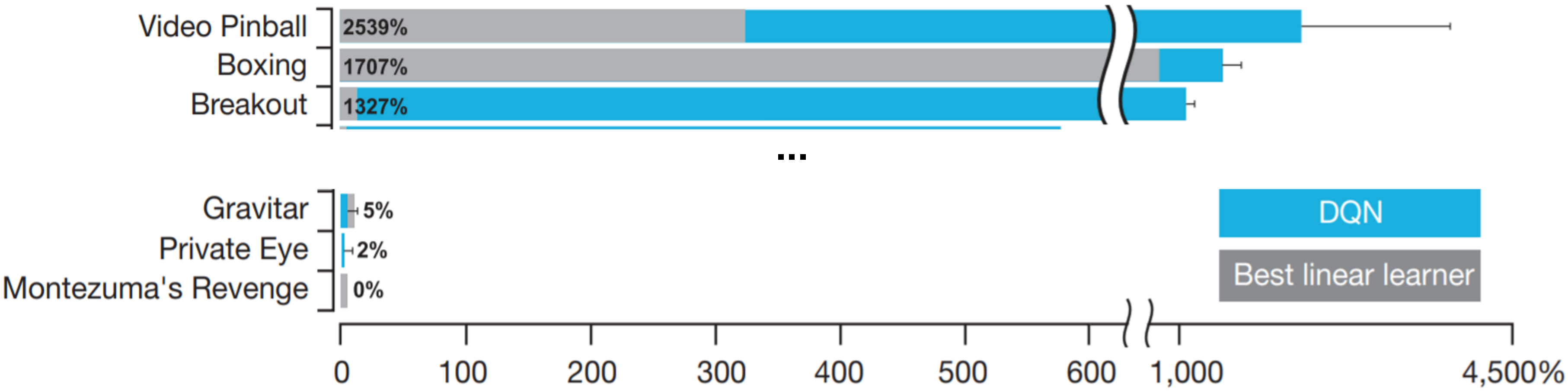
Never Give Up
Learning Directed Exploration Strategies

Accepted in ICLR 2020

Kyeong Ryeol, Go
M.S. Candidate of OSI Lab

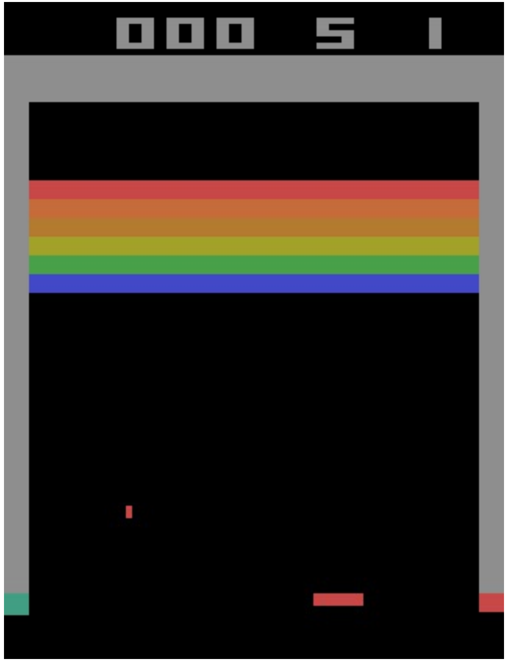
Overview

- Motivation
  - In sparse reward setting, exploration strategy is crucial

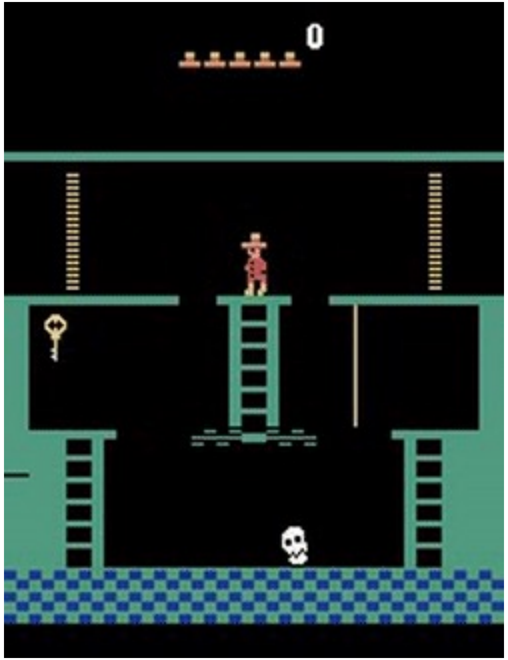


$$100*(DQN score - randomplay score)/(human score - randomplay score)$$

Overview



Break out



Montezuma's
revenge

Pitfalls!

Overview

- Intrinsic reward
  - Quantifies the novelty of the experience to respond to agent's curiousity
- Previous work
  - Visitation counts : ignore the downstream learning opportunity
  - Prediction error : expensive, error prone, generalization issue
- Main idea
  - Designing the intrinsic rewards to encourage the agents to visit diverse states **within** and **across** episodes
  - Learn a range of policies with varying the exploration and exploitation trade-offs

Contents

1. Never Give Up Intrinsic reward

2. Model architecture & Loss function

3. Experiment

4. Conclusion

5. Appendix

Never Give Up Intrinsic reward

NGU Intrinsic Reward

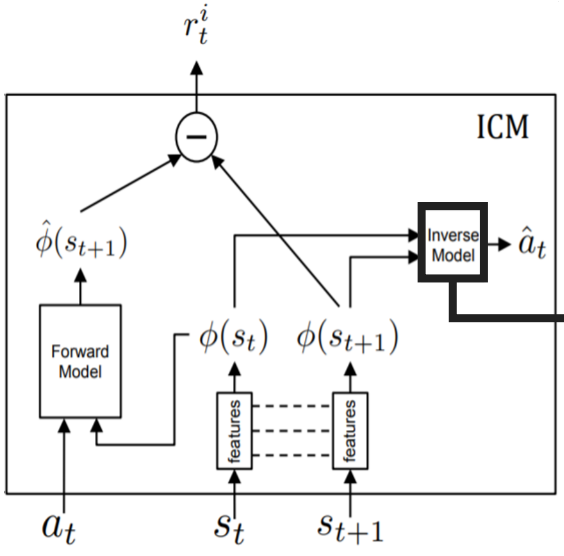
- Reward at time t : $r_t = r_t^e + \beta r_t^i$
    - Linear combination of extrinsic reward and intrinsic reward
    - β is the balancing parameter

- 3 Properties of intrinsic reward
    - The notion of state ignores aspects of an environment that are not influenced by an agent. (Embedding function)
    - It rapidly discourages revisiting the same state within the same episode. (Episodic Novelty Module)
    - It slowly discourages visiting to states where visited many times across episodes. (Life-long Novelty Module)

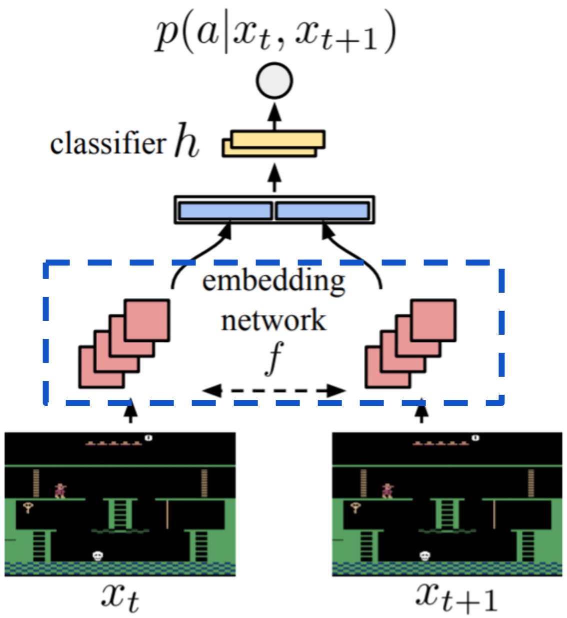
NGU Intrinsic Reward - Embedding function

- Embedding function f
  - The notion of state ignores aspects of an environment that are not influenced by an agent.
    - Input : two consecutive observations
    - Output : In-between action

Robust to inherent stochasticity

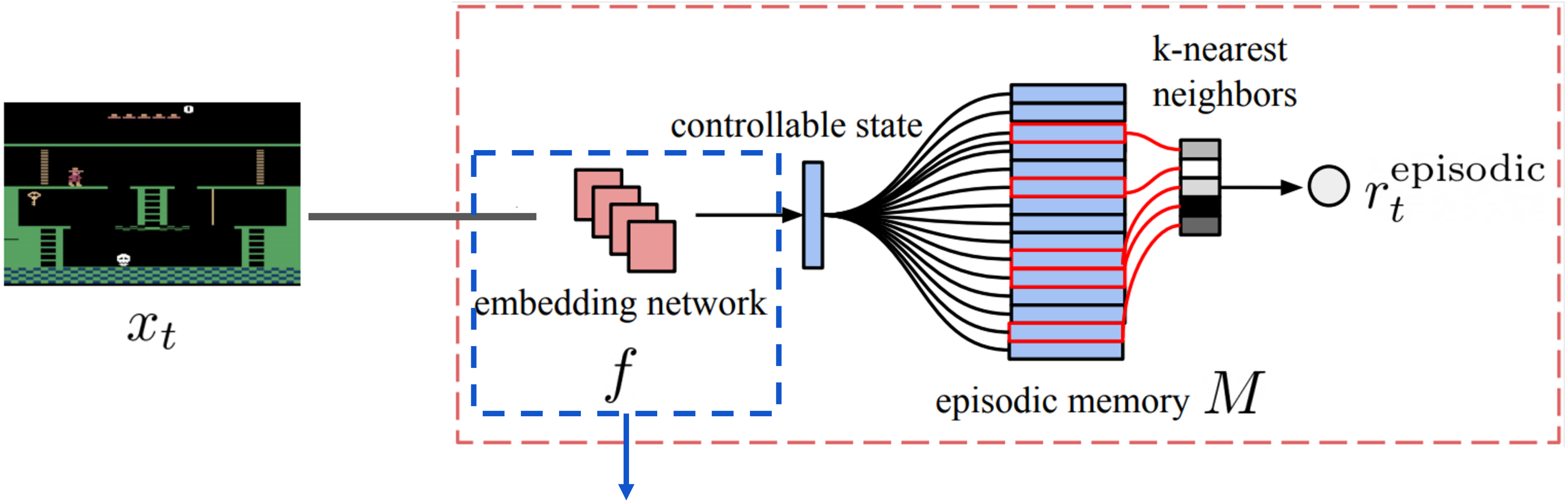


Intrinsic Curiosity Module (ICM)



NGU Intrinsic reward - Episodic novelty module

- Episodic novelty module $r_t^{episodic}$
  - It rapidly discourages revisiting the same state within the same episode.



Embedding function f

NGU Intrinsic reward - Episodic novelty module

- Episodic novelty module $r_t^{episodic}$
  - Episodic memory M : At every step, agent computes an episodic intrinsic reward $r_t^{episodic}$ and appends the controllable state corresponding to the current observation to the memory M

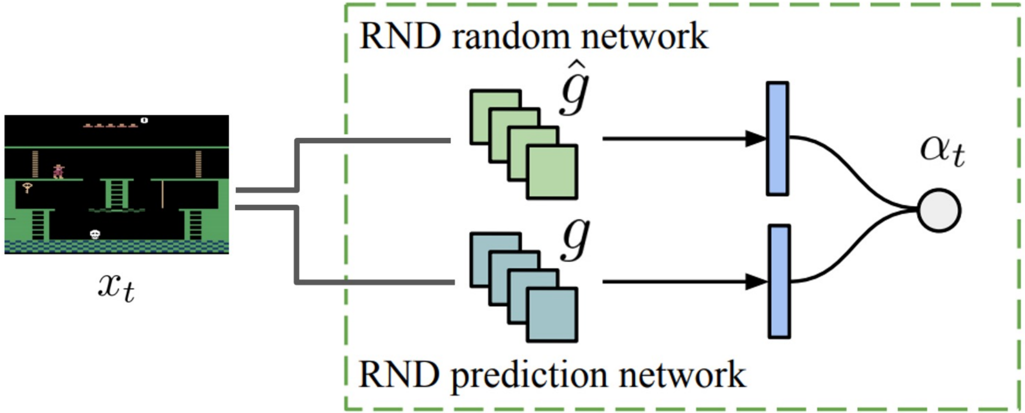
$$\boxed{r_t^{\text{episodic}} = \frac{1}{\sqrt{n(f(x_t))}}} \approx \frac{1}{\sqrt{\sum_{f_i \in N_k} K(f(x_t), f_i) + c}} \qquad K(x, y) = \frac{\epsilon}{\frac{d^2(x,y)}{d_m^2} + \epsilon}$$

From UCB

  - embedding function $f : O \rightarrow \mathbb{R}^p$: mapping the current observation to a p-dimensional vector corresponding to its controllable state
  - episodic novelty : Promotes the agent to visit as many different states as possible within a single episode

NGU Intrinsic Reward - Life-long novelty module

- Life-long novelty module α_t
  - It slowly discourages visiting to states where visited many times across episodes.
  - SOTA for Montezuma's revenge at the time



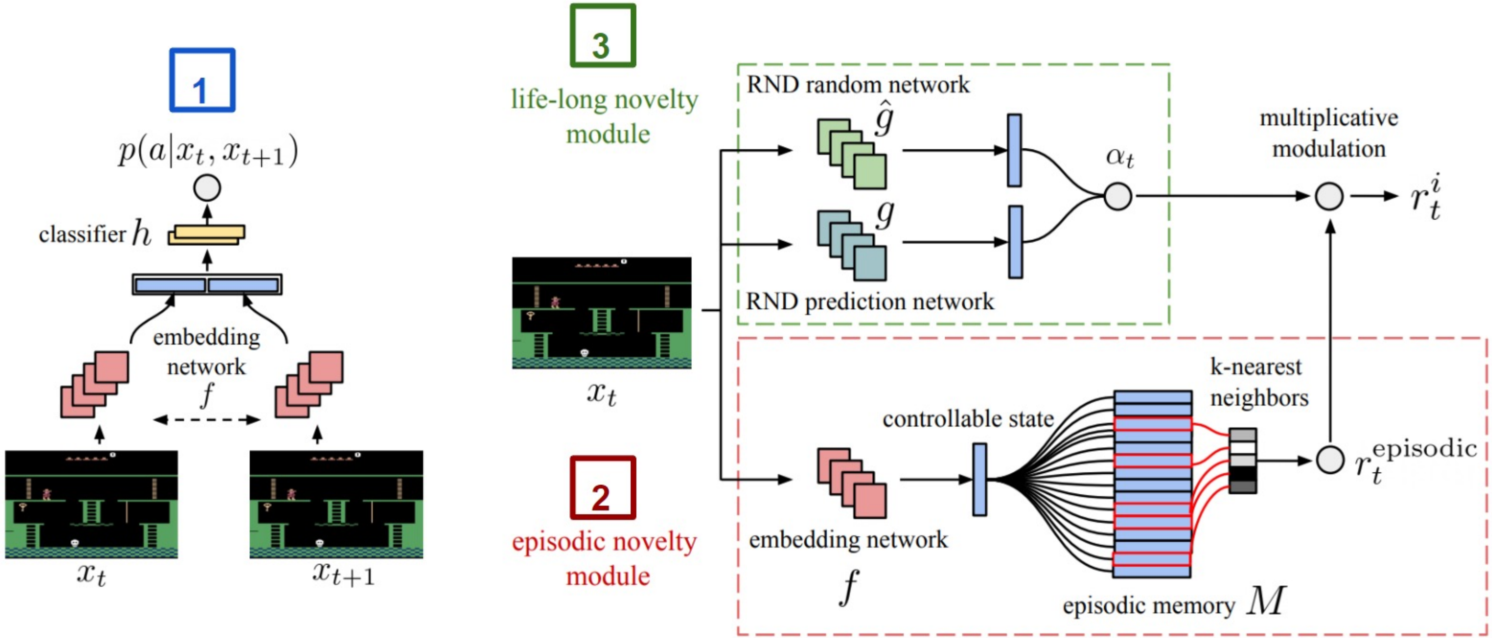
$$\mathrm{err}(x_t) = ||\hat{g}(x_t;\theta) - g(x_t)||^2$$

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

Error would be small if similar experience to the current one would have been accumulated

NGU Intrinsic Reward

- Intrinsic reward $r_t^i = r_t^{episodic} \cdot min\{max\{\alpha_t, 1\}, L\}\}$
  - Episodic intrinsic reward is modulated by Life-long Intrinsic reward



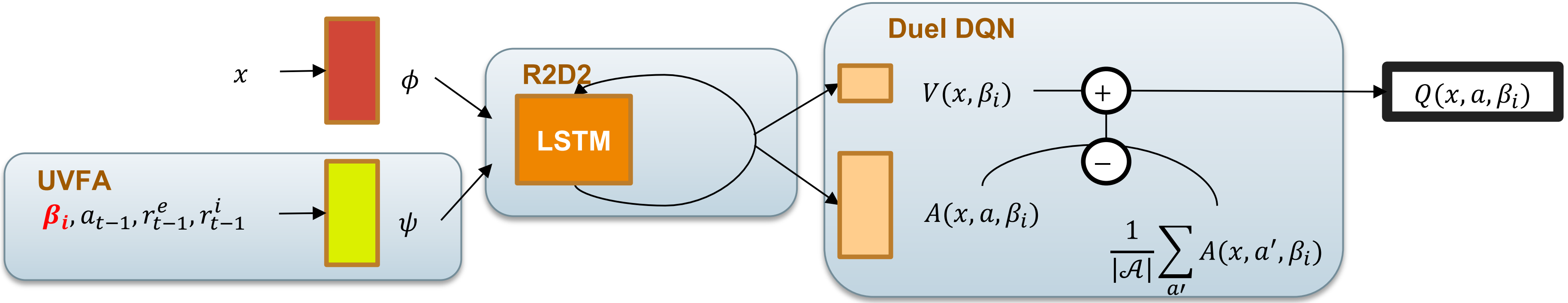
Model architecture & Loss function

NGU - Proposed architecture

- Train the function approximator for $Q(x, a)$
  - Utilize UVFA to simultaneously consider a range of policies varying the weights on the intrinsic reward for overall reward design $\{\beta_i\}_{i=0}^{N-1}$: $r_t^{\beta_i} = r_t^e + \beta_i r_t^i$

※ Recurrent Replay Distributed DQN (R2D2)
: DQN with LSTM module with distributed prioritized experience replay from Ape-X

※ Duel Deep Q-Network (Duel DQN)
: Devise two stream model whose outputs are $V(s)$ and $A(s, a)$



NGU - RL Loss function (1)

- Retrace(λ)
  - safe and efficient off-policy algorithm by using $c_k = \lambda \min\left(1, \frac{\pi(a_k, x_k)}{\mu(a_k, x_k)}\right)$
    - π : target policy, μ : behavior policy

    (Used along with Double q-learning and multi-step return to lower the variance)

Arbitrary behavior policy is fine

$$y_t^{Retrace(\lambda)} = E_{\boxed{a_{t+1},\dots,a_{t+s}\sim\mu}}\left[\sum_{s=0}^{k}\gamma^s\left(\prod_{i=1}^{s} c_{t+i}\right)\delta_s + Q(x_t, a_t; \boxed{\theta^-})\right]$$

$$\delta_s = r_{t+s} + \gamma E_{\boxed{a_{t+s+1}\sim\pi}}[Q(x_{t+s+1}, a_{t+s+1}; \boxed{\theta^-})] - Q(x_t, a_t; \boxed{\theta^-})$$

Select action increasingly greedy w.r.t. $Q(x, a; \theta)$ Evaluate target on $Q(x, a; \theta^-)$

NGU - RL Loss function (2)

- Transformed Retrace(λ)
  - Squash the scale of the action-value function

$$y_t^{T\_Retrace(\lambda)} = E_{a_{t+1},\dots,a_{t+s}\sim\mu}\left[h\left(\sum_{s=0}^{k}\gamma^s\left(\prod_{i=1}^{s}c_{t+i}\right)\delta_s^h + h^{-1}(Q(x_t,a_t;\theta^-))\right)\right]$$

$$\delta_s^h = r_{t+s} + \gamma E_{a_{t+s+1}\sim\pi}\left[h^{-1}(Q(x_{t+s+1},a_{t+s+1};\theta^-))\right] - h^{-1}(Q(x_t,a_t;\theta^-))$$

$$h(x) = sign(x)\left(\sqrt{|x+1|}-1\right) + \epsilon x,$$

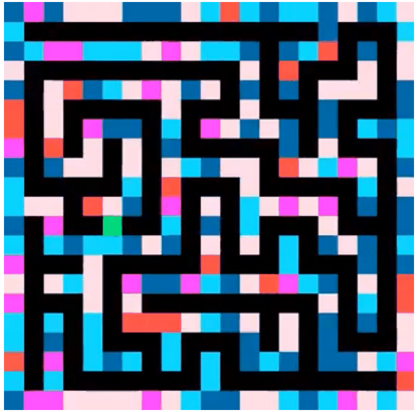
$$h^{-1}(x) = sign(x)\left(\left(\frac{\sqrt{1+4\epsilon(|x|+1+\epsilon}-1}{2\epsilon}\right)-1\right)$$

$$\boxed{L(\theta) = E_{(s,a,r,s')\in B}\left[\left(y^{T\_Retrace(\lambda)} - Q(s,a;\theta)\right)^2\right]}$$

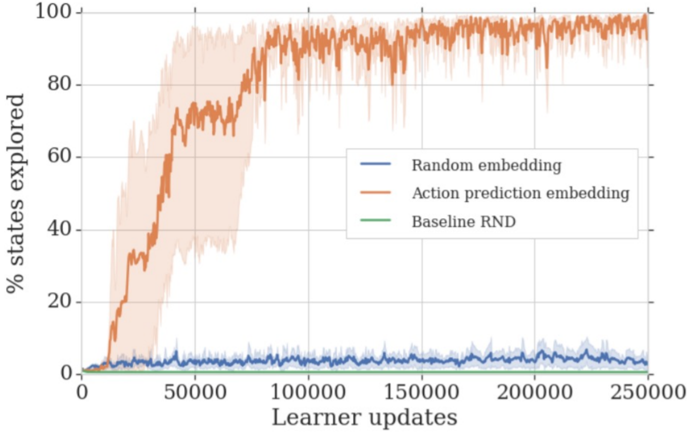
Experiment

Experiments

- Random Disco Maze
    - Things to check
        1. Effectiveness of the purely exploratory policy
        2. Effectiveness of the controllable state representation

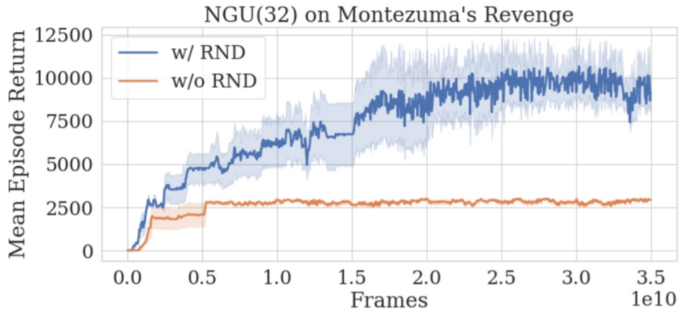


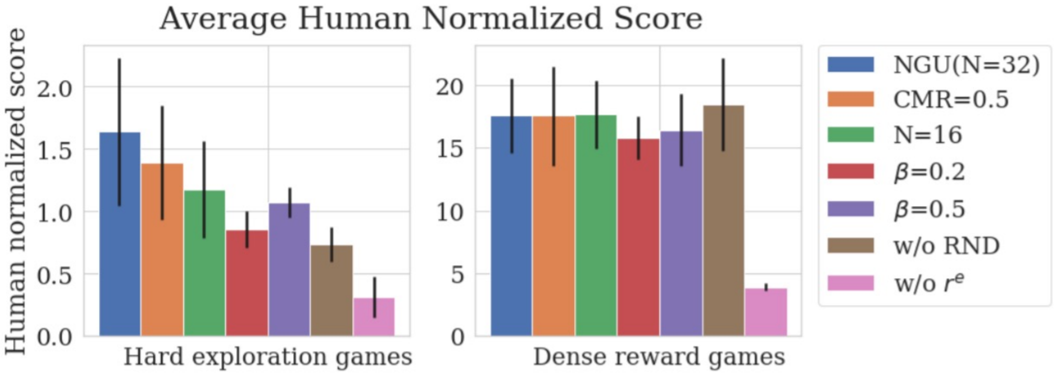
Black – pathways, **Green** – agent
Random - wall



Experiments

- Ablation study on ATARI
  - N ≈ Number of beta values (Default = 32)
  - CMR ≈ Proportion of off-policy experiences (Default = 0)
  - β ≈ Magnitude of the maximum beta value (Default = 0.3)
  - RND ≈ Whether the life-long novelty module is used or not (Default = with)
  - r^e ≈ Whether the extrinsic reward is used or not (Default = with)





Justification
- UVFA for varying β
- RND for modulation
- Augmented reward

Experiments

- Comparing to several baselines

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

Dense Reward Games

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

Conclusion

Conclusion

- Summary
  - Designing the intrinsic rewards to encourage the agents to visit diverse states **within** and **across** episodes
    (*Embedding function*, *Episodic Novelty Module*, *Life-long Novelty Module*)
  - Learn a range of policies with varying the exploration and exploitation trade-offs
    (*UVFA framework*)

- Extension
  - Agent 57 : Outperforming the Atari human benchmark
    (https://arxiv.org/abs/2003.13350)
    - Utilizing the meta controller to choose which beta to be utilized for each episode

Thank you for your attention

Appendix

DQN variants

- Deep Q-Network (DQN)
    - Utilize DNN as the function approximator for Q-value function
    - Utilize the experience replay and target network to stabilize learning

$$L(\theta) = E_{(s,a,r,s')\in B}\left[\left(y^{DQN} - Q(s,a;\theta)\right)^2\right] \ where \ y^{DQN} = r + \gamma \max_{a'} Q(s',a';\theta^-)$$

- Double Deep Q-Network (DDQN)
    - Resolve the overestimation by selecting the action by the online network and estimating Q-value by the target network

$$y^{DDQN} = r + \gamma Q\left(s', \arg\max_{a} Q(s',a';\theta); \theta^-\right)$$

- Dueling Deep Q-Network (Duel DQN)
    - Two streams of network to compute $V(s)$ and $A(s,a)$ for computing $Q(s,a)$

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a')\right)$$

Sampling with Priority

- Prioritized Experience Replay
  - Impose priority on the experiences by the absolute TD error
  - Use importance sampling weight to correct the bias

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha} \quad where\ p_i = |\delta_i| + \epsilon \ \ or\ \ p_i = \frac{1}{rank(|\delta_i|)}$$

$$\delta = r + \gamma Q\left(s', \arg\max_a Q(s', a'; \theta)\, ; \theta^-\right) - Q(s, a; \theta)$$

$$\theta \leftarrow \theta + \eta \cdot \frac{w_i}{\max_i w_i} \cdot \delta_i \cdot \nabla_\theta Q(s_i, a_i) \ \ where\ \ i \sim P(i)\ and\ \ w_i = \left(\frac{1}{N \cdot P(i)}\right)^\beta$$

(continued)

- Distributed Prioritized Experience Replay (Ape-X DQN)
  - Extend prioritized experience replay to the distributed setting
    - Actors : Select actions in the environment and store them in a buffer
      - Set the priority by the absolute TD error
      - Periodically synchronize the parameter of the learner
    - Learner : Sample experiences with priority and update the policy parameter
      - Update the priority again by the absolute TD error with the updated parameter
  - Use double q-learning and multi-step target

$$y_t^{Ape-X} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n Q\left(s_{t+n}, \arg\max_{a'} Q(s_{t+n}, a'; \theta); \theta^-\right)$$

Recurrent Replay Distributed DQN

- Recurrent Replay Distributed DQN (R2D2)
  - Resolve the representation drift and recurrent state staleness
    - Storing recurrent state in replay memory
    - Allow a burn-in period by the portion of replay memory

  (verifies its effectiveness by checking the Q-value discrepancy)
  - Impose priority by a mixture of max and mean absolute n-step TD-error

$$p(i) = \eta \cdot \max_i \delta_i + (1 - \eta) \cdot mean_i |\delta_i|$$

  - Modify Ape-X DQN target by rescaling

$$y_t^{R2D2} = \sum_{k=0}^{n-1} \gamma^k r_{t+k} + \gamma^n h^{-1}\left(Q\left(s_{t+n}, \arg\max_{a'} Q(s_{t+n}, a'; \theta); \theta^-\right)\right)$$

$$h(x) = sign(x)\left(\sqrt{|x+1|} - 1\right) + \epsilon x, \qquad h^{-1}(x) = sign(x)\left(\left(\frac{\sqrt{1 + 4\epsilon(|x| + 1 + \epsilon} - 1}{2\epsilon}\right) - 1\right)$$