PlaNet

Learning Latent Dynamics for Planning from Pixels

PlaNet Deep Planning Network

- Scalable Model-based RL
- Efficint planning in latent space with large batch size
- Reaches top performance using 200X fewer episodes

PlaNet Deep Planning Network

■ Recurrent State Space Model (RSSM)

Deterministic & Stochastic Components

■ Latent Overshooting

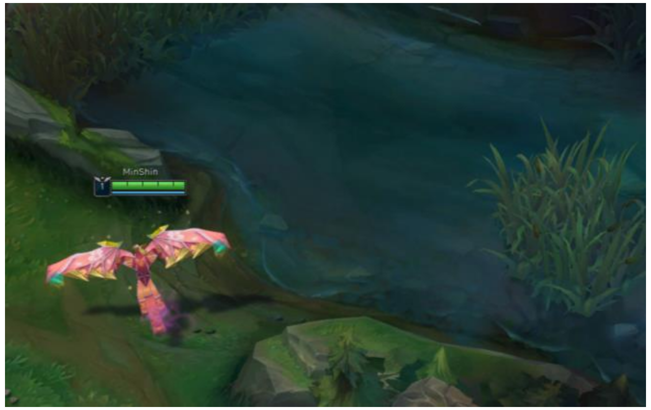
Latent Space Result로 Latent Sequence Model을 굴리는게 장기예측에 적합

Recurrent State Space Model

Recurrent State Space Model (RSSM)

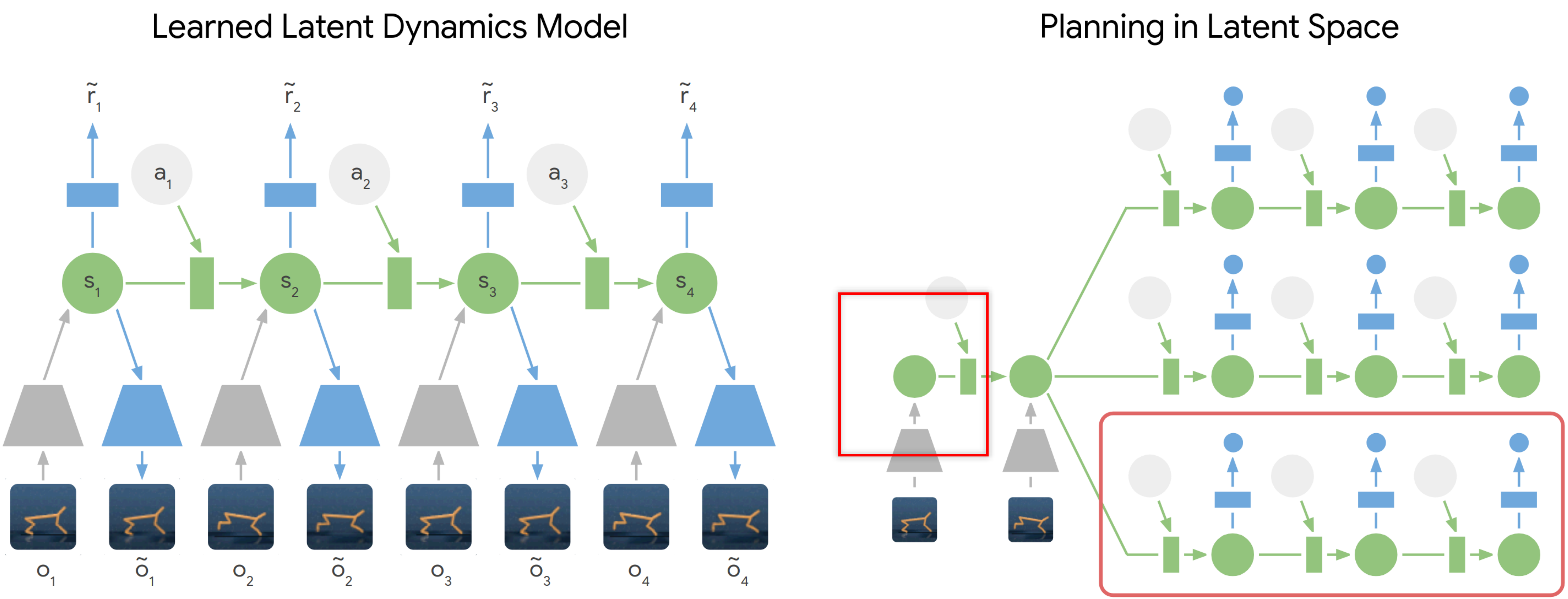
보이는게 State의 전부가 아닐 수 있다. (Non-Marcov)

Partially Observable Markov Decision Process를 Marcov Decision Process로 바꾸는 기법



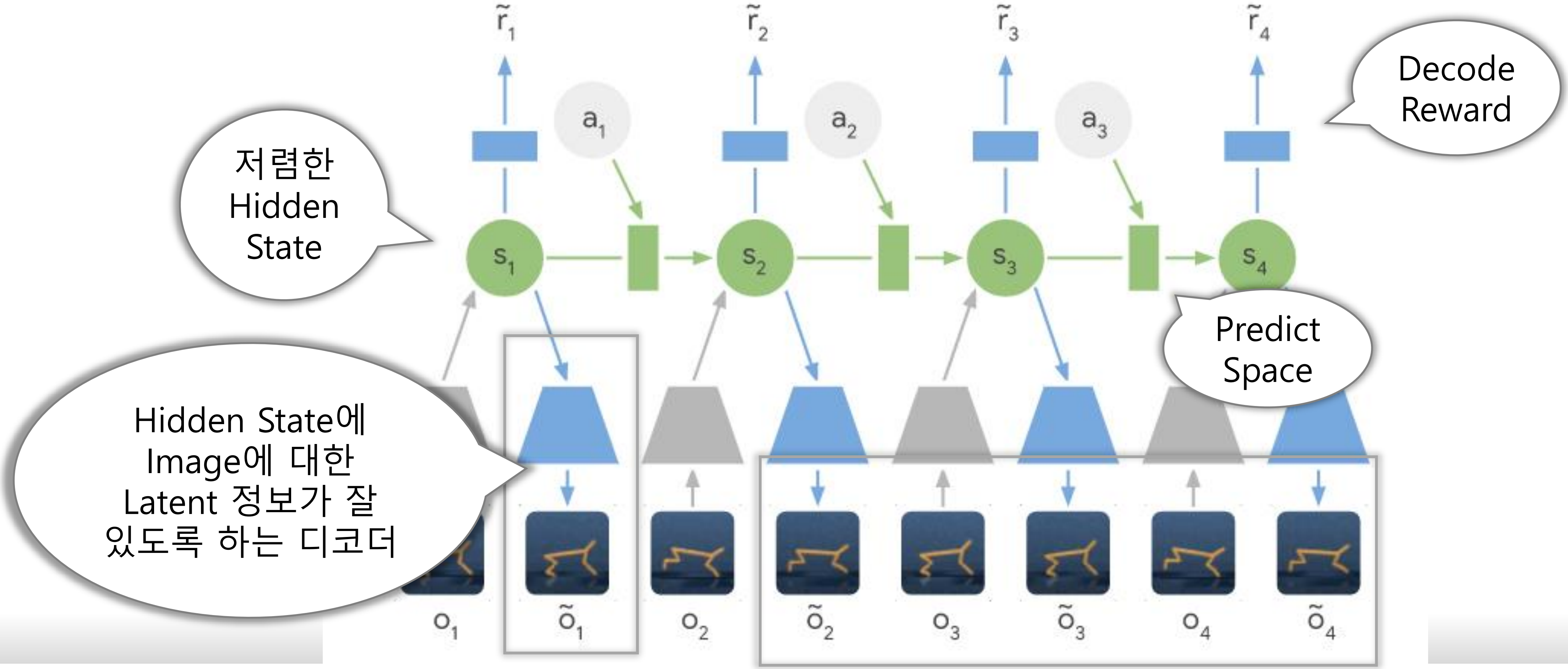
DQN에서는 비슷하게 **이미지 네 장을 묶어 하나의 State**로 취급하는 것과 유사하게 (Non-Marcov를 Marcov하게 바꾸는 기법으로 제시하는 것이다.

Recurrent State Space Model (RSSM)



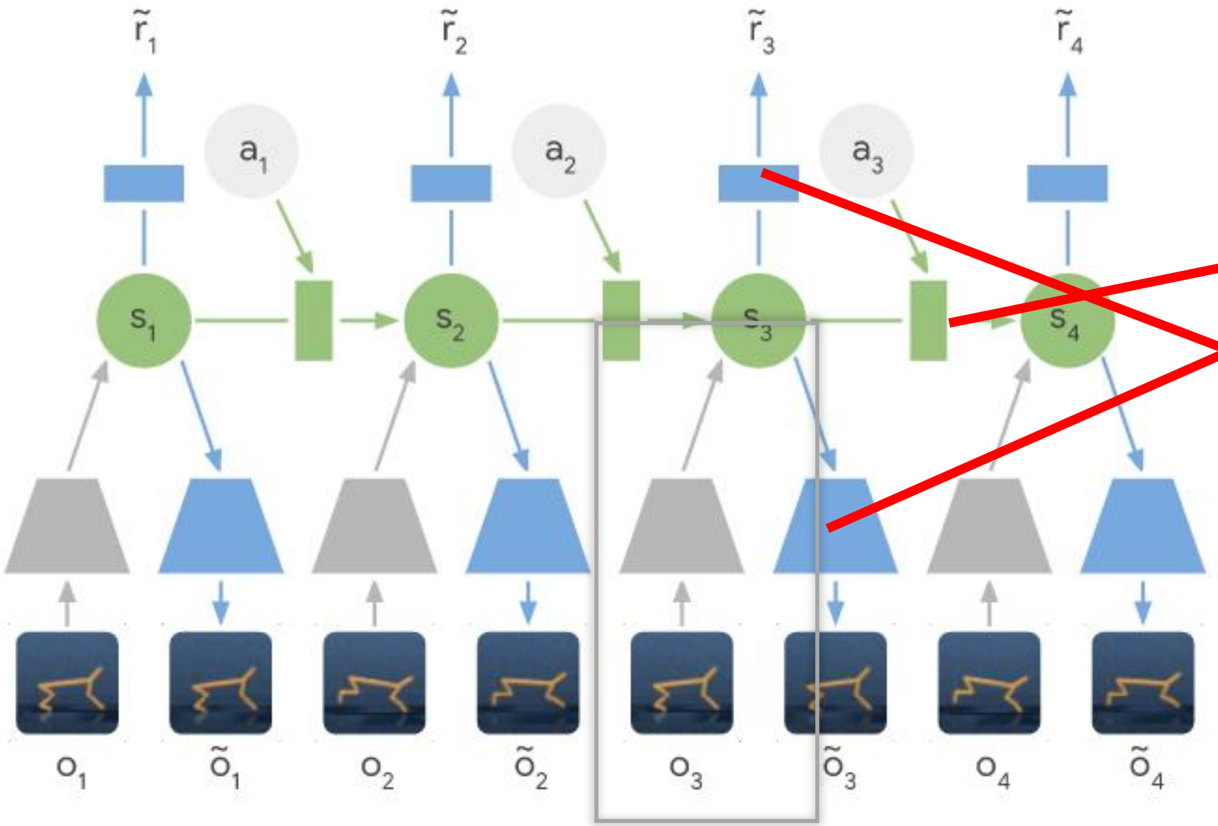
Recurrent State Space Model (RSSM)

■ Learned Latent Dynamics Model



Recurrent State Space Model (RSSM)

■ Learned Latent Dynamics Model

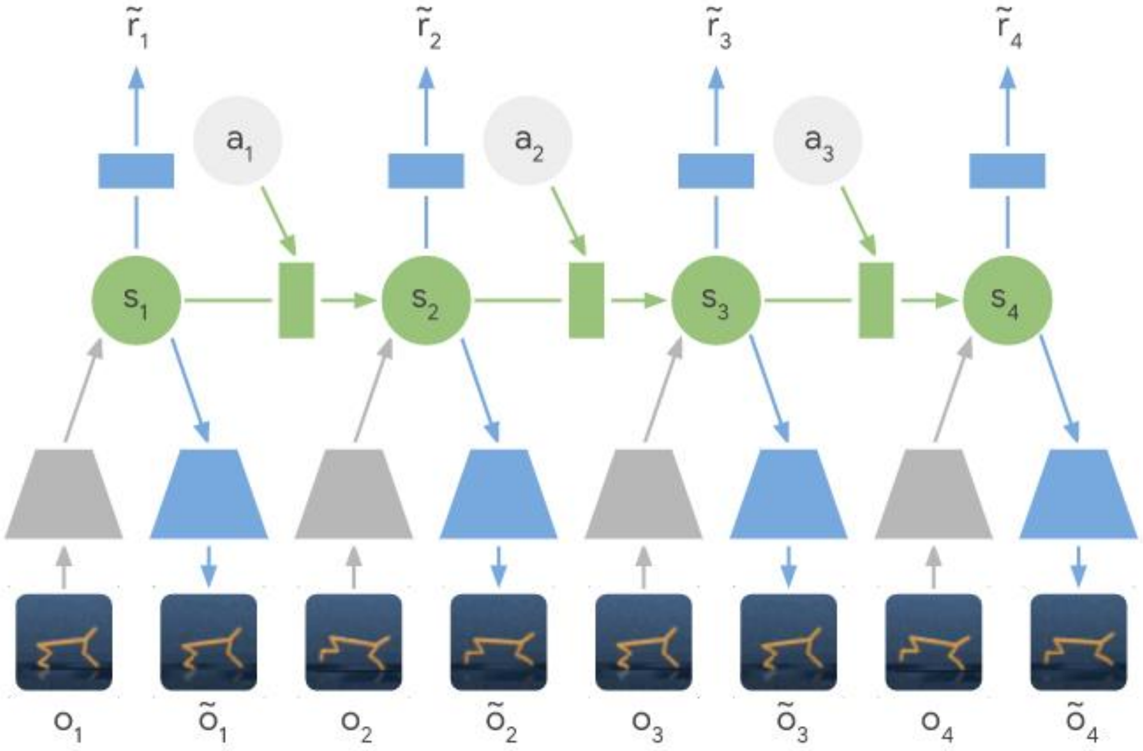


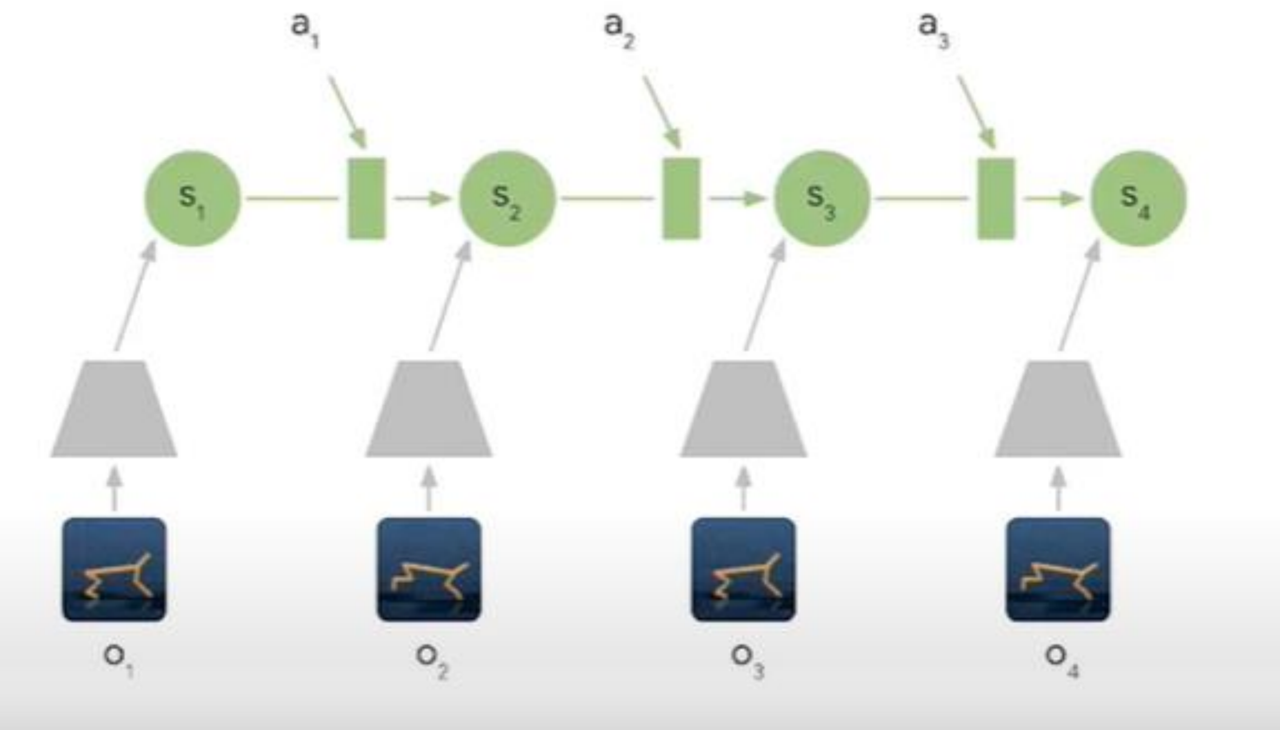
$$
\begin{aligned}
&\text{Transition function:} && s_t \sim \mathrm{p}(s_t \mid s_{t-1}, a_{t-1}) \\
&\text{Observation function:} && o_t \sim \mathrm{p}(o_t \mid s_t) \\
&\text{Reward function:} && r_t \sim \mathrm{p}(r_t \mid s_t) \\
&\boxed{\text{Policy:} \quad a_t \sim \mathrm{p}(a_t \mid o_{\leq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

Policy는?
Model-free처럼 Policy Net을 사용하는 게 아니고
굴려온 RNN이용해서 그때그때 Planning한다.

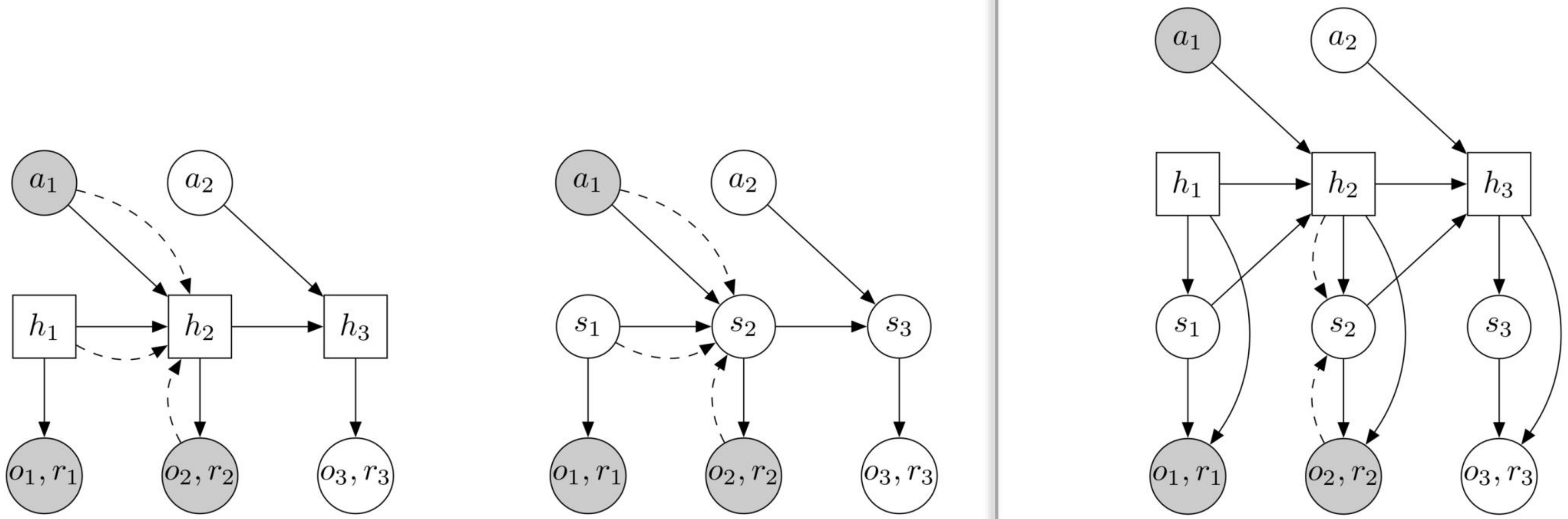
Recurrent State Space Model (RSSM)

■ Learned Latent Dynamics Model





Recurrent State Space Model (RSSM)



(a) Deterministic model (RNN) (b) Stochastic model (SSM) (c) Recurrent state-space model (RSSM)

Recurrent State Space Model (RSSM)

RSSM에서 State는 Deterministic & Stochastic Components 으로 이루어져 있다.

Deterministic Components

Discrete Time step	t
Hidden states	s_t
Image observations	o_t
Continuous action vectors	a_t
Scalar rewards	r_t

Stochastic Components

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

Recurrent State Space Model (RSSM)

Motivation

Deterministic

Stochastic transitions make it difficult for the transition model to reliably remember information for multiple time steps.

Stochastic

Split the state into stochastic and deterministic parts, allowing the model to robustly learn to predict multiple futures.

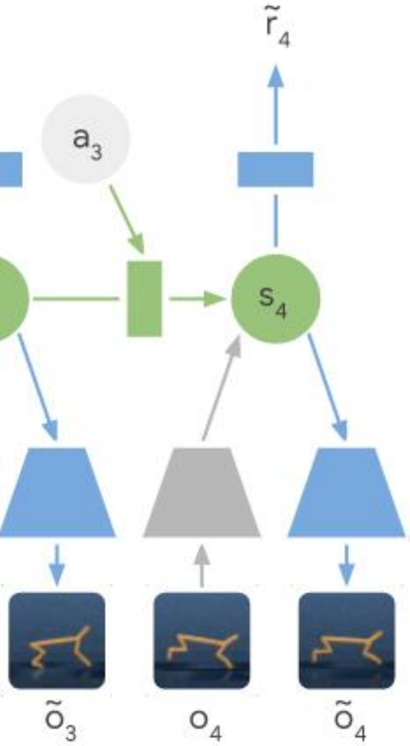
Detail

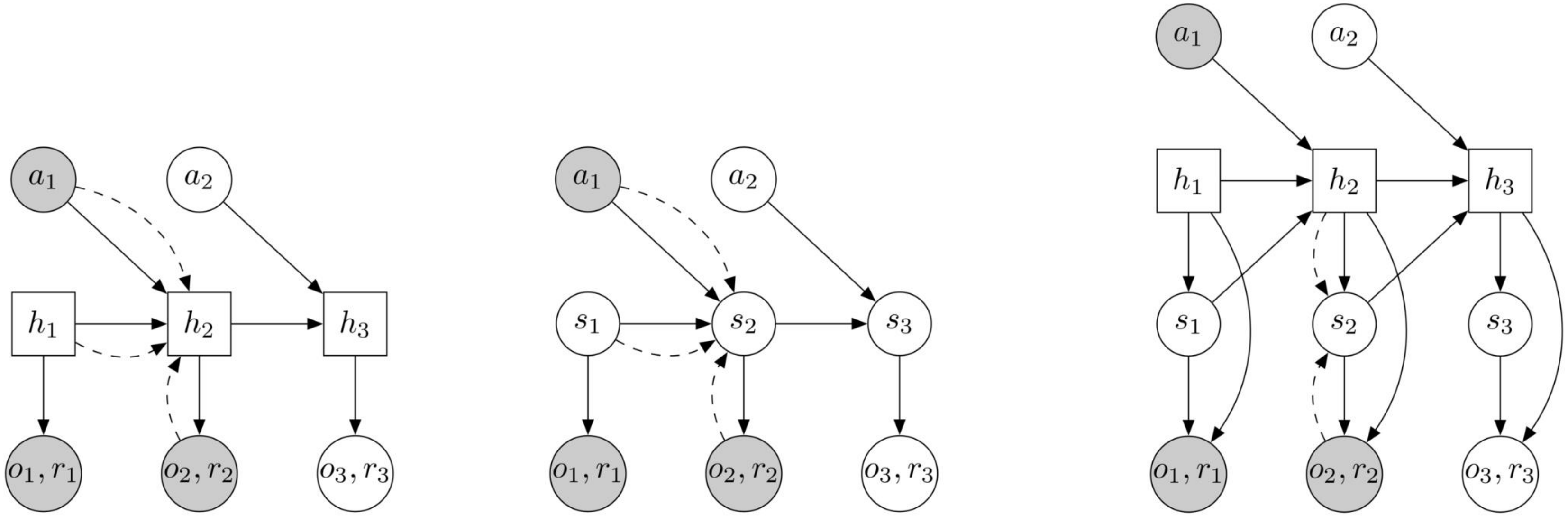
All information about the observations must pass through the sampling step of the encoder

to avoid a deterministic shortcut from inputs to reconstructions.

Latent Dynamics (PlaNet, Deep Planning Network)

■ Recurrent State Space Model (RSSM)





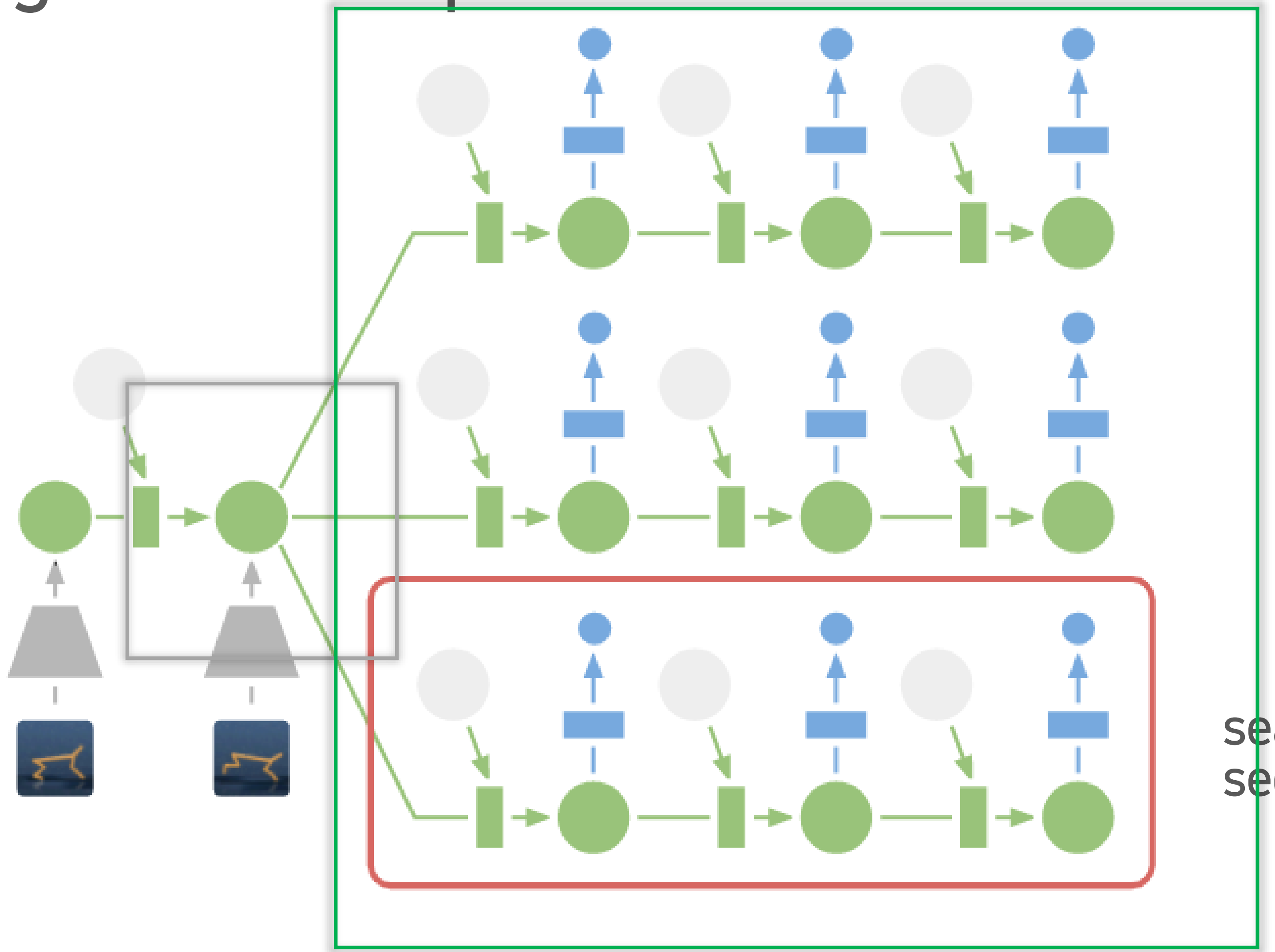
(a) Deterministic model (RNN) (b) Stochastic model (SSM) (c) Recurrent state-space model (RSSM)

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

$$
\begin{aligned}
&\text{Deterministic state model:} && h_t = f(h_{t-1}, s_{t-1}, a_{t-1}) \\
&\text{Stochastic state model:} && s_t \sim p(s_t \mid h_t) \\
&\text{Observation model:} && o_t \sim p(o_t \mid h_t, s_t) \\
&\text{Reward model:} && r_t \sim p(r_t \mid h_t, s_t),
\end{aligned}
\tag{4}
$$

Latent Dynamics (PlaNet, Deep Planning Network)

■ Planning in Latent Space



searches for the best
sequence of future actions.

Latent Dynamics (PlaNet, Deep Planning Network)

Algorithm 1: Deep Planning Network (PlaNet)

Input :

R Action repeat		$p(s_t \mid s_{t-1}, a_{t-1})$	Transition model
S Seed episodes		$p(o_t \mid s_t)$	Observation model
C Collect interval		$p(r_t \mid s_t)$	Reward model
B Batch size		$q(s_t \mid o_{\leq t}, a_{<t})$	Encoder
L Chunk length		$p(\epsilon)$	Exploration noise
α Learning rate			

Initialize dataset $\mathcal{D}$ with S random seed episodes.
Initialize model parameters θ randomly.
while *not converged* **do**
    `// Model fitting`
    for *update step* $s = 1..C$ **do**
        Draw sequence chunks $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathcal{D}$ uniformly at random from the dataset.
        Compute loss $\mathcal{L}(\theta)$ from Equation 8.
        Update model parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$.
    `// Data collection`
    $o_1 \leftarrow$ `env.reset()`
    for *time step* $t = 1..\lceil \frac{T}{R} \rceil$ **do**
        Infer belief over current state $q(s_t \mid o_{\leq t}, a_{<t})$ from the history.
        $a_t \leftarrow$ `planner`$(q(s_t \mid o_{\leq t}, a_{<t}), p)$, see Algorithm 2 in the appendix for details.
        Add exploration noise $\epsilon \sim p(\epsilon)$ to the action.
        for *action repeat* $k = 1..R$ **do**
            $r_t^k, o_{t+1}^k \leftarrow$ `env.step`(a_t)
        $r_t, o_{t+1} \leftarrow \sum_{k=1}^{R} r_t^k, o_{t+1}^R$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$

PlaNet은 Planning을 위해
Cross Entropy Method (CEM) 을 사용했다.

Algorithm 2: Latent planning with CEM

Input :

H	Planning horizon distance	$q(s_t \mid o_{\leq t}, a_{<t})$	Current state belief
I	Optimization iterations	$p(s_t \mid s_{t-1}, a_{t-1})$	Transition model
J	Candidates per iteration	$p(r_t \mid s_t)$	Reward model
K	Number of top candidates to fit		

Initialize factorized belief over action sequences $q(a_{t:t+H}) \leftarrow \text{Normal}(0, \mathbb{I})$.
for *optimization iteration* $i = 1..I$ **do**
    `// Evaluate J action sequences from the current belief.`
    for *candidate action sequence* $j = 1..J$ **do**
        $a_{t:t+H}^{(j)} \sim q(a_{t:t+H})$
        $s_{t:t+H+1}^{(j)} \sim q(s_t \mid o_{1:t}, a_{1:t-1}) \prod_{\tau=t+1}^{t+H+1} p(s_\tau \mid s_{\tau-1}, a_{\tau-1}^{(j)})$
        $R^{(j)} = \sum_{\tau=t+1}^{t+H+1} \mathrm{E}[p(r_\tau \mid s_\tau^{(j)})]$
    `// Re-fit belief to the K best action sequences.`
    $\mathcal{K} \leftarrow \operatorname{argsort}(\{R^{(j)}\}_{j=1}^{J})_{1:K}$
    $\mu_{t:t+H} = \frac{1}{K} \sum_{k \in \mathcal{K}} a_{t:t+H}^{(k)}, \quad \sigma_{t:t+H} = \frac{1}{K-1} \sum_{k \in \mathcal{K}} |a_{t:t+H}^{(k)} - \mu_{t:t+H}|.$
    $q(a_{t:t+H}) \leftarrow \text{Normal}(\mu_{t:t+H}, \sigma_{t:t+H}^2 \mathbb{I})$
return *first action mean* μ_t.

Latent Dynamics (PlaNet, Deep Planning Network)

■ Objective Function

$$
\begin{aligned}
\ln p(o_{1:T} \mid a_{1:T}) &\triangleq \ln \int \prod_t p(s_t \mid s_{t-1}, a_{t-1}) p(o_t \mid s_t)\, ds_{1:T} \\
&\geqslant \sum_{t=1}^{T} \Big(\underbrace{\mathrm{E}_{q(s_t \mid o_{\leqslant t}, a_{<t})} [\ln p(o_t \mid s_t)]}_{\text{reconstruction}} \hookleftarrow \\
&\quad - \underbrace{\mathop{\mathrm{E}}_{q(s_{t-1} \mid o_{\leqslant t-1}, a_{<t-1})} \big[\mathrm{KL}[q(s_t \mid o_{\leqslant t}, a_{<t}) \parallel p(s_t \mid s_{t-1}, a_{t-1})] \big]}_{\text{complexity}} \Big).
\end{aligned}
\tag{3}
$$

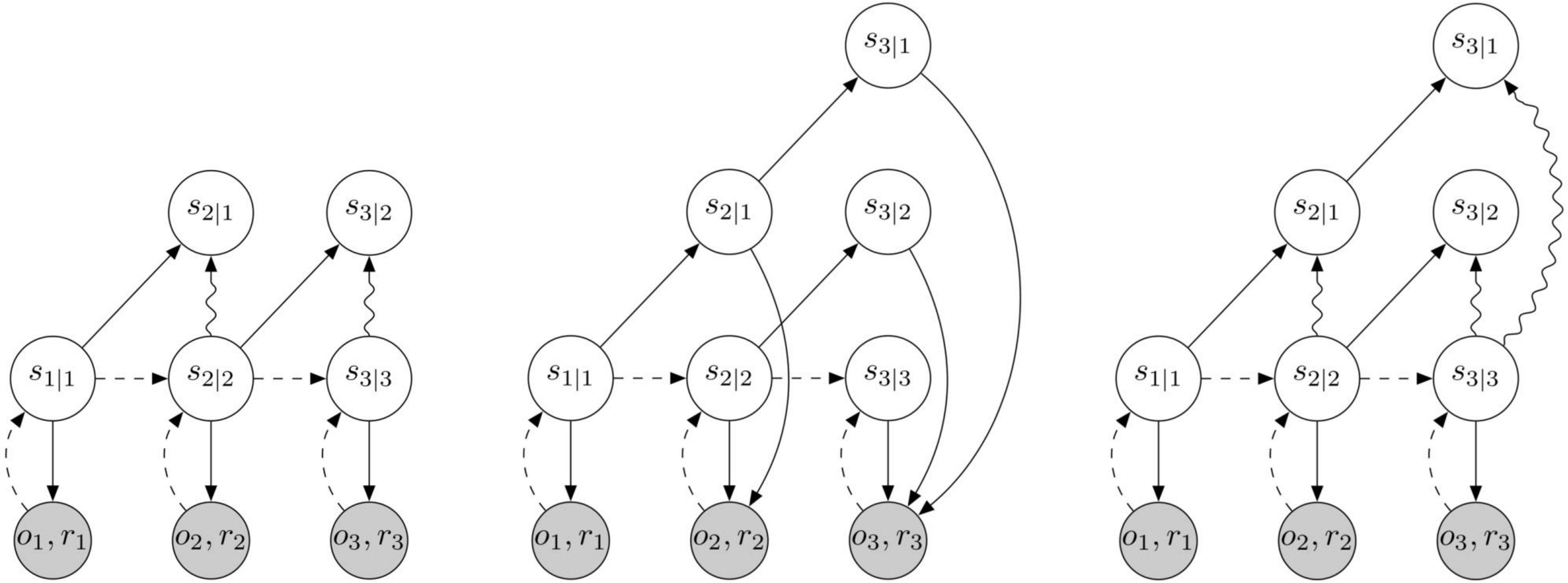
Limitation Gradient가 한 step만 흐르기 때문에, 1-step prediction만 보장한다

Latent Overshooting

Latent OverShooting

Motivation

multi-step prediction은 추가 image를 가하지 않았을 때 **latent space에서의 손실**로 개선할 수 있다.



(a) Standard variational bound (b) Observation overshooting (c) Latent overshooting

Latent OverShooting

Define Multi Step Prediction

$$p(s_t \mid s_{t-d}) \triangleq \int \prod_{\tau=t-d+1}^{t} p(s_\tau \mid s_{\tau-1})\, ds_{t-d+1:t-1} \qquad (5)$$
$$= \mathrm{E}_{p(s_{t-1}|s_{t-d})}[p(s_t \mid s_{t-1})].$$

d = 1 이면 1-step prediction

Objective Function (given distance d)

$$\ln p_d(o_{1:T}) \triangleq \ln \int \prod_{t=1}^{T} p(s_t \mid s_{t-d}) p(o_t \mid s_t)\, ds_{1:T}$$
$$\geqslant \sum_{t=1}^{T} \Big(\underbrace{\mathrm{E}_{q(s_t|o_{\leqslant t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} \qquad (6)$$
$$- \underbrace{\mathrm{E}_{p(s_{t-1}|s_{t-d})q(s_{t-d}|o_{\leqslant t-d})}\Big[\mathrm{KL}[q(s_t \mid o_{\leqslant t}) \,\|\, p(s_t \mid s_{t-1})]\Big]}_{\text{multi-step prediction}} \Big).$$

와 ~ Marcov하다 !

Latent OverShooting

■ Objective Function (all distances up to the planning horizon.)

$$\frac{1}{D}\sum_{d=1}^{D}\ln p_d(o_{1:T}) \geqslant \sum_{t=1}^{T}\Bigg(\underbrace{\mathrm{E}_{q(s_t|o_{\leqslant t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} \hookleftarrow$$
$$-\underbrace{\frac{1}{D}\sum_{d=1}^{D}\beta_d\,\mathrm{E}_{p(s_{t-1}|s_{t-d})q(s_{t-d}|o_{\leqslant t-d})}\Big[\mathrm{KL}[q(s_t \mid o_{\leqslant t}) \,\|\, p(s_t \mid s_{t-1})]\Big]}_{\text{latent overshooting}}\Bigg). \quad (7)$$

Latent OverShooting

■ Objective Function (all distances up to the planning horizon.)

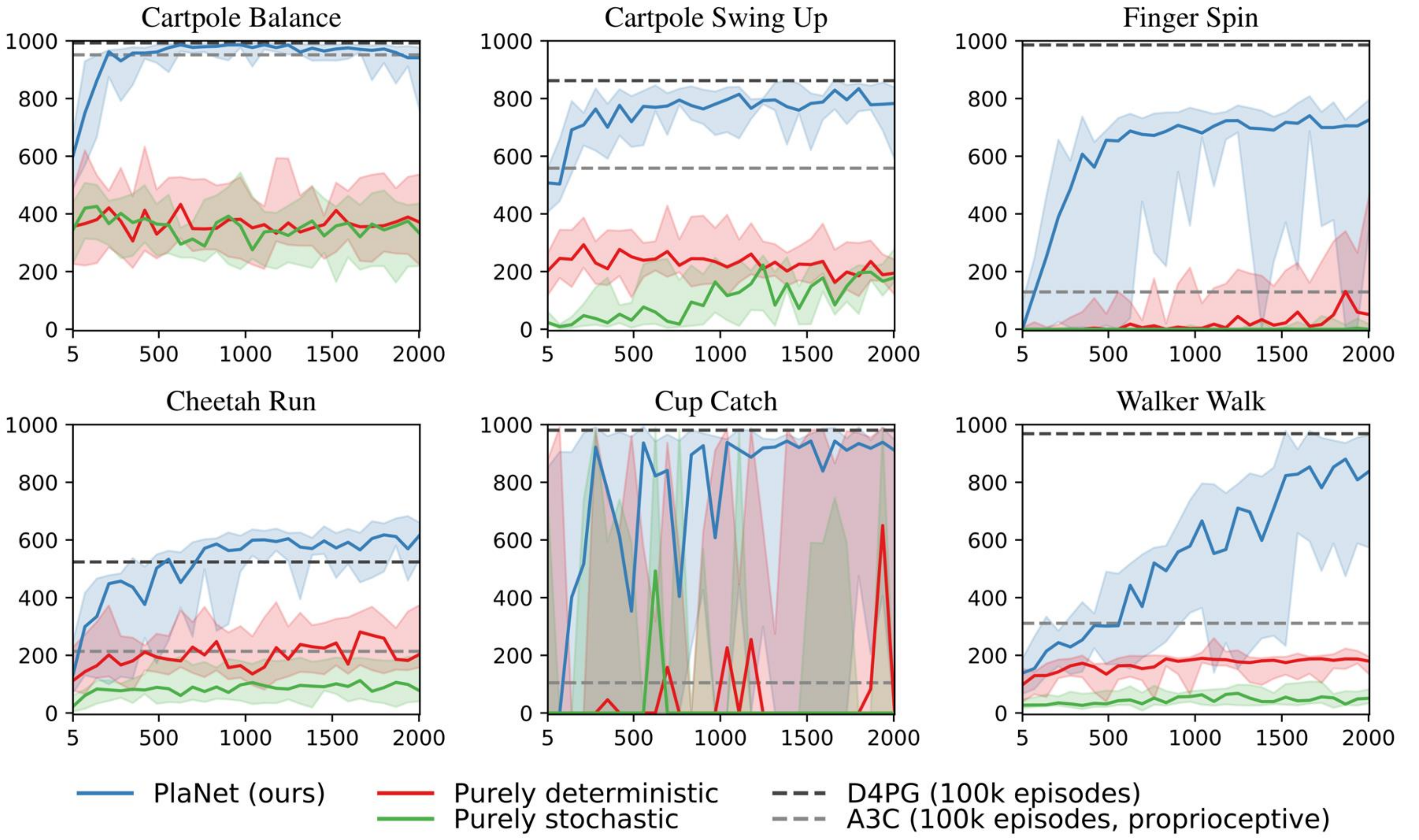
$$\frac{1}{D}\sum_{d=1}^{D}\ln p_d(o_{1:T}) \geqslant \sum_{t=1}^{T}\Bigg(\underbrace{\mathrm{E}_{q(s_t|o_{\leqslant t})}[\ln p(o_t \mid s_t)]}_{\text{reconstruction}} \hookleftarrow$$
$$\underbrace{-\frac{1}{D}\sum_{d=1}^{D}\beta_d\,\mathrm{E}_{p(s_{t-1}|s_{t-d})q(s_{t-d}|o_{\leqslant t-d})}\Big[\mathrm{KL}[q(s_t \mid o_{\leqslant t}) \parallel p(s_t \mid s_{t-1})]\Big]}_{\text{latent overshooting}}\Bigg). \quad (7)$$

Experiments

Comparison (Model-Free Method)

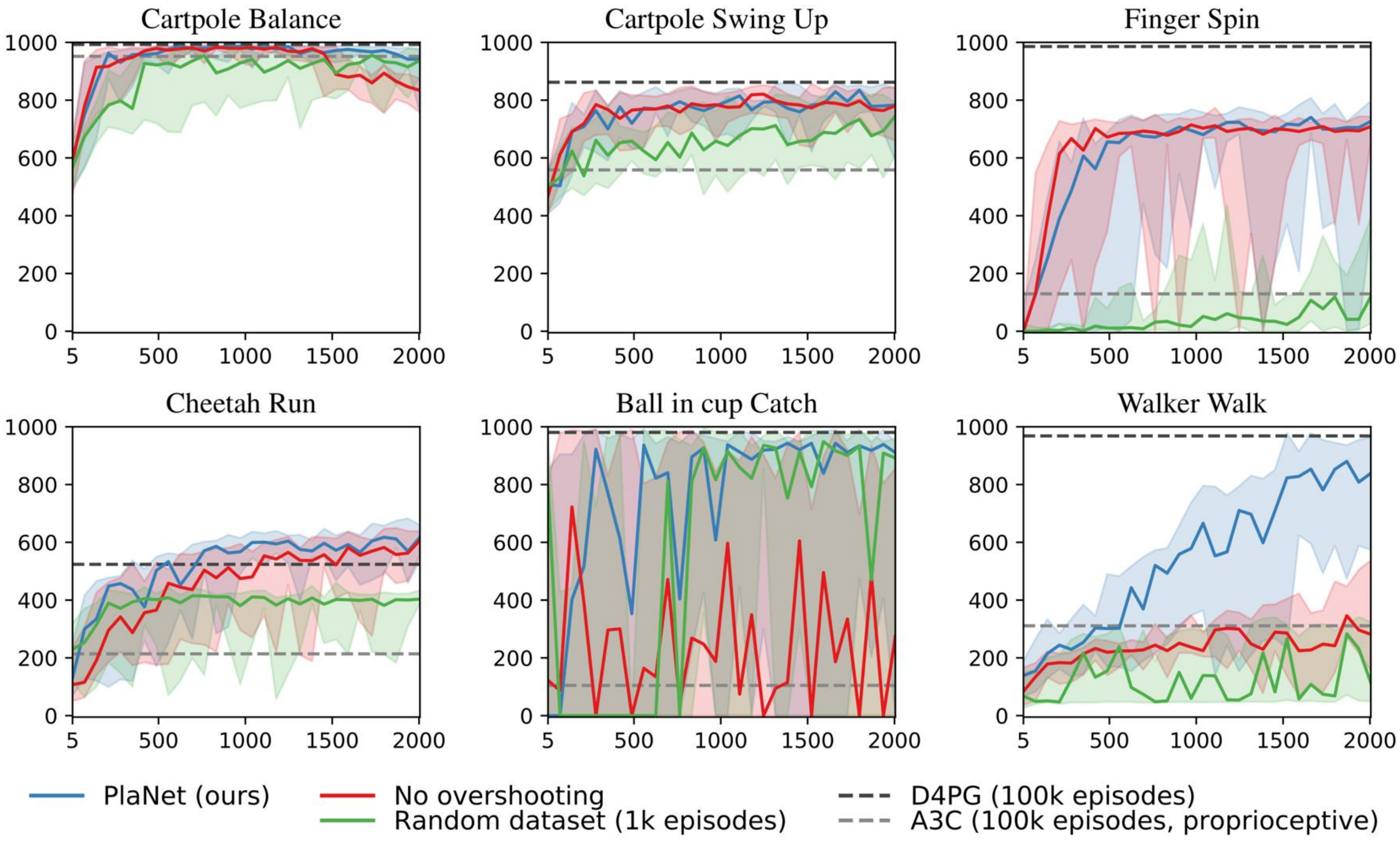
Method	Modality	Episodes	Cartpole Balance	Cartpole Swingup	Finger Spin	Cheetah Run	Ball in cup Catch	Walker Walk
A3C	proprioceptive	100,000	952	558	129	214	105	311
D4PG	pixels	100,000	993	862	985	524	980	968
PlaNet (ours)	pixels	2,000	986	831	744	650	914	890
CEM + true simulator	simulator state	0	998	850	825	656	993	994
Data efficiency gain PlaNet over D4PG (factor)			100	180	16	50+	20	11

Model



Cartpole Balance
Cartpole Swing Up
Finger Spin
Cheetah Run
Cup Catch
Walker Walk
1000
800
600
400
200
0
5
500
1000
1500
2000
PlaNet (ours)
Purely deterministic
Purely stochastic
D4PG (100k episodes)
A3C (100k episodes, proprioceptive)

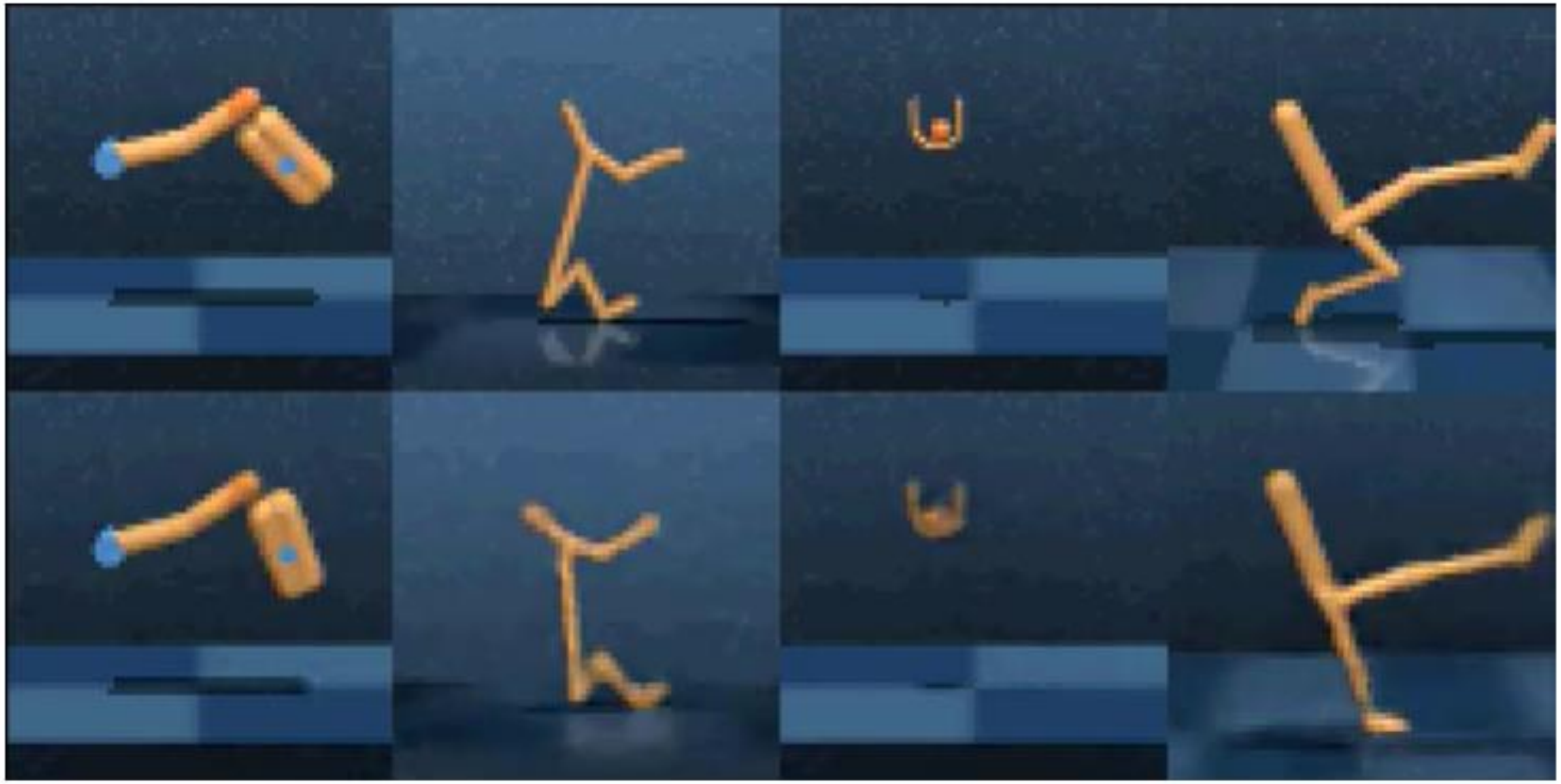
Agent


Cartpole Balance
Cartpole Swing Up
Finger Spin
Cheetah Run
Ball in cup Catch
Walker Walk
0
200
400
600
800
1000
5
500
1000
1500
2000
PlaNet (ours)
No overshooting
Random dataset (1k episodes)
D4PG (100k episodes)
A3C (100k episodes, proprioceptive)

One agent all tasks

Episode

Prediction



Relative Works

Relative Works

1. **Planning in state space**
2. **Hybrid agents**
3. **Multi-step predictions**
4. **Latent sequence models**
5. **Video prediction**

End !