Fraunhofer IIS
FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Memory-based Exploration

Christopher Mutschler

Agenda

- Motivation, Problem Definition & Multi-Armed Bandits
- Classic Exploration Strategies
  - Epsilon Greedy
  - (Bayesian) Upper Confidence Bounds
  - Thomson Sampling
- **Exploration in Deep RL**
  - Count-based Exploration: Density Models, Hashing
  - Prediction-based Exploration:
    - Forward Dynamics
    - Random Networks
    - Physical Properties
  - **Memory-based Exploration:**
    - Episodic Memory
    - Direct Exploration
- Summary and Outlook

Memory-based Exploration

- Reward-based exploration works well in many applications

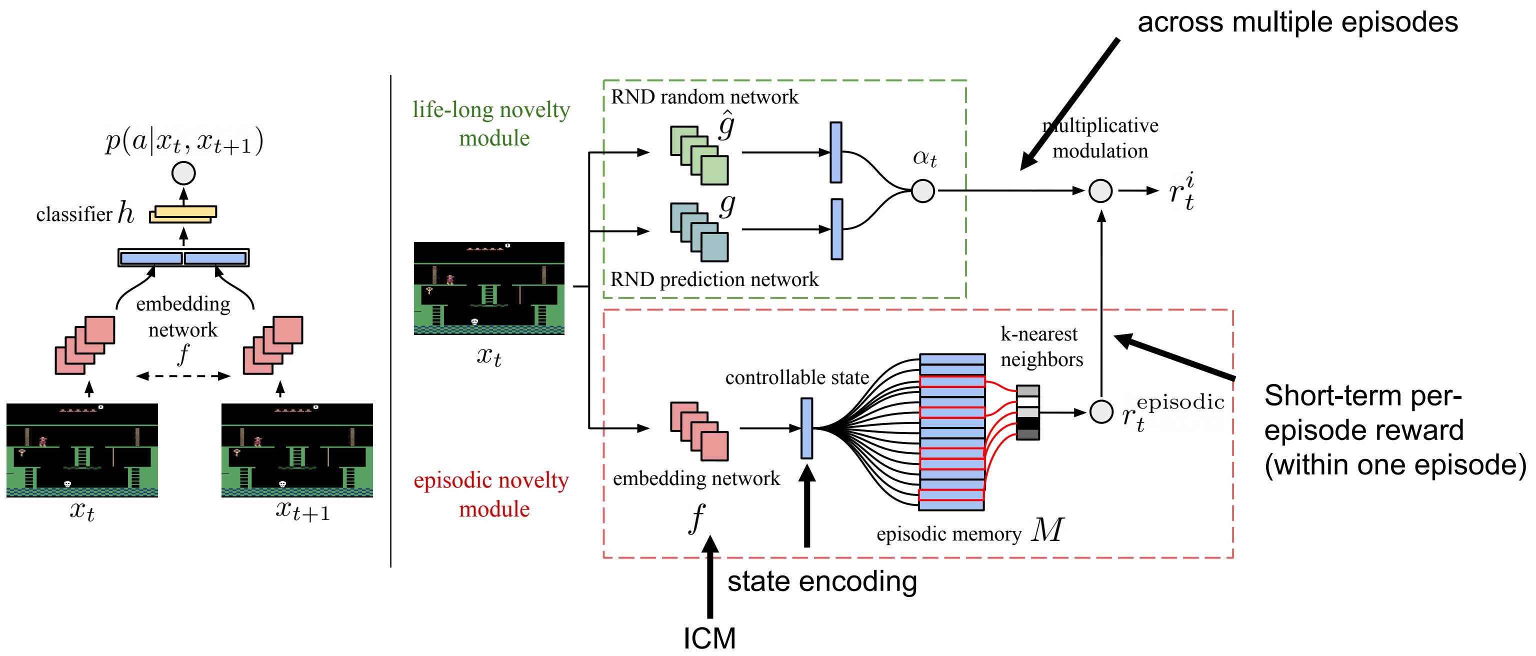
- However, it suffers from several disadvantages:
  - Function approximation is slow
  - Exploration bonus is non-stationary
  - Knowledge fading: states are no longer novel in time and do no longer provide intrinsic reward signals

Idea of memory-based exploration:

→ *Use external memories in combination to resolve such disadvantages!*

Episodic Memory: Never Give Up (NGU)[1]

- So far: RND works great but suffers from episodic settings
- Idea: use two modules:
  1. RND as a lifelong novelty module, and
  2. an episodic novelty module for rapid in-episode adaptation



[1] *Badia et al.: Never Give Up: Learning Directed Exploration Strategies. ICLR 2020.*

Episodic Memory: Never Give Up (NGU)[1]

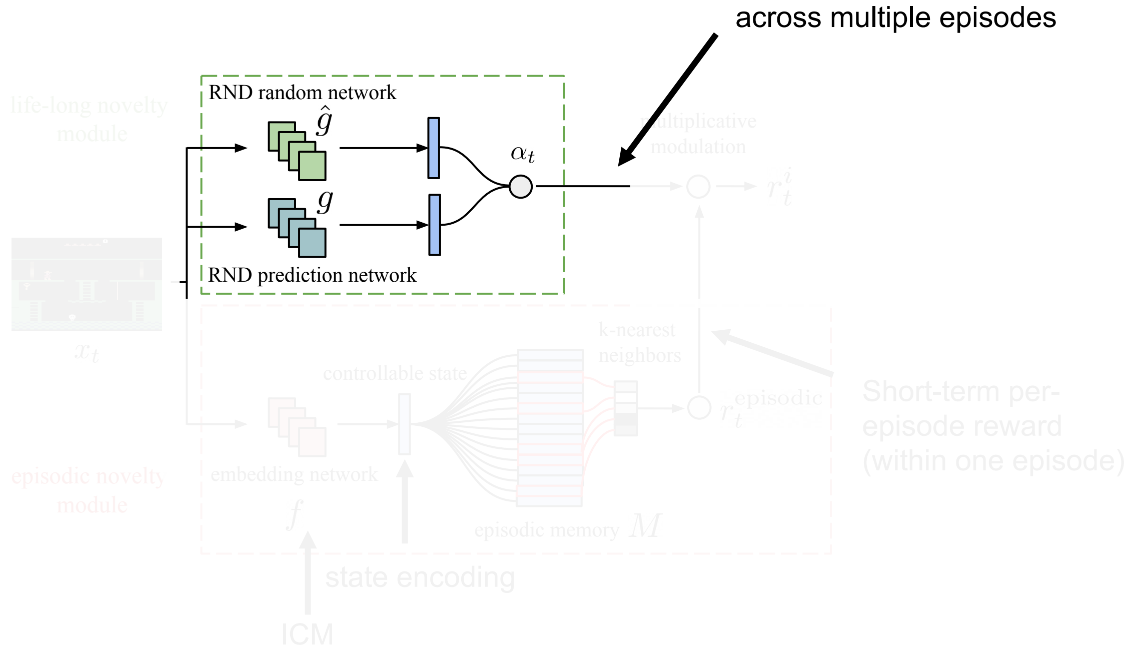
- So far: RND works great but suffers from episodic settings
- Idea: use two modules:
  1. RND as a lifelong novelty module, and
  2. an episodic novelty module for rapid in-episode adaptation

- RND derives a life-long novelty bonus
- The exploration bonus is given as

$$\alpha_t = 1 + \frac{e^{RND}(s_t) - \mu_e}{\sigma_e}, \text{ with}$$

  - the running mean μ_e , and
  - the running standard deviation σ_e

  of the RND error $e^{RND}(s_t)$



[1] Badia et al.: Never Give Up: Learning Directed Exploration Strategies. ICLR 2020.

Episodic Memory: Never Give Up (NGU)[1]

- So far: RND works great but suffers from episodic settings
- Idea: use two modules:
  1. RND as a lifelong novelty module, and
  2. an episodic novelty module for rapid in-episode adaptation

- Add $\phi(s_t)$ into M
- Compare $\phi(s_t)$ to the other content in M:

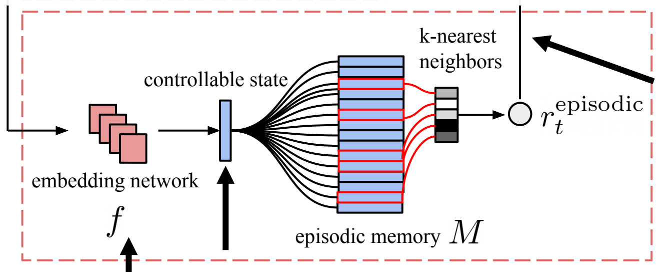
$$r_t^e = \frac{1}{\sqrt{\sum_{\phi_i \in N_k} K(\phi(x_t), \phi_i) + c}}, \text{ with}$$

  - a kernel function $K(x, y)$
  - # nearest neighbors N_k, and
  - a constant c to avoid a zero-sum

Originally, the paper proposes

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

- Euclidean distance d between two samples
- squared Euclidean distance d_m^2 of the k-th nearest neighbors → more robust
- a small constant ϵ



Short-term per-episode reward (within one episode)

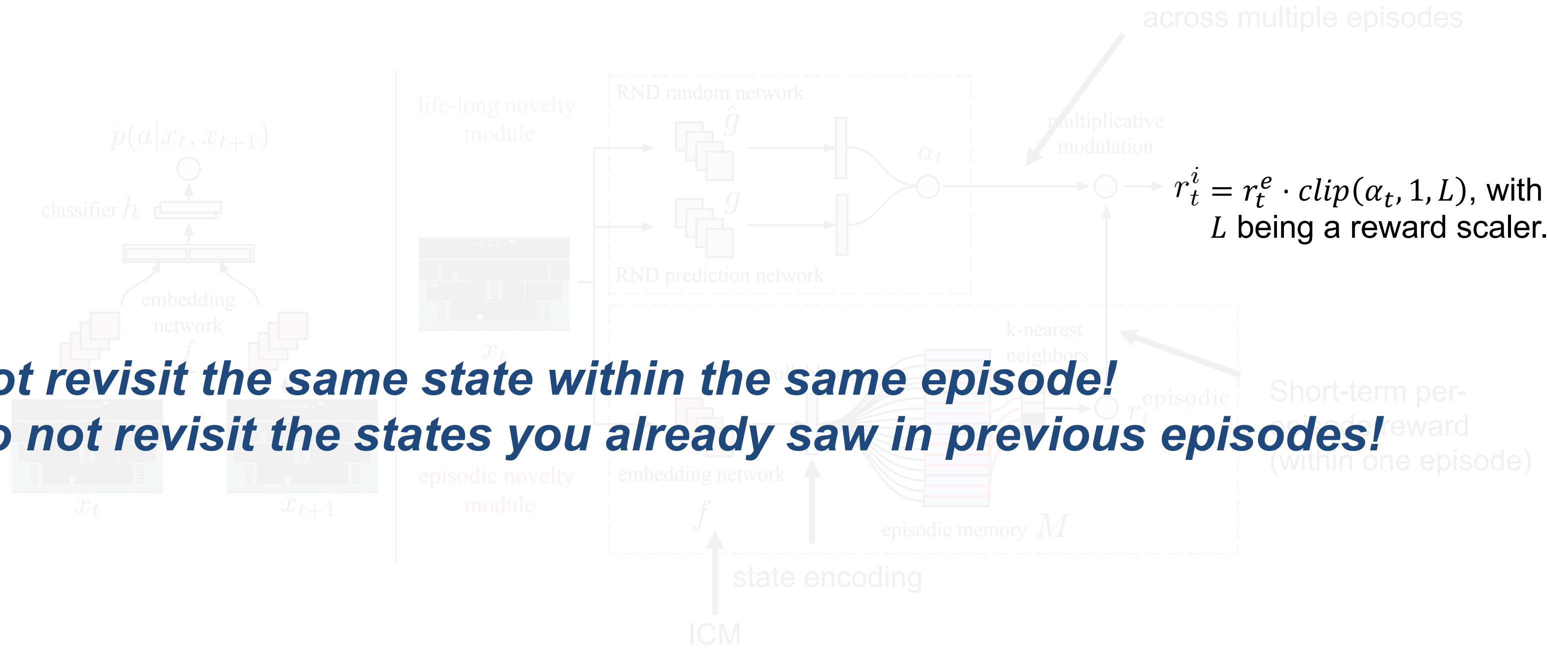
state encoding

ICM

[1] Badia et al.: Never Give Up: Learning Directed Exploration Strategies. ICLR 2020.

Episodic Memory: Never Give Up (NGU)[1]

- So far: RND works great but suffers from episodic settings
- Idea: use two modules:
  1. RND as a lifelong novelty module, and
  2. an episodic novelty module for rapid in-episode adaptation



$r_t^i = r_t^e \cdot clip(\alpha_t, 1, L)$, with L being a reward scaler.

→ ***Do not revisit the same state within the same episode!***
→ ***Try to not revisit the states you already saw in previous episodes!***

[1] *Badia et al.: Never Give Up: Learning Directed Exploration Strategies. ICLR 2020.*

Agent57[1]

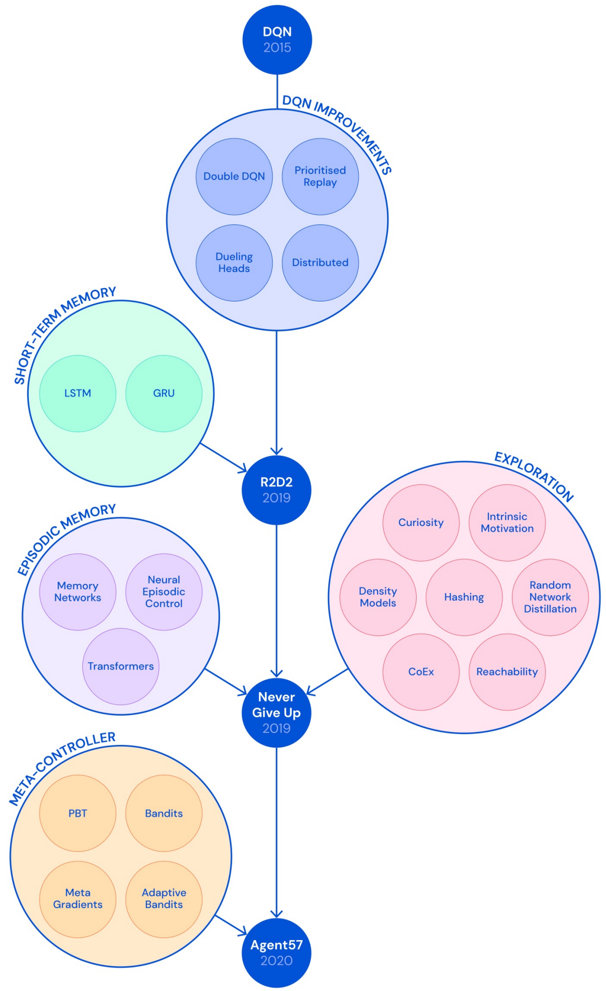
- Agent57 is the first RL agent who beats Atari57 consistently
- Two main improvements over NGU:

1. Population of policies:
    - Each policy has its own pair of exploration parameters $\{(\beta_j, \gamma_j)\}_{j=1}^{N}$
    - Policies with high β_j (and lower γ_j) make more progress at early stages
    - Policies with high γ_j (and lower β_j) make more progress at later stages
    - A meta-controller (sliding window UCB) is trained to select from the policies

2. Re-Parameterization of Q-value function:
    - Q-function is decomposed into intrinsic and extrinsic influence:

$$Q(s, a; \theta_j^i) = Q(s, a; \theta_j^e) + \beta_j Q(s, a; \theta_j^i)$$

- During training both parameter sets (θ^e and θ^i) are optimized separately with rewards r_j^e and r_j^i, respectively

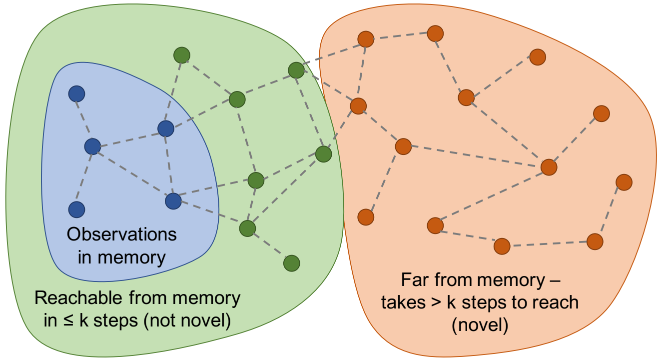


https://deepmind.com/blog/article/Agent57-Outperforming-the-human-Atari-benchmark

[1] *Badia et al.: Agent 57: Outperforming the Atari Human Benchmark. ICML 2020.*

* In fact: this paper was published before the NGU paper!

Episodic Curiosity through Reachability[1]

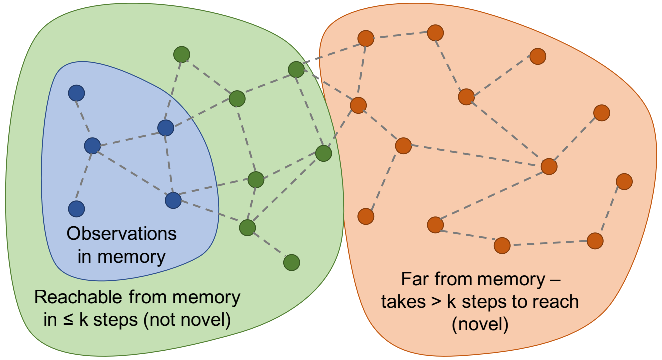


- There is a better thing than using the Euclidean distance*
- Episodic Curiosity (EC) module:
  - Measure the number of steps needed to transit between two states
  - The novelty than depends on the *reachability* between states
- General idea/steps:
  1. Clear episodic memory M on environment reset
  2. At each step t until the episode ends:
     1. Compare s_t with all the states in the memory
     2. If it takes more than k steps to reach s_t → agent gets a bonus
     3. If the novelty bonus is large enough → add s_t to M
  - But how can we estimate the *reachability*?

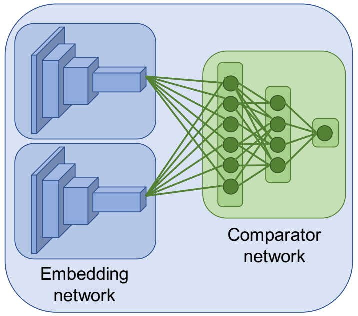
[1] *Savinov et al.: Episodic Curiosity through Reachability. ICLR 2019.*

Episodic Curiosity through Reachability[1]

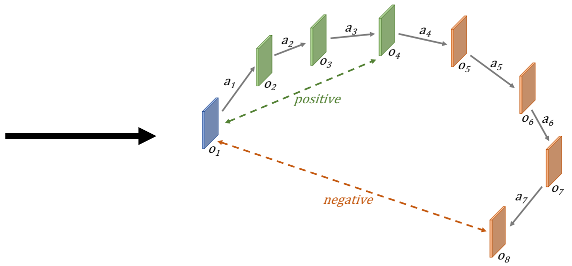
- Ideally, we would have access to a transition graph
    1. Not possible to build up (due to limited memory)
    2. Hard to build



- Solution: Train a Siamese neural network that predicts how far two states are apart



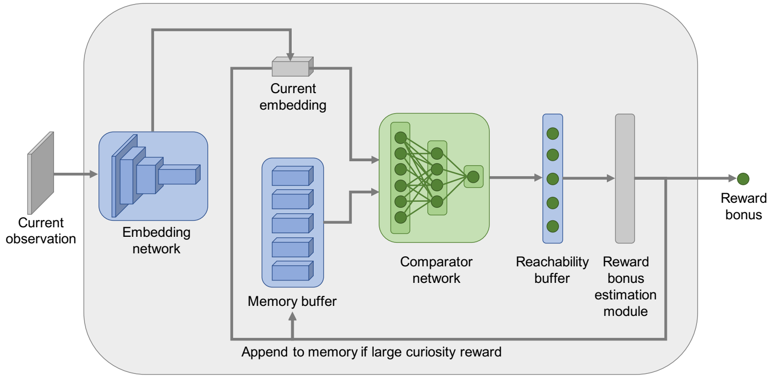
    - Embedding network E: $\mathcal{O} \rightarrow \mathbb{R}^n$
      (encodes states to feature vectors)
    - Comparator network C: $\mathbb{R}^n \times \mathbb{R}^n \rightarrow [0,1]$
      (reachability within k steps: 0 (not reachable) to 1 (reachable))
    - "R-network" as a classifier trained with logistic regression loss, based on trajectory data:
      → Hence: $R(o_i, o_j) = C\left(E(o_i), E(o_j)\right)$



[1] Savinov et al.: Episodic Curiosity through Reachability. ICLR 2019.

Episodic Curiosity through Reachability

- Putting all together: **Episodic Curiosity (EC) Module**
- At every time step



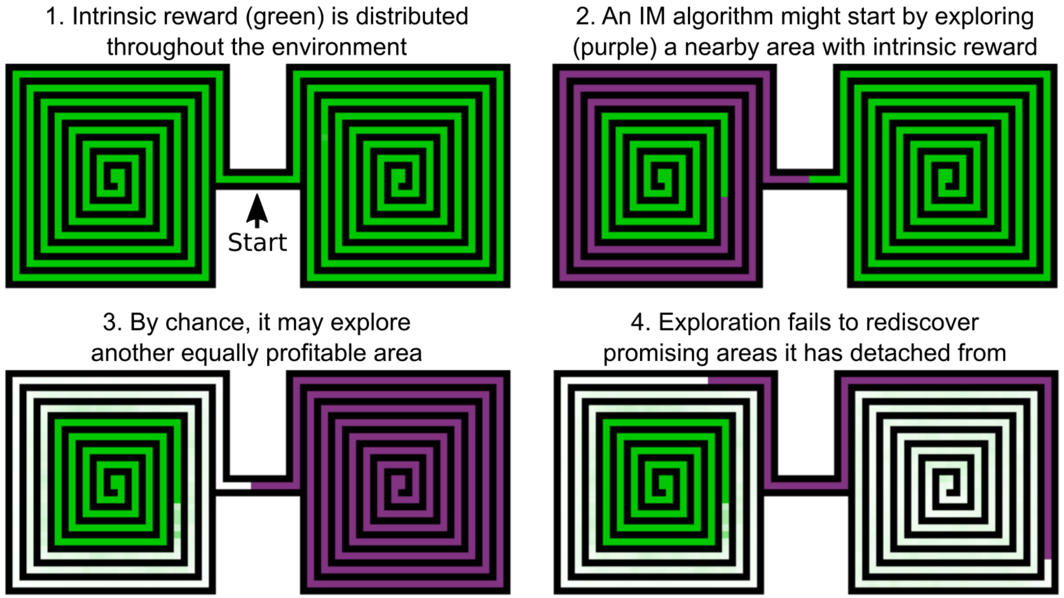
1. Embedding network processes o_t → embedding vector $e = E(o_t)$
2. Compare e with all embeddings in the buffer $M = \langle e_1, \ldots, e_{|M|} \rangle$ via C
   → fills the reachability buffer with values
   $$c_i = C(e_i, e), \quad i = 1, \ldots, M.$$
3. Compute the similarity score between e and the memory buffer M as $C(M, e) = F\left(c_1, \ldots, c_{|M|}\right) \in [0,1]$, where $F(\cdot)$ is a hyperparameter (function).
   - $\max(\cdot)$ would theoretically be a good choice but is prone to outliers
   - 90th percentile works better in experiments.
4. Compute the curiosity bonus as $b = B(M, e) = \alpha\left(\beta - C(M, e)\right)$
   - α tunes scale of task rewards
   - β defines the sign of the reward (0.5 works well for fixed-duration episodes)

[1] *Savinov et al.: Episodic Curiosity through Reachability. ICLR 2019.*

Direct Exploration

Go-Explore[1]

- The problems stemming from sparse rewards and intrinsic motivation are two-fold:
  1. Detachment
     - Intrinsic rewards are nearly always a consumable resource: short-term focus lies on such areas but with time they become less interesting to the agent
     - *Catastrophic forgetting: we forget things that happened far in the past*



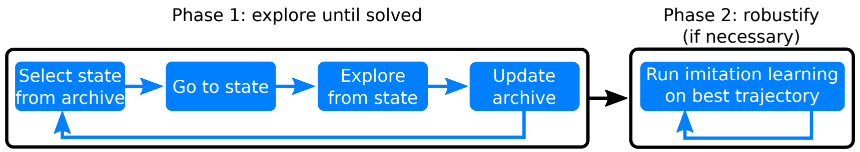
  2. Derailment
     - Describes the problem of re-visiting an interesting state again, in order to further explore from there
     - Previous work runs the policy again (with stochastic perturbation) in a "hope" to reach the desired state again

→ with naïve perturbations in complex environments this becomes highly unlikely

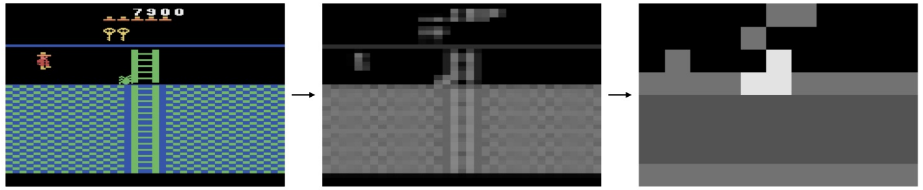
see also: https://towardsdatascience.com/a-short-introduction-to-go-explore-c61c2ef201f0

[1] Adrian Ecoffet et al.: Go-Explore: a new Approach for Hard-Exploration Problems. 2020.

Direct Exploration: Go-Explore



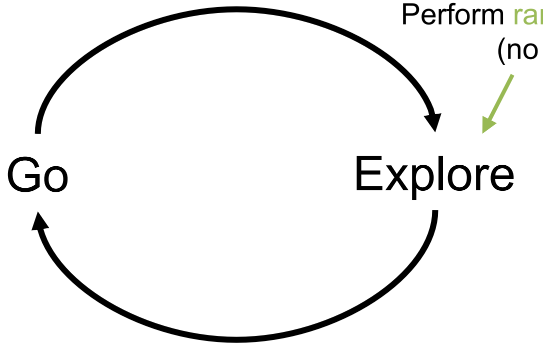
Go-Explore[1] addresses *detachment* and *derailment* using two phases:



1. Explore until solved

- No ML and NNs involved here. Just random (or semi-guided) exploration
- Main goal: find **interesting cells**
  - Newly discovered & high reward obtained to reach them
- For each interesting cell we store the
  1. full trajectory to get there
  2. a snapshot of the environment state
  3. total reward of the trajectory
  4. length of the trajectory
- If we revisit a state
  - Update the entry if it is better (i.e., short trajectory, higher reward)

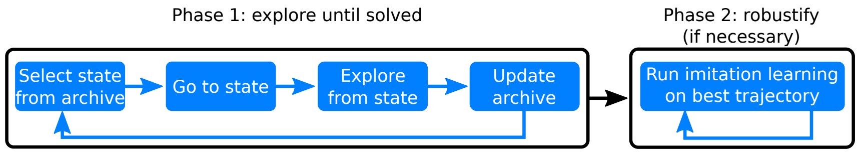
Perform random actions from thereon (no policy, no network)



Use a heuristic to choose a good cell and go there (i.e., best reward, least-visited,…)

[1] *Adrian Ecoffet et al.: Go-Explore: a new Approach for Hard-Exploration Problems. 2020.*

Direct Exploration: Go-Explore



Go-Explore[1] addresses *detachment* and *derailment* using two phases:

- Limitation of 1st phase: go to a cell is only "easy" in deterministic environments!

2. Robustify (if needed): "Backward" algorithm
   - Consider a sequence $c_1, c_2, \dots, c_{n-1}, c_n$, where c_n is the "go" cell
   - Algorithm:
     1. Initialize $c_i = c_{n-1}$
     2. Set environment to the snapshot of c_i and train the agent to reach c_n
     3. When agent finds a trajectory with an equal or higher reward:
        → set $i \leftarrow i - 1$
        → go to step 2
     4. Stop when $i = 1$

   - How to make policies more robust to non-determinism in Atari?
     - No ops
     - Sticky actions

[1] *Adrian Ecoffet et al.: Go-Explore: a new Approach for Hard-Exploration Problems. 2020.*

Direct Exploration: Go-Explore

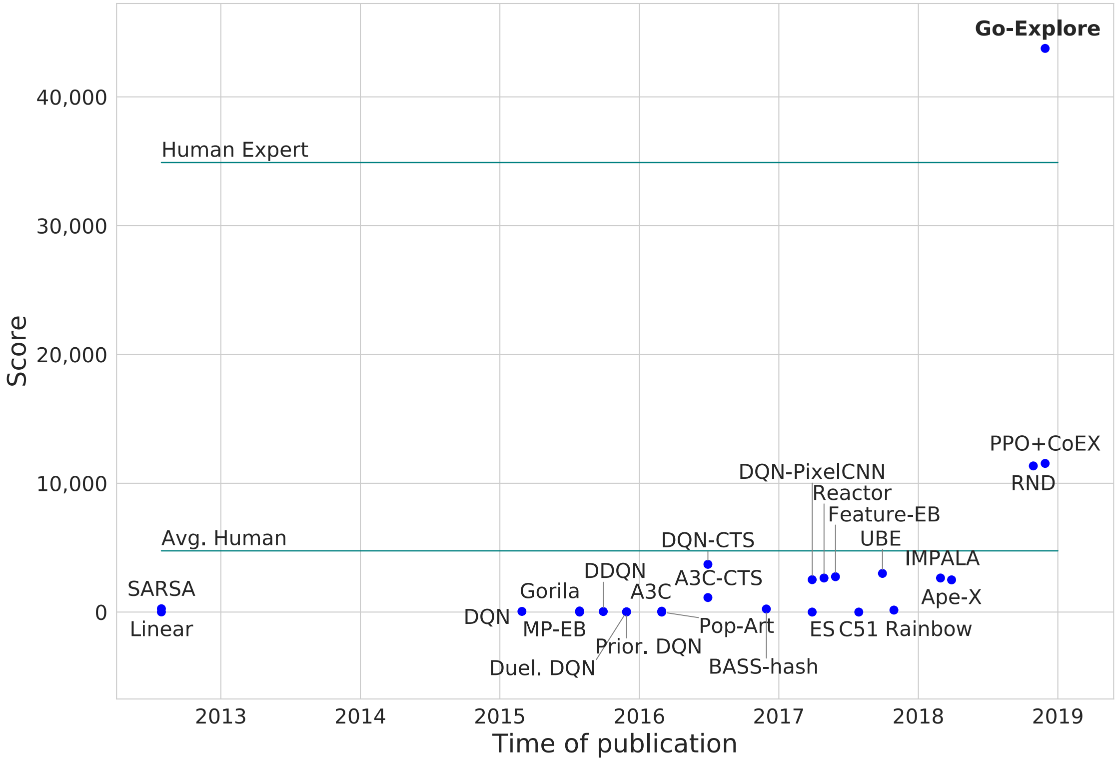
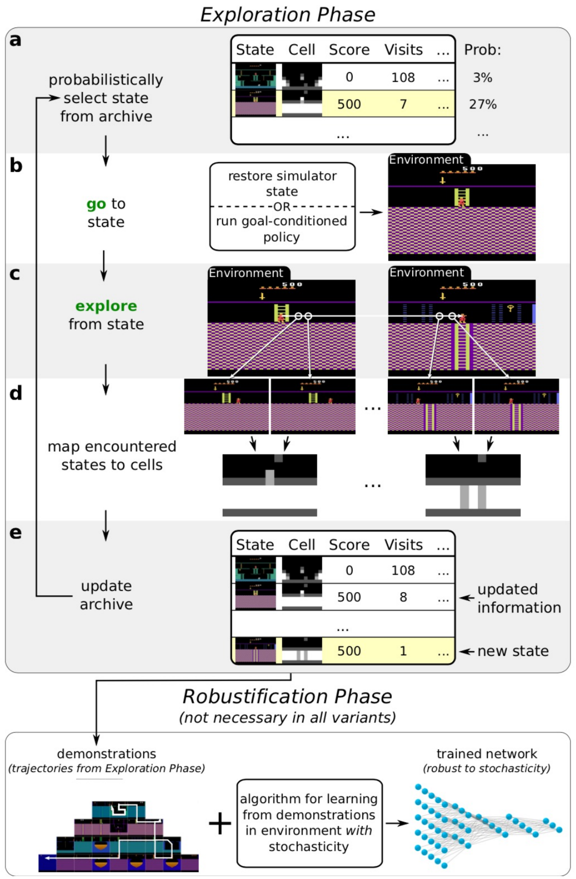


Figure 6: **History of progress on Montezuma's Revenge vs. the version of Go-Explore that does not harness domain knowledge.** Go-Explore significantly improves on the prior state of the art.

[1] *Adrian Ecoffet et al.: Go-Explore: a new Approach for Hard-Exploration Problems. 2020.*

Direct Exploration: Go-Explore (Improvements)

- “First return, then explore”[1]: *policy-based Go-Explore*
  - Instead of resetting the simulator: learn a goal-conditioned policy to reach a state
    → mainly trained to follow the best trajectory so far
  - Self-Imitation Learning to extract more information from successful trajectories



- “Memory based Trajectory-conditioned Policies for Learning from Sparse Reward
  - Like policy-based go-explore:
    - Maintain a memory of demonstrations collected during training
    - Use them to train a trajectory-conditioned policy via Self-Imitation Learning
  - Prioritize trajectories that end with a rare state during sampling.

[1] *Adrian Ecoffet et al.: First return, then explore. 2021.*
[2] *Yijie Guo et al.: Memory Based Trajectory-conditioned Policies for Learning from Sparse Rewards. 2021.*

Agenda

- Motivation, Problem Definition & Multi-Armed Bandits
- Classic Exploration Strategies
  - Epsilon Greedy
  - (Bayesian) Upper Confidence Bounds
  - Thomson Sampling
- Exploration in Deep RL
  - Count-based Exploration: Density Models, Hashing
  - Prediction-based Exploration:
    - Forward Dynamics
    - Random Networks
    - Physical Properties
  - Memory-based Exploration:
    - Episodic Memory
    - Direct Exploration
- **Summary and Outlook**

Summary

- We studied different families of algorithms and settings in this lecture:
  1. Multi-armed Bandits and their theoretical assumptions
  2. Challenges that arise from going from small MDPs to high-dimensional POMDPs
  3. We found different families of methods to guide exploration in Deep RL:
     - Count-based Exploration
     - Prediction-based Exploration
     - Memory-based Exploration

- Exploration is **really hot topic** in current RL research
  - Proving theoretical assumption and bound from (contextual) bandits on small MDPs, as well as
  - Exploration in Deep RL

References

- Lilian Weng: The Multi-Armed Bandit Problem and Its Solutions. lilianweng.github.io/lil-log, 2018. https://lilianweng.github.io/lil-log/2018/01/23/the-multi-armed-bandit-problem-and-its-solutions.html
- CS229 Supplemental Lecture notes. http://cs229.stanford.edu/extra-notes/hoeffding.pdf
- UCL Course by David Silver – Lecture 9: XX.
- Oliver Chapelle and Lihong Li: An empirical evaluation of Thompson Sampling. NIPS2011.
- Daniel Russo et al.: A Tutorial on Thompson Sampling. arXiv:1707.02038. 2017.
- https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/