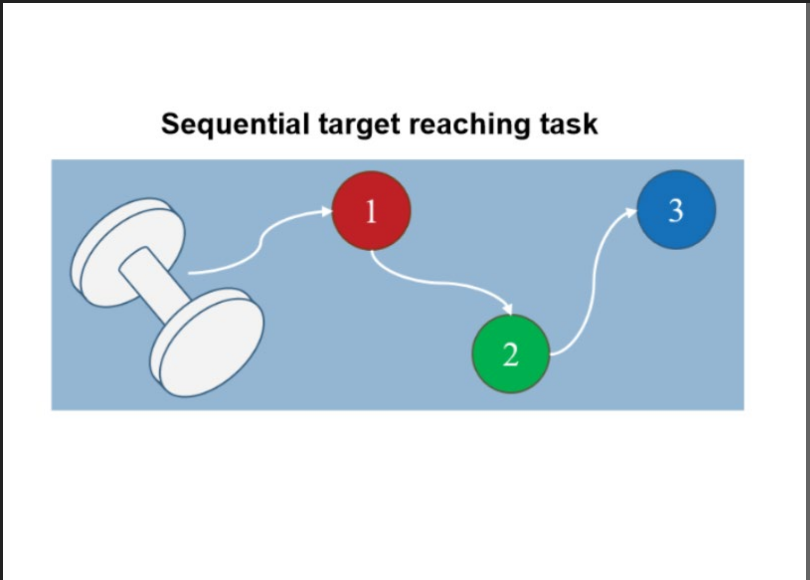
VRM **SAC-MLP** **SAC-RNN** **SLAC**



Variational Recurrent Models

Results

VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**



Goal

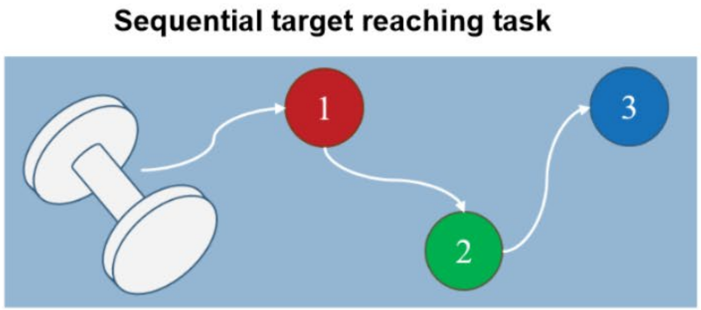
An agent needs to reach 3 different targets in a certain sequence

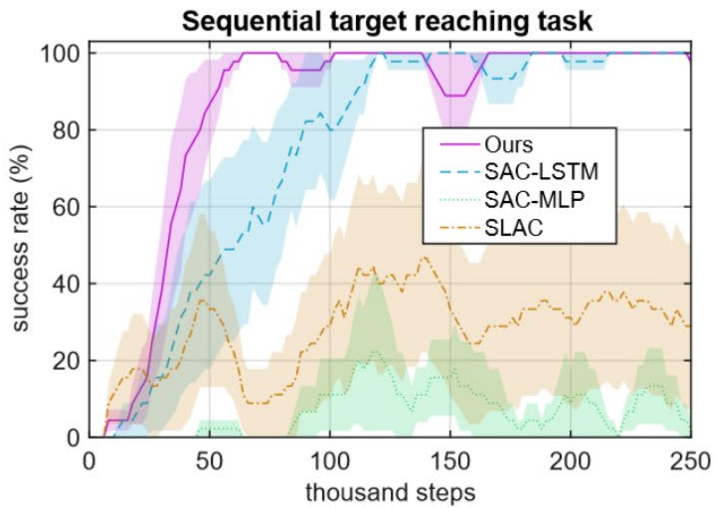
Requires long-term memorization of past events

Variational Recurrent Models

Results

VRM | **SAC-MLP** | **SAC-RNN** | **SLAC**





Variational Recurrent Models (ICLR 2020)
Summary

Advantages

+ Simple to implement
+ Gaussian instead of Particle Filter
  - ➢ More sample efficient
  - ➢ Includes prior knowledge on belief distribution
+ Can make use of advancements in Fully Observable MDP tasks

Weaknesses

- The training of a RNN is still data intensive
- The use of a Gaussian distribution makes the model less suitable for general POMDPs
- The model was only tested against Model-free algorithms

Variational Recurrent Models (ICLR 2020)

Summary

Advantages

+ ...
+ ...

...includes prior knowledge on belief distribution

+ Can make use of advancements in Fully Observable MDP tasks

Weaknesses

- ...ta
- ...on ...or
- The model was only tested against Model-free algorithms

Personal note

VRM seems to aim at solving a POMDP where the underlying state is not stochastic in nature. The variational part in the VRNN may mostly help to get a continuous latent variable like it is the case for a VAE.

POMDP

Approaches

Model free

RNN
DRQN
ADRQN

Explicit Belief tracking

DVRL
DPFRL

Implicit Belief tracking

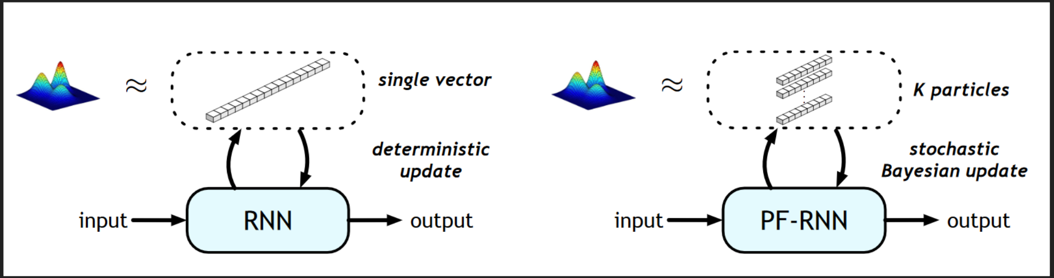
VRN

Belief

VRN	DVRL	DPFRL
Belief tracked by RNN	Belief tracked by Particle and summarized by RNN	Belief tracked by Particle and summarized by MGF
Belief updated using a Gaussian	Belief updated using a latent representation of the observation	Belief updated using discriminative function

POMDP

Further discussion



Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

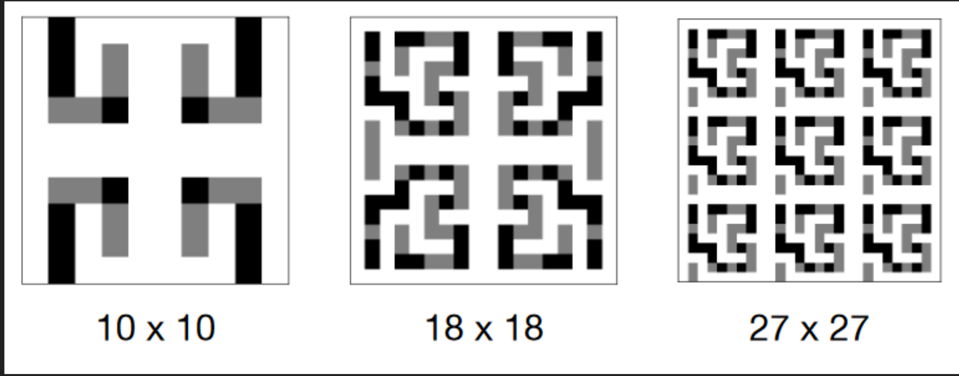
Introduction

Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

thus increasing the number of network param
amount of data required for training.

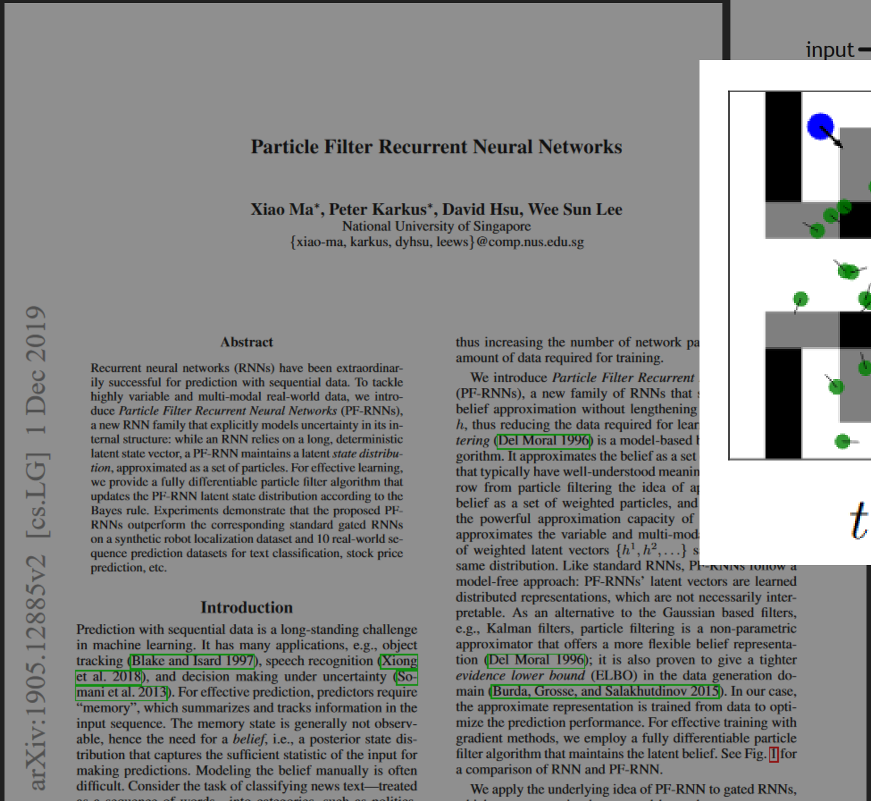
We introduce *Particle Filter Recurrent Neu*
(PF-RNNs), a new family of RNNs that seek
belief approximation without lengthening the
h, thus reducing the data required for learning
tering (Del Moral 1996) is a model-based belie
gorithm. It approximates the belief as a set of s
that typically have well-understood meaning. P
row from particle filtering the idea of appro
belief as a set of weighted particles, and con
the powerful approximation capacity of RN
approximates the variable and multi-modal b
of weighted latent vectors $\{h^1, h^2, \ldots\}$ samp
same distribution. Like standard RNNs, PF-R
model-free approach: PF-RNNs' latent vector
distributed representations, which are not nec
pretable. As an alternative to the Gaussian
e.g., Kalman filters, particle filtering is a no
approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,



POMDP

Further discussion



Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

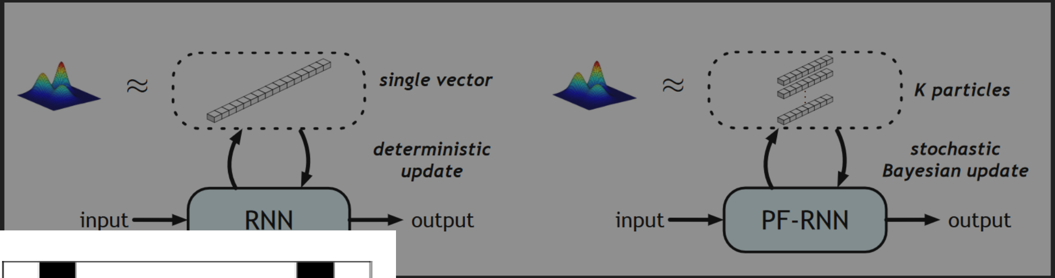
Introduction

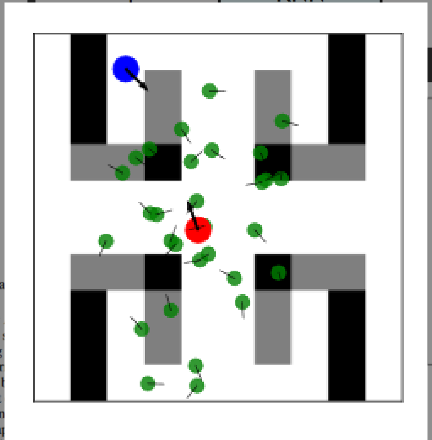
Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

thus increasing the number of network pa
amount of data required for training.

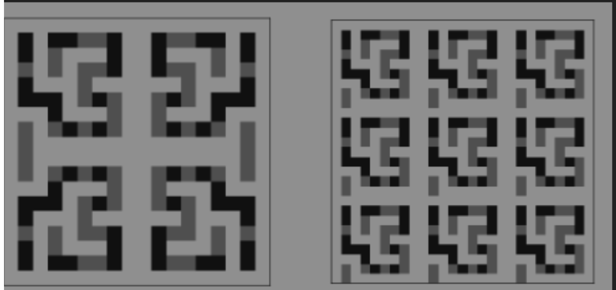
We introduce *Particle Filter Recurrent*
(PF-RNNs), a new family of RNNs that
belief approximation without lengthening
h, thus reducing the data required for lear
tering (Del Moral 1996) is a model-based
gorithm. It approximates the belief as a set
that typically have well-understood meanin
row from particle filtering the idea of ap
belief as a set of weighted particles, and
the powerful approximation capacity of
approximates the variable and multi-mod
of weighted latent vectors $\{h^1, h^2, \ldots\}$ s
same distribution. Like standard RNNs, PF-RNNs follow a model-free approach: PF-RNNs' latent vectors are learned distributed representations, which are not necessarily interpretable. As an alternative to the Gaussian based filters, e.g., Kalman filters, particle filtering is a non-parametric approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,





$t = 0$



18 x 18

27 x 27

POMDP

Further discussion

arXiv:1905.12885v2 [cs.LG] 1 Dec 2019

Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

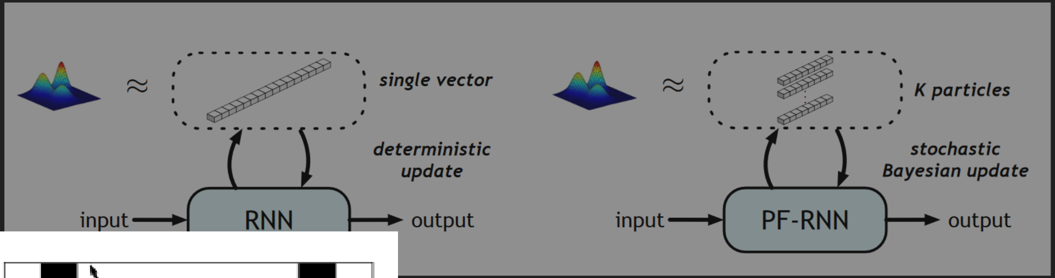
Introduction

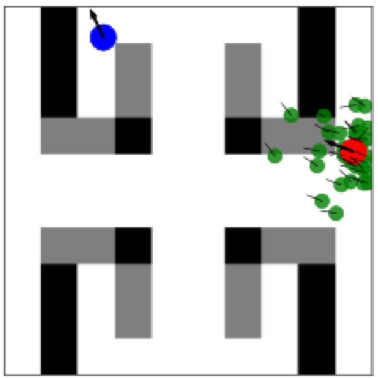
Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

thus increasing the number of network pa amount of data required for training.

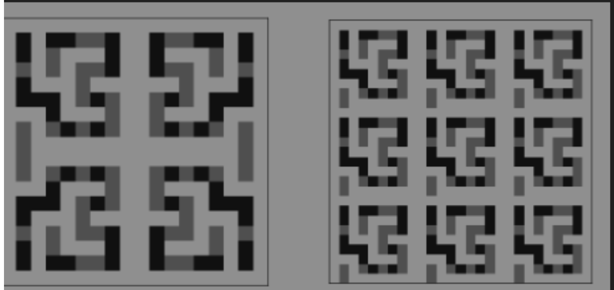
We introduce *Particle Filter Recurrent* (PF-RNNs), a new family of RNNs that belief approximation without lengthening h, thus reducing the data required for lear *tering* (Del Moral 1996) is a model-based gorithm. It approximates the belief as a set that typically have well-understood meanin row from particle filtering the idea of a belief as a set of weighted particles, and the powerful approximation capacity of approximates the variable and multi-mod of weighted latent vectors $\{h^1, h^2, \ldots\}$ same distribution. Like standard RNNs, PF-RNNs follow a model-free approach: PF-RNNs' latent vectors are learned distributed representations, which are not necessarily interpretable. As an alternative to the Gaussian based filters, e.g., Kalman filters, particle filtering is a non-parametric approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,





$t = 6$



18 x 18

27 x 27

POMDP

Further discussion

arXiv:1905.12885v2 [cs.LG] 1 Dec 2019

Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

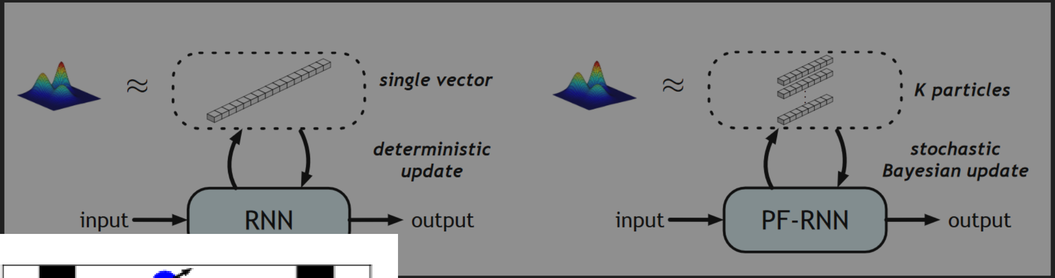
Introduction

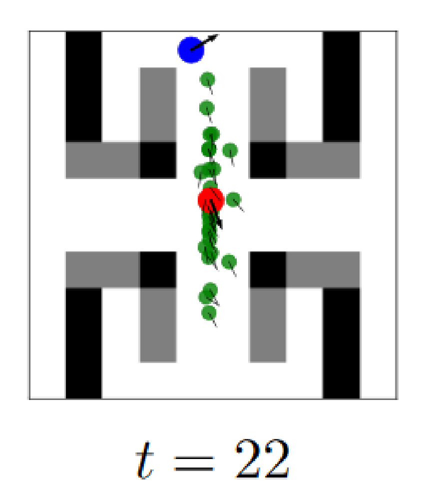
Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

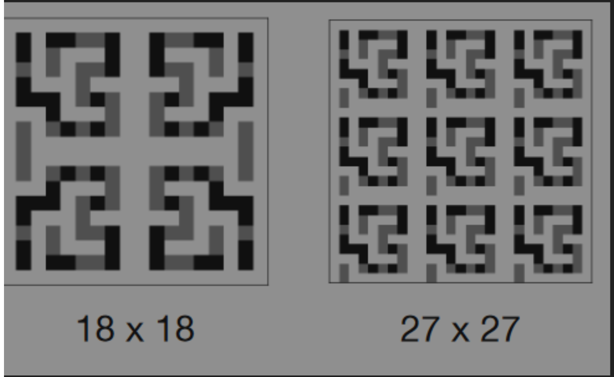
thus increasing the number of network pa
amount of data required for training.

We introduce *Particle Filter Recurrent*
(PF-RNNs), a new family of RNNs that
belief approximation without lengthening
h, thus reducing the data required for lear
tering (Del Moral 1996) is a model-based
gorithm. It approximates the belief as a set
that typically have well-understood meanin
row from particle filtering the idea of ap
belief as a set of weighted particles, and
the powerful approximation capacity of
approximates the variable and multi-mod
of weighted latent vectors $\{h^1, h^2, \ldots\}$ s
same distribution. Like standard RNNs, PF-RNNs follow a model-free approach: PF-RNNs' latent vectors are learned distributed representations, which are not necessarily interpretable. As an alternative to the Gaussian based filters, e.g., Kalman filters, particle filtering is a non-parametric approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,







POMDP

Further discussion

arXiv:1905.12885v2 [cs.LG] 1 Dec 2019

Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

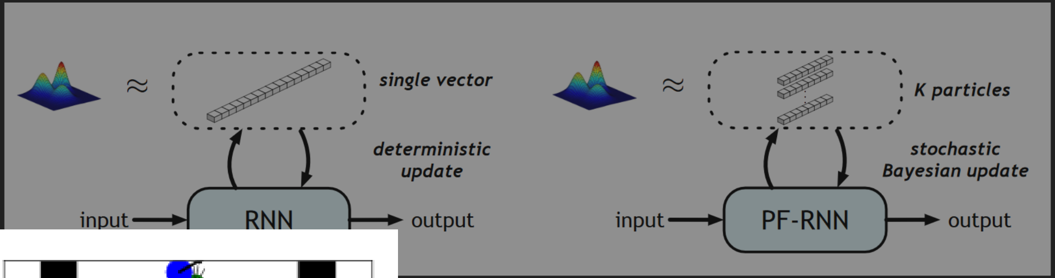
Introduction

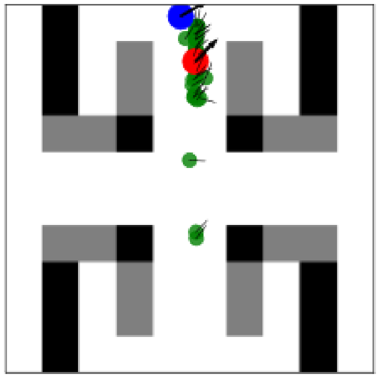
Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

thus increasing the number of network pa
amount of data required for training.

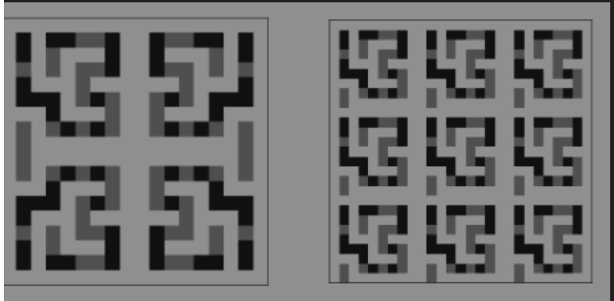
We introduce *Particle Filter Recurrent*
(PF-RNNs), a new family of RNNs that
belief approximation without lengthening
h, thus reducing the data required for lear
tering (Del Moral 1996) is a model-based
gorithm. It approximates the belief as a set
that typically have well-understood meanin
row from particle filtering the idea of ap
belief as a set of weighted particles, and
the powerful approximation capacity of
approximates the variable and multi-mod
of weighted latent vectors $\{h^1, h^2, \ldots\}$
same distribution. Like standard RNNs, PF-RNNs follow a model-free approach: PF-RNNs' latent vectors are learned distributed representations, which are not necessarily interpretable. As an alternative to the Gaussian based filters, e.g., Kalman filters, particle filtering is a non-parametric approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,





$t = 24$



18 x 18

27 x 27

POMDP

Further discussion

Particle Filter Recurrent Neural Networks

Xiao Ma*, Peter Karkus*, David Hsu, Wee Sun Lee
National University of Singapore
{xiao-ma, karkus, dyhsu, leews}@comp.nus.edu.sg

Abstract

Recurrent neural networks (RNNs) have been extraordinarily successful for prediction with sequential data. To tackle highly variable and multi-modal real-world data, we introduce *Particle Filter Recurrent Neural Networks* (PF-RNNs), a new RNN family that explicitly models uncertainty in its internal structure: while an RNN relies on a long, deterministic latent state vector, a PF-RNN maintains a latent *state distribution*, approximated as a set of particles. For effective learning, we provide a fully differentiable particle filter algorithm that updates the PF-RNN latent state distribution according to the Bayes rule. Experiments demonstrate that the proposed PF-RNNs outperform the corresponding standard gated RNNs on a synthetic robot localization dataset and 10 real-world sequence prediction datasets for text classification, stock price prediction, etc.

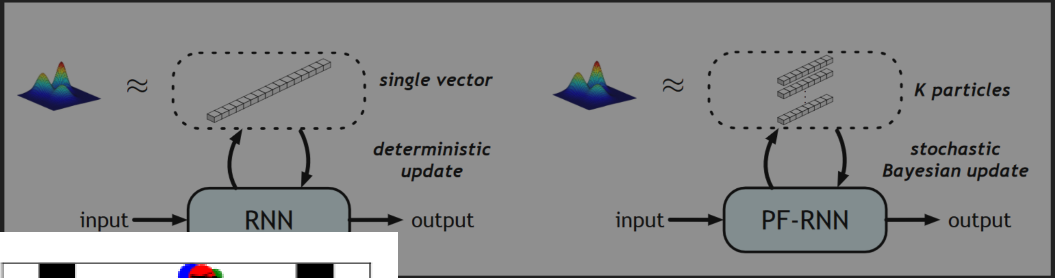
Introduction

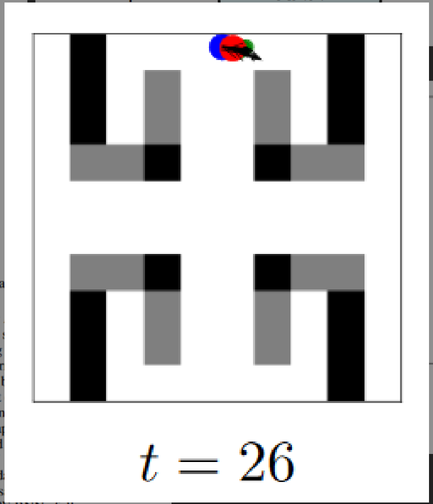
Prediction with sequential data is a long-standing challenge in machine learning. It has many applications, e.g., object tracking (Blake and Isard 1997), speech recognition (Xiong et al. 2018), and decision making under uncertainty (Somani et al. 2013). For effective prediction, predictors require "memory", which summarizes and tracks information in the input sequence. The memory state is generally not observable, hence the need for a *belief*, i.e., a posterior state distribution that captures the sufficient statistic of the input for making predictions. Modeling the belief manually is often difficult. Consider the task of classifying news text—treated

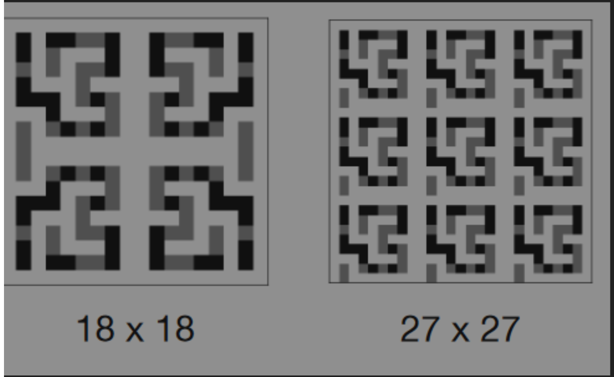
thus increasing the number of network pa
amount of data required for training.

We introduce *Particle Filter Recurrent*
(PF-RNNs), a new family of RNNs that
belief approximation without lengthening
h, thus reducing the data required for lear
tering (Del Moral 1996) is a model-based
gorithm. It approximates the belief as a set
that typically have well-understood meanin
row from particle filtering the idea of ap
belief as a set of weighted particles, and
the powerful approximation capacity of
approximates the variable and multi-mod
of weighted latent vectors $\{h^1, h^2, \ldots\}$ s
same distribution. Like standard RNNs, PF-RNNs follow a model-free approach: PF-RNNs' latent vectors are learned distributed representations, which are not necessarily interpretable. As an alternative to the Gaussian based filters, e.g., Kalman filters, particle filtering is a non-parametric approximator that offers a more flexible belief representation (Del Moral 1996); it is also proven to give a tighter *evidence lower bound* (ELBO) in the data generation domain (Burda, Grosse, and Salakhutdinov 2015). In our case, the approximate representation is trained from data to optimize the prediction performance. For effective training with gradient methods, we employ a fully differentiable particle filter algorithm that maintains the latent belief. See Fig. 1 for a comparison of RNN and PF-RNN.

We apply the underlying idea of PF-RNN to gated RNNs,

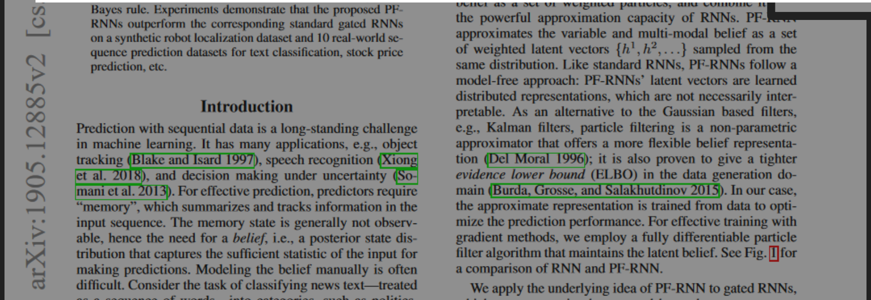


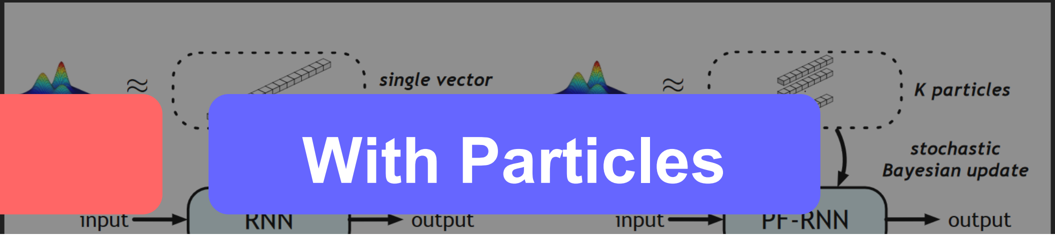




POMDP

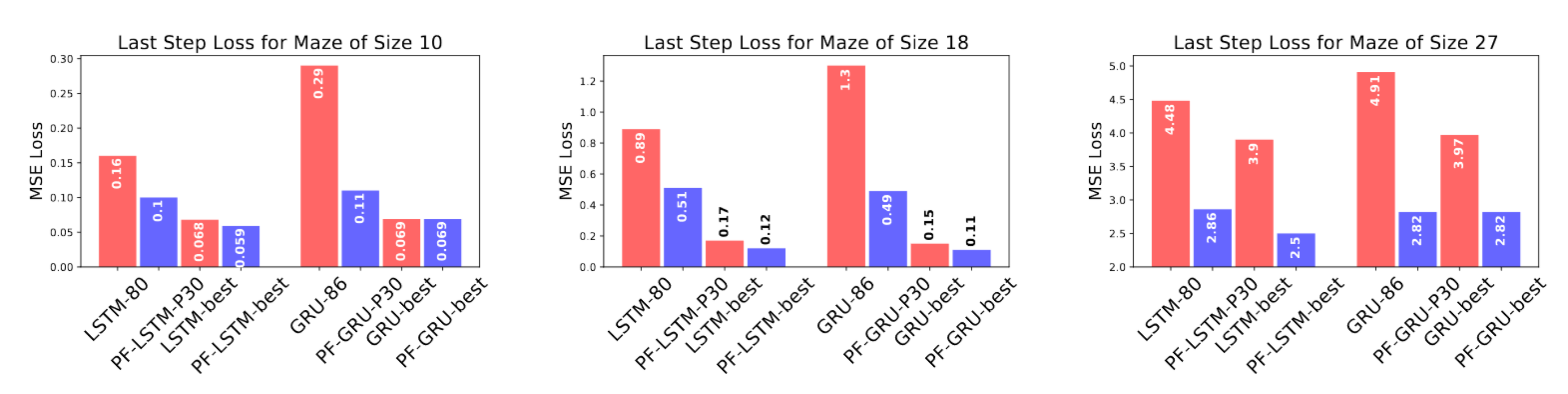
Further disc





Classic

With Particles



Thank you for your attention

Discussion

1) Are POMDPs with a deterministic state transition a field worth more research?

VRM is a model based RL algorithm. However the results where not always very disincentive to RNNs.

2) How can we be confident that an algorithm truly learned a model?