Advanced Exploration

Acknowledgement: Most slides were contributed by 林九州、何國豪、陳昱丞 etc., and organized by 陳昱丞.

References

- Intrinsic Curiosity Module
  - Curiosity-driven Exploration by Self-supervised Prediction
    - Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." Proceedings of the 34 th International Conference on Machine Learning, Sydney, Australia, 2017
- Random Network Distillation
  - Exploration by Random Network Distillation
    - *Burda, Yuri, et al. "Exploration by random network distillation." arXiv preprint arXiv:1810.12894 (2018).*
- Never Give Up
  - Never Give Up: Learning Directed Exploration Strategies
    - Puigdomènech Badia, A., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, B., and Blundell, C. Never give up: Learning directed exploration strategies. In International Conference on Learning Representations, 2020.
- Agent57
  - Agent57: Outperforming the Atari Human Benchmark
    - Puigdomènech Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., Blundell, C. Agent57: Outperforming the Atari Human Benchmark. arXiv:2003.13350 (2020)

Advanced Exploration

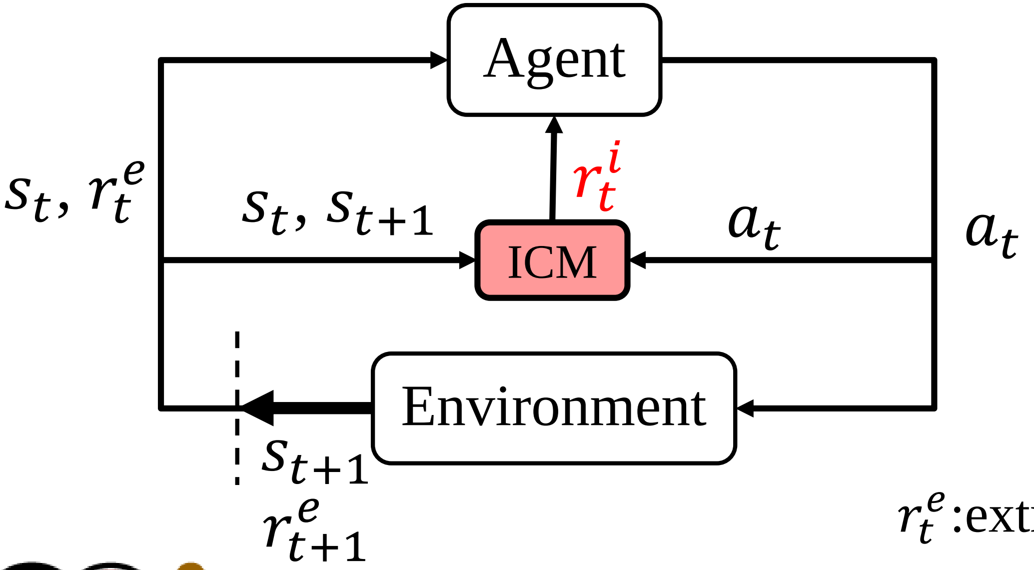
- Intrinsic Curiosity Module (ICM)
- Random Network Distillation (RND)
- Never Give Up (NGU)
- Agent57

Intrinsic Curiosity Module (ICM)

- Sparse reward environment
  - Almost every state-action pair causes no reward
    - E.g., Navigation in maze: Only the actions lead to the terminal state have rewards
  - Hard for a RL agent to learn a good policy
    - Usually need certain auxiliary tasks to help, like grid-cell agent
- Intrinsic motivation
  - Child can entertain himself/herself without any reward signal
  - Intrinsic reward signal enable the agent to explore the environment and discover novel states

Intrinsic Curiosity Module (ICM)

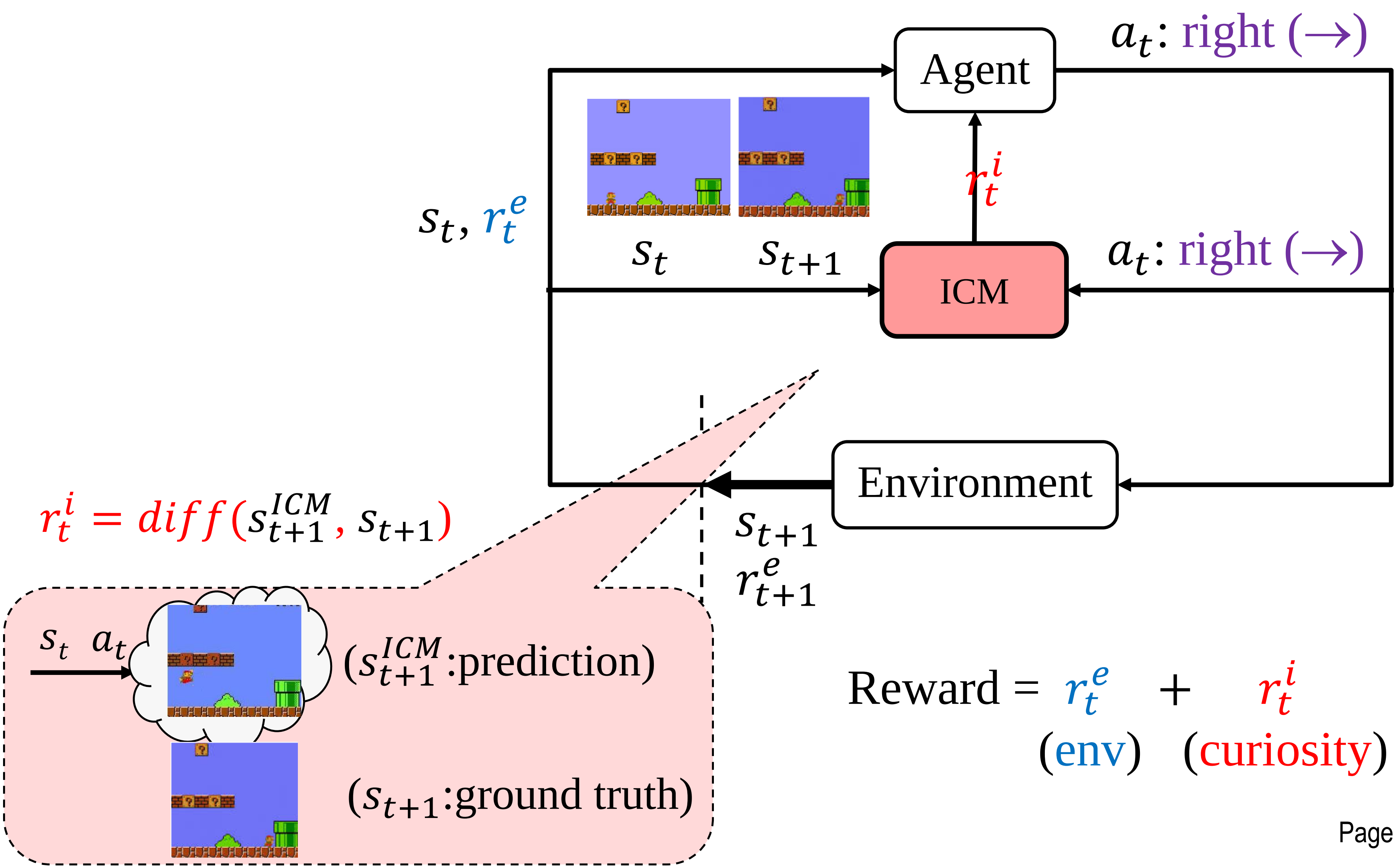
- Agent can learn from "curiosity reward" signal even if there is no "extrinsic reward"
- Curiosity reward signal is produced by Intrinsic Curiosity Module (ICM)



r_t^e:extrinsic (environment) reward

r_t^i:intrinsic reward

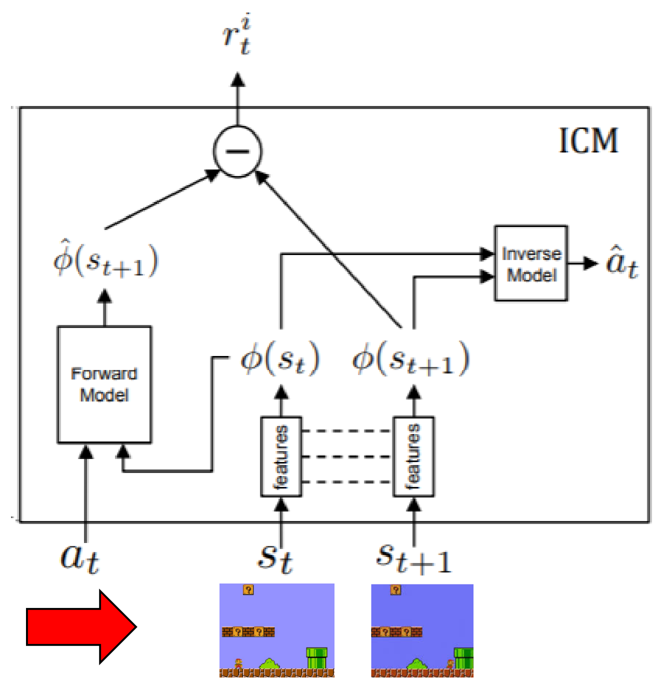
ICM Example



Reward = r_t^e + r_t^i
(env) (curiosity)

Intrinsic Curiosity Module (ICM)

- Architecture of ICM
  - Input: action (a_t), state (s_t) and next state (s_{t+1}) from replay buffer
  - Feature extractor for state and next state
    - $\phi(s_t)$: feature of the state (s) for time t
  - Inverse dynamics model
    - Surmise the taken action ($\hat{a}_t$)
  - Forward model
    - Prediction the feature of next state $\hat{\phi}(s_{t+1})$
  - Output: intrinsic reward (${r^i}_t$)
    - Curiosity reward = feature prediction error
  - Prediction of raw image will suffer from noise

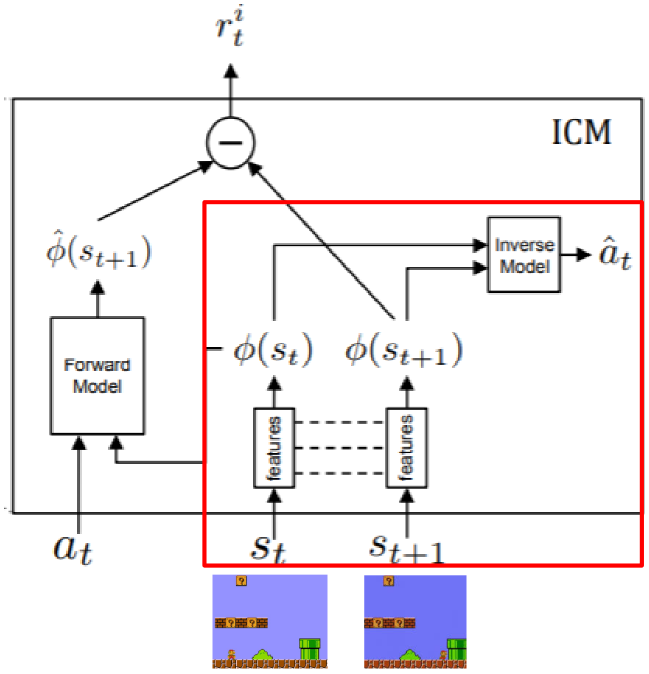


Intrinsic Curiosity Module (ICM)

- Inverse dynamics model
  - Give two consecutive states to surmise the taken action
  - Loss: Discrepancy of the action

$$\min_{\theta_I} L_I(\hat{a}_t, a_t)$$

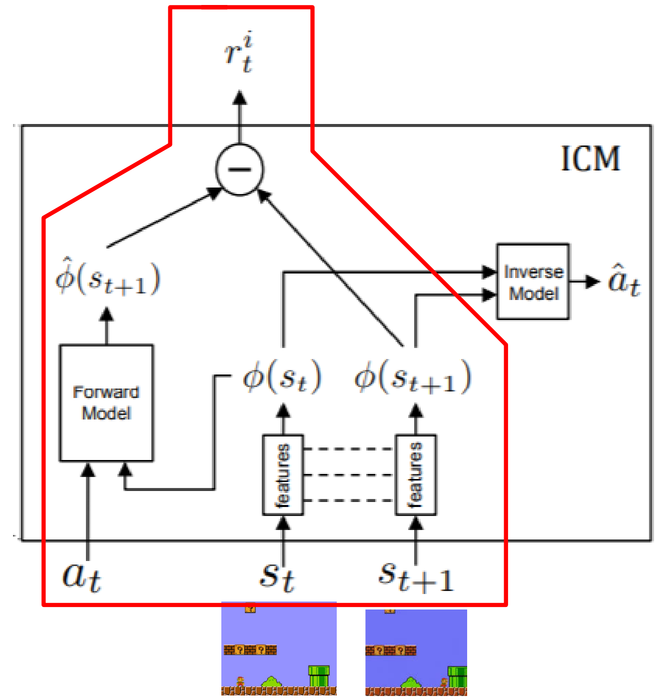
  - Learns a feature space that encodes information relevant for predicting the agent's actions only



Intrinsic Curiosity Module (ICM)

- Forward model
  - Give a state feature and an action to predict next state feature
  - Loss: Discrepancy of the features

$$L_F\left(\phi(s_t), \hat{\phi}(s_{t+1})\right) = \frac{1}{2} \| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \|_2^2$$



- Total loss $L = L_F + L_I$
- Intrinsic Reward
  - For calculating curiosity reward, η is a scaling factor (>0)

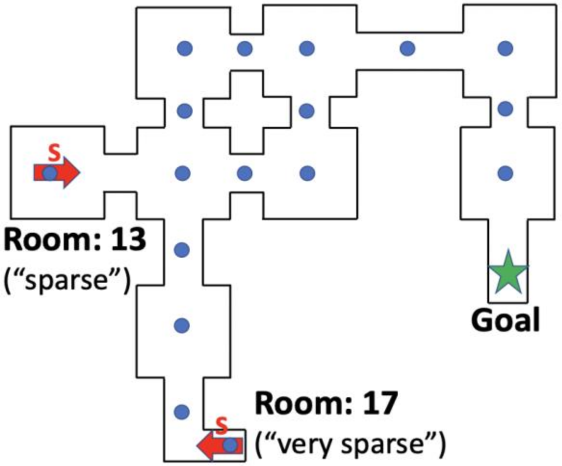
$$r_t^i = \frac{\eta}{2} \| \hat{\phi}(s_{t+1}) - \phi(s_{t+1}) \|_2^2$$

  - Encourage agent to take actions for exploration

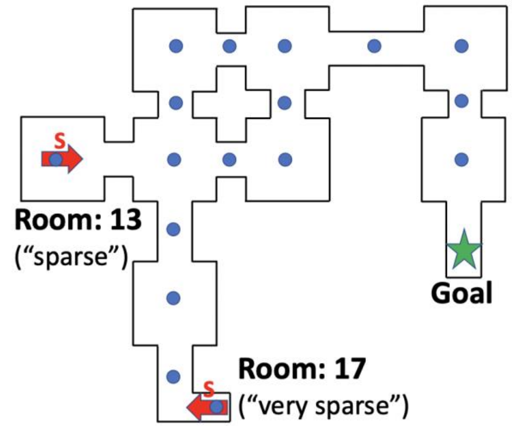
Experiments - ICM

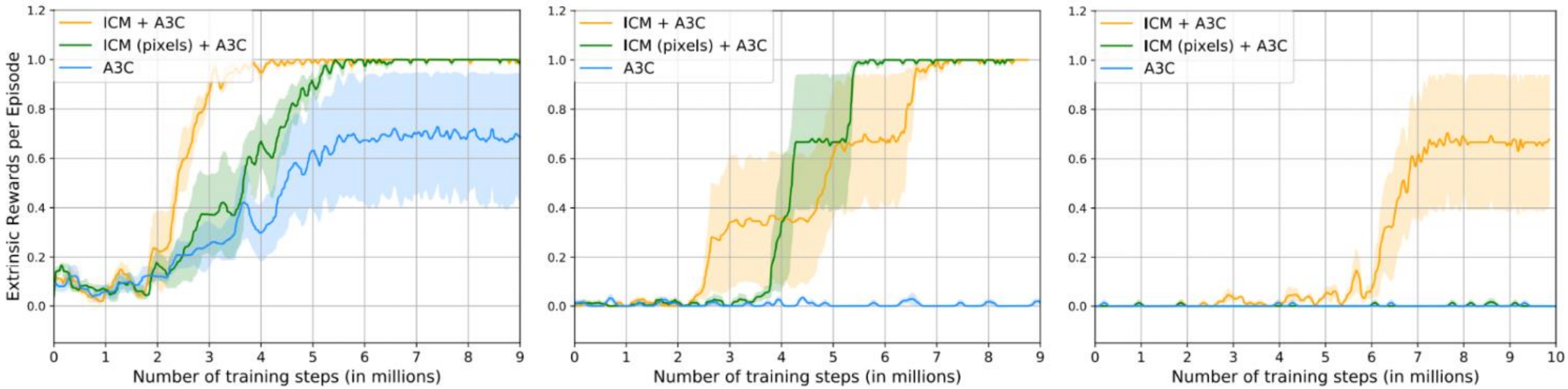
- VizDoom
- 4 action space
  - Move forward / left / right and no action
- Terminal:
  - Find goal: +1 reward
  - > 2100 steps

Setting	Location space	Optimal step
Dense	Blue point	<250
Sparse	Room 13	270
Very sparse	Room 17	370



Experiments - ICM





(a) "dense reward" setting (b) "sparse reward" setting (c) "very sparse reward" setting

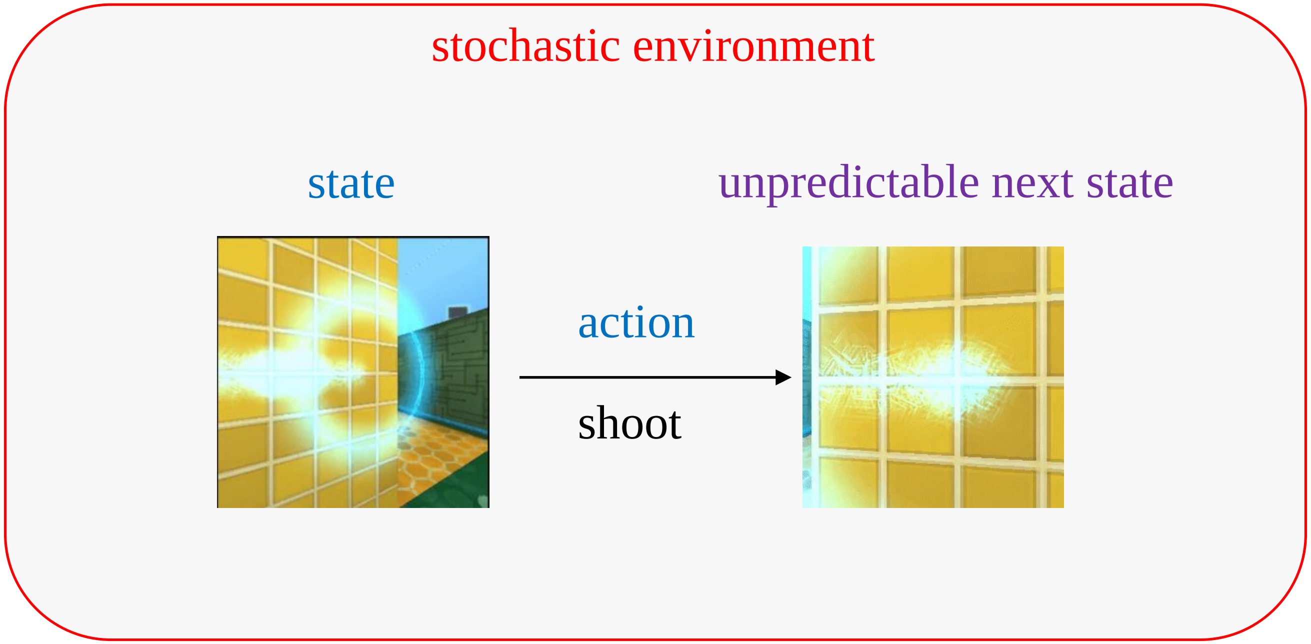
ICM + A3C: full algorithm
ICM (pixels) + A3C: ICM without the inverse model

Noisy TV Problems

- Next-state prediction agents (e.g. ICM) can be attracted by stochastic (random) or noisy elements in the environment.

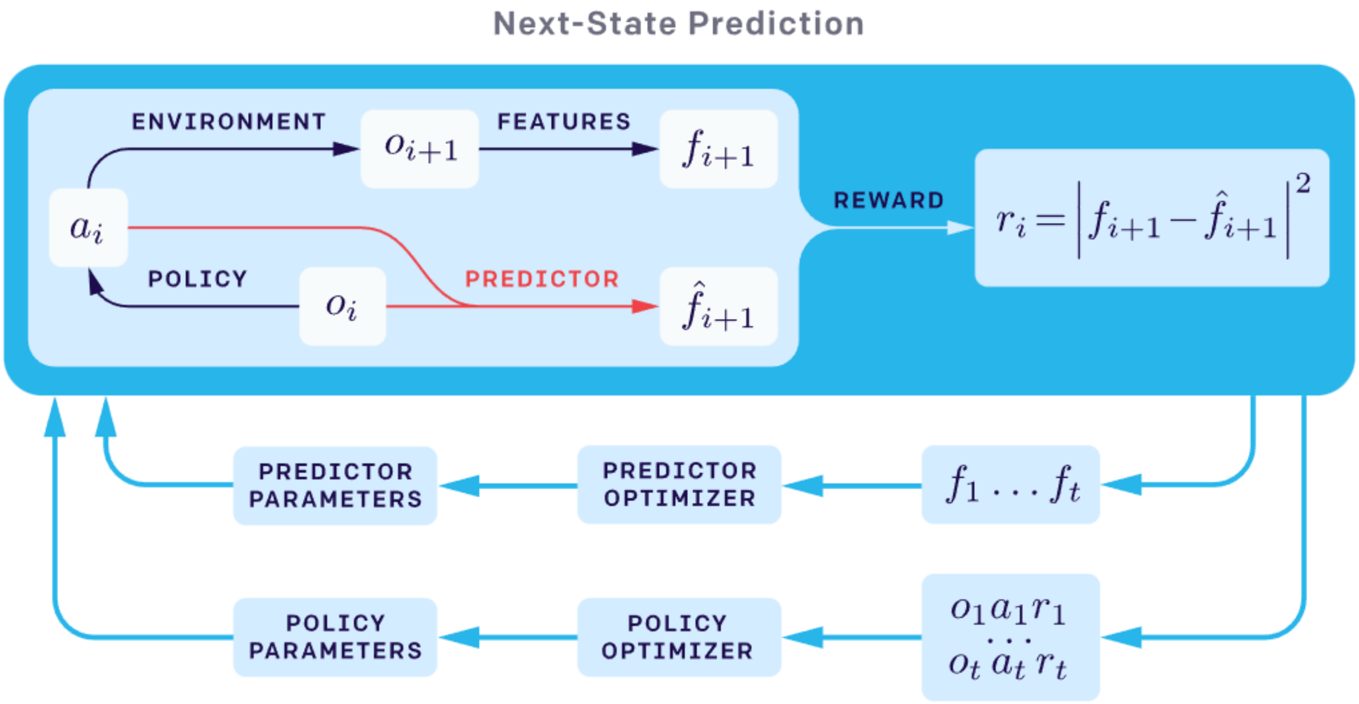
Noisy TV Problems

- At every timestamp, the curiosity reward will be high because it will be very hard to predict the next state (since the next state is random).



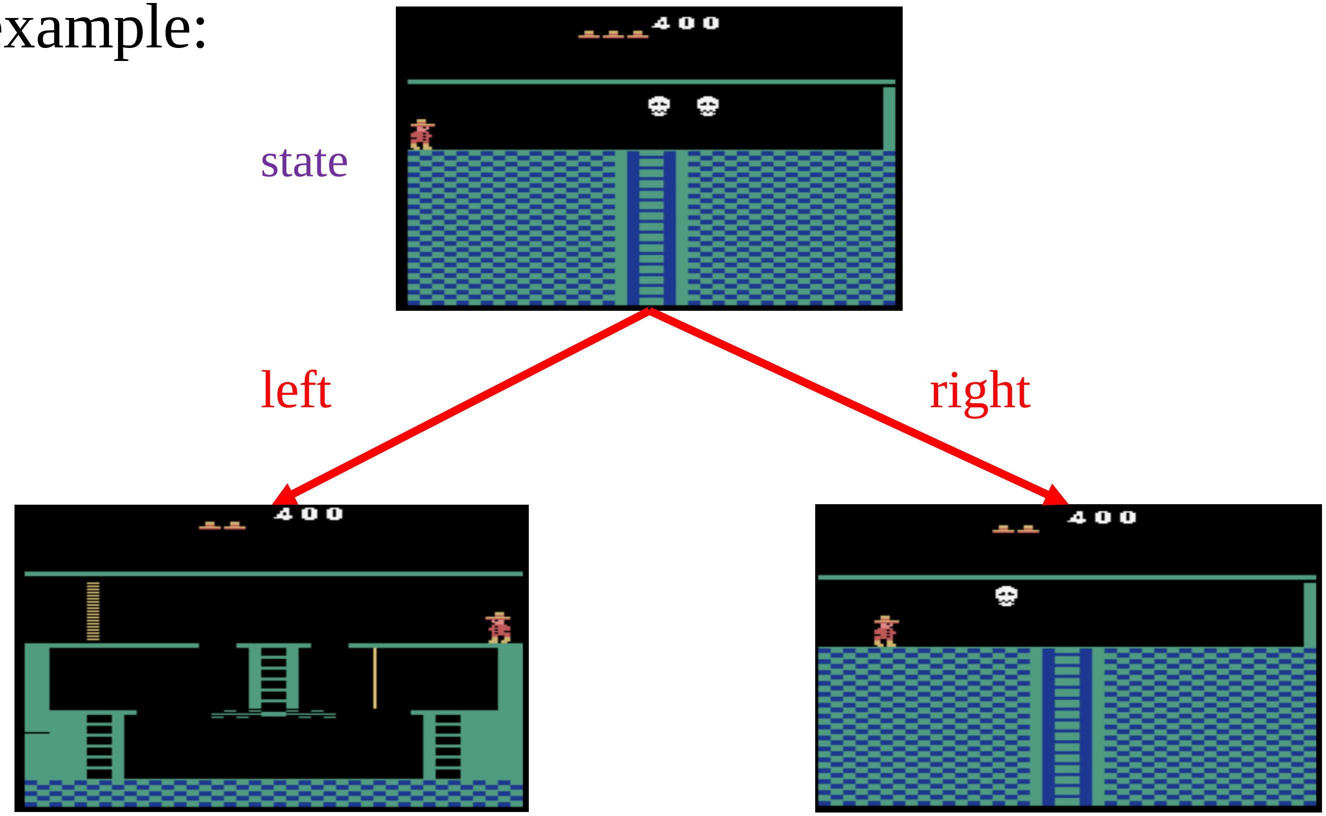
Noisy TV Problems

- Next-state prediction agents tend to visit the state that he can’t predict.



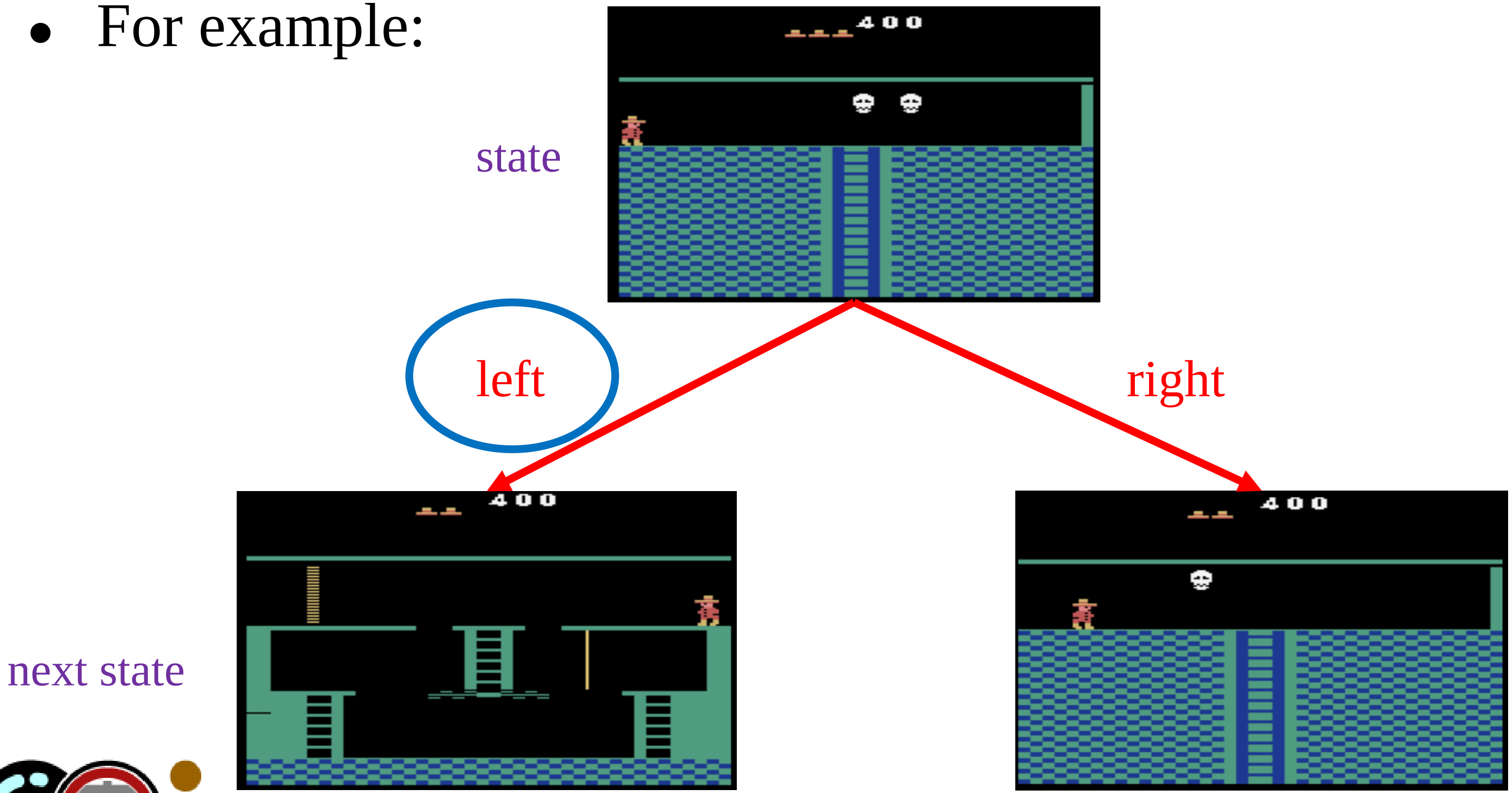
Noisy TV Problems

- Next-state prediction agents tend to visit the state that he can’t predict.
- For example:



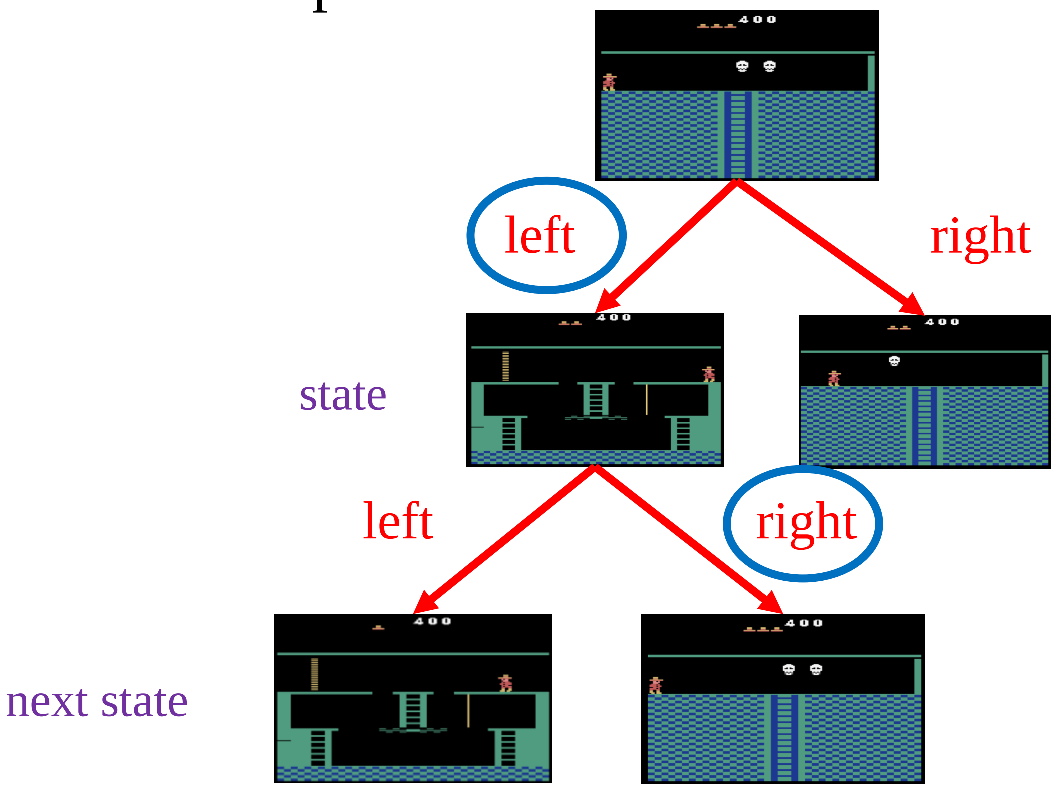
Noisy TV Problems

- Next-state prediction agents tend to visit the state that he can’t predict.
- For example:



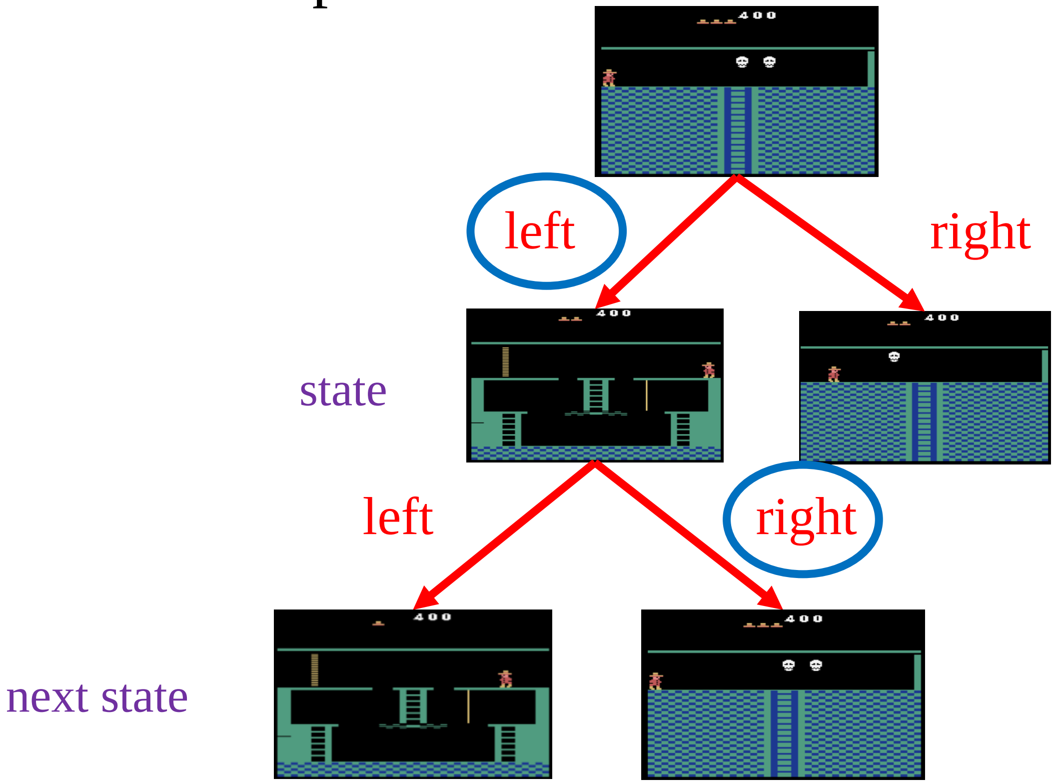
Noisy TV Problems

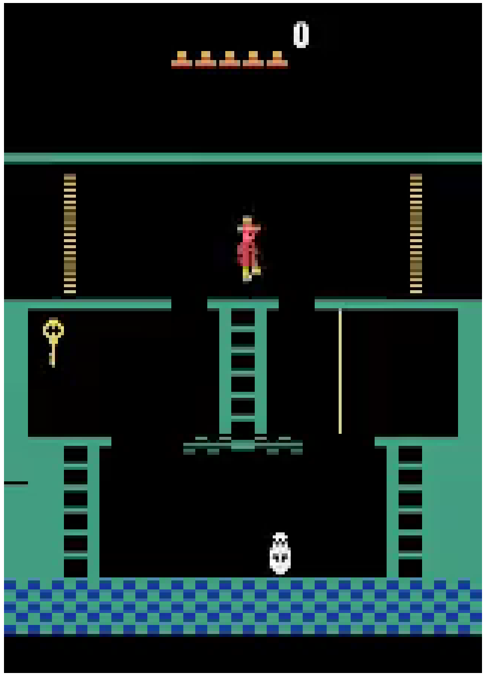
- For example:



Noisy TV Problems

- For example:

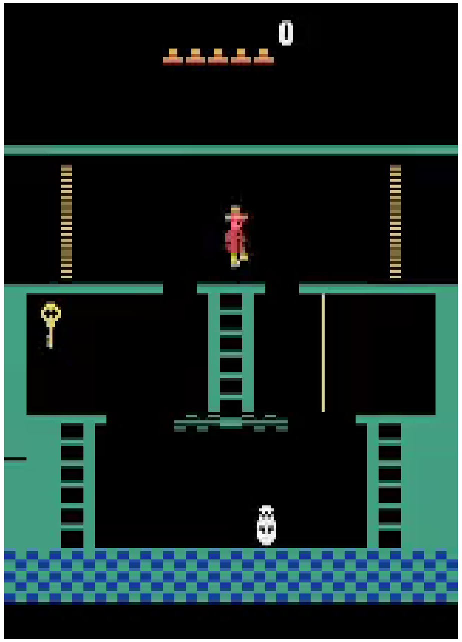




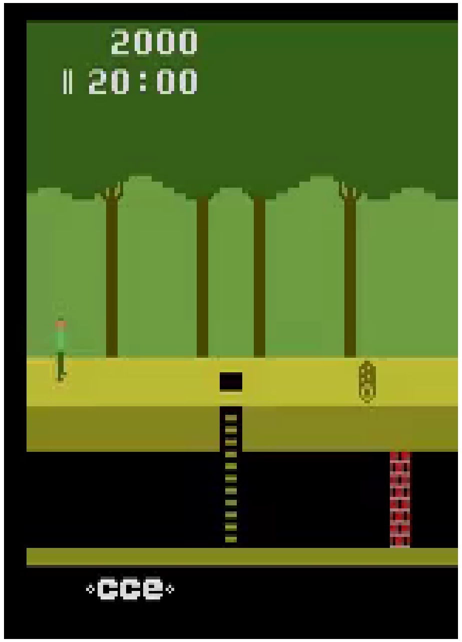
Noisy TV Problems

- More example (video from OpenAI website):

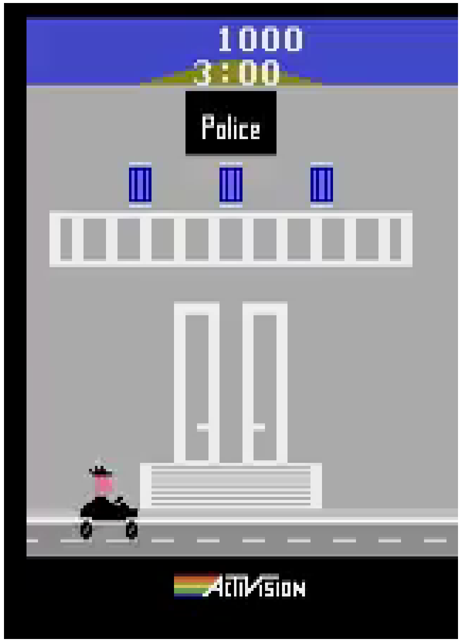
Montezuma



Pitfall



Private Eye



https://openai.com/research/reinforcement-learning-with-prediction-based-rewards

Advanced Exploration

- Intrinsic Curiosity Module (ICM)
- Random Network Distillation (RND)
- Never Give Up (NGU)
- Agent57

Random Network Distillation (RND)

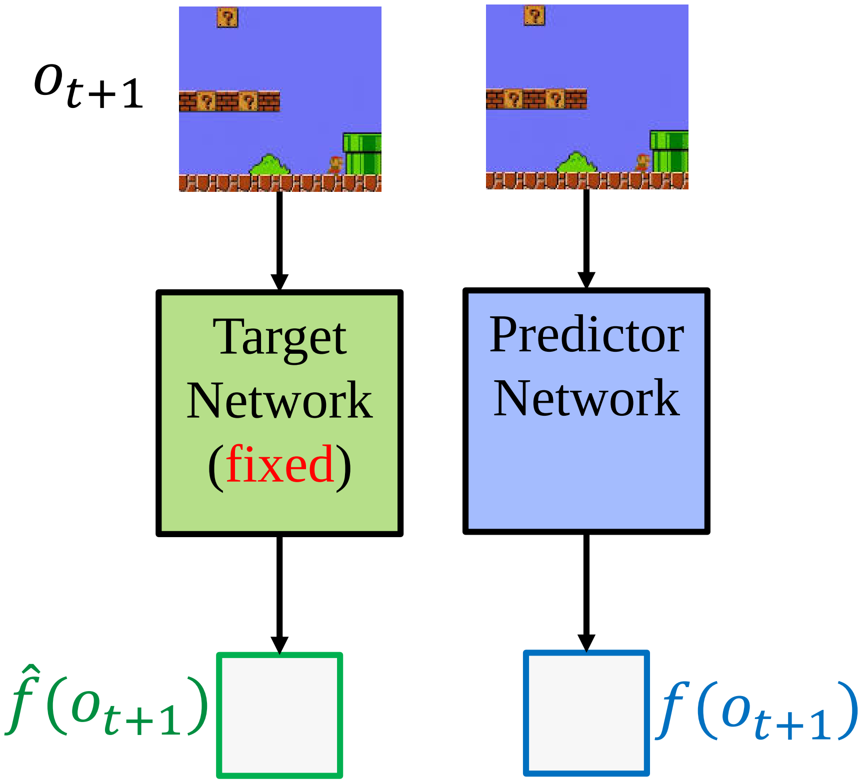
- Handle RL in sparse reward environments
  - With random exploration, RL can find a good (converged) policy with dense reward environment
  - However, it's hard to find a good policy in sparse reward environments
- Random network distillation
  - Exploration bonus
  - Prediction error on features between predictor network and target network
    - Predictor network: a updating network
    - Target network: a fixed and randomly initialized network
- Results
  - State-of-the-art in Montezuma's Revenge (exceeded by Go-Explore)
    - Find all 24 rooms in the game
    - Exceed all previous methods and human's average score

Sources of Prediction Errors

- Amount of training data.
  - Prediction error is high where few similar examples were seen by the predictor (epistemic uncertainty).
- Stochasticity.
  - Prediction error is high because the target function is stochastic (aleatoric uncertainty).
    - Stochastic transitions are a source of such error for forward dynamics prediction. (Noisy TV)
- Model misspecification.
  - Prediction error is high because necessary information is missing, or the model class is too limited to fit the complexity of the target function.
- Learning dynamics.
  - Prediction error is high because the optimization process fails to find a predictor in the model class that best approximates the target function.

RND's idea

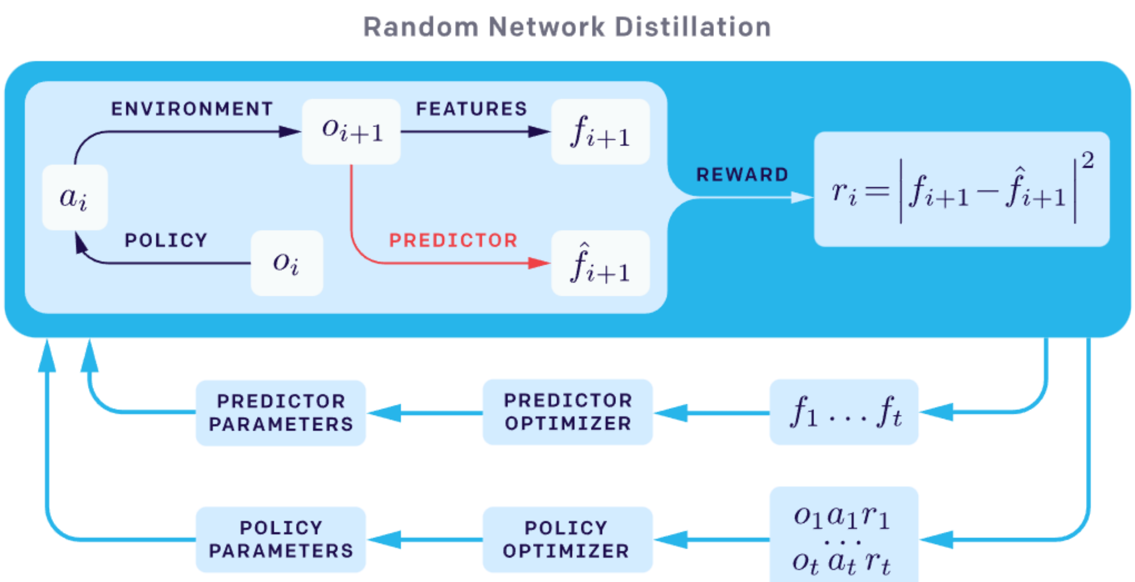
- The reward (r_t) for agent is separated as environment reward (e_t) and exploration bonus (i_t) $r_t = e_t + i_t$
- Feature prediction error should be
  - Higher for novel states
  - Lower for those states have been trained on
- Two networks
  - Target network
    Randomly initialized and fixed
  - Predictor network
    minimize the MSE with target network



$$i_t = MSE(\hat{f}(o_{t+1}), f(o_{t+1}))$$

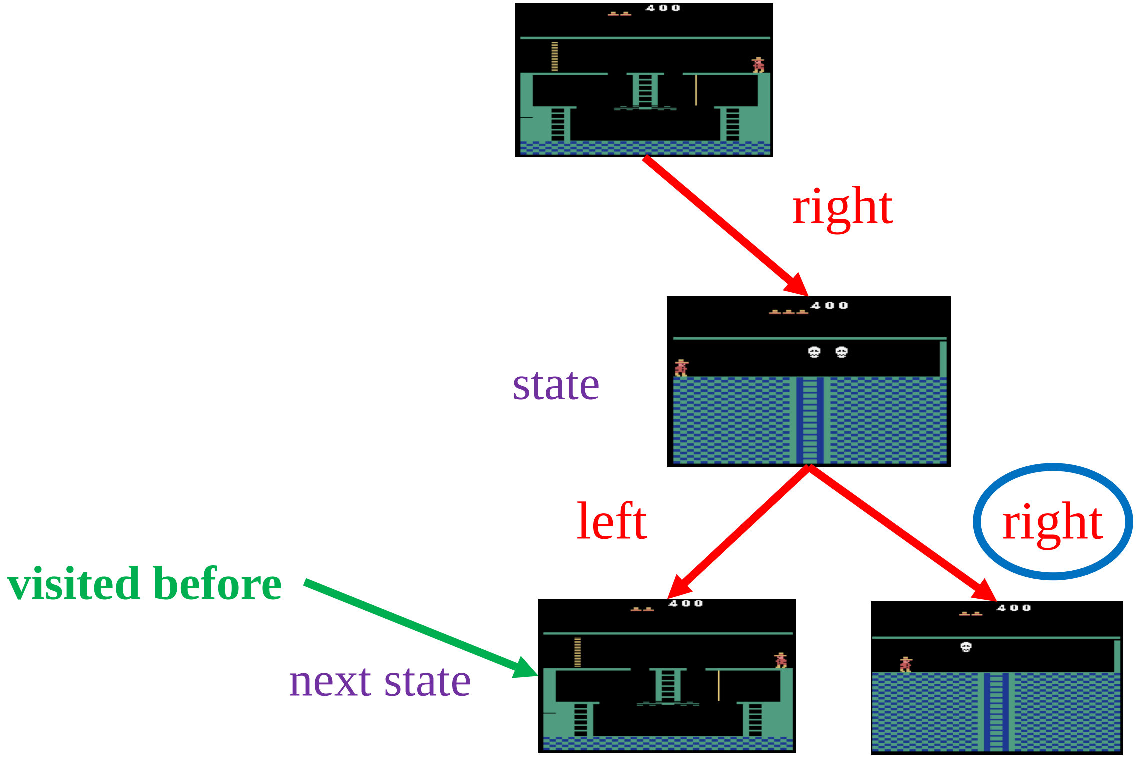
Noisy TV Problems

- RND agents tend to visit the state that he has never visited before.
- As the prediction error decreases, the agent becomes less attracted to the noisy states than to other unexplored states. This reduces the Noisy TV error.



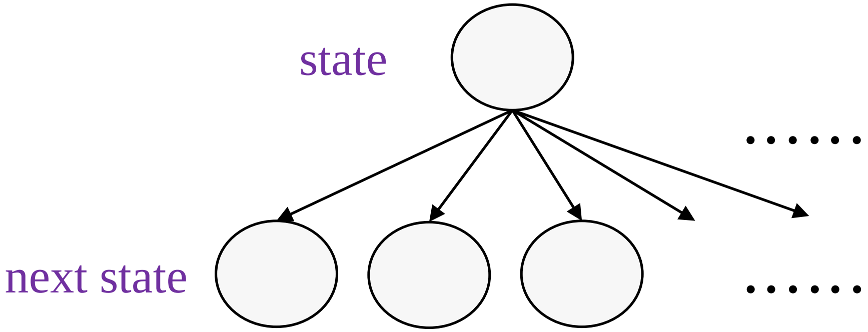
Noisy TV Problems

- For example:



ICM vs. RND

- ICM tends to visit the state that he can't predict.
  - Cause the Noisy TV problem and other problems.
- RND tends to visit the state that he has never visited before.
  - Can alleviate the Noisy TV problem and other problems.
- RND is easier to implement.



Noisy TV with infinite channel

Algorithm 1 RND pseudo-code

$N \leftarrow$ number of rollouts
$N_{\text{opt}} \leftarrow$ number of optimization steps
$K \leftarrow$ length of rollout
$M \leftarrow$ number of initial steps for initializing observation normalization
$t = 0$
Sample state $s_0 \sim p_0(s_0)$
for $m = 1$ **to** M **do**
  sample $a_t \sim \text{Uniform}(a_t)$
  sample $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$
  Update observation normalization parameters using s_{t+1}
  t += 1
end for
for $i = 1$ **to** N **do**
  for $j = 1$ **to** K **do**
    sample $a_t \sim \pi(a_t|s_t)$
    sample $s_{t+1}, e_t \sim p(s_{t+1}, e_t|s_t, a_t)$
    calculate intrinsic reward $i_t = \|\hat{f}(s_{t+1}) - f(s_{t+1})\|^2$
    add $s_t, s_{t+1}, a_t, e_t, i_t$ to optimization batch B_i
    Update reward normalization parameters using i_t
    t += 1
  end for
  Normalize the intrinsic rewards contained in B_i
  Calculate returns $R_{I,i}$ and advantages $A_{I,i}$ for intrinsic reward
  Calculate returns $R_{E,i}$ and advantages $A_{E,i}$ for extrinsic reward
  Calculate combined advantages $A_i = A_{I,i} + A_{E,i}$
  Update observation normalization parameters using B_i
  for $j = 1$ **to** N_{opt} **do**
    optimize θ_π wrt PPO loss on batch B_i, R_i, A_i using Adam
    optimize $\theta_{\hat{f}}$ wrt distillation loss on B_i using Adam
  end for
end for

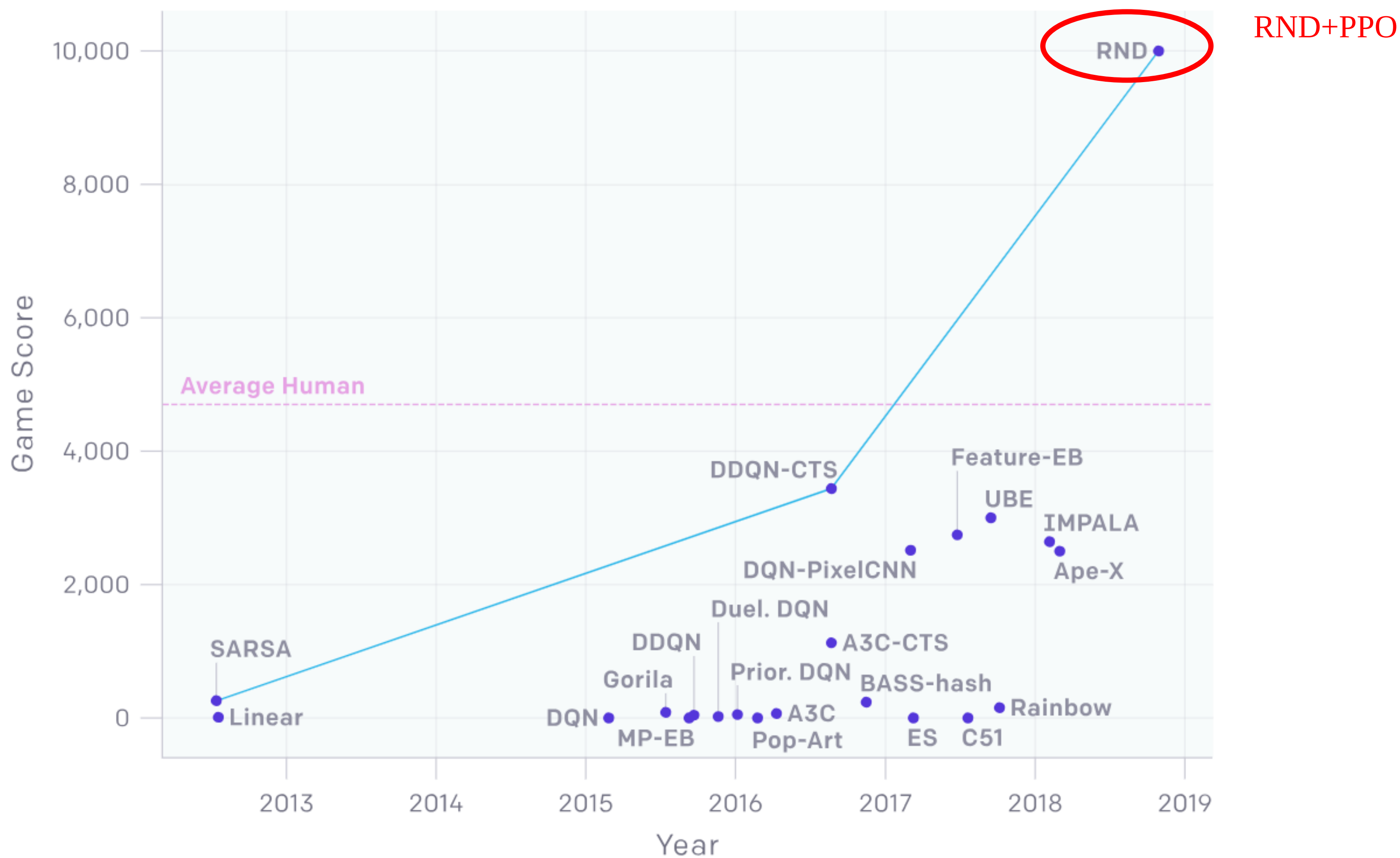
Initialized observation normalization parameters

Collect K transitions to batch B_i

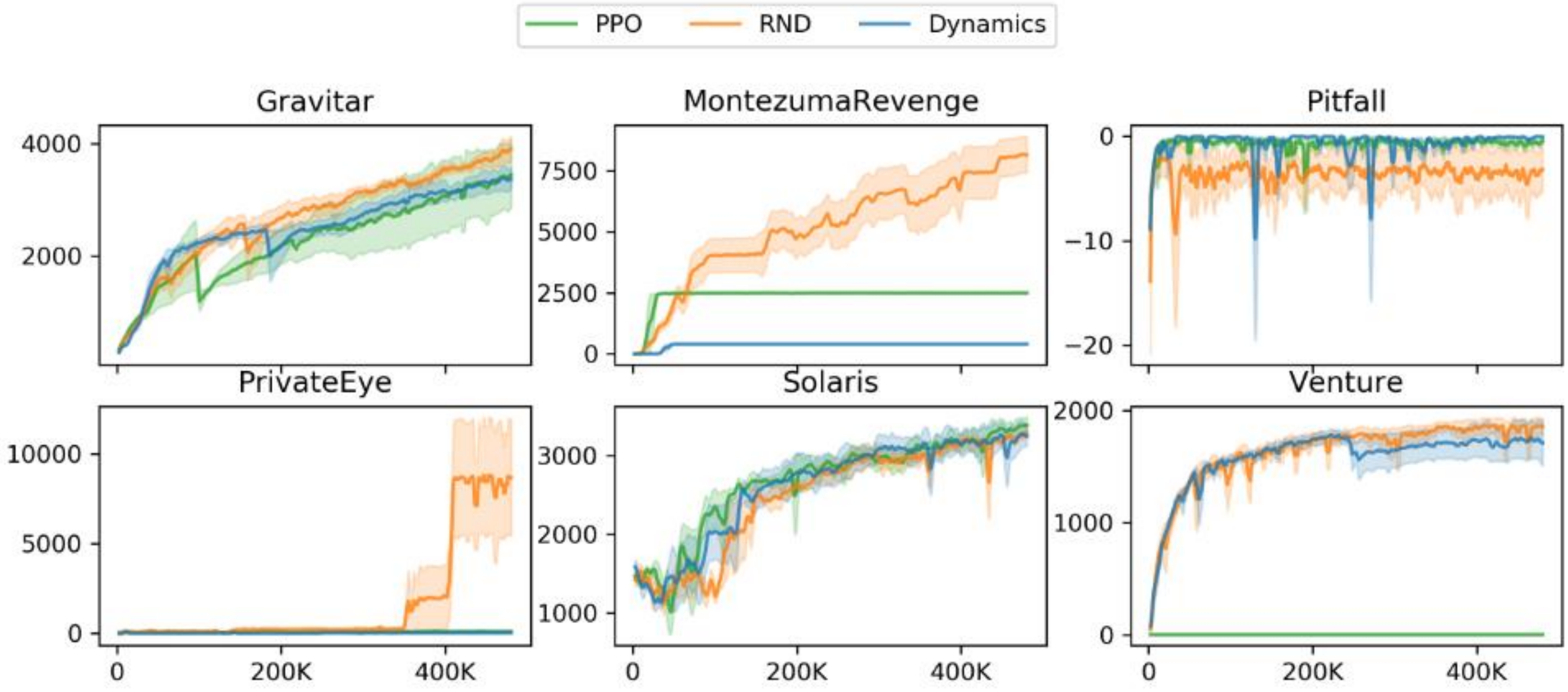
Calculate returns and advantages

Update policy with batch B_i, R_i, A_i
Update predictor using B_i

RND in Montezuma's Revenge



Experiments - RND



Dynamics: dynamics prediction-based exploration method

Advanced Exploration

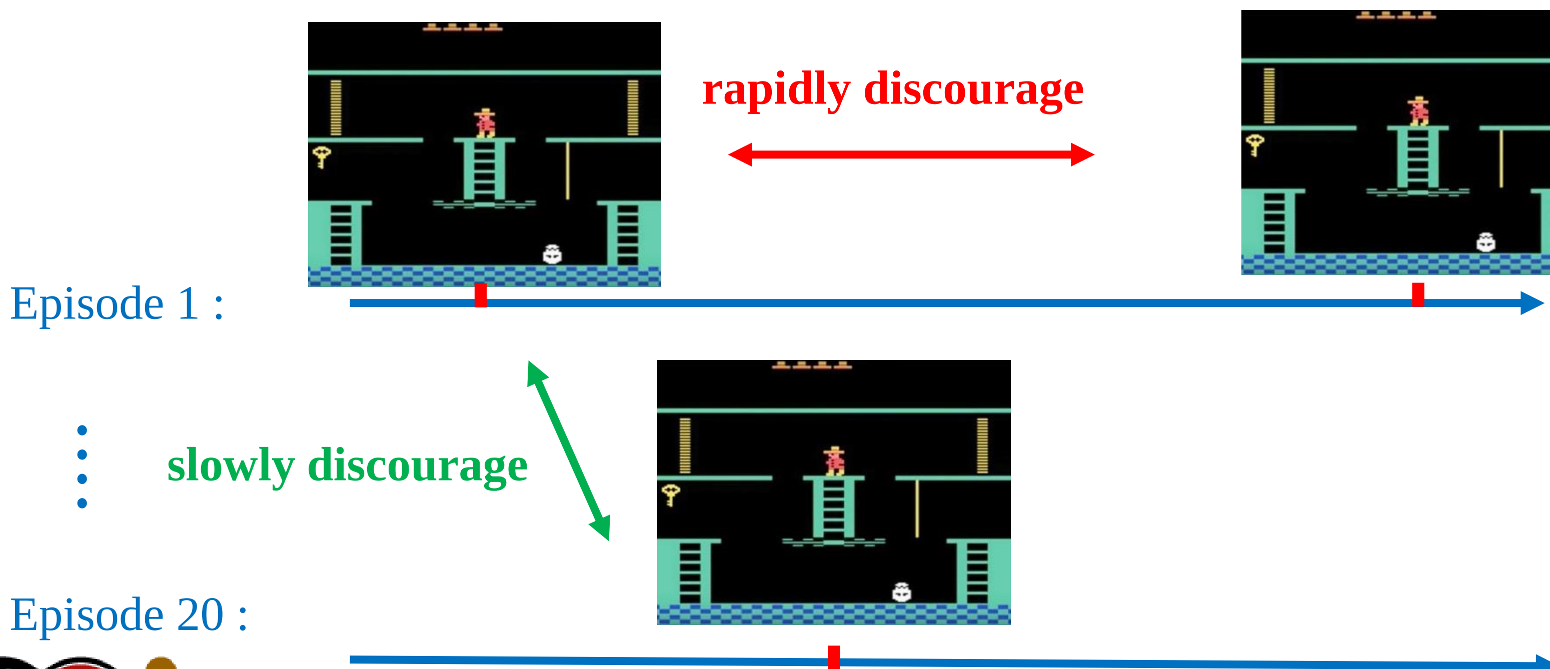
– Intrinsic Curiosity Module (ICM)
– Random Network Distillation (RND)
– Never Give Up (NGU)
– Agent57

NGU - Define Reward

- Define reward as $r_t^{\beta_i} = r_t^e + \beta_i r_t^i$
  - r_t^e: extrinsic reward
  - r_t^i: intrinsic reward
  - β_i: exploration rate

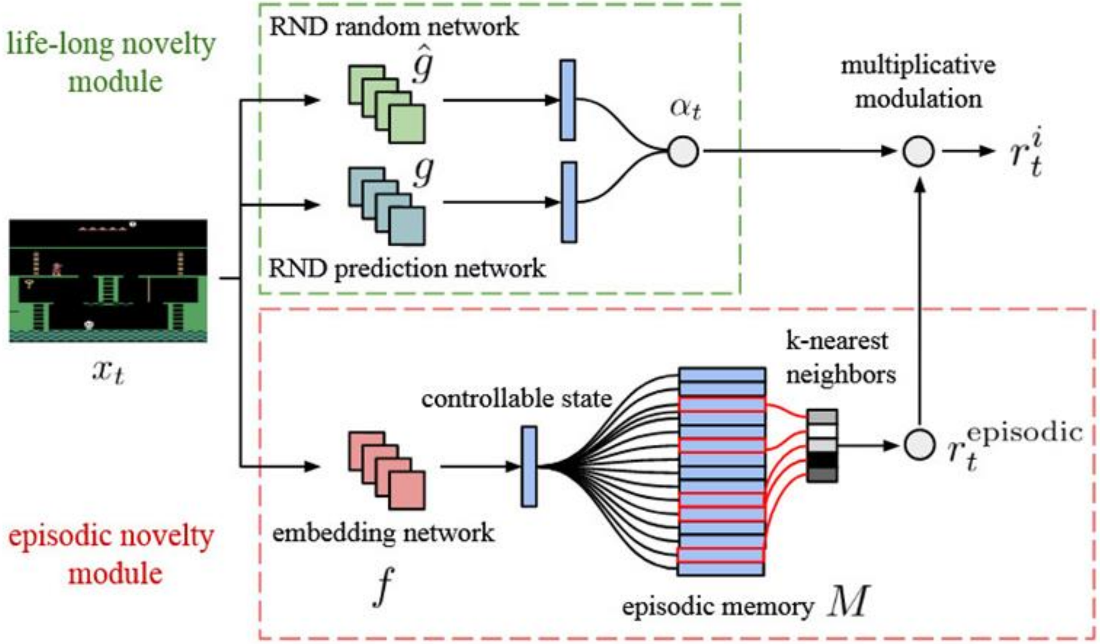
NGU - Intrinsic Reward

- **Life-long**: slowly discourage visiting the states visited many times across episodes.
- **Episodic**: rapidly discourage revisiting the same state within the same episode.

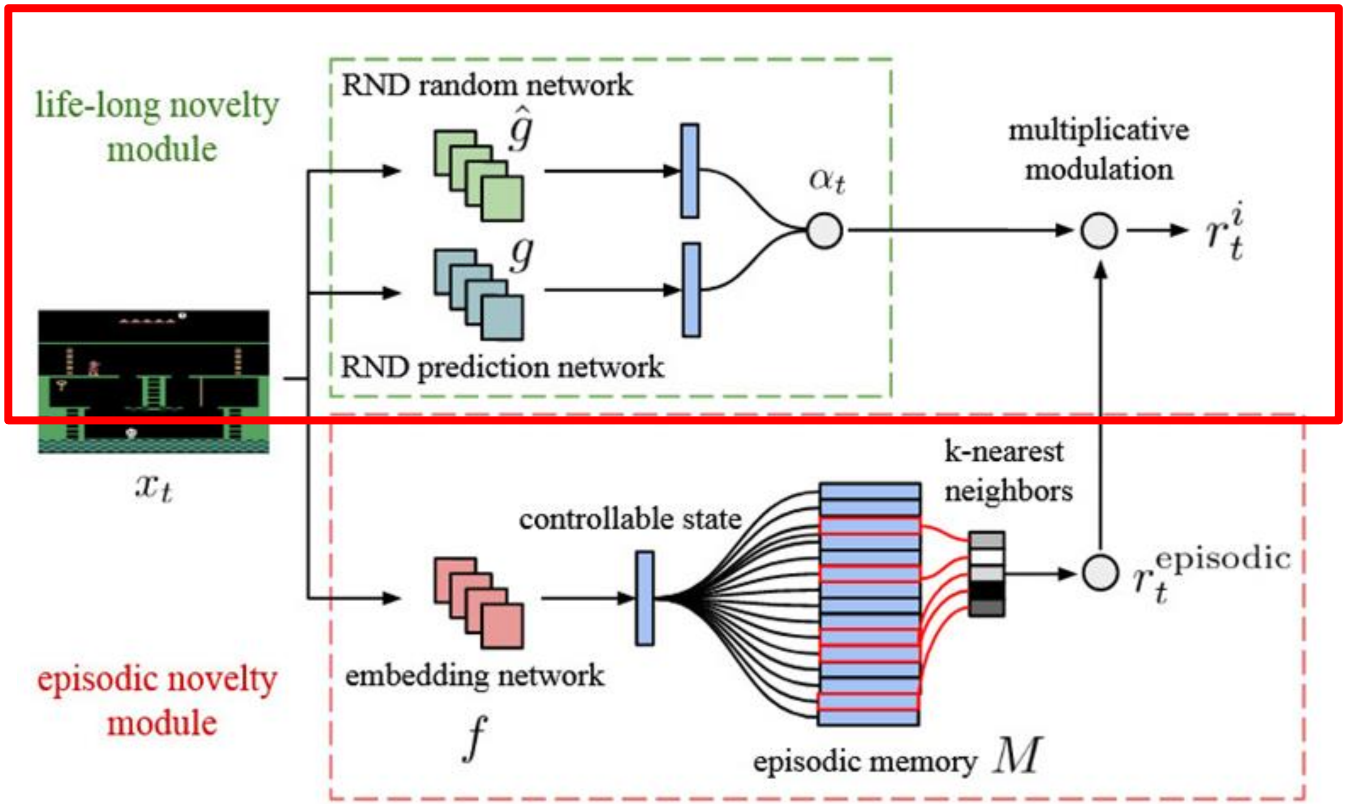


NGU - Intrinsic Reward

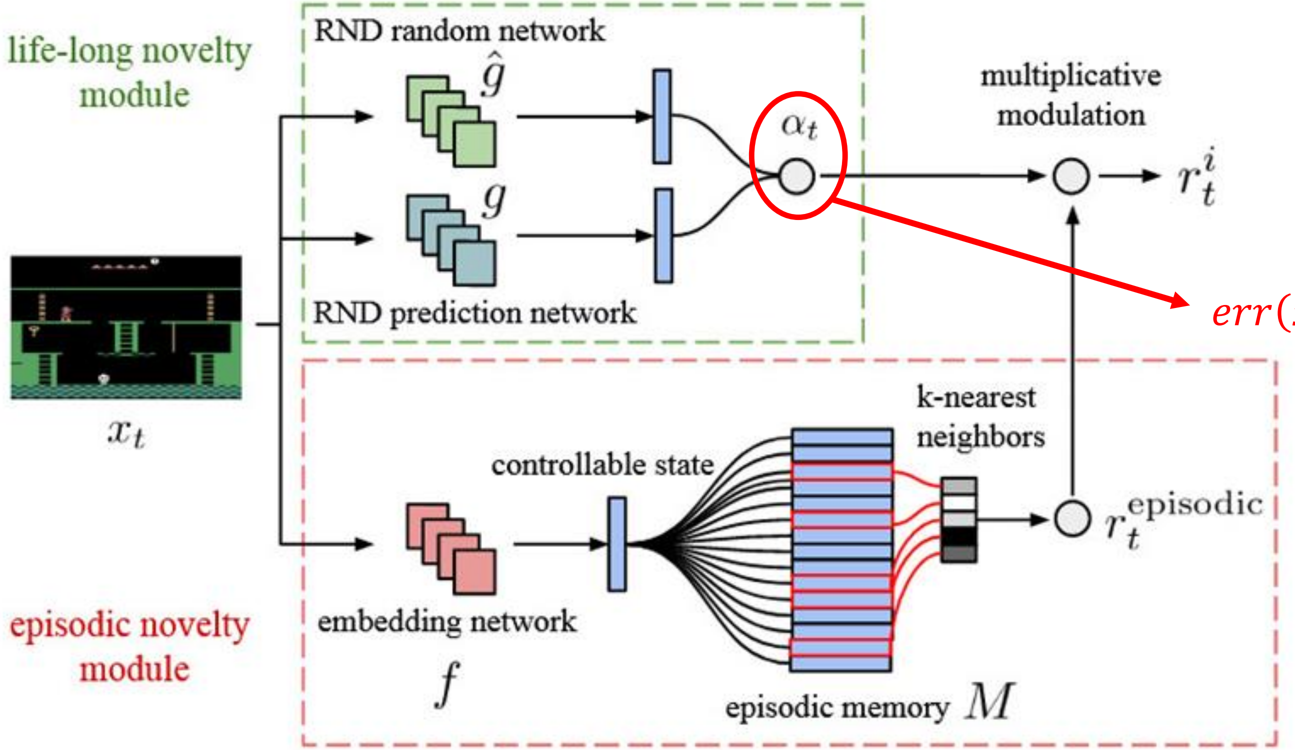
- $r_t^i = r_t^{episodic} \cdot \min(\max(\alpha_t, 1), 5)$
- **Life-long**: slowly discourage visiting the states visited many times across episodes.
- **Episodic**: rapidly discourage revisiting the same state within the same episode.



NGU - Intrinsic Reward



NGU - Intrinsic Reward



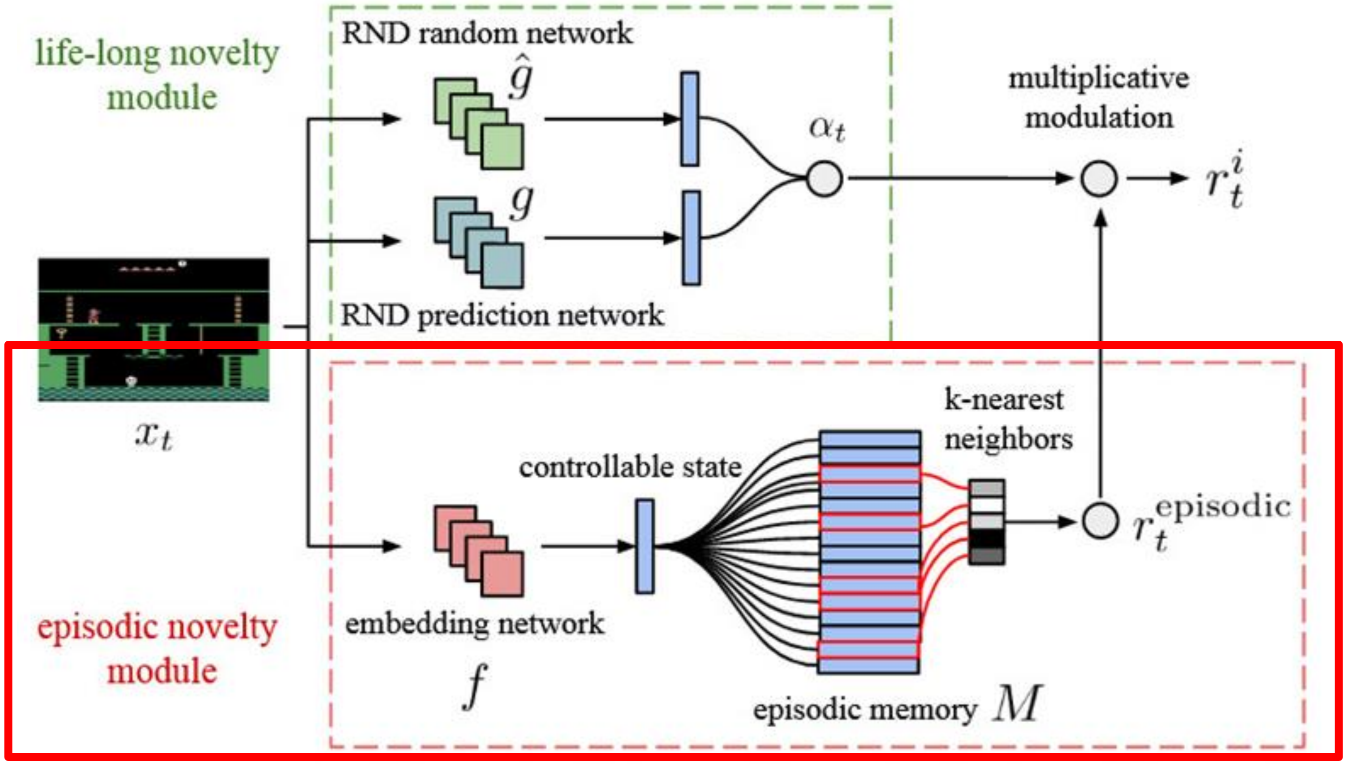
$$err(x_t) = \left\|\hat{g}(x_t;\hat{\theta}) - g(x_t;\theta)\right\|^2$$

$$\alpha_t = 1 + \frac{err(x_t) - \mu_e}{\sigma_e}$$

μ_e : mean for err

σ_e : standard deviation for err

NGU - Intrinsic Reward



NGU - Episodic Novelty Model

- Rapidly discourage revisiting the same state within the same episode.
- Workflow

state

↓ embedding

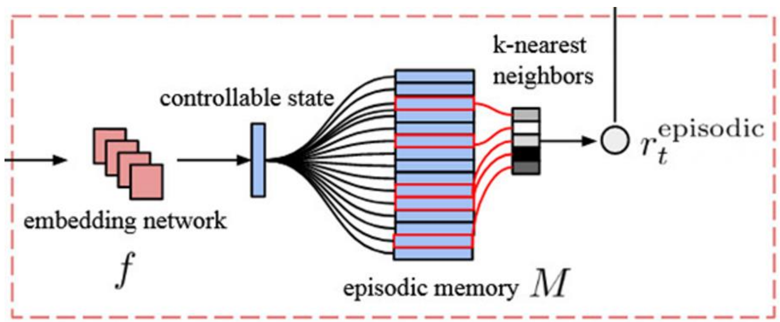
controllable state

↓

calculate $r_t^{episodic}$ from M

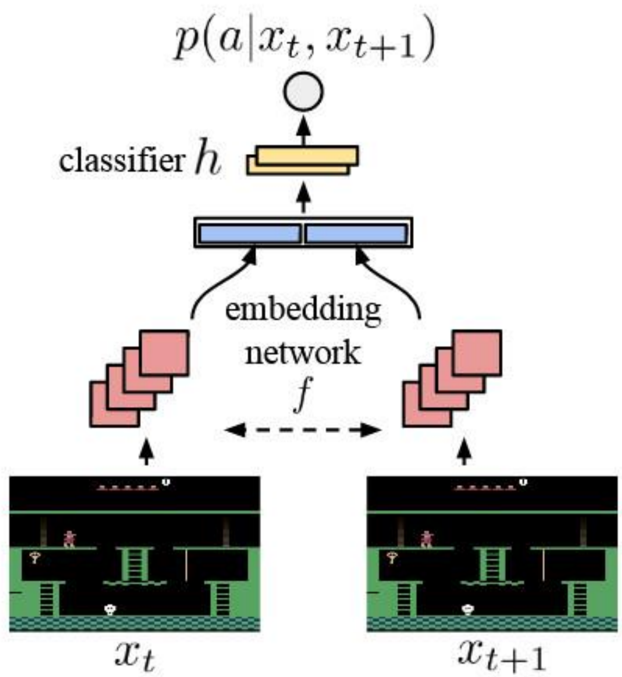
↓

store controllable state in M



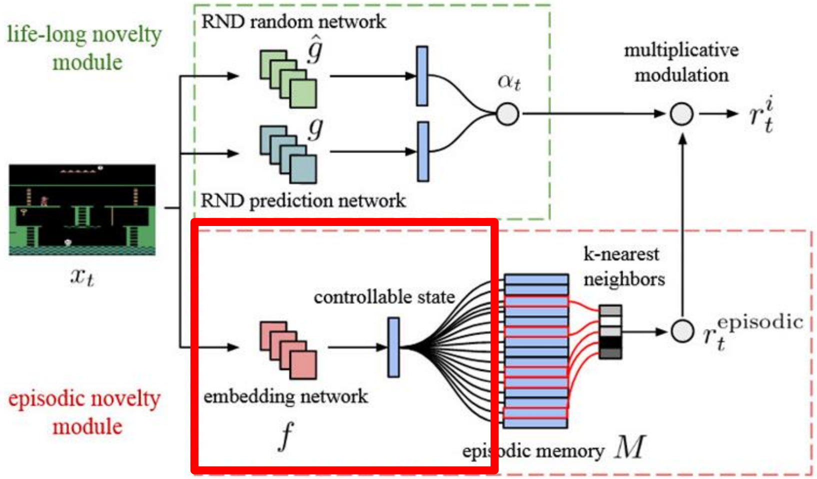
NGU - Inverse Dynamic Model

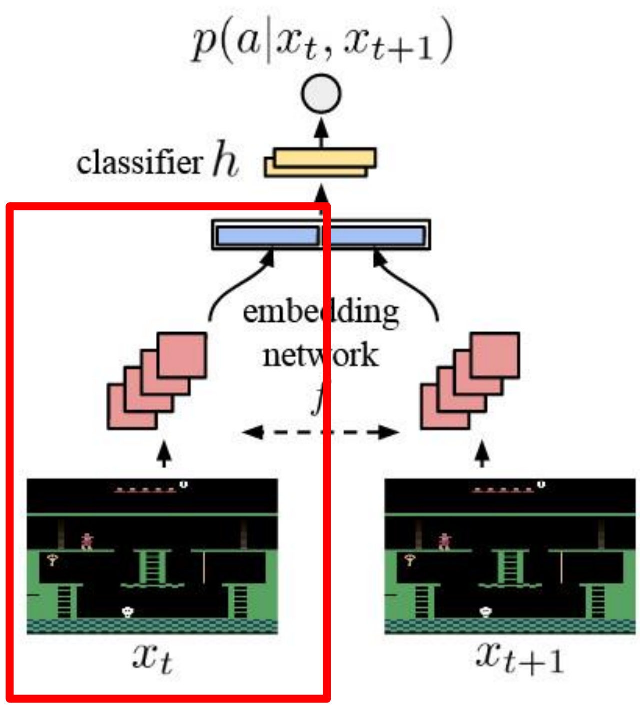
- Given a state, next state, predict the action between these two.
- Curiosity exploration uses the features from this model.
- The latent codes from this model can reserve features related to actions.



NGU - Inverse Dynamic Model

- Given a state, next state, predict the action between these two.
- Curiosity exploration uses the features from this model.
- The latent codes from this model can reserve features related to actions.





NGU - Episodic Novelty Model

state

↓ embedding

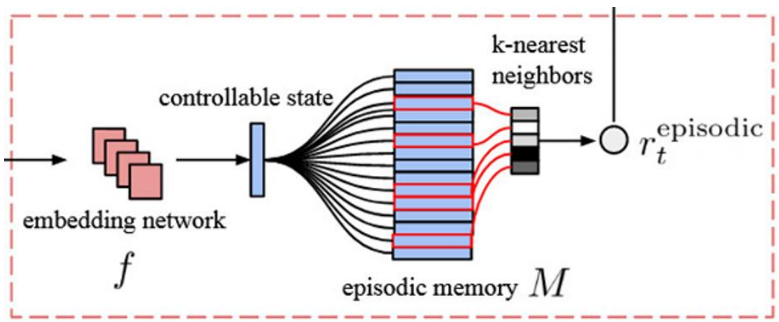
controllable state

↓

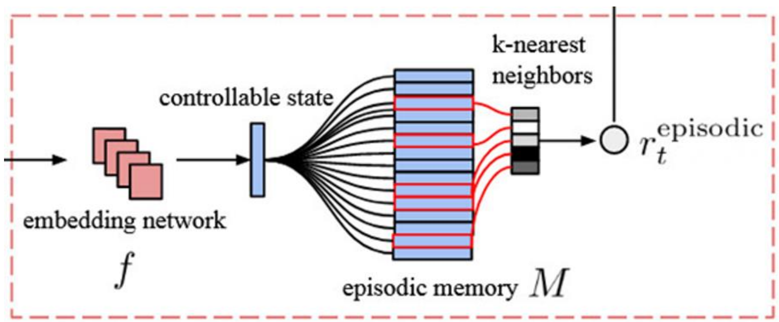
calculate $r_t^{episodic}$ from M

↓

store controllable state in M



NGU - Episodic Novelty Model



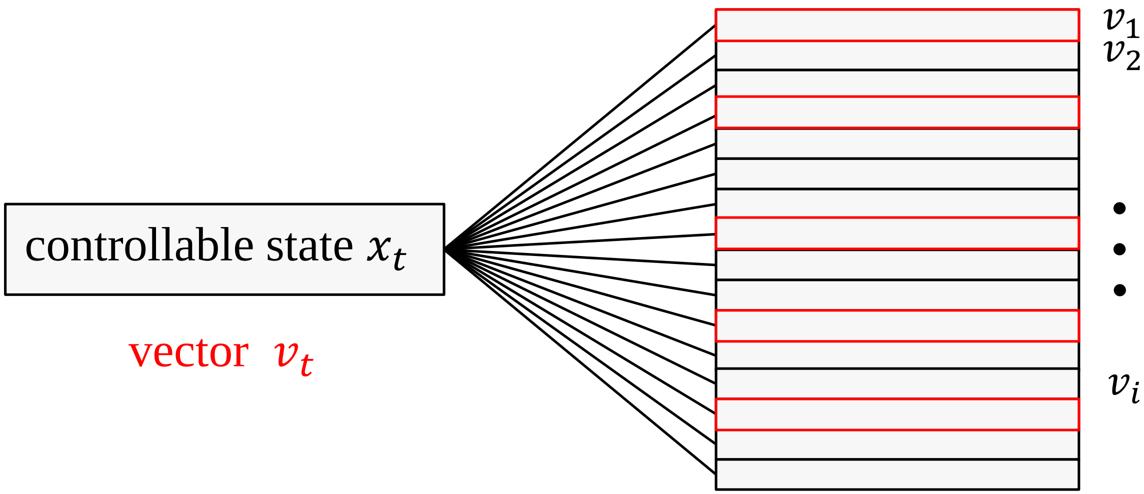
Euclidean distance $d(v_t, v_i)$

10
12
14
4
20
18
23
2
27
41
7
63
54
5
21
17

K-nearest neighbors
K=5

N_k

10
4
2
7
5



episodic memory M

NGU - Episodic Novelty Model

N_k

10
4
2
7
5

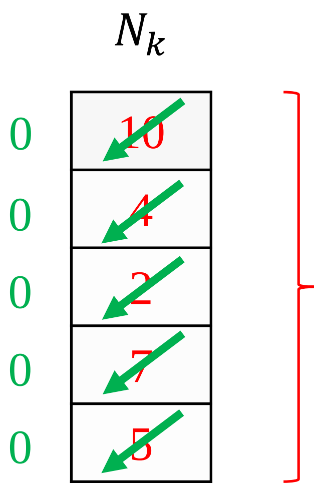
$$K(x,y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$$

small constant, 10^{-3}

Euclidean distance of x and y

Running average of Euclidean distance

NGU - Episodic Novelty Model



$$K(x,y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon} \approx 1$$

When $f(x_t), f_i$ is similar (close), then $K(f(x_t), f_i)$ close to 1.

$\sum_{f_i \in N_k} K(f(x_t), f_i) + c \approx$ pseudo visit count of state $f(x_t)$.

NGU - Episodic Novelty Model

N_k

10
4
2
7
5

$$K(x,y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$$

small constant, 10^{-3}

Euclidean distance of x and y

Running average of Euclidean distance

$$r_t^{episodic} = \frac{1}{\sqrt{n(f(x_t))}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i)} + c}$$

pseudo visit count of state $f(x_t)$

K-nearest neighbors of $f(x_t)$

NGU - Episodic Novelty Model

Algorithm 1: Computation of the episodic intrinsic reward at time t: r_t^{episodic}.

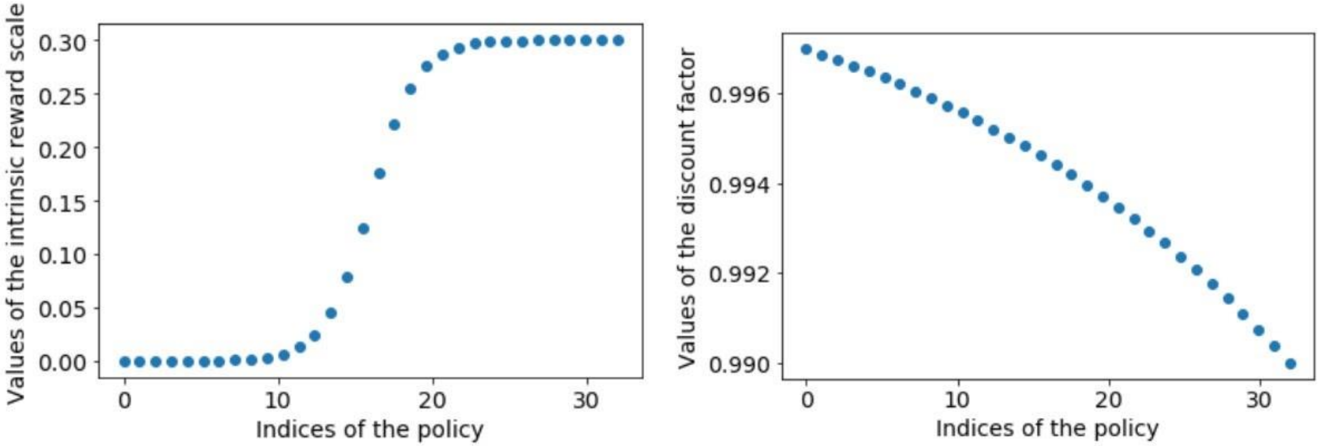
Input : M; k; $f(x_t)$; c; ϵ; ξ; s_m; d_m^2
Output : r_t^{episodic}

Compute the k-nearest neighbours of $f(x_t)$ in M and store them in a list N_k
Create a list of floats d_k of size k
/* The list d_k will contain the distances between the embedding $f(x_t)$ and its neighbours N_k. */
for $i \in \{1, \dots, k\}$ **do**
  $d_k[i] \leftarrow d^2(f(x_t), N_k[i])$
end
Update the moving average d_m^2 with the list of distances d_k
/* Normalize the distances d_k with the updated moving average d_m^2. */
$d_n \leftarrow \frac{d_k}{d_m^2}$
/* Cluster the normalized distances d_n i.e. they become 0 if too small and 0_k is a list of k zeros. */
$d_n \leftarrow \max(d_n - \xi, 0_k)$
/* Compute the Kernel values between the embedding $f(x_t)$ and its neighbours N_k. */
$K_v \leftarrow \frac{\epsilon}{d_n + \epsilon}$
/* Compute the similarity between the embedding $f(x_t)$ and its neighbours N_k. */
$s \leftarrow \sqrt{\sum_{i=1}^{k} K_v[i]} + c$
/* Compute the episodic intrinsic reward at time t: r_t^i. */
if $s > s_m$ **then**
  $r_t^{\text{episodic}} \leftarrow 0$
else
  $r_t^{\text{episodic}} \leftarrow \frac{1}{s}$

NGU - A Family of Policies

- Pairs of exploration rates β_i and discount factor γ_i .
- Long term horizons (high values of γ_j) for exploitative policies (low values of β_j) and vice versa.
  - $(\beta_i \downarrow, \gamma_i \uparrow) \rightarrow$ exploitation $\rightarrow$ long term
  - $(\beta_i \uparrow, \gamma_i \downarrow) \rightarrow$ exploration $\rightarrow$ short term

Reward: $r_t^{\beta_i} = r_t^e + \beta_i r_t^i$



(a) Values taken by the $\{\beta_i\}_{i=0}^{N-1}$ (b) Values taken by the $\{\gamma_i\}_{i=0}^{N-1}$

NGU - A Family of Policies

- During each training step, select different level of exploration.
- By doing this, we can observe data from various levels of exploration, increasing the diversity of data in the replay buffer.
- From the reward definition, we know that
  - $\beta = 0$: exploitation
  - $\beta > 0$: exploration

Reward: $r_t^{\beta_i} = r_t^e + \beta_i r_t^i$

NGU - A Family of Policies

Number of policies N = 32

(β_j, γ_j)

(β_0, γ_0)

(β_1, γ_1)

(β_2, γ_2)

$\vdots$

$(\beta_{31}, \gamma_{31})$

uniformly select ➡ j

current state x

value function network

RNN

$Q(x, a, j; \theta)$

NGU - A Family of Policies

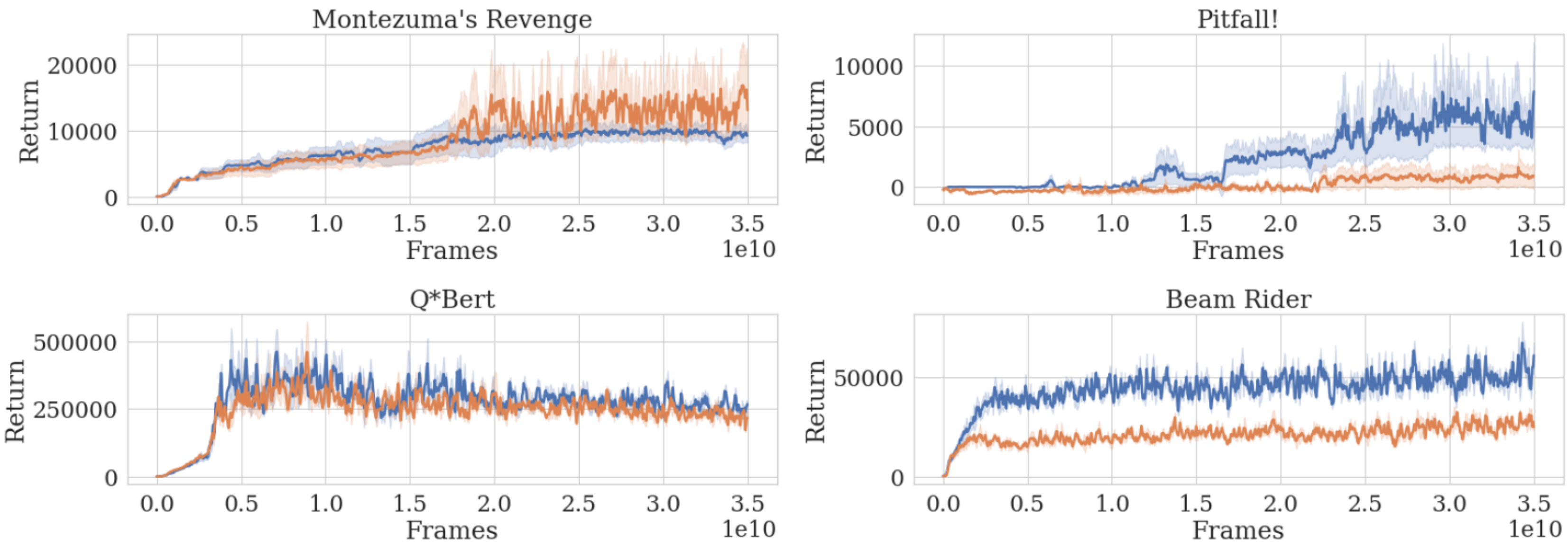


Figure 6: NGU(N=32) behavior for β_0 (blue) and β_{31} (orange).

https://sites.google.com/view/nguiclr2020

NGU - Experiments

Algorithm	Gravitar	MR	Pitfall!	PrivateEye	Solaris	Venture
Human	3.4k	4.8k	6.5k	69.6k	**12.3k**	1.2k
Best baseline	**15.7k**	**11.6k**	0.0	11k	5.5k	2.0k
RND	3.9k	10.1k	-3	8.7k	3.3k	1.9k
R2D2+RND	15.6k±0.6k	10.4k±1.2k	-0.5±0.3	19.5k±3.5k	4.3k±0.6k	**2.7k±0.0k**
R2D2(Retrace)	13.3k±0.6k	2.3k±0.4k	-3.5±1.2	32.5k±4.7k	6.0k±1.1k	2.0k±0.0k
NGU(N=1)-RND	12.4k±0.8k	3.0k±0.0k	**15.2k±9.4k**	40.6k±0.0k	5.7k±1.8k	46.4±37.9
NGU(N=1)	11.0k±0.7k	8.7k±1.2k	9.4k±2.2k	60.6k±16.3k	5.9k±1.6k	876.3±114.5
NGU(N=32)	14.1k±0.5k	10.4k±1.6k	8.4k±4.5k	**100.0k±0.4k**	4.9k±0.3k	1.7k±0.1k

NGU - Experiments

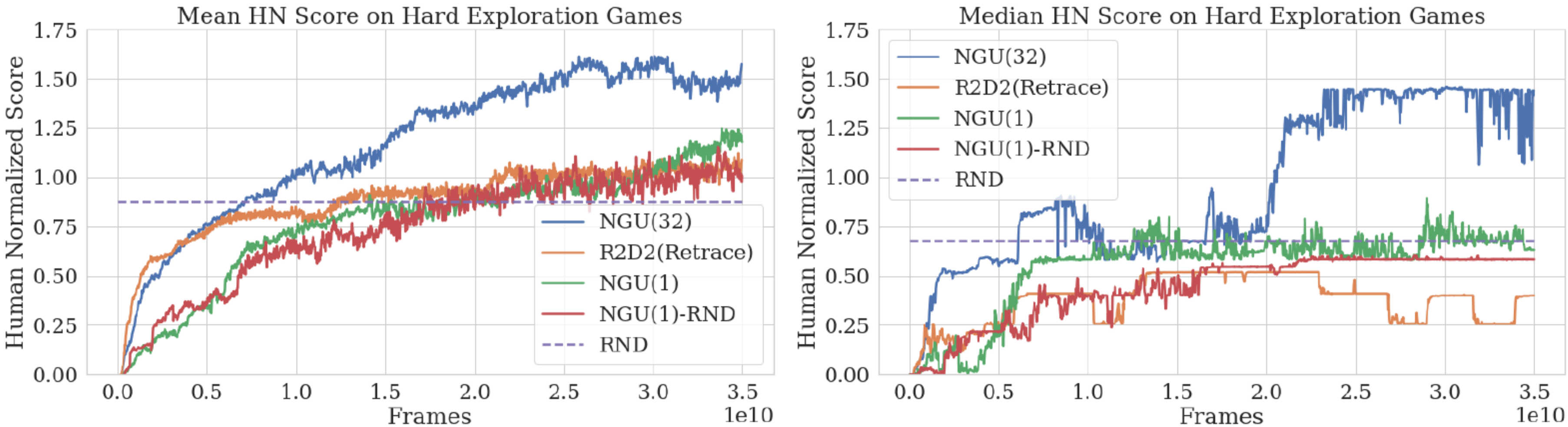


Figure 4: Human Normalized Scores on the 6 hard exploration games.

Advanced Exploration

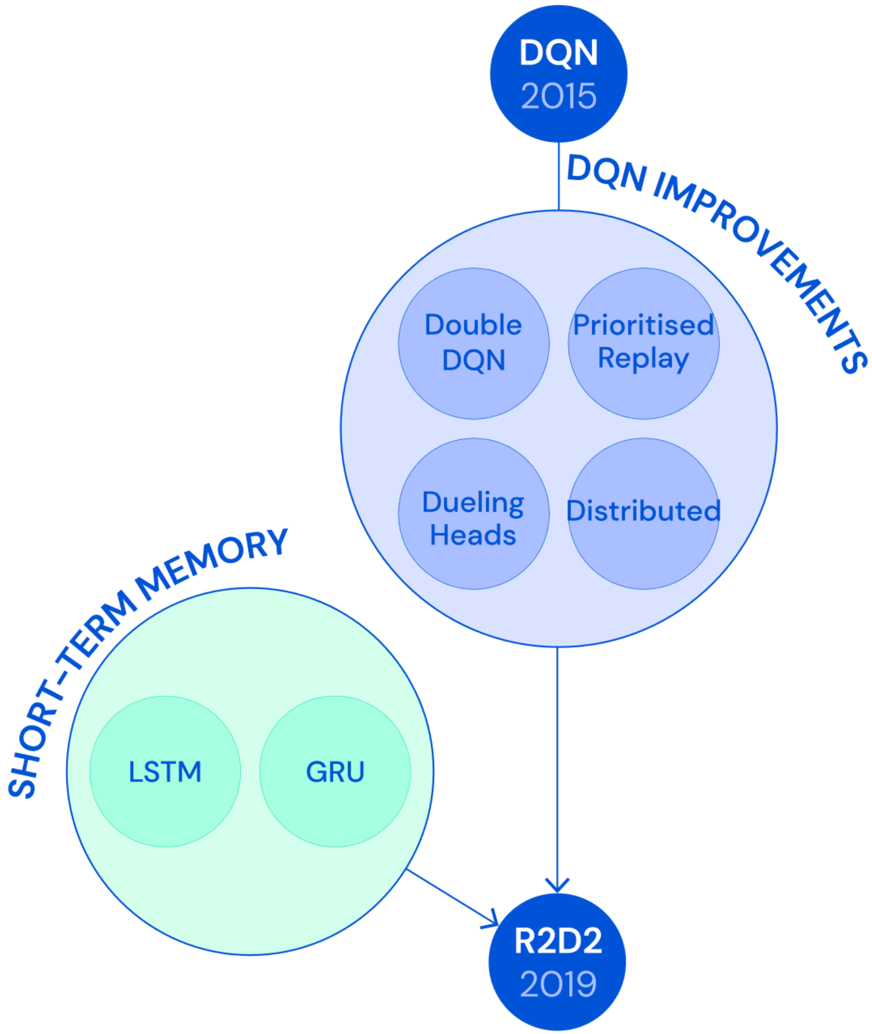
- Intrinsic Curiosity Module (ICM)
- Random Network Distillation (RND)
- Never Give Up (NGU)
- Agent57

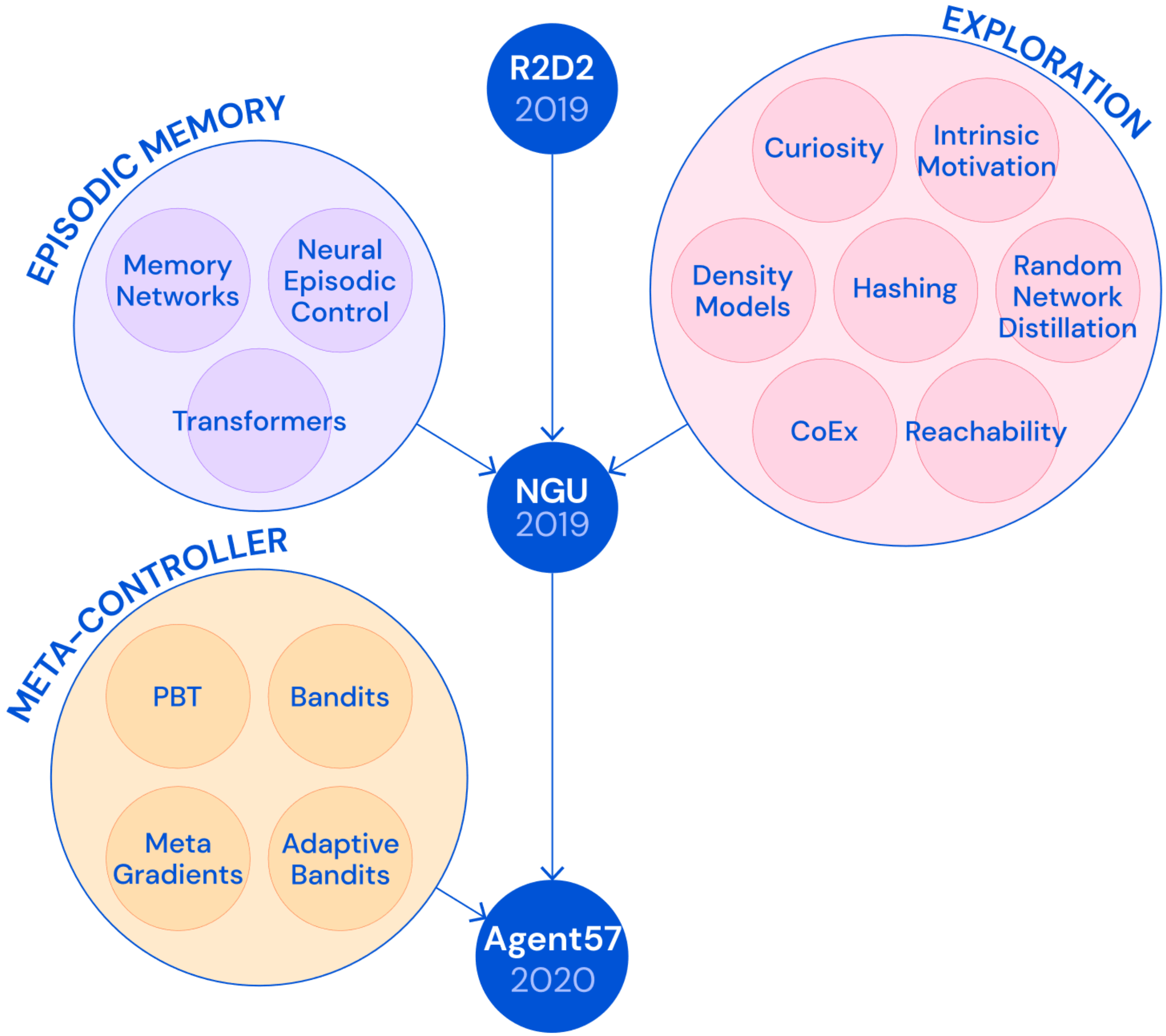
Agent57 - Introduction

- Achievement: the first deep RL agent that outperforms the standard human benchmark on all 57 Atari games
- Method: propose some improvements to Never Give Up (NGU)

Statisitics	Agent57	NGU	R2D2	MuZero
Number of games > human	57	51	52	51
Mean HNS	4766.25%	3421.80%	4622.09%	5661.84%
Median HNS	1933.49%	1359.78%	1935.86%	2381.51%
5th percentile of HNS	116.67%	64.10%	50.27%	0.03%

HNS: Human Normalized Scores


DQN
2015
DQN IMPROVEMENTS
Double DQN
Prioritised Replay
Dueling Heads
Distributed
SHORT-TERM MEMORY
LSTM
GRU
R2D2
2019


R2D2
2019
EPISODIC MEMORY
Memory Networks
Neural Episodic Control
Transformers
EXPLORATION
Curiosity
Intrinsic Motivation
Density Models
Hashing
Random Network Distillation
CoEx
Reachability
NGU
2019
META-CONTROLLER
PBT
Bandits
Meta Gradients
Adaptive Bandits
Agent57
2020

Agent57 - Introduction

- The development of general, domain-independent agents is a long-term goal of AI
- Atari 57 is an achievable stepping stone testing general competency
- All best agents (MuZero, R2D2, Rainbow) fail on the same subset of games with these important issues:
  - Exploration
  - Long-term credit assignment

Agent57 - Introduction

- Exploration
  - Efficient exploration can be critical to effective learning
  - E.g., Montezuma’s Revenge and Pitfall!
- Long-term credit assignment
  - Which decisions are most deserving of credit for the outcomes?
  - E.g., Solaris and Skiing

Montezuma’s Revenge

Pitfall!

Solaris

Skiing

Agent57 - Introduction

- Agent57 builds on top of the Never Give Up agent
- Never Give Up is not the most general agent
  - NGU has shown significant recent promise in improving performance on hard exploration games
  - NGU is unable to cope with long-term credit assignment problems

Problems of NGU

- All policies are trained equally, regardless of their contribution to the learning progress

  → Use a meta-controller to adaptively select the policy to use

Agent57 - Improvements to NGU

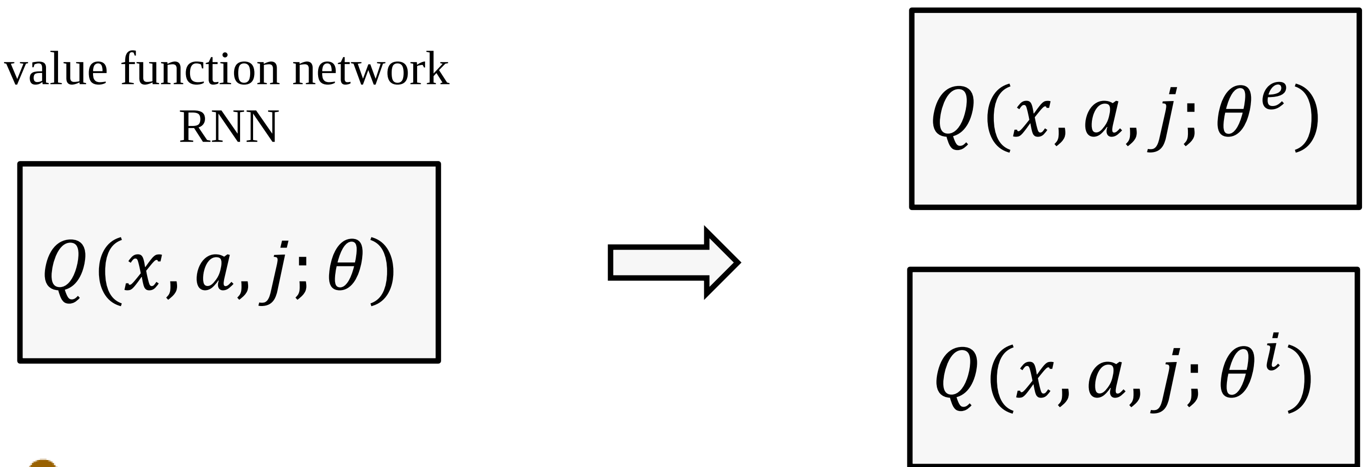
- Separate networks for intrinsic and extrinsic values
  - Significantly increase the training stability
- Use a **meta-controller** to select which of the policy to prioritize
  - Allow the agent to control the exploration/exploitation trade-off
- Use a longer backprop through time window
  - Lead to superior long-term credit assignment

Agent57 - Separated Networks

- The architectural improvement consists in splitting the state-action value function into extrinsic and intrinsic components :

$$Q(x,a,j;\theta) = Q(x,a,j;\theta^e) + \beta_j Q(x,a,j;\theta^i)$$

- The two components optimized separately in the learner with rewards r^e and r^i respectively.



Agent57 - Meta Controller

- Use Non-Stationary Multi-armed Bandit Algorithm to choose policy pair (β_j, γ_j).

Number of policies N = 32

(β_j, γ_j)

value function network
RNN

(β_0, γ_0)
(β_1, γ_1)
(β_2, γ_2)
$\vdots$
$(\beta_{31}, \gamma_{31})$

current state x

~~uniformly select~~ ➡ j

Non-Stationary UCB

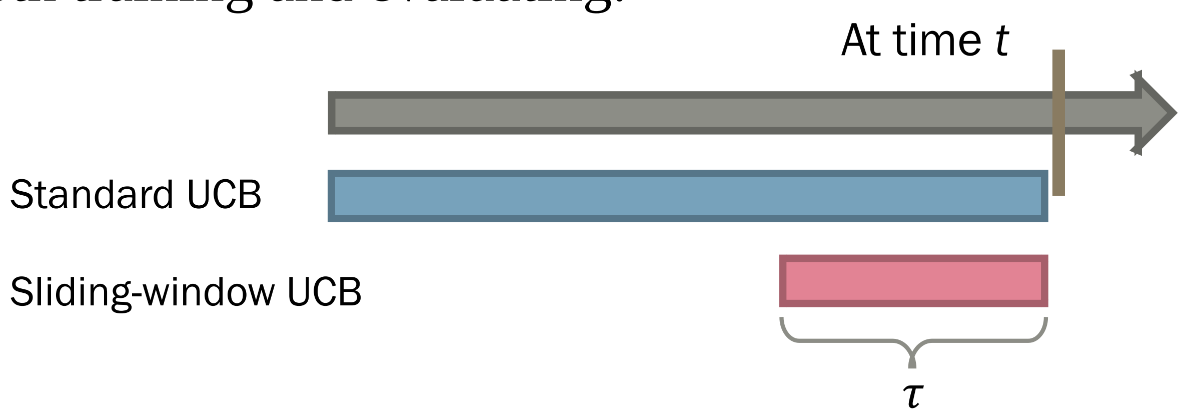
$Q(x, a, j; \cancel{\theta})$ θ^i

Agent57 - Meta Controller

- Use Non-Stationary Multi-armed Bandit Algorithm to choose policy pair (β_j, γ_j).
  - Use sliding-window UCB
  - Only use τ last plays

Hyperparameter	**Value**
Bandit window size τ	$\{160, 224, 320, 640\}$

- Use in both training and evaluating.



Agent57 - Meta Controller

- Use undiscounted extrinsic episodic returns as a reward signal.

0.5 1

$$\begin{cases} \forall 0 \le k \le N-1, & A_k = k \\ \forall N \le k \le K-1 \text{ and } U_k \ge \epsilon_{\text{UCB}}, & A_k = \arg\max_{0 \le a \le N-1} \hat{\mu}_{k-1}(a,\tau) + \beta \sqrt{\frac{1}{N_{k-1}(a,\tau)}} \\ \forall N \le k \le K-1 \text{ and } U_k < \epsilon_{\text{UCB}}, & A_k = Y_k \end{cases}$$

k: episode number

N: # of possible arms

A_k: arm at time k

τ: sliding window length

U_k: drawn uniformly from [0,1]

Y_k: a random action drawn uniformly from $\{0, \ldots, N-1\}$

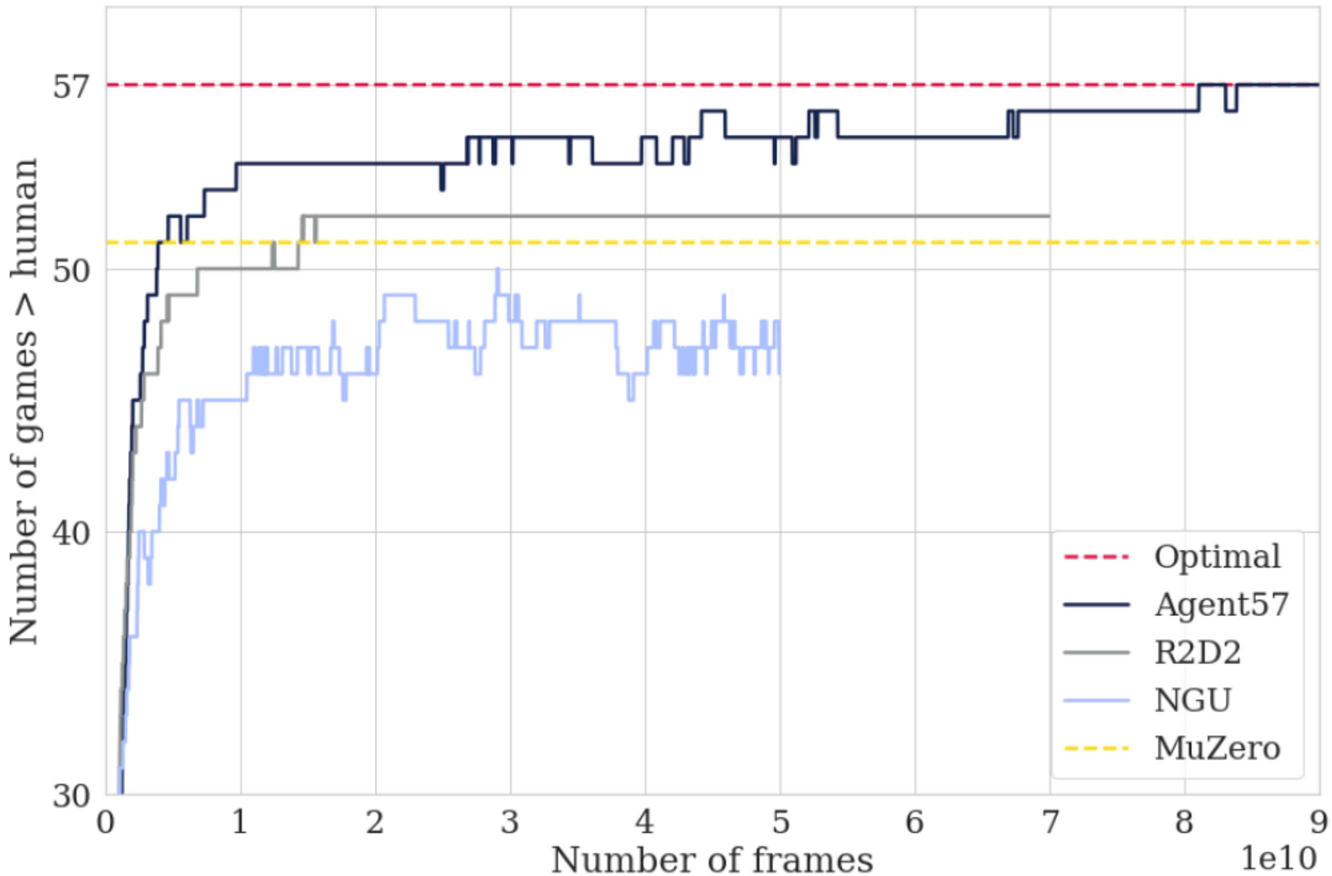
$\hat{\mu}_k(a,\tau)$: the empirical mean of an arm a after k steps for a window of length τ

$N_k(a,\tau)$: number of times an arm a has been played after k steps for a window of length τ

Agent57 - Longer Backprop

- The authors compare using backprop through time window sizes of 80 (default in R2D2) versus 160.
- Using a longer backprop through time window results in better overall stability and slightly higher final score.

Agent57 - Experiments



Agent57 - Experiments

Table 1. Number of games above human, mean capped, mean and median human normalized scores for the 57 Atari games.

Statistics	Agent57	R2D2 (bandit)	NGU	R2D2 (Retrace)	R2D2	MuZero
Capped mean	**100.00**	96.93	95.07	94.20	94.33	89.92
Number of games > human	**57**	54	51	52	52	51
Mean	4766.25	5461.66	3421.80	3518.36	4622.09	**5661.84**
Median	1933.49	2357.92	1359.78	1457.63	1935.86	**2381.51**
40th Percentile	1091.07	**1298.80**	610.44	817.77	1176.05	1172.90
30th Percentile	614.65	**648.17**	267.10	420.67	529.23	503.05
20th Percentile	**324.78**	303.61	226.43	267.25	215.31	171.39
10th Percentile	**184.35**	116.82	107.78	116.03	115.33	75.74
5th Percentile	**116.67**	93.25	64.10	48.32	50.27	0.03