RL 논문 리뷰 스터디
Week2

Model Based Reinforcement Learning For Atari (SimPLe)

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, Henryk Michalewski

Presented by 유성동

Objectives

“ How “ this paper appeared?

“ Which” technique this paper used?

“ What “ does this paper mean?

Preliminaries

What is Model-Based?

Preliminaries

- Model Free

    algorithm which does **not** use the *transition probability distribution*

    $$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

    ex) DQN, A3C, TRPO, PPO, DDPG, SAC …

- Model Based

    algorithm which does use the *transition probability distribution*

    ex) AlphaZero, MCTS


92.1%
67.0%
72.8%
AlphaZero
A General Reinforcement Learning Algorithm that Masters Chess, Shogi and Go through Self-Play

Actually…

How about “Getting” a model with **Trial and Error?**

Model Based 혐오를 멈춰주세요..

Preliminaries

	Model-based RL	Model-free RL
Pros	Sample efficiency & Scalability	Don’t need a model
Cons	Computation cost	Training Data

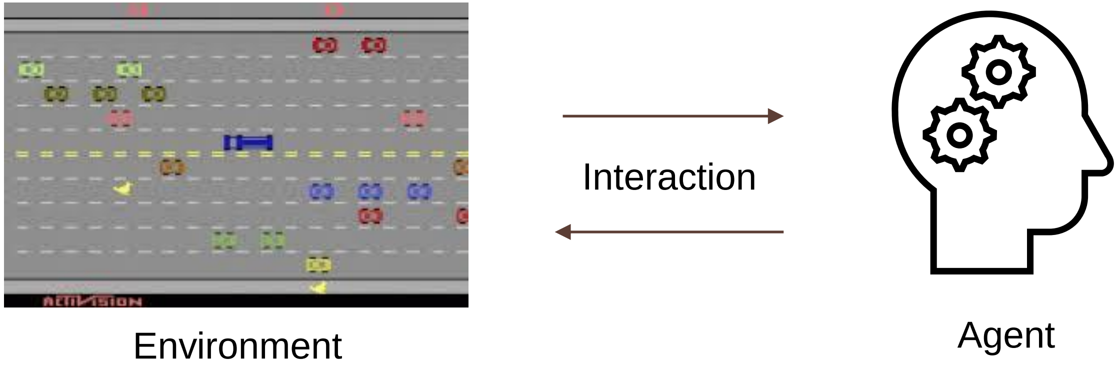
Introduction

Introduction

Model free Reinforcement Learning

Great performance in Atari games!

However, requires too many interactions (= Too many times) !!!



Then… How about Human?

What's Next?

Introduction

Human possess an **intuitive understanding** of the physical processes

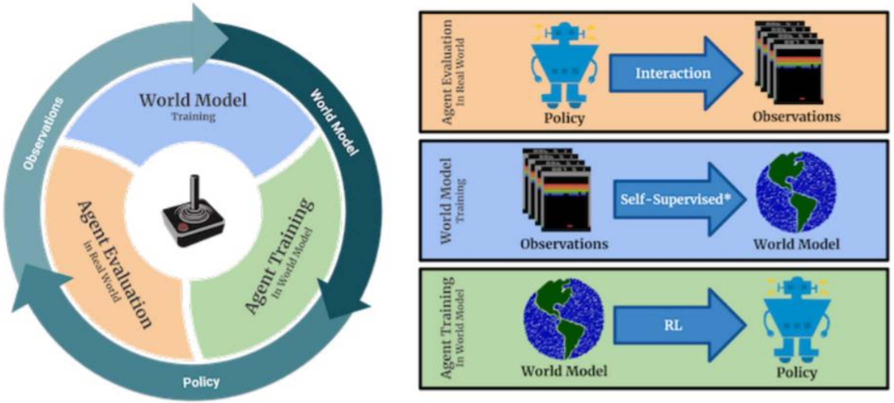
We know planes can fly, ball can roll, and bullet can destroy aliens!

Let's make our agent learns intuitive understanding!

How?

Video Prediction!

Introduction



*World Model:
우리가 생각하는 가상의 simulation 환경!
실제 env와 interaction 없이도 가능한 Network

1. Interacting with the real environment following the latest policy
2. Collected observations will be used to train the current world model
3. Update the policy by acting inside the world model

Introduction

SIMulated Policy Learning

SimPLe

Utilize video prediction techniques and trains a policy to play the game within the learned model

여기서 Simulation은 “Neural Network” 라는 것이 포인트!

Related Work

Related Work _Model free

DQN

Double DQN

Dueling DQN

Priortized DQN

Rainbow DQN

A3C

REINFORCE

A2C

TRPO

PPO

DDPG

Good Performance in Atari,
but remains far higher than the **amount of experience required** for human players to learn each game

Related Work_Video prediction

(2015) Action conditional video prediction using deep networks in atari games (Oh et al)

(2017) Recurrent environment simulators (Chiappa et al)

(2016) A deep learning approach for joint video frame and reward prediction in Atari games (Leibfried et al)

This paper focus on using video prediction in the context of learning how to play the game well and positively verify that learned simulators can be used to train a policy useful in original environments

Related Work_Other works

(2017) Value Prediction Network (Oh et al)

→ use a model of reward to augment model-free learning, but not aim to model or predict future

(2019) Recurrent world models facilitate policy evaluation (Sodhani et al)

→ present a way to compose VAE with RNN, but need enough exploration

MCTS, Dyna-DQN, Generative Adversarial Tree Search(GATS), …

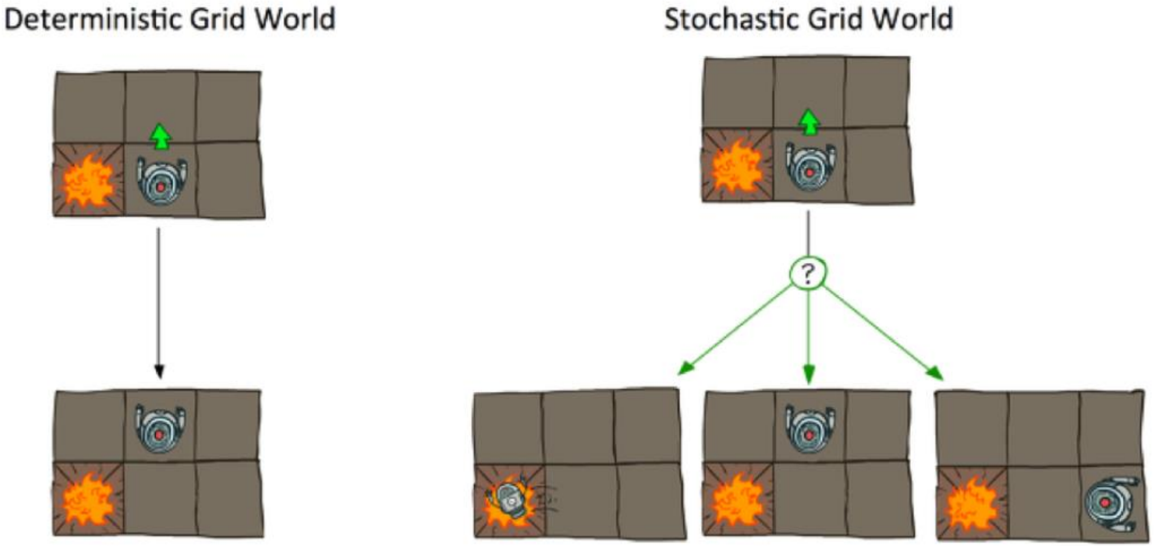
No prior work has successfully demonstrated Model-based control via predictive models that achieve competitive results with model-free RL!

Model

Deterministic Model / Stochastic Model

Model

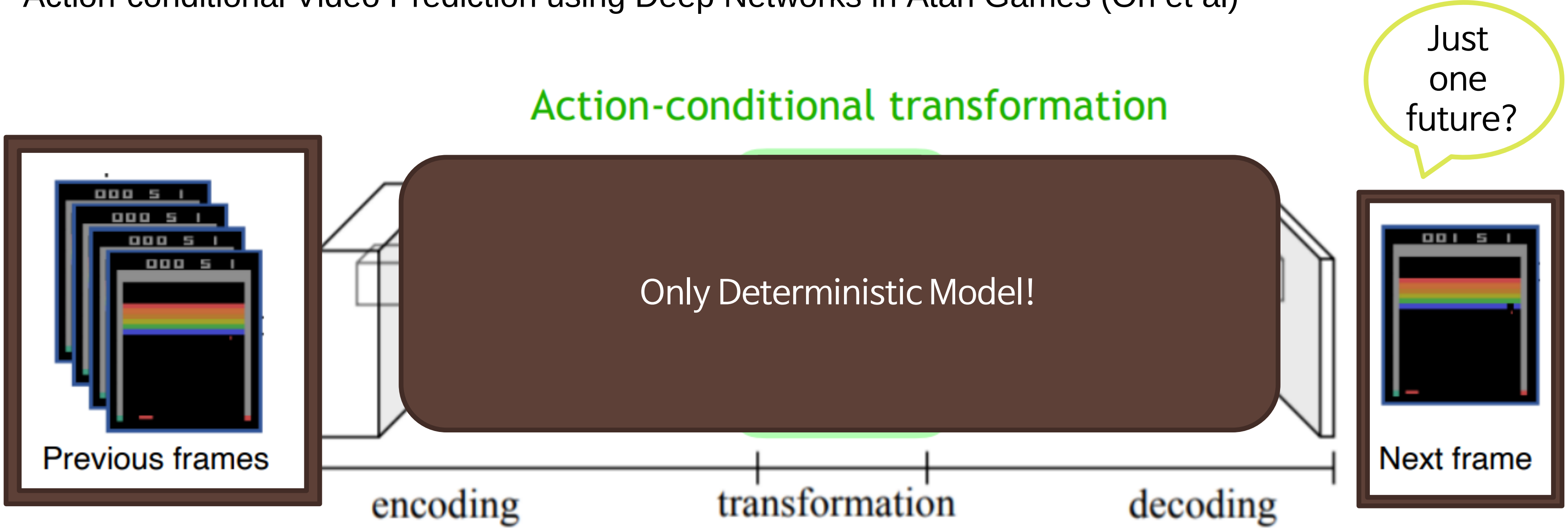
Stochastic Model & Deterministic Model



출처 : UC Berkeley CS188

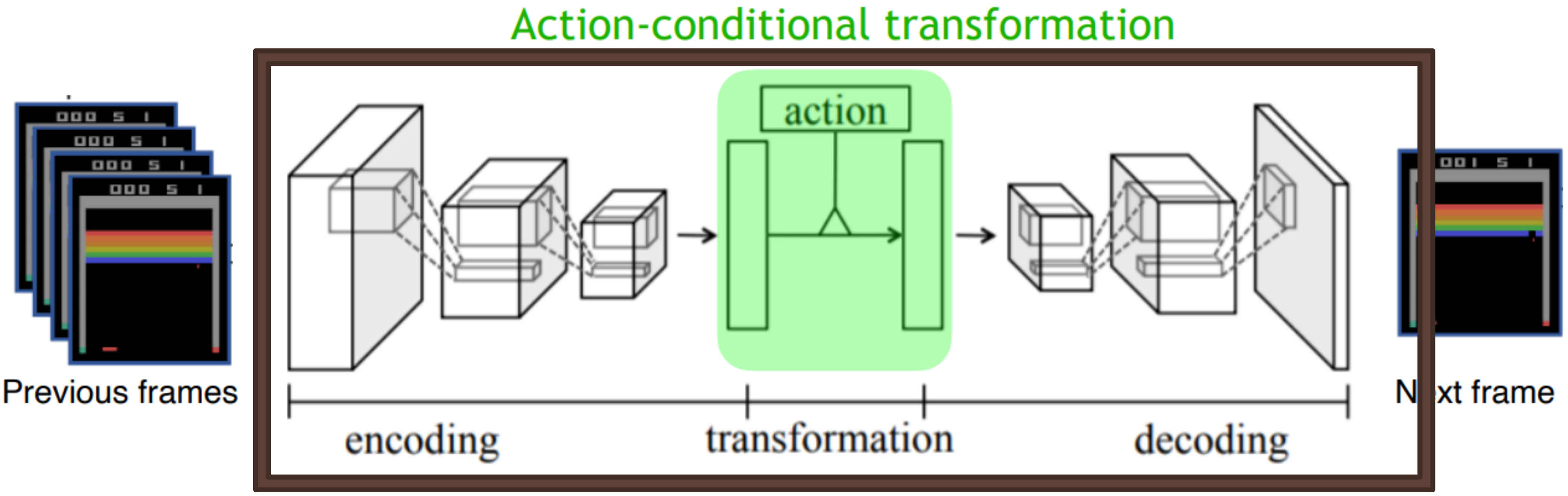
Model

Action-conditional Video Prediction using Deep Networks in Atari Games (Oh et al)



https://arxiv.org/pdf/1507.08750.pdf

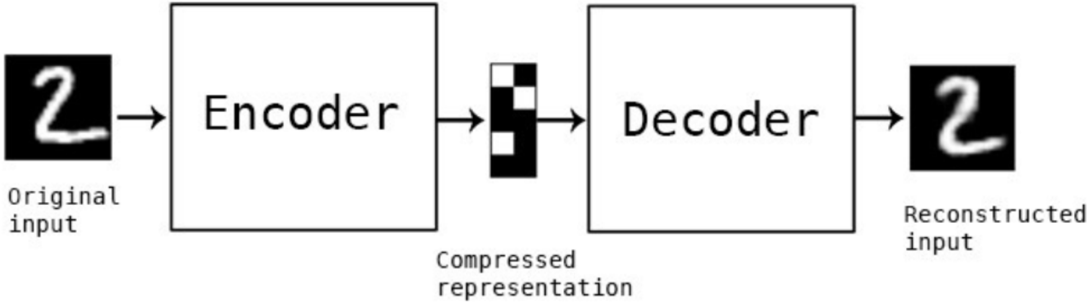
Model



Wait.. Have you seen this structure before?

Model

Autoencoder

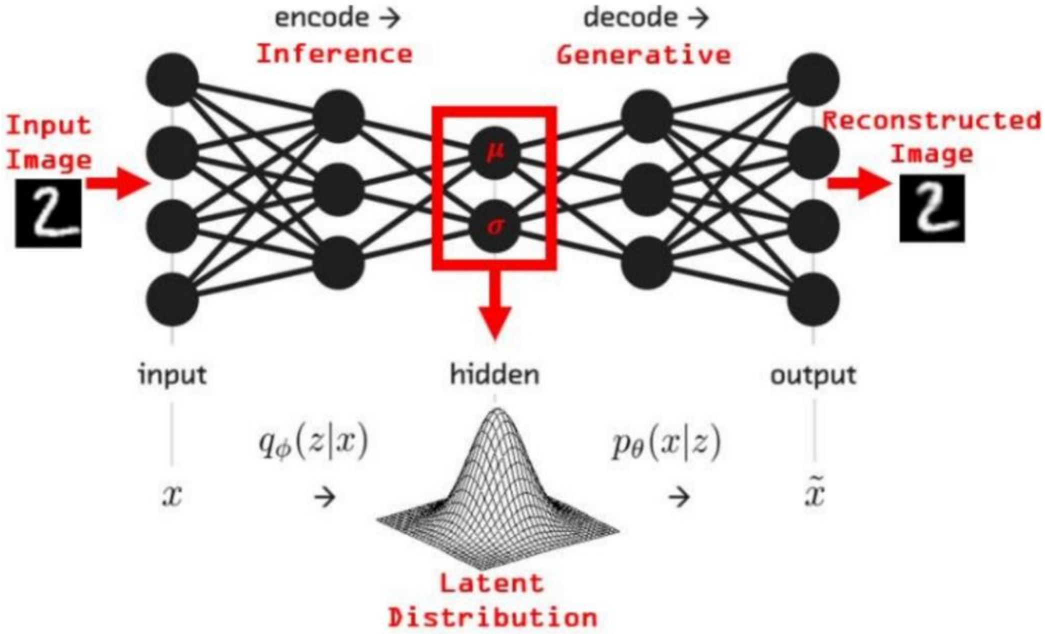


Only one output image(frame)…

출처 : https://medium.com/@realityenginesai/understanding-variational-autoencoders-and-their-applications-81a4f99efc0d

Model

Variational Autoencoder



Distribution 형태의 latent variable z를 통해 x를 generate 하고 싶다!

$$x = g_\theta(z)$$

$$p(x) = p(x|z)$$

$$p(x) = E_{z\sim p_\theta(z)}[p(x|z)]$$

$$p(x) = E_{z\sim p_\theta(z|x)}[p(x|z)]$$
$$\approx E_{z\sim q_\theta(z|x)}[p(x|z)]$$

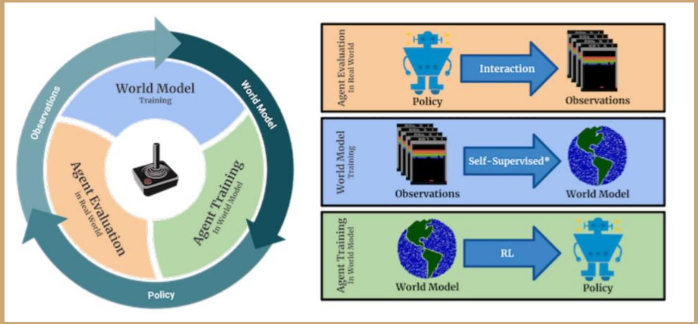
자세한 증명은 생략

출처 : https://medium.com/@realityenginesai/understanding-variational-autoencoders-and-their-applications-81a4f99efc0d

Model

Algorithm 1: Pseudocode for SimPLe

```
Initialize policy π
Initialize model parameters θ of env′
Initialize empty set D
while not done do
    ▷ collect observations from real env.
    D ← D ∪ COLLECT(env, π)
    ▷ update model using collected data.
    θ ← TRAIN_SUPERVISED(env′, D)
    ▷ update policy using world model.
    π ← TRAIN_RL(π, env′)
end while
```

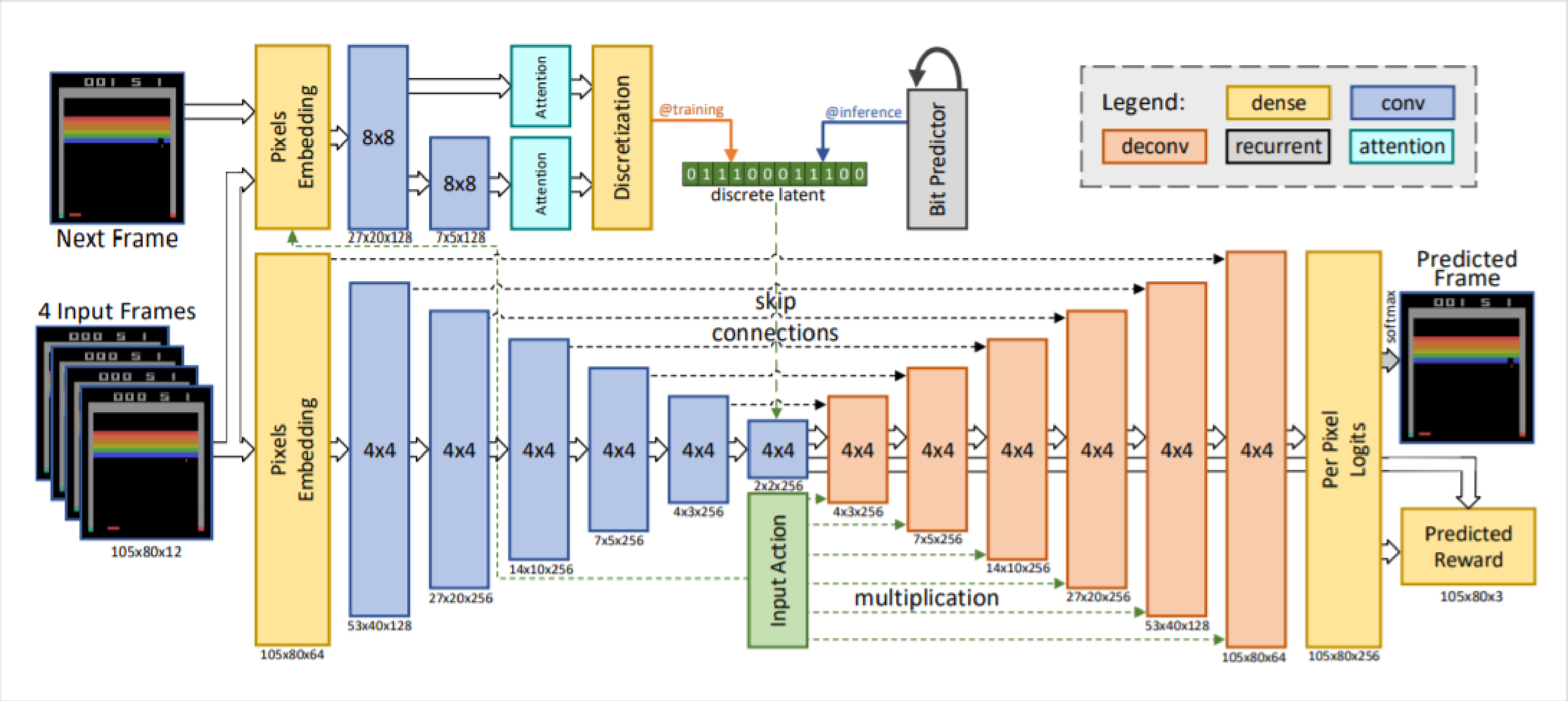


Neural Network simulated environment, ***env’***

Our purpose is to train a policy π using *env’* so that π achieves good performance in the original environment *env*

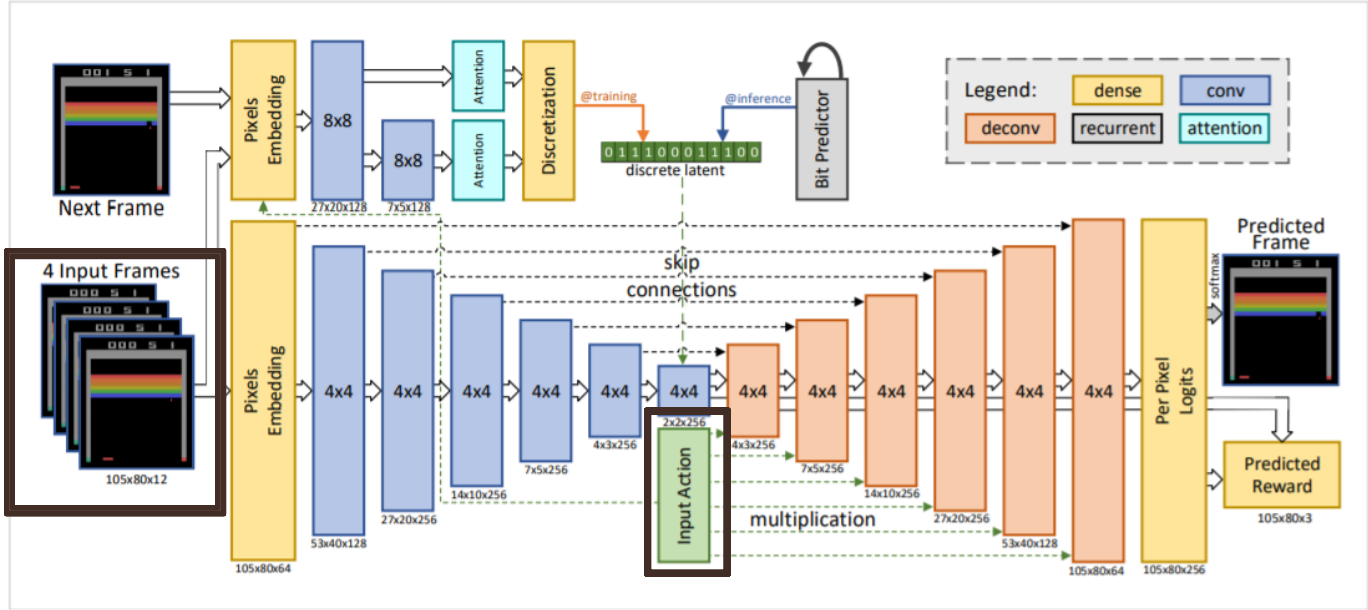
We aim to use as few interactions with *env as possilbe*

Model



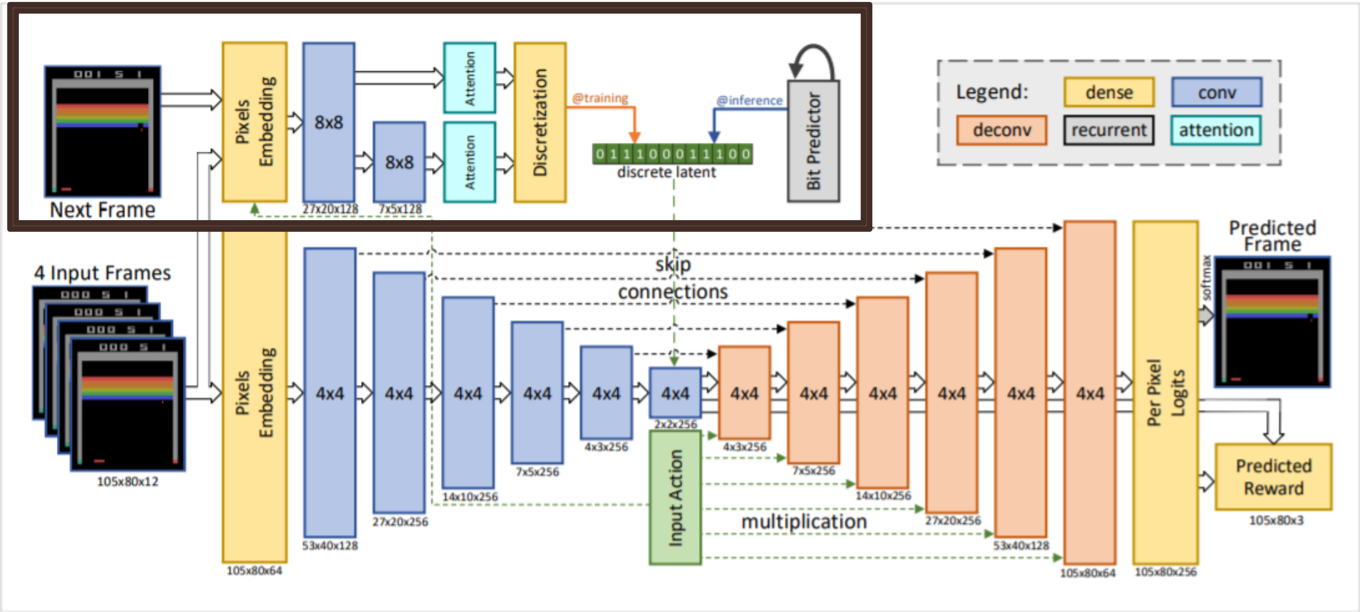
Next Frame
4 Input Frames
105x80x12
Pixels Embedding
105x80x64
8x8
27x20x128
8x8
7x5x128
Attention
Attention
Discretization
@training
@inference
0 1 1 1 0 0 0 1 1 1 0 0
discrete latent
Bit Predictor
Legend:
dense
conv
deconv
recurrent
attention
4x4
53x40x128
27x20x256
14x10x256
7x5x256
4x3x256
2x2x256
skip
connections
Input Action
multiplication
105x80x64
Per Pixel Logits
105x80x256
softmax
Predicted Frame
Predicted Reward
105x80x3

Model



Input : four consecutive game frames and action a

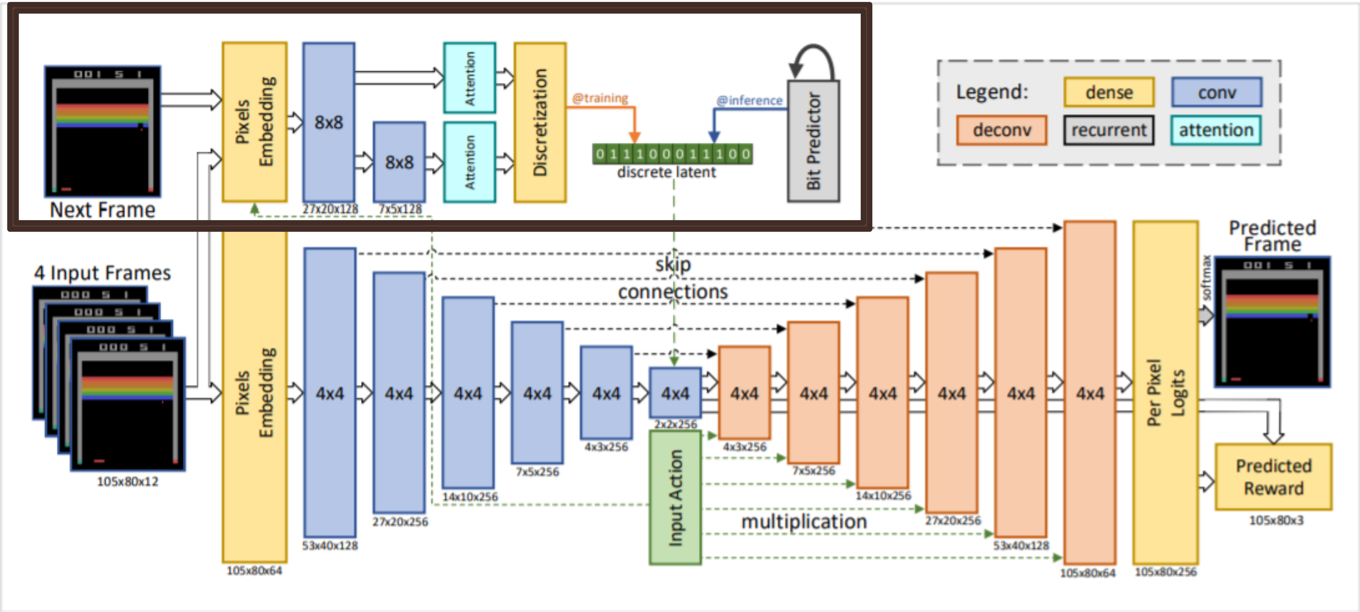
Model



Input : receives the future target frame as input and approximates the distribution of the posterior $p(z|x)$

At test time, the latent values are sampled from an assumed prior

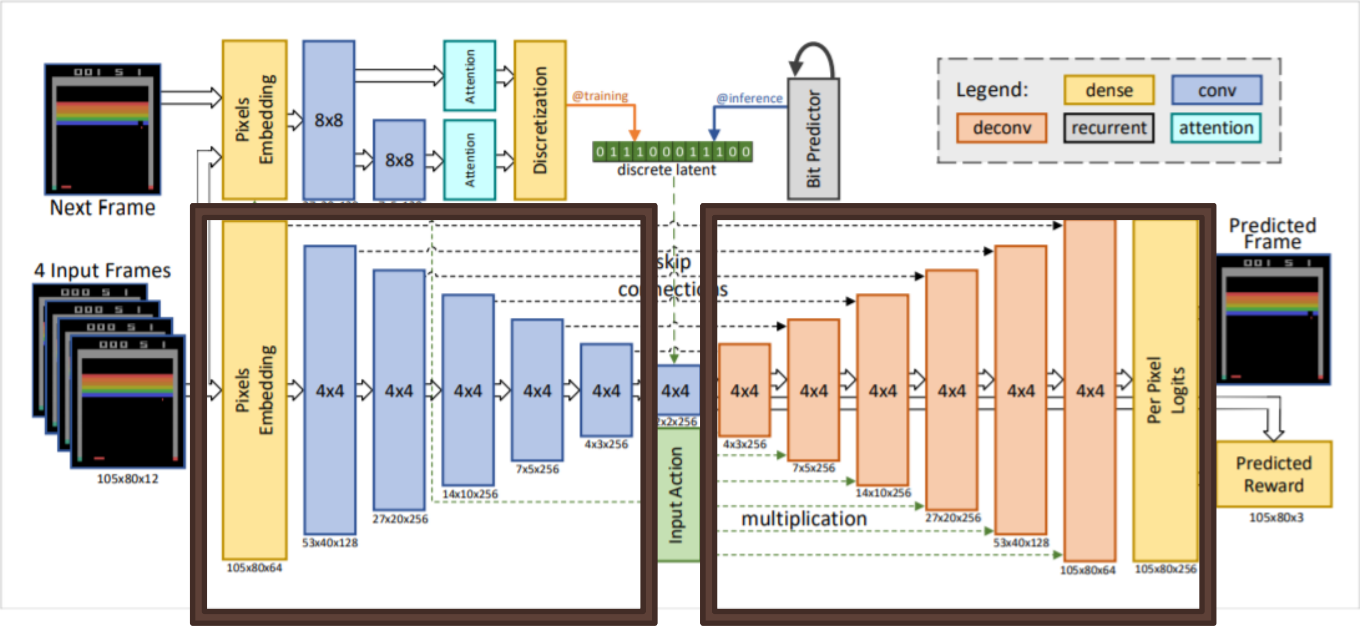
Model



Why discrete latent?

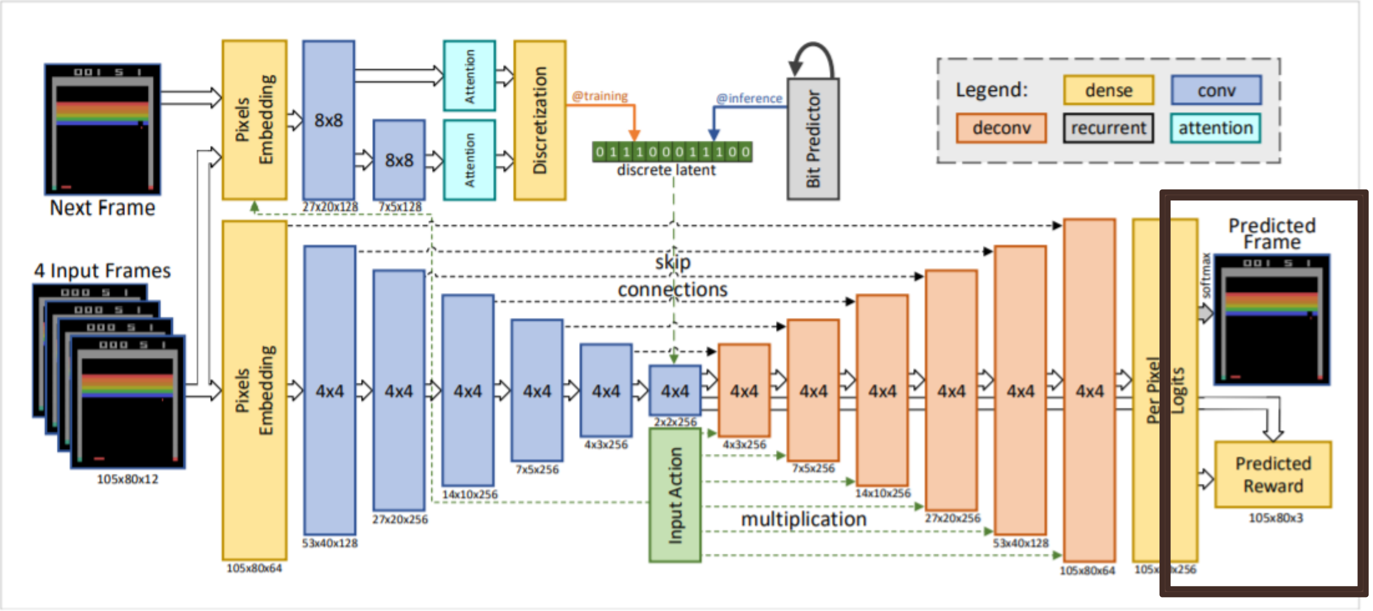
1. The weight of the KL divergence is game dependent
2. The weight is usually a very small number, $[10^{-3}, 10^{-5}]$

Model



Action is vector which is multiplied channel-wise with the output of the convolution layer + Skip Connections!

Model



Next Frame
4 Input Frames
105x80x12
Pixels Embedding
105x80x64
8x8
27x20x128
7x5x128
Attention
Attention
Discretization
@training
@inference
0 1 1 1 0 0 0 1 1 1 0 0
discrete latent
Bit Predictor
Legend:
dense
conv
deconv
recurrent
attention
skip
connections
4x4
53x40x128
27x20x256
14x10x256
7x5x256
4x3x256
2x2x256
Input Action
multiplication
105x80x64
Per Pixel
Logits
softmax
Predicted Frame
Predicted Reward
105x80x3

Model

Loss function

Use the *clipped loss,* $\max(Loss, C)$

It may make model to concentrate on small but important areas (e.g. the ball in Pong)

C = 10 for L2 loss, 0.03 for softmax loss

Policy Training

The algorithm generates rollouts in the *env' and uses them to improve policy* π

Using PPO algorithm for learning policy

Imperfections of the model compounding over time

→ Short rollout (N=50)

→ degrading effect of PPO algorithm

→ add to the reward

Experiment

Experiment Setting

Main loop : 15 times
World model : 45K steps in first iteration, and 15K steps in others

PPO
- 16 parallel agents collecting different steps from the *env'*

Training Data
- Data from real environment:

  6400 interactions + 6400 interactions x 15 iterations

  = 102,400 interactions = 409,600 frames

- Data from simulated environment:

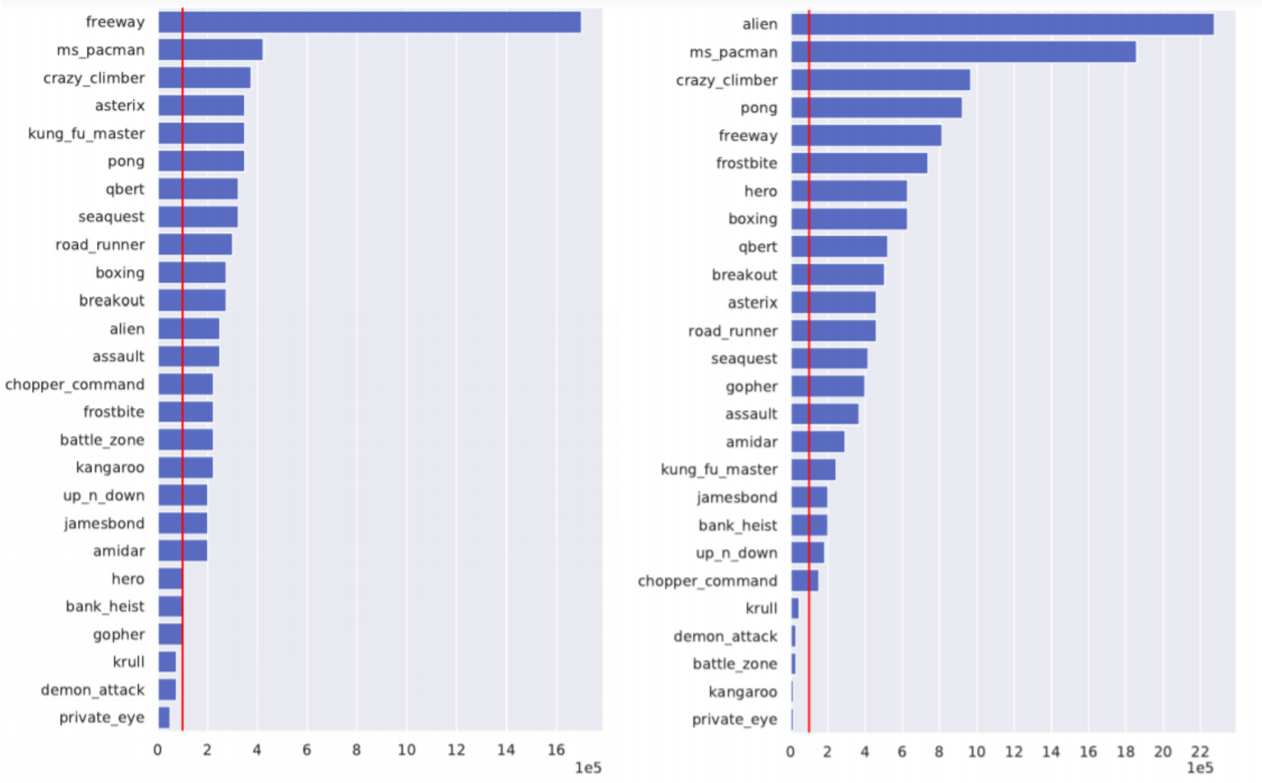
  1,000,000 interactions x 15 iterations = 15,000,000 interactions

26 games, comparisons are Rainbow, PPO

Algorithm 1: Pseudocode for SimPLe

Initialize policy $\boldsymbol{\pi}$
Initialize model parameters $\boldsymbol{\theta}$ of env'
Initialize empty set $\mathbf{D}$
while not done **do**
  ▷ collect observations from real env.
  $\mathbf{D} \leftarrow \mathbf{D} \cup \text{COLLECT}(env, \pi)$
  ▷ update model using collected data.
  $\boldsymbol{\theta} \leftarrow \text{TRAIN\_SUPERVISED}(env', \mathbf{D})$
  ▷ update policy using world model.
  $\boldsymbol{\pi} \leftarrow \text{TRAIN\_RL}(\boldsymbol{\pi}, env')$
end while

Experiment

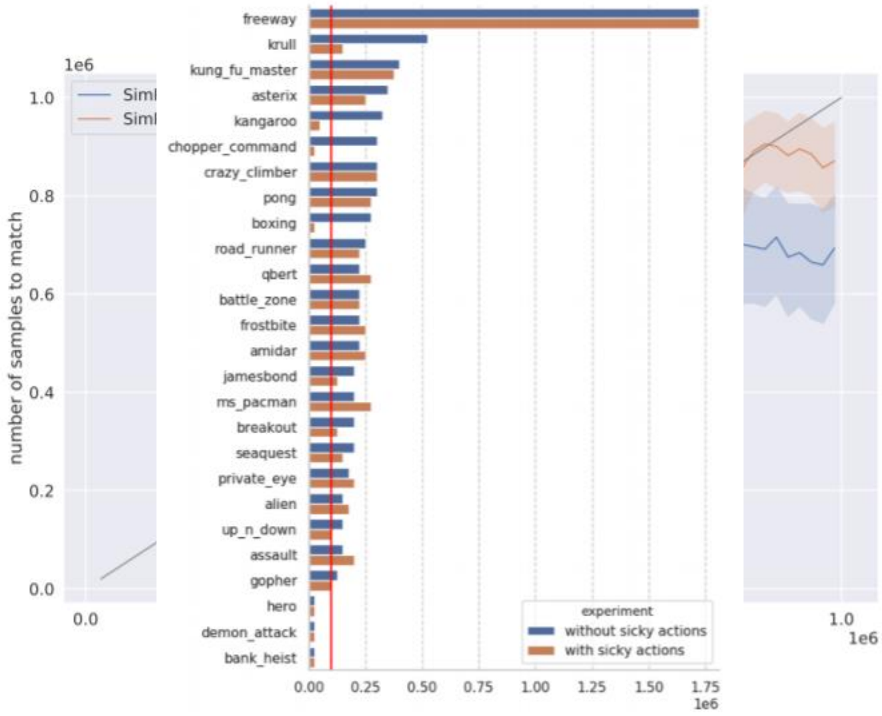


Model-based RL provides an <u>effective approach</u> to learning tasks!

For the 6 games, it exceeds the avg human score (future work)

In some cases, we observed high variance when <u>mismatch the model and the real env</u>

Experiment



SimPLe excels in a low data regime, but its <u>advantage disappears with a bigger amount of data</u>

Also, It's useful initialization for model-free PPO training

SimPLe is also effective in stochastic environments.

Conclusion

Conclusion

This paper present SimPle, a model-based RL approach that operates directly on raw pixel observations and learns effective policies to play games in the Atari Learning Environment

It can be applied in highly stochastic environments

The representation learned by model is likely be more meaningful than the raw pixel observations

Limitations

1. Final scores are lower than SOTA model-free methods

2. The performance of our method generally varied between different runs on the same game
   → capture uncertainty via Bayesian parameter posteriors or ensembles

3. Computational and time requirement of training are substantial
   → developing lighter models & applied to other environments