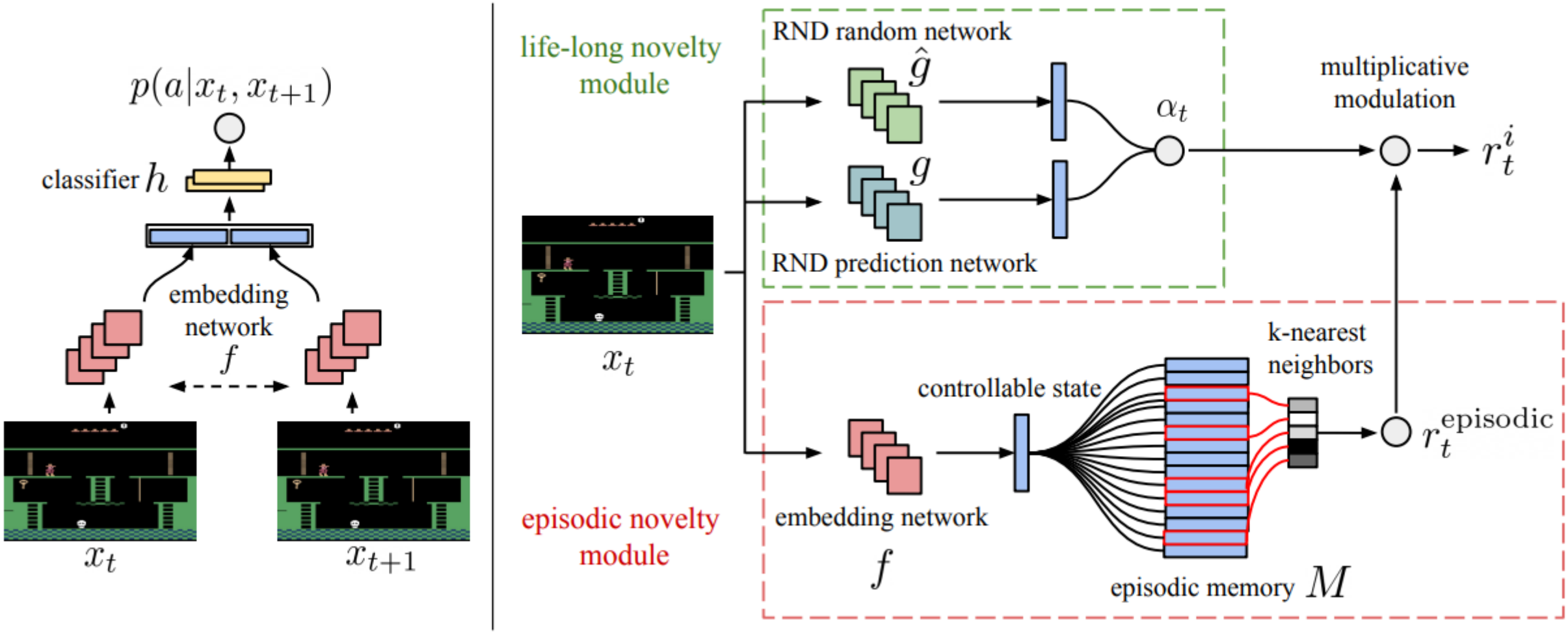
NEVER GIVE UP: LEARNING DIRECTED EXPLORATION STRATEGIES

Adrià Puigdomènech Badia* **Pablo Sprechmann*** **Alex Vitvitskyi** **Daniel Guo**

Bilal Piot **Steven Kapturowski** **Olivier Tieleman** **Martín Arjovsky**

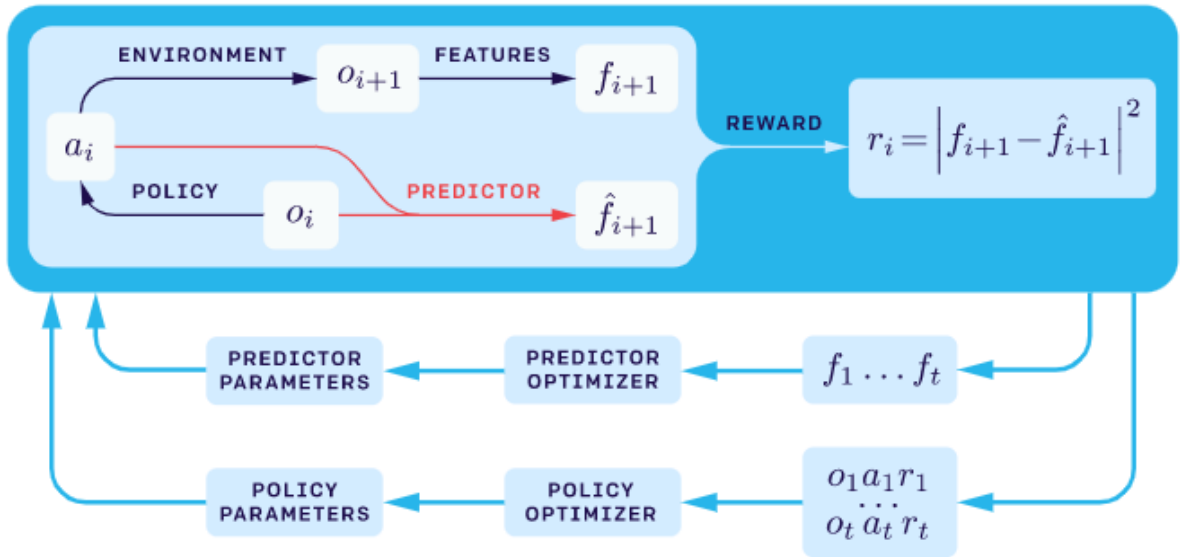
Alexander Pritzel **Andew Bolt** **Charles Blundell**

DeepMind {adriap, psprechmann, avlife, danielguo,
piot, skapturowski, tieleman,
apritzel, abolt, cblundell}@google.com

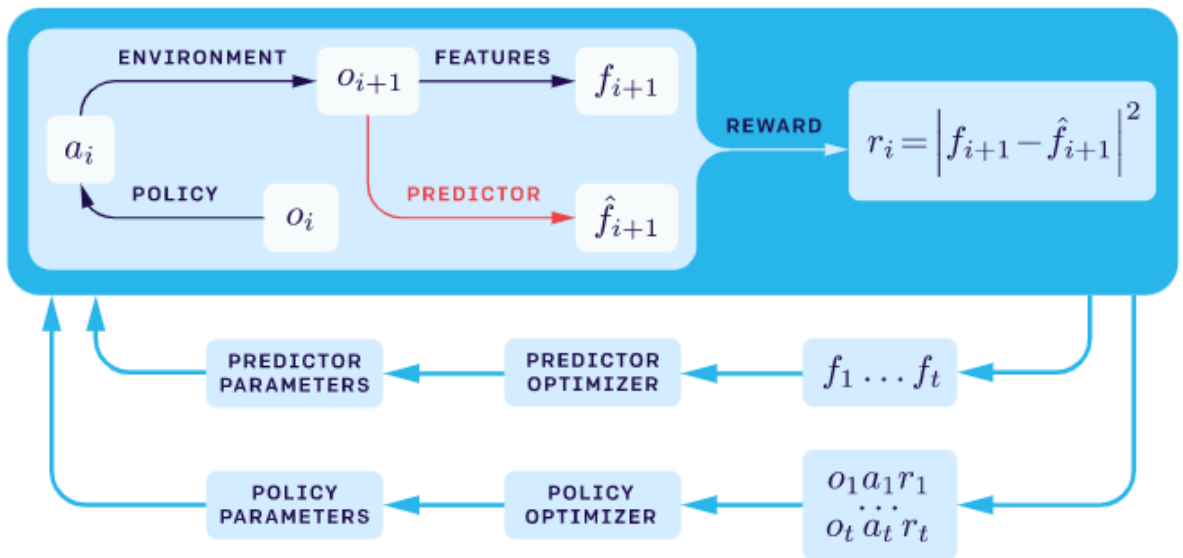

$p(a|x_t, x_{t+1})$
classifier h
embedding network f
x_t
x_{t+1}
life-long novelty module
RND random network
$\hat{g}$
g
α_t
multiplicative modulation
r_t^i
RND prediction network
x_t
episodic novelty module
k-nearest neighbors
controllable state
r_t^{episodic}
embedding network f
episodic memory M

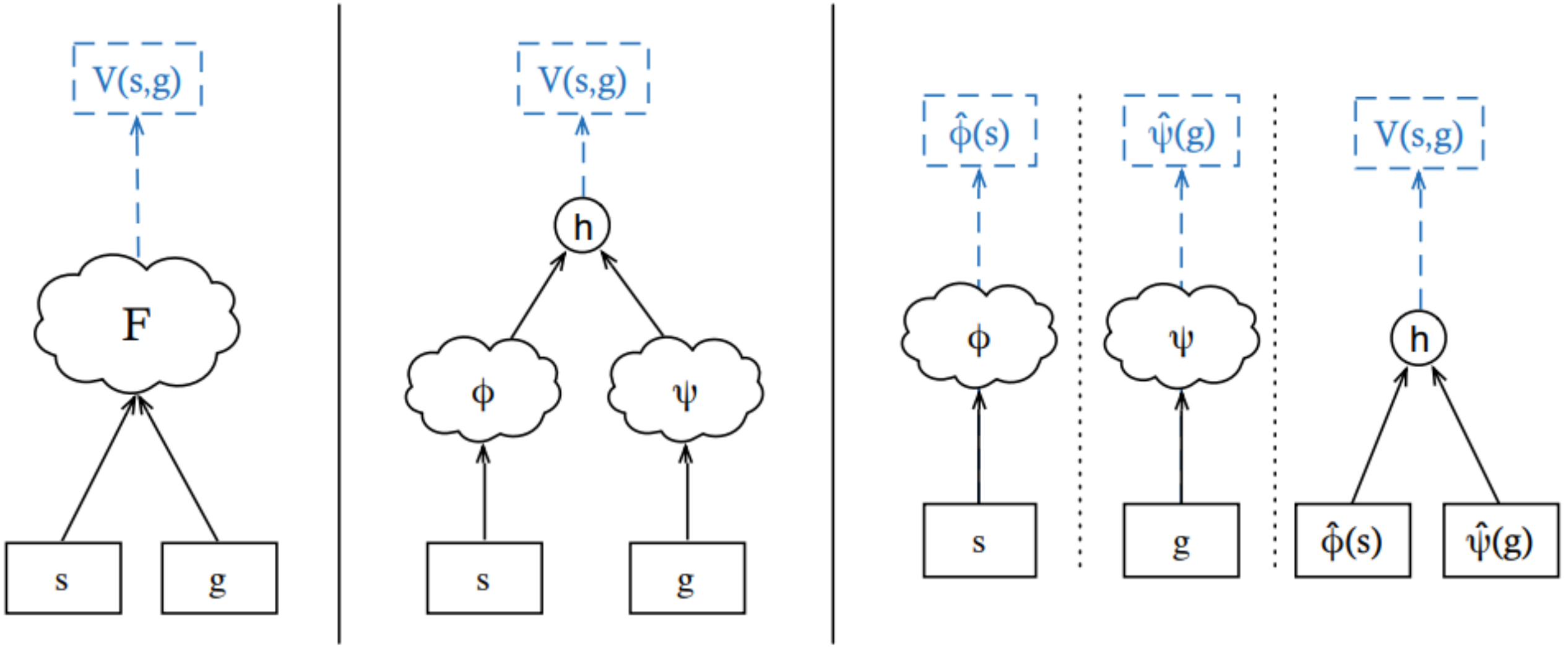
Comparison of Next-State Prediction with RND

Next-State Prediction



Random Network Distillation




V(s,g)
F
s
g
V(s,g)
h
ϕ
ψ
s
g
ϕ̂(s)
ψ̂(g)
V(s,g)
ϕ
ψ
h
s
g
ϕ̂(s)
ψ̂(g)

Introduction

충분한 exploration 이 확보되지 않은 채로 greedy정책을 사용하여 suboptimal policy로 수렴되고 데이터를 더 이상 모으지 않는 문제

epsilon-greedy나 Boltzmann exploration은 결국 tabular에서 optimal policy를 학습하지만, 매우 비효율적이고 상태공간에 따라 필요한 step이 기하급수적으로 늘어난다.

하지만 sparse reward setting에서는 temporally-extended exploration(=deep exploration)이 very few rewarding states를 찾는데 중요하기 때문에 결국 학습에 실패할 수 있다.

Introduction

최근(논문 작성일자 2020년)에는 탐험을 유도하기 위해 agent에게 intrinsic reward를 주는 것을 제안한다.

- 이 intrinsic reward는 기존에 방문한 상태들과 얼마나 다른 지를 중요도로 수량화한 것에 비례한다.
- 방문횟수가 늘어나며 익숙해지면 exploration bonus는 줄어들고 extrinsic reward를 통한 학습만 하게 된다.

이는 very hard exploration tasks에서 매우 좋은 성과를 달성했지만, 이런 알고리즘들은 기초적인 한계에 직면했다.

- 상태의 새로움이 사라진 후, 에이전트는 그것을 다시 방문하도록 encouraged 되지 않는다. 그것이 허용하는 downstream learning 기회에도 불구하고 말이다.

Introduction

주요 아이디어는 동일한 네트워크에서 파생된 별도의 exploration 및 exploitation 정책을 공동으로 학습하는 것이다.

exploitation 정책은 외부 보상(가까이에 있는 일 해결)을 최대화하는 데 집중할 수 있는 반면

탐색 정책은 궁극적으로 undirected policy를 줄이지 않고 exploration을 유지할 수 있다.

Introduction

controllable state : Environment의 Background 등 같은(Noise) 것에 영향을 받지 않는 Feature State

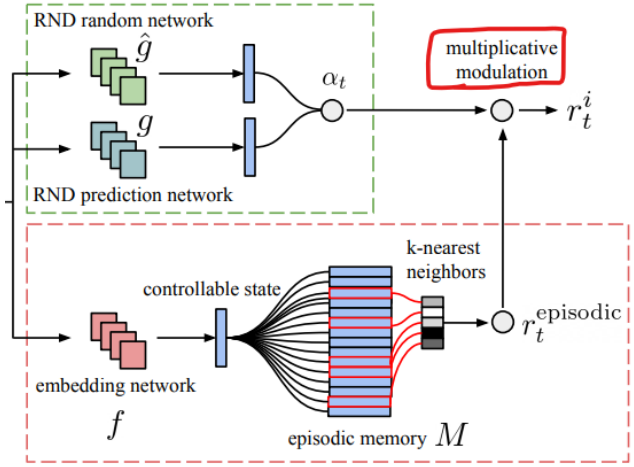
우리는 episode동안 반복적으로 환경의 모든 controllable states를 방문하도록 encourage 하기 위해 per-episode 와 life-long novelty를 결합하는 intrinsic reward를 제안한다.

Episodic novelty 는 같은 episode가 아닌 several episode에서 완전히 탐색되지 않았지만 익숙한 상태들에 대해 다시 방문하도록 encourage한다.

Life-long novelty는 많은 에피소드에 걸쳐 점진적으로 더 친숙해지는 상태를 점진적으로 하향조정한다.

Introduction

$$r_t^i = r_t^{\text{episodic}} \cdot \min\left\{\max\left\{\alpha_t, 1\right\}, L\right\}$$



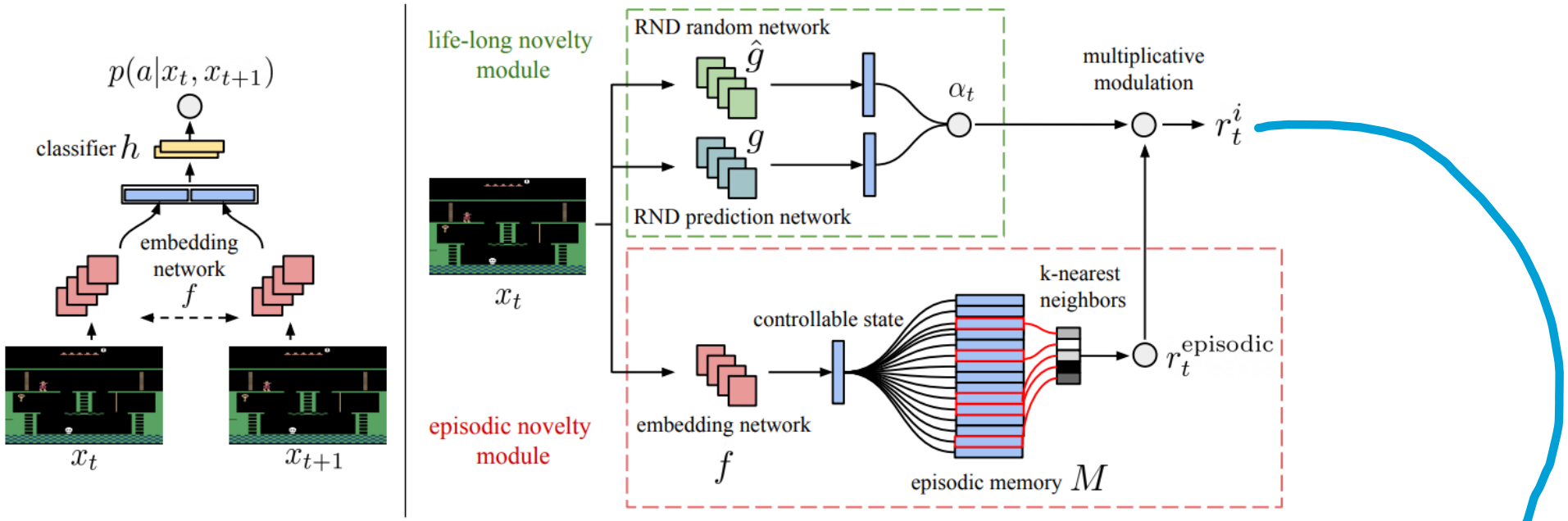
Episodic novelty는 최근에 방문된 상태들로 채워진 episodic memory를 사용하고 이전에 저장된 state들과의 유사성으로 정의된다.

Life-long novelty는 episodic similiarity signal을 곱셈으로 modulate하고 RND에 의해 유도된다.
Episodic novelty와는 반대로 life-long novelty는 GD optimisation에 의존한다.

이 결합된 novelty 개념은 복잡하고 고차원이어서 다시 방문되지 않는 상태공간을 일반화할 수 있고 episode마다 한 결같은 exploration을 유지할 수 있다.

+사전 정보나 특별한 지도가 없었음을 얘기한다.

THE NEVER-GIVE-UP INTRINSIC REWARD



Deep RL은 r_t를 기반으로 학습되고

성능은 r_t^e를 기반으로 측정된다.

extrinsic intrinsic

$$r_t = r_t^e + \beta r_t^i$$

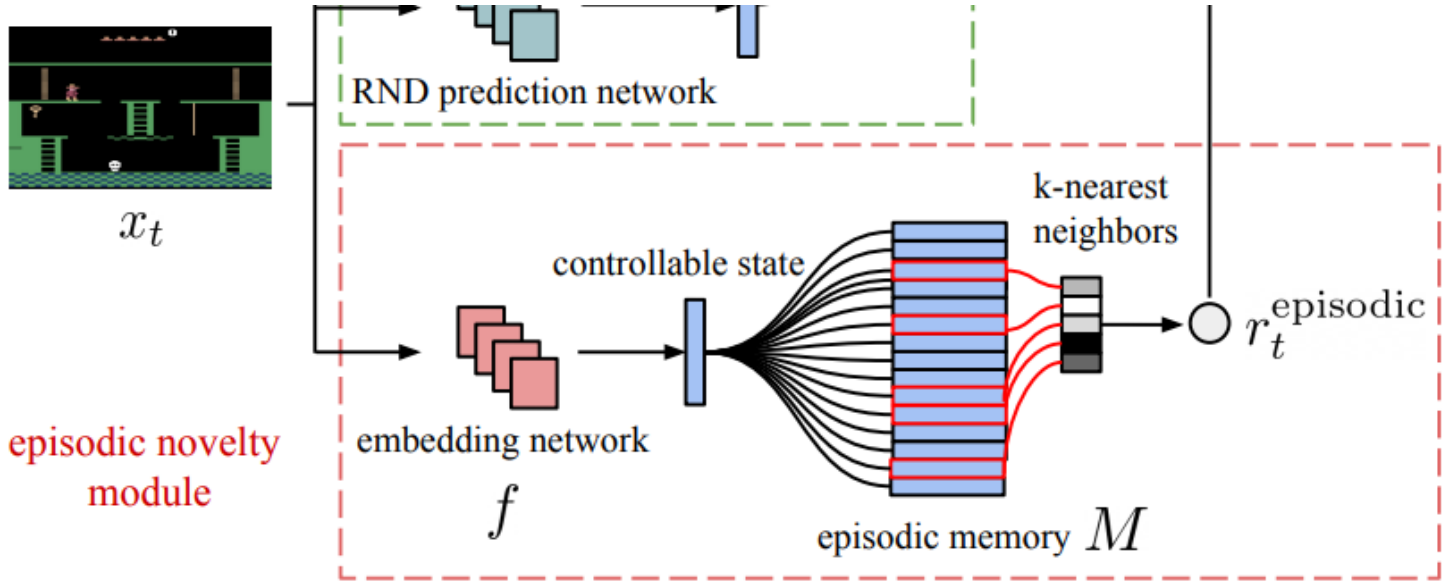
THE NEVER-GIVE-UP INTRINSIC REWARD

r_t^i

intrinsic reward는 다음 세가지 속성을 만족한다.

1. 같은 episode에서 같은 state를 다시 방문하는 것을 rapidly discourage한다.
2. episode간(across episode)에 많이 방문했던 상태를 slowly discourage한다.
3. the notion of state ignores aspects of an environment that are not influenced by an agent's actions.

   (상태라는 개념은 에이전트의 행동에 영향을 받지 않는 환경의 측면은 무시한다.)



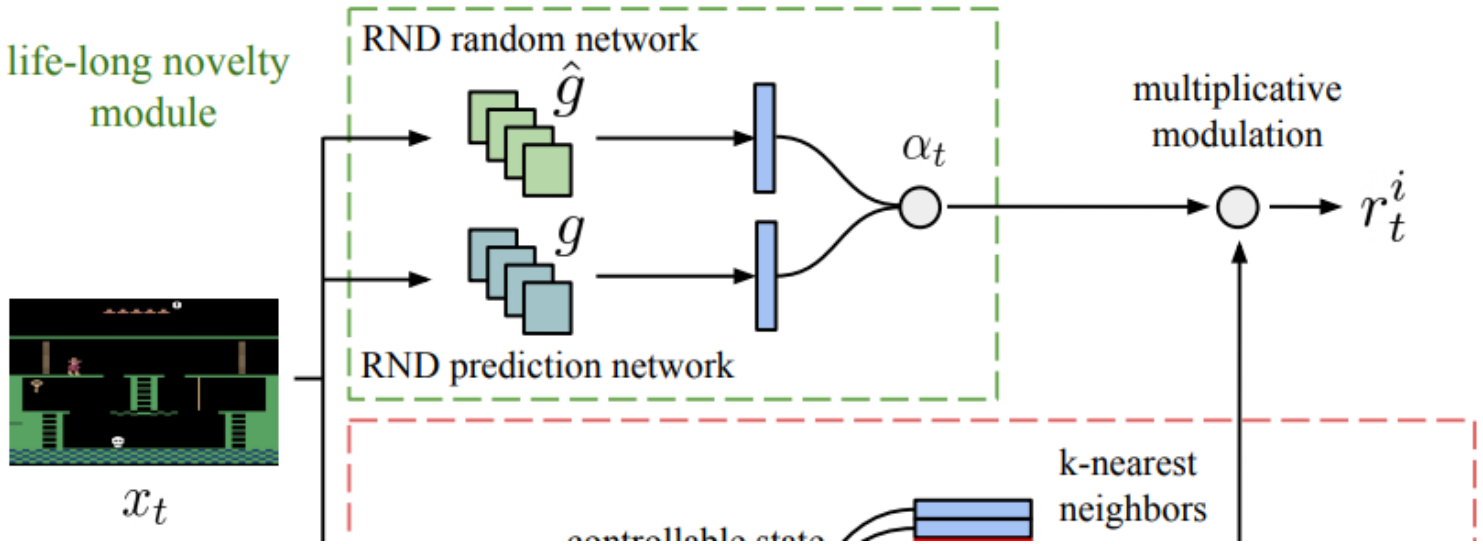
Our episodic novelty uses an episodic memory filled with all previously visited states, encoded using the self-supervised objective of Pathak et al. (2017) to avoid uncontrollable parts of the state space.

Curiosity-driven Exploration by Self-supervised Prediction

큰 차이는 큰 episodic intrinsic reward를 만든다.

Episodic intrinsic reward $r_t^{episodic}$ 는 한 에피소드에서 최대한 다른 상태를 방문하도록 촉진한다.

(다른 episodic와 독립적이다)



Life-long (or inter-episodic) novelty module은 에피소드간(across episode)에 탐험의 양을 statefully control하는 신호를 제공한다.

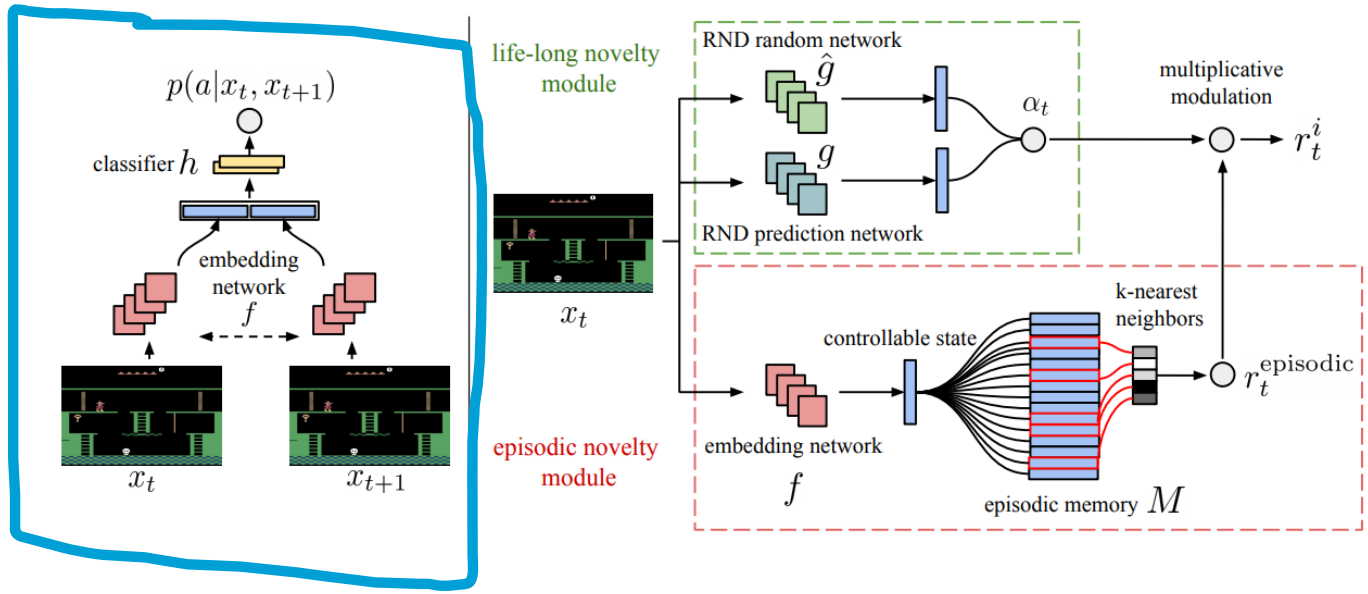
where L is a chosen maximum reward scaling (we fix $L = 5$ for all our experiments).

$$r_t^i = r_t^{\text{episodic}} \cdot \min\left\{\max\left\{\alpha_t, 1\right\}, L\right\}$$

이 modulation(alpha_t)은 시간이 지남에 따라 사라지고 modulation되지 않은 보상을 사용하는 방법이 줄어듭니다.

Note that this modulation will vanish over time, reducing our method to using the non-modulated reward.

Embedding network: $f : \mathcal{O} \rightarrow \mathbb{R}^p$



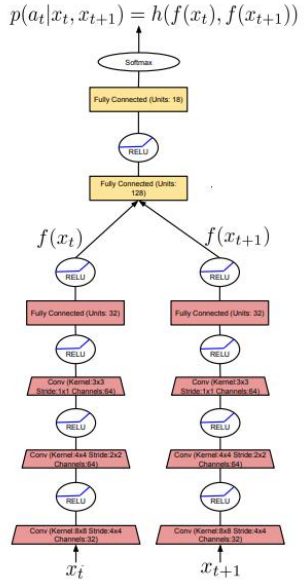


Figure 15: Embedding Network Architecture.

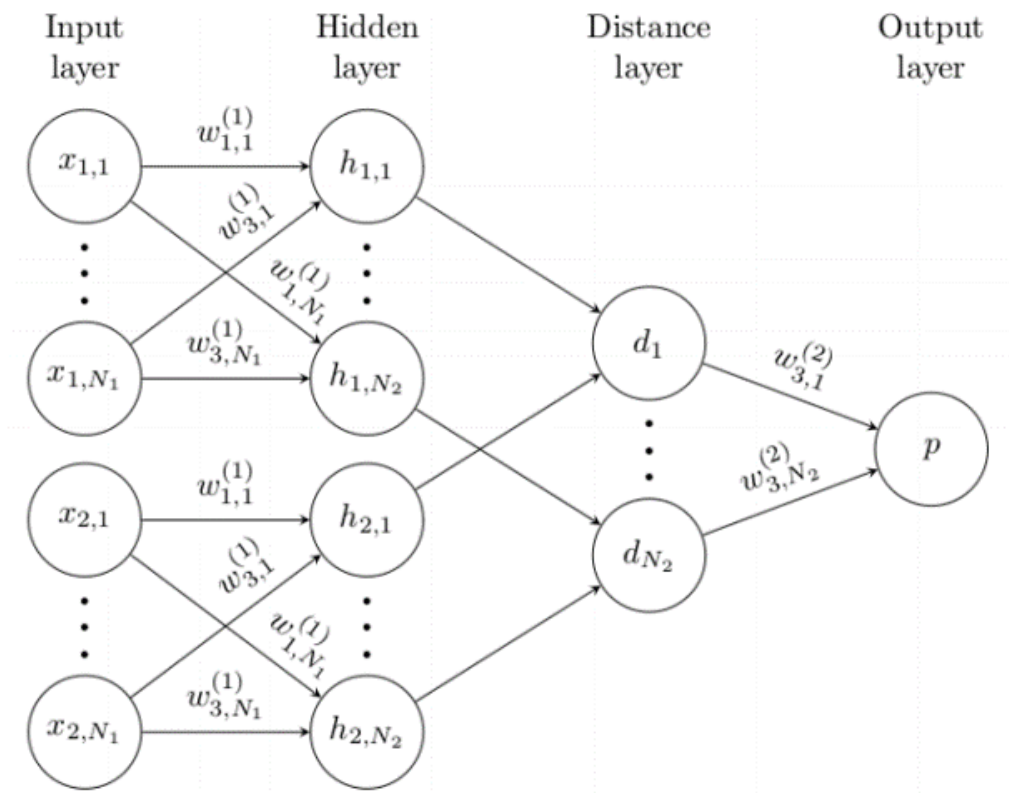
Current observation -> p-dimensional vector corresponding to its controllable state

예를 들어 차와 사람이 많은 도시 한가운데 서있다면 아무 행동을 하지 않아도 정말 많은 state를 탐험하게 된다(많은 intrinsic reward를 모은다) . 하지만 이것은 탐험으로써 의미있는 형태가 아니다.

이런 의미없는 탐험을 피하기 위해 우리는 연속된 두 observation이 주어지고 Siamese network f를 훈련하여 한 observation에서 다음 행동을 예측하도록 한다.

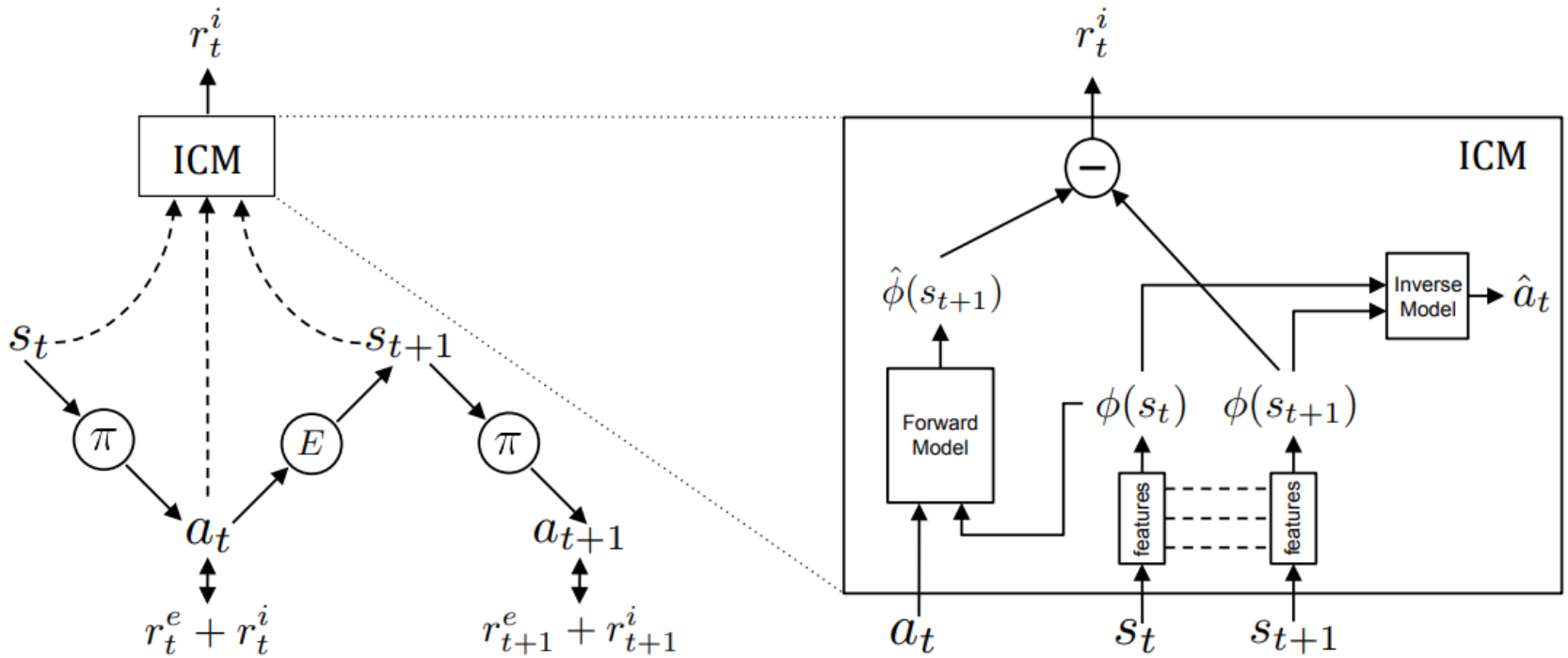
Siamese Neural Networks for One-shot Image Recognition

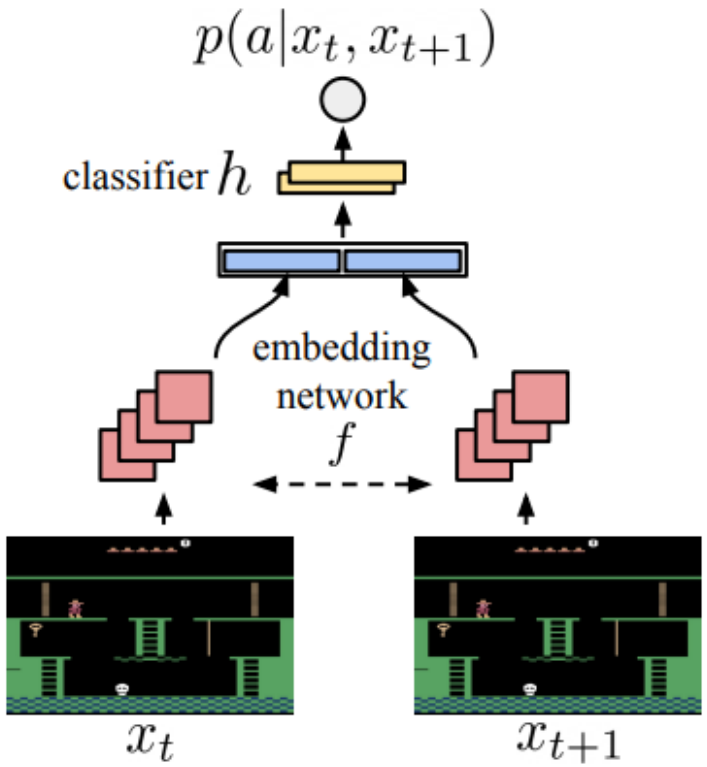
Gregory Koch GKOCH@CS.TORONTO.EDU
Richard Zemel ZEMEL@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.



Curiosity-driven Exploration by Self-supervised Prediction

Deepak Pathak [1] **Pulkit Agrawal** [1] **Alexei A. Efros** [1] **Trevor Darrell** [1]





직관적으로 에이전트의 행동에 의해 영향받지 않은 환경의 변동성은 예측에 있어서 유용하지 않다.

$\{x_t, a_t, x_{t+1}\}$ 을 가지고 조건부확률 $p(a|x_t, x_{t+1}) = h\big(f(x_t), f(x_{t+1})\big)$을 parameterise한다.

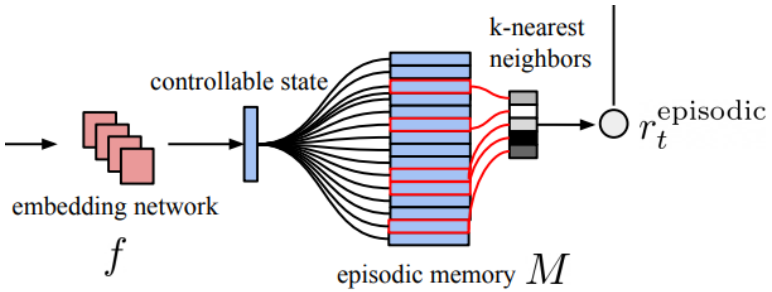
where h is a one hidden layer MLP followed by a softmax

h와 f는 Maximum likelihood을 통해 훈련된다.

이 구조는 Siamese network 위에 레이어 하나가 있는 것으로 생각될 수 있다.

State와 Next State를 Embedding Network를 통해 Feature State를 추출한다. 추출한 Feature state를 MLP에 넣어 Action을 예측한다. Action을 잘 예측할 수록 Embedding Network과 Background에 강건한(Noise)등 Feature를 추출할 수 있게 된다.

Episodic memory and intrinsic reward:



Episodic memory는 최근 episode에서 방문된 controllable states를 포함한다. $\{f(x_0), f(x_1), \ldots, f(x_{t-1})\}$

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

abstract state $f(x_t)$에 대한 방문횟수

우리는 이 $n(f(x_t))$를 메모리 M의 요소들에 대한 kernel function K를 이용한 유사도의 합으로 근사한다.

실제로는 메모리 M에서 $f(x_t)$의 k-nearest neighbor를 계산한다, $N_k = \{f_i\}_{i=1}^{k}$

K가 Dirac delta function일 때 approximation은 정확해지지만 결국 큰 상태공간에서 탐험의 일반화는 제공하지 못한다.

Episodic memory and intrinsic reward:

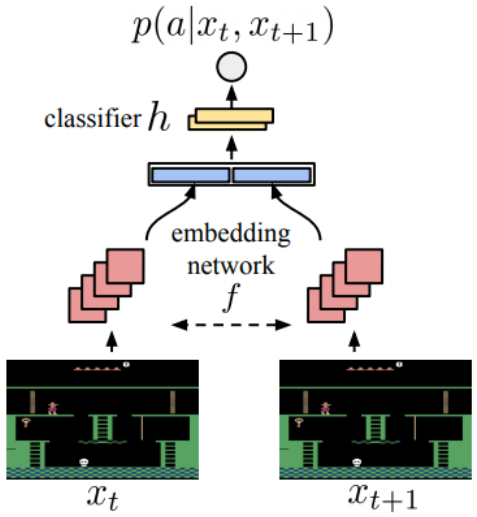
$$K(x, y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$$

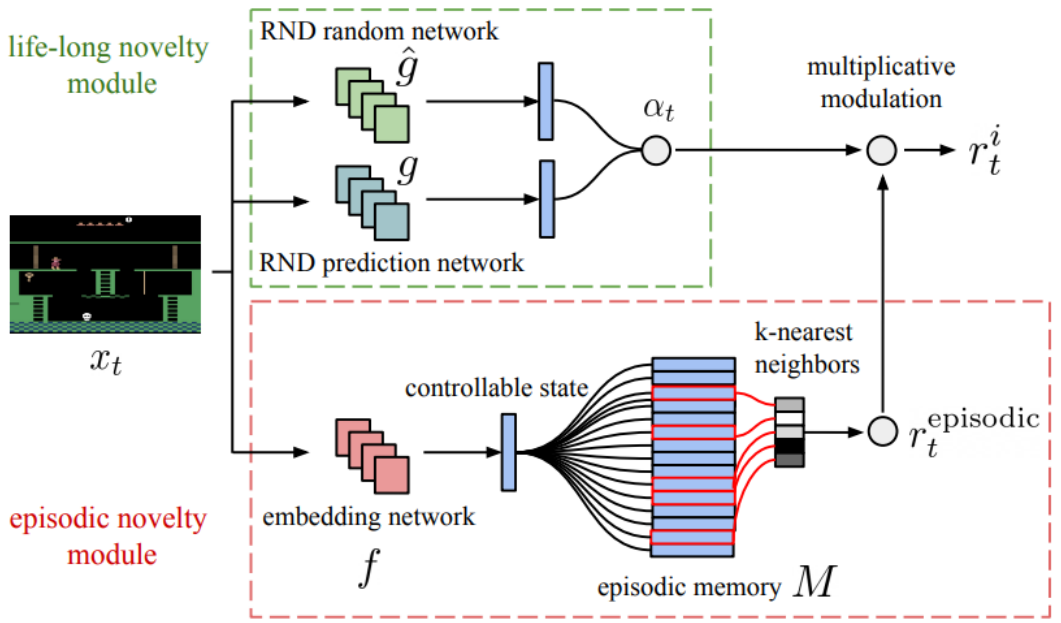
we use the inverse kernel for K

d is the Euclidean distance

d_m^2 is a running average of the squared Euclidean distance of the k-th neighbor

다른 tasks는 학습된 임베딩 사이의 일반적인 거리가 다를 수 있으므로 이 running average는 해결중인 작업에 대해 커널을 보다 robust하게 만드는데 사용된다. (Normalize)

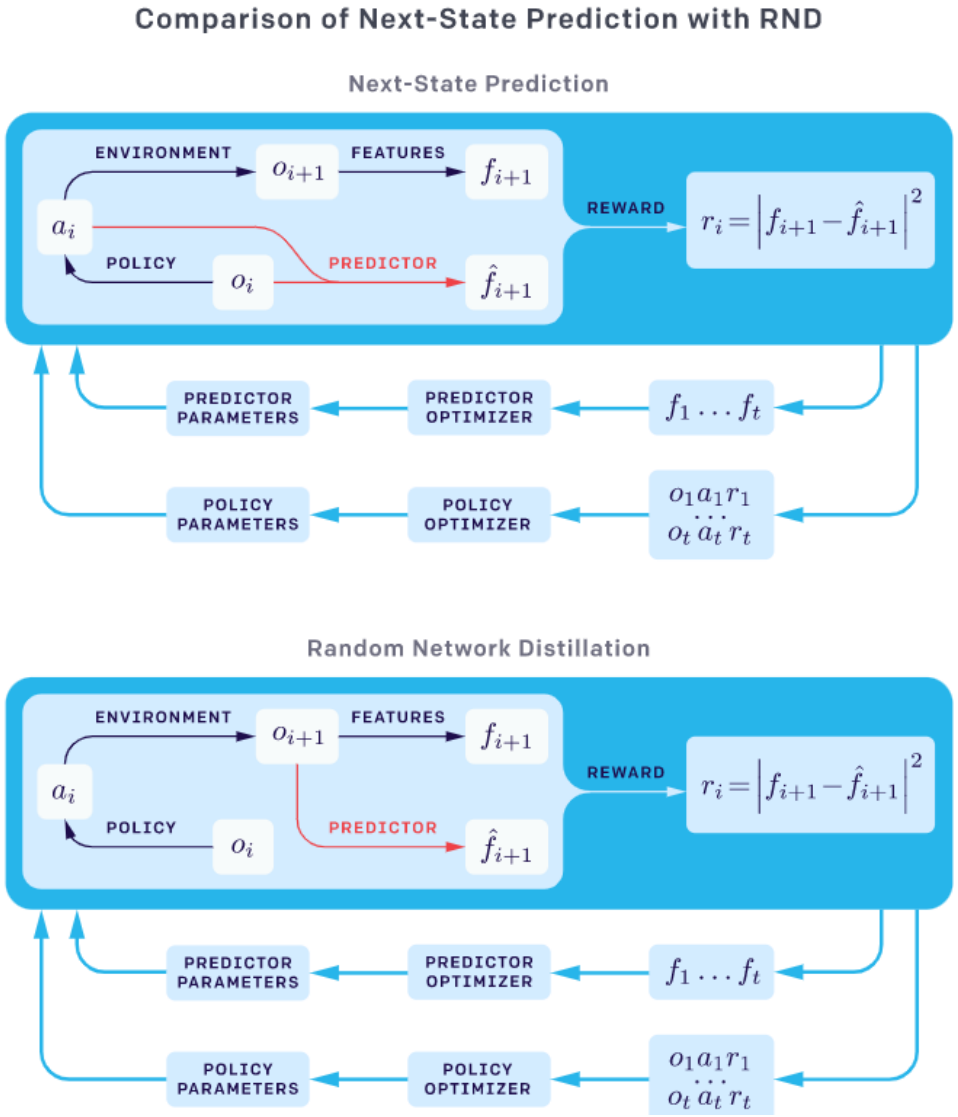




$$r_t^{\text{episodic}} = \frac{1}{\sqrt{n(f(x_t))}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i)} + c}$$

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

Integrating life-long curiosity:



$\alpha_t = 1 + \frac{err(x_t) - \mu_e}{\sigma_e}$, where σ_e and μ_e are running standard deviation and mean for $\text{err}(x_t)$

THE NEVER-GIVE-UP AGENT

exploration의 수단으로 intrinsic reward를 사용하면 reward 가 action, state에 따라 예측 불가능하게 바뀌면서 MDP가 아니라 POMDP가 된다.

이럴 경우 MDP를 풀기가 훨씬 힘들어지는데 이 복잡도를 피하기 위해 두가지 접근을 이용한다.

1. intrinsic reward는 agent에게 직접 전달된다.
2. 에이전트는 에피소드의 모든 입력(s, a, r) history를 요약하는 내부 상태표현을 유지한다.

-> recurrent state, experience replay, off-policy value learning and distributed training을 포함하는 R2D2를 사용한다.

THE NEVER-GIVE-UP AGENT

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

이전의 많은 intrinsic reward와 달리, NGU의 intrinsic reward는 시간이 흐르면서 사라지지 않고 그것에 의해 정책은 언제나 부분적으로 유도될 것이다. (episode는 계속 리필되기 때문에)

게다가 제안된 탐험적 행동(intrinsic reward에 의한)은 value function에 직접 인코딩되어 있고 따라서 쉽게 꺼질 수 없다.

Furthermore, the proposed exploratory behaviour is directly encoded in the value function and as such it cannot be easily turned off.

이 문제를 해결하기 위해 (당장의 task의) extrinsic reward에 의해 유도되는 명시적 exploitative policy를 공동으로 학습하는 것을 제안한다.

THE NEVER-GIVE-UP AGENT

Proposed architecture:

$Q(x, a, \beta_i)$

$r_t^{\beta_i} = r_t^e + \beta_i r_t^i$ 형태의 augmented reward 모임 각각에 대해 동시에 근사하기 위해 UVFA를 사용할 것을 제안한다.

N개의 discrete number 들을 사용하며 $\{\beta_i\}_{i=0}^{N-1}$ $\beta_0 = 0$이고 $\beta_{N-1} = \beta$ (β=선택된 최댓값) 이다.

이 방법으로 하면 $\beta = 0$ 일 때 $Q(x, a, \beta_i)$ 에 대해 greedily하게 행동하게 된다.

원칙적으로 $\beta_0 = 0, \beta_1 > 0$ 단 두 정책만가지는 구조를 생각할수도 있다.

많은 수의 정책을 훈련시키는 것은 exploitative and exploratory policies가 상당히 다른 행동 관점을 가질수 있다는 사실에서 기인한다. 부드럽게 바뀌는 많은 정책들을 가지는 것은 학습을 더 효율적으로 만든다.

THE NEVER-GIVE-UP AGENT

RL Loss functions:

training loss -> transformed Retrace double Q-learning loss.

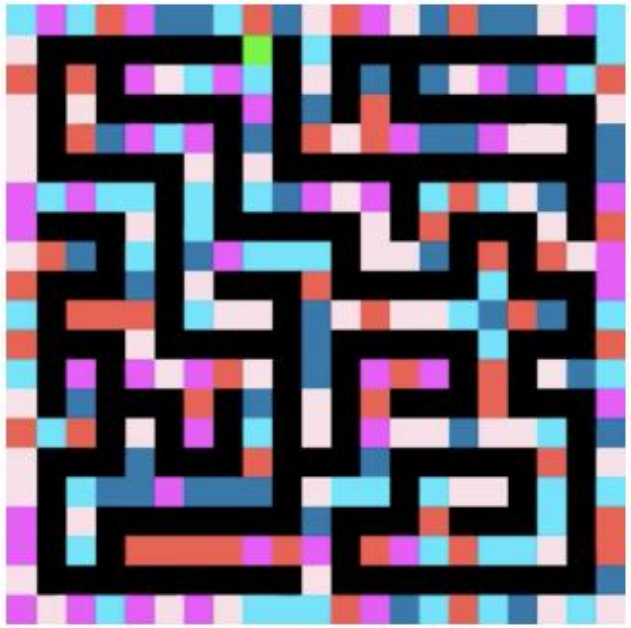
$\gamma_0 = 0.997$, and $\gamma_{N-1} = 0.99$

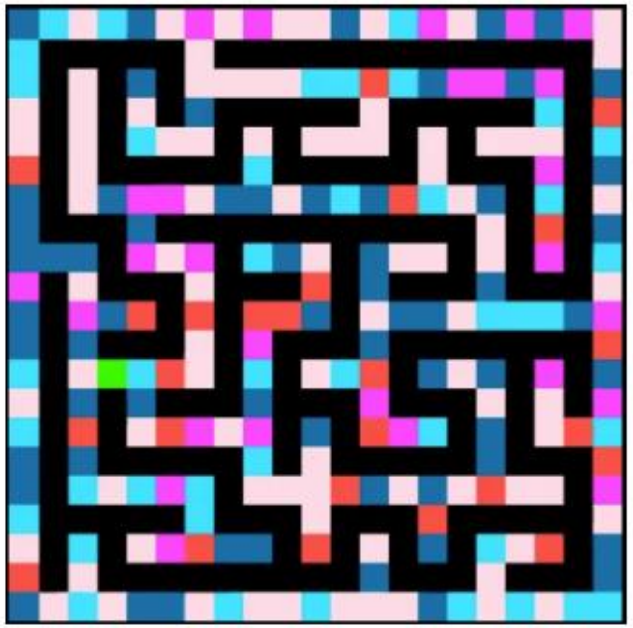
β_0-> highest discount factor $\gamma_0 = \gamma_{\max}$: to be as close as possible from optimizing the undiscounted return

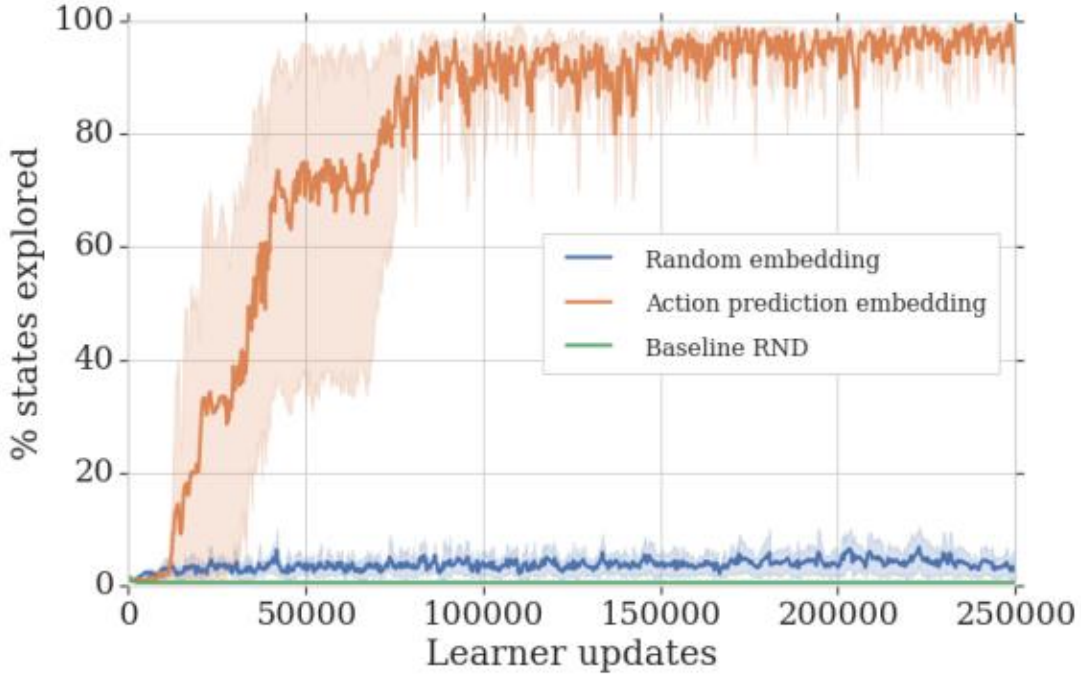
β_{N-1} -> smallest discount factor $\gamma_0 = \gamma_{\min}$: exploratory policies -> intrinsic reward is dense

->range of value is small

Distributed training:







https://youtu.be/9HTY4ruPrHw

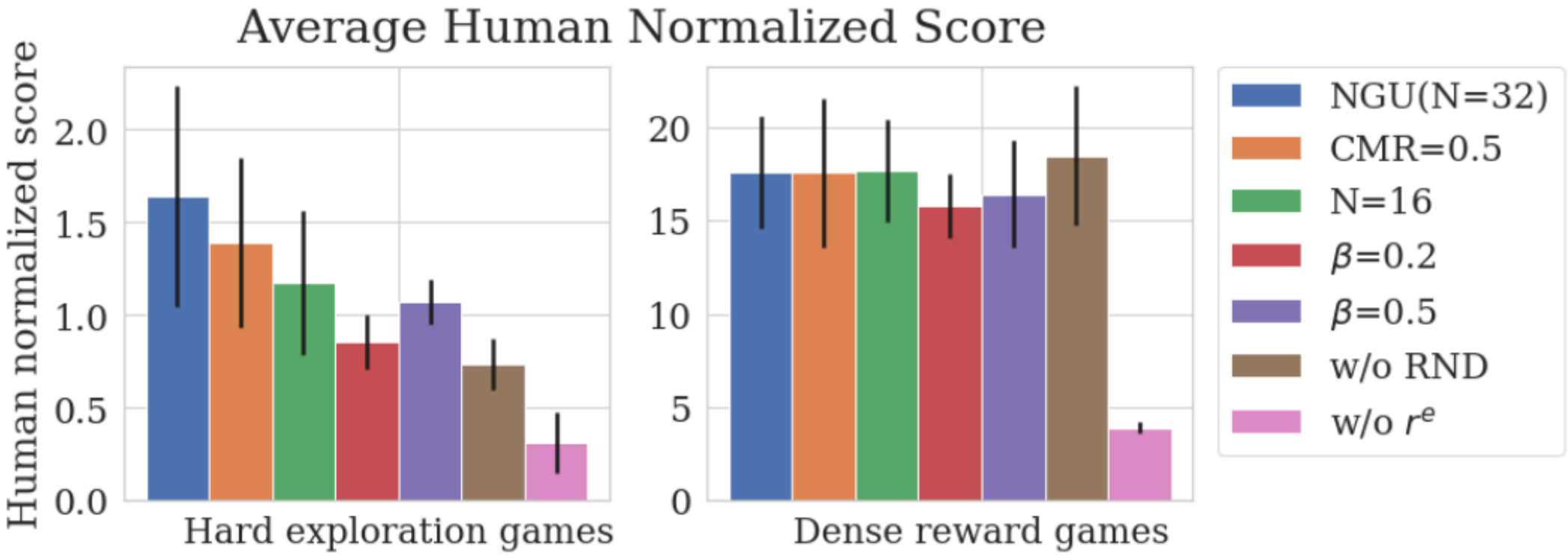


Figure 3: Human Normalized Scores on dense reward and hard exploration games.

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

Table 1: Results against exploration algorithm baselines. Best baseline takes the best result among R2D2 (Kapturowski et al., 2019), DQN + PixelCNN (Ostrovski et al., 2017), DQN + CTS (Bellemare et al., 2016), RND (Burda et al., 2018b), and PPO + CoEx (Choi et al., 2018) for each game.

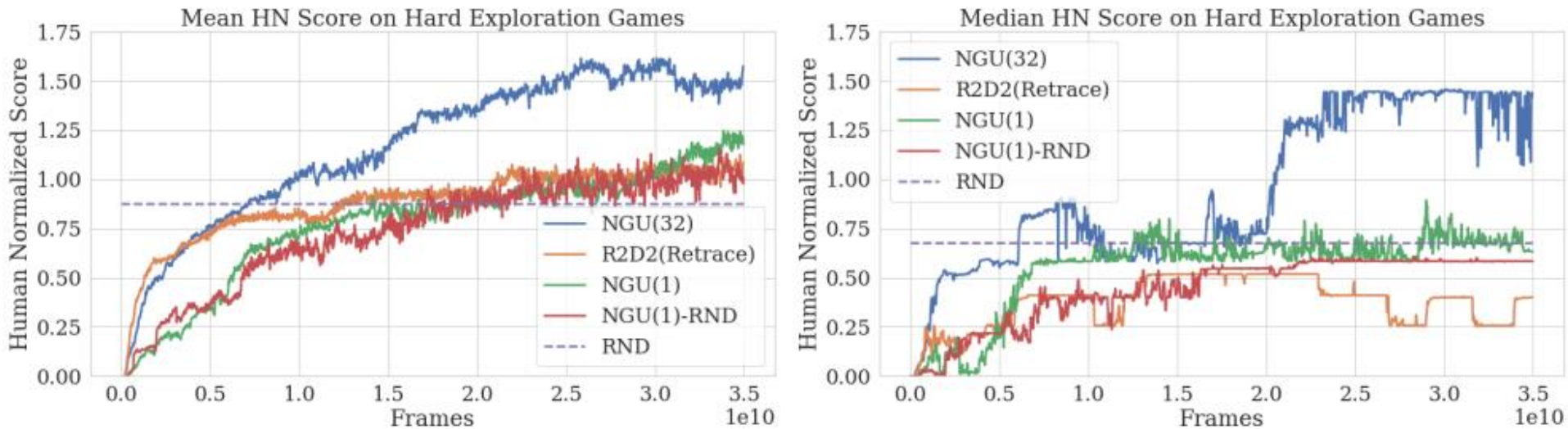


Figure 4: Human Normalized Scores on the 6 hard exploration games.

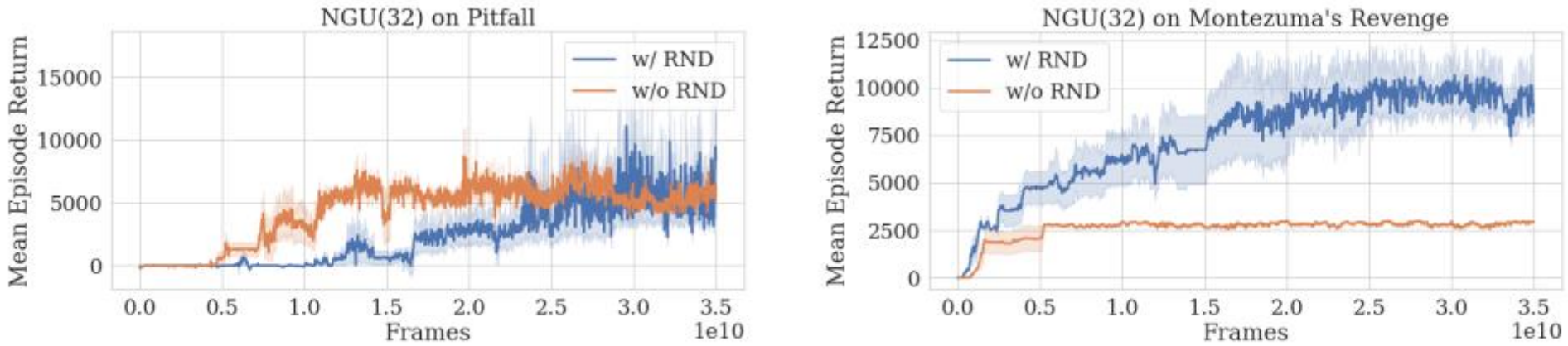


Figure 5: Mean episodic return for agents trained (left) *Pitfall!* and (right) *Montezuma's Revenge*.

Algorithm	Pong	QBert	Breakout	Space Invaders	Beam Rider
Human	14.6	13.4k	30.5	1.6k	16.9k
R2D2	**21.0**	408.8k	837.7	43.2k	**188.2k**
R2D2+RND	20.7±0.0	353.5k±41.0k	815.8±5.3	**54.5k±2.8k**	85.7k±9.0k
R2D2(Retrace)	20.9±0.0	415.6k±55.8k	838.3±7.0	35.0k±13.0k	111.1k±5.0k
NGU(N=1)-RND	-8.1±1.7	647.1k±50.5k	**864.0±0.0**	45.3k±4.9k	166.5k±8.6k
NGU(N=1)	-9.4±2.6	**684.7k±8.8k**	**864.0±0.0**	43.0k±3.9k	114.6k±2.3k
NGU(N=32)	19.6±0.1	465.8k±84.9k	532.8±16.5	44.6k±1.2k	68.7k±11.1k

Table 2: Results against baselines on dense reward games.

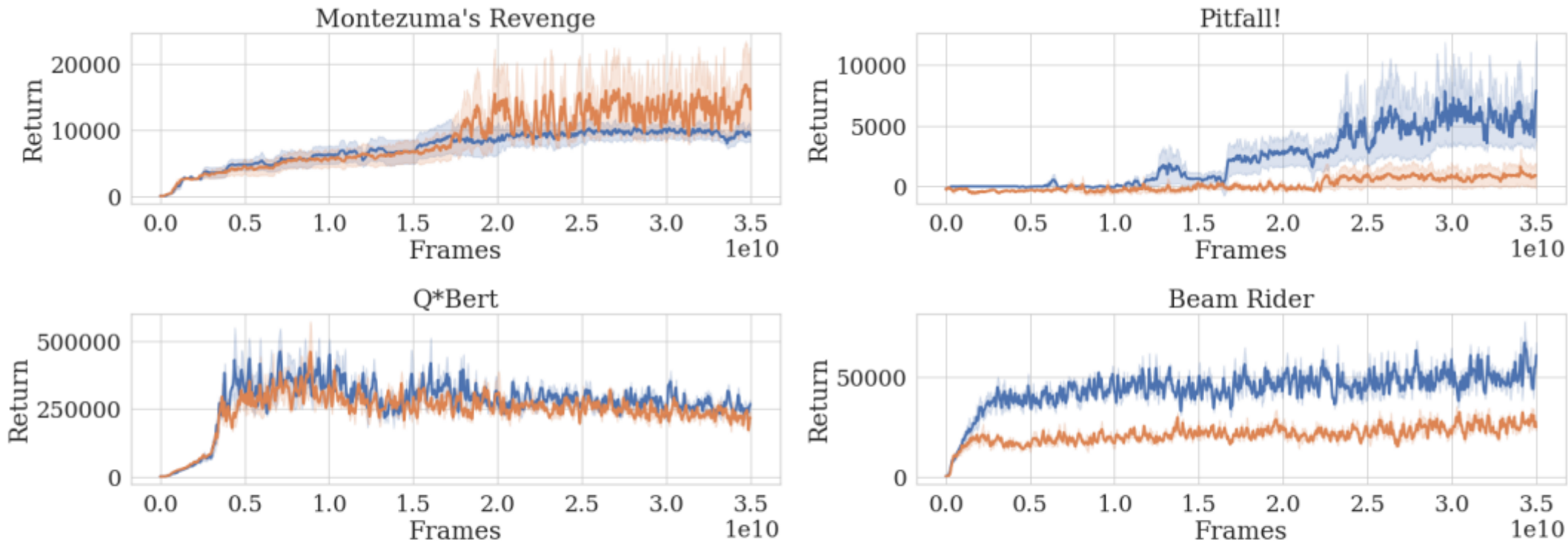


Figure 6: NGU(N=32) behavior for β_0 (blue) and β_{31} (orange).