Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

Reinforcement Learning

—

Lecture 12: Exploration in Deep RL

Christopher Mutschler

Exploration in Deep RL

Agenda

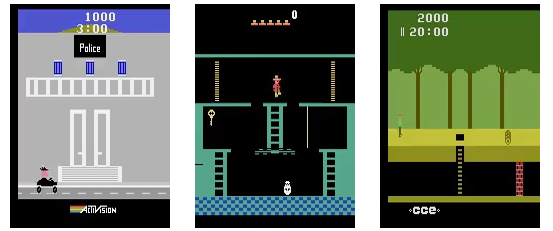
- Motivation, Problem Definition & Multi-Armed Bandits
- Classic Exploration Strategies
  - Epsilon Greedy
  - (Bayesian) Upper Confidence Bounds
  - Thomson Sampling
- **Exploration in Deep RL**
  - Count-based Exploration: Density Models, Hashing
  - Prediction-based Exploration:
    - Forward Dynamics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

Exploration in Deep RL

Key Exploration Problems in Deep RL

1. The “Hard-Exploration” Problem

- Exploration in environments with very sparse or even *deceptive* rewards
- Random exploration is prone to failure as it will rarely find successful states or obtain meaningful feedback from the environment
- Montezuma’s Revenge is one of such examples



problems with sticky actions

2. The Noisy-TV Problem

- Initially proposed by Burda et al.[1]: An agent seeks for novelty in the environment and finds a TV with uncontrollable & unpredictable output (e.g., Gaussian noise)
  → this will attract the agent’s attention forever!
- The agent obtains new rewards from the noisy TV consistently but fails to make any meaningful progress and becomes a “couch potato”

Agent in maze with TV

Agent in maze without TV

Images taken from https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/

[1] *Yuri Burda et al.: Exploration by Random Network Distillation. ICLR 2019.*

Exploration in Deep RL

Exploration in Deep RL

- But is it all so much different from what we studied with bandits?
- Recap: classes of exploration methods:
  - Optimistic exploration:
    - A new state is always a good state
    - We must estimate the state visitation frequencies or novelty
    - Typically realized by means of exploration bonuses
  - Thompson sampling style algorithms:
    - Learn distribution over Q-functions or policies
    - Sample and act according to sample
  - Information gain style algorithms
    - We reason about information gain from visiting new states
  - Entropy-loss & Noise-based algorithms:
    - Implicit exploration through induction of noise

We talked about this

Not our focus here

Exploration in Deep RL

- Let us revisit Upper Confidence Bounds:

$$a_t^{UCB} = \arg\max_{a \in \mathcal{A}} Q(a) + \underbrace{c \cdot \sqrt{\frac{2 \log t}{N_t(a)}}}_{\text{Exploration Bonus}}$$

- We can make use of several exploration bonus functions
  (don't worry about all the elements, most important is that it decreases with $N(a)$!)
- Open question: how can we make use of such methods in an MDP?
  - Idea: Count-based Exploration:
    - use $N(s, a)$ or $N(s)$, and
    - add an *exploration bonus!*

Exploration in Deep RL

Intrinsic Rewards as Exploration Bonuses

- Instead of $r(s,a)$ we provide $r^+(s,a) = r(s,a) + \mathcal{B}\big(N(s)\big)$

  decreases with $N(s)$

- We can give this to any model-free agent!
- A general formulation looks like this:

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

  - β is a hyperparameter that adjusts the balance between exploitation and exploration
  - r_t^e is called the extrinsic reward form the environment at time t
  - r_t^i is called the intrinsic reward, i.e., the exploration bonus at time t

- The intrinsic reward is/can be inspired intrinsic motivation[1] and we can transfer those findings to RL too:
  1. Discovery of novel states
  2. Improvement of the agent's knowledge about the environment

[1] *Pierre-Yves Oudeyer and Frederic Kaplan: How can we define intrinsic motivation? 8th Intl. Conf. Epigenetic Robotics*

Exploration in Deep RL

Agenda

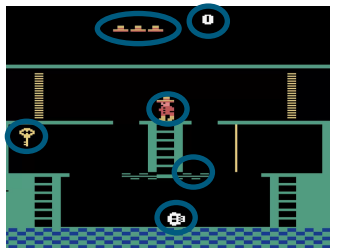
- Motivation, Problem Definition & Multi-Armed Bandits
- Classic Exploration Strategies
  - Epsilon Greedy
  - (Bayesian) Upper Confidence Bounds
  - Thomson Sampling
- **Exploration in Deep RL**
  - **Count-based Exploration: Density Models, Hashing**
  - Prediction-based Exploration:
    - Forward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

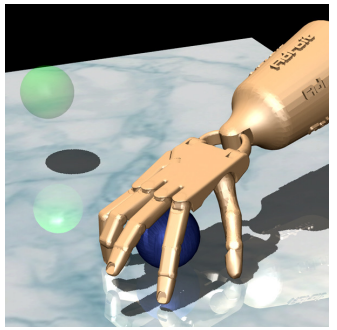
Exploration in Deep RL

Count-based Exploration

- We want novel states to surprise the agent
  → we need a metric that measures how novel a state appears to us
- Intuitive idea: count how many times we see a particular state & apply the bonus accordingly
  → Let $N_n(s)$ be the empirical count of visits of a state s in the sequence $s_{1:n}$.

- But wait, as we deal with **high-dimensional** or **continuous state spaces**, we still have 2 problems:
  1. Many states we will never see at all
  2. Many states we will never see again
  → So, what is a "count" after all?
  → Counting will become somehow "useless"…





- Side-note: we need a non-zero count for most states, even if we haven't seen them before

- But some states are more similar than others!
  → This might be useful to exploit!

Exploration in Deep RL

Count-based Exploration: Density Models

- Idea: Density Models
  1. Fit a density model $p(s;\theta)$ to approximate the frequency of visits
  2. Derive a pseudo count from the model

true density at time step T:

$$p(s) = \frac{N(s)}{n}$$

$\longrightarrow$

true density at time step T+1 after observing s:

$$p'(s) = \frac{N(s)+1}{n+1}$$

- Can we get $p(s;\theta)$ and $p(s;\theta')$ to satisfy the above?

Marc G. Bellemare et al.: Unifying Count-Based Exploration and Intrinsic Motivation. NIPS 2016.

Exploration in Deep RL

Count-based Exploration: Density Models

- Idea: Density Models
  1. Fit a density model $p(s;\theta)$ to approximate the frequency of visits
  2. Derive a pseudo count from the model

true density at time step T:

$$p(s;\theta) = \frac{\hat{N}(s)}{\hat{n}}$$

true density at time step T+1 after observing s:

$$p(s;\theta') = \frac{\hat{N}(s)+1}{\hat{n}+1}$$

- Can we get $p(s;\theta)$ and $p(s;\theta')$ to satisfy the above?
- Sure:
  1. Fit model $p(s;\theta)$ to all states $\mathcal{D}$ seen so far
  2. Take a step T and observe s_T
  3. Fit new model $p(s;\theta')$ to $\mathcal{D} \cup s_T$
  4. Use $p(s;\theta)$ and $p(s;\theta')$ to estimate $\hat{N}(s)$
  5. Set $r_i^+ = r_i + \mathcal{B}\left(\hat{N}(s)\right)$
  6. Repeat.

$\rightarrow \hat{N}(s) = \hat{n} \cdot p(s;\theta)$

$\rightarrow \hat{n} = \frac{\hat{N}(s)+1-p(s;\theta')}{p(s;\theta')}$

but how?

$\rightarrow$ solve linear system

$\rightarrow \hat{N}(s) = \frac{p(s;\theta)[1-p(s;\theta')]}{p(s;\theta')-p(s;\theta)}$

Marc G. Bellemare et al.: Unifying Count-Based Exploration and Intrinsic Motivation. NIPS 2016.

Exploration in Deep RL

Count-based Exploration: Density Models

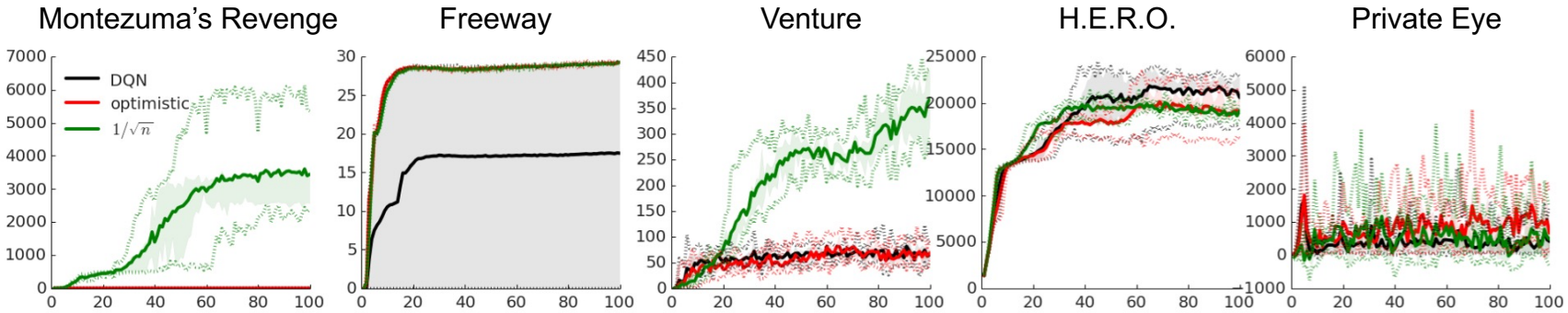
- **Open issue #1: What bonus $\mathcal{B}(\hat{N}(s))$ could we choose?**
  - Upper Confidence Bounds: $\mathcal{B}(\hat{N}(s)) = \sqrt{\frac{2 \log t}{\hat{N}(s)}}$
  - MBIE-EB[1,2]: $\mathcal{B}(\hat{N}(s)) = \sqrt{\frac{1}{\hat{N}(s)}}$
  - BEB[3]: $\mathcal{B}(\hat{N}(s)) = \frac{1}{1+\hat{N}(s)}$

- **Open issue #2: What density model could we choose?**
  - Note: we only need rough densities (no need for accuracy or normalization, and we do also not need to sample from it (such in GANs or VAEs!) – it only needs to get up for states that have higher density
  - Context Switching Trees (CTS)[2,4]
  - PixelCNN[5,6]
  - Gaussian Mixture Models (GMM)[7]
  - …

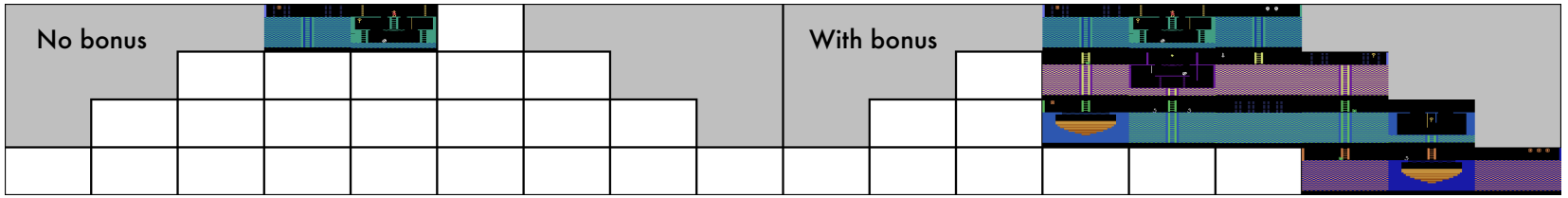
[1] *Strehl & Littman: An analysis of model-based Interval Estimation for Markov Decision Processes. 2008.*
[2] *Marc G. Bellemare et al.: Unifying Count-Based Exploration and Intrinsic Motivation. NIPS 2016.*
[3] *Kolter & Ng: Near-Bayesian Exploration in Polynomial Time. ICML 2009.*
[4] *Marc G. Bellemare et al.: Skip Context Tree Switching. ICML 2014.*
[5] *Georg Ostrovski et al.: Count-Based Exploration with Neural Density Models. ICML 2017.*
[6] *Arron van den Oord et al.: Conditional Image Generation with PixelCNN Decoders. NIPS 2016.*
[7] *Zhao & Tresp: Curiosity-Driven Experience Prioritization via Density Estimation. NIPS Deep RL Workshop. 2018.*

Exploration in Deep RL

Count-based Exploration: Density Models



Average training score with and without exploration bonus or optimistic initialization in 5 Atari 2600 games. Shaded areas denote inter-quartile range, dotted lines show min/max scores



"Known world" of a DQN agent trained for 50 million frames with (right) and without (left) count-based exploration bonuses, in Montezuma's Revenge.

Marc G. Bellemare et al.: Unifying Count-Based Exploration and Intrinsic Motivation. NIPS 2016.

Exploration in Deep RL

Counting after Hashing

- Alternative idea:
  - Map high-dimensional states into a k-bit hash code via $\phi(s)$ and count $N(\phi(s))$ instead of $N(s)$
  - → Shorter codes = more hash collisions
  - → Similar states = similar hashes?

- **Locality-Sensitive Hashing (LSH)[1]**
  - Hashing scheme that preserves the distancing information between data points
    → close vectors obtain similar hashes, distant vectors have different hashes
  - SimHash[2]:

$$\phi(s) = \mathrm{sgn}(Ag(s)) \in \{-1,1\}^k, \text{ where}$$

    - $A \in \mathbb{R}^{k \times D}$ is a matrix with each entry drawn from $\mathcal{N}(0,1)$, and
    - $g: \mathcal{S} \to \mathbb{R}^D$ is an optional preprocessing function
  - Larger k's lead to higher granularity and fewer collisions.

[1] *Haoran Tang et al.: Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. NIPS 2017.*
[2] *Moses Charikar: Similarity Estimation Techniques from Rounding Algorithms. STOC 2002.*

Exploration in Deep RL

Counting after Hashing

- Alternative idea:
  - Map high-dimensional states into a k-bit hash code via $\phi(s)$ and count $N(\phi(s))$ instead of $N(s)$
  - → Shorter codes = more hash collisions
  - → Similar states = similar hashes?

- **Locality-Sensitive Hashing (LSH)[1]**

Algorithm 1: Count-based exploration through static hashing, using SimHash

Define state preprocessor $g : \mathcal{S} \to \mathbb{R}^D$
(In case of SimHash) Initialize $A \in \mathbb{R}^{k \times D}$ with entries drawn i.i.d. from the standard Gaussian distribution $\mathcal{N}(0, 1)$
Initialize a hash table with values $n(\cdot) \equiv 0$
for each iteration j **do**
  Collect a set of state-action samples $\{(s_m, a_m)\}_{m=0}^{M}$ with policy π
  Compute hash codes through any LSH method, e.g., for SimHash, $\phi(s_m) = \text{sgn}(Ag(s_m))$
  Update the hash table counts $\forall m : 0 \leq m \leq M$ as $n(\phi(s_m)) \leftarrow n(\phi(s_m)) + 1$
  Update the policy π using rewards $\left\{ r(s_m, a_m) + \frac{\beta}{\sqrt{n(\phi(s_m))}} \right\}_{m=0}^{M}$ with any RL algorithm

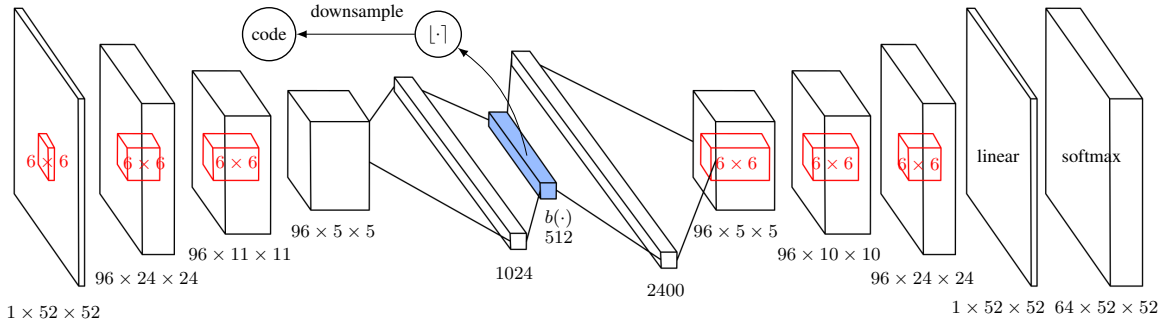
[1] Haoran Tang et al.: Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. NIPS 2017.

Exploration in Deep RL

Counting after Hashing

- Alternative idea:
  - Map high-dimensional states into a k-bit hash code via $\phi(s)$ and count $N(\phi(s))$ instead of $N(s)$
  - → Shorter codes = more hash collisions
  - → Similar states = similar hashes?

- **Learning Hash-Codes[1]**
  - SimHash works poorly on high-dimensional input with complex structure (such as images) as measuring the similarity on pixel-level fails to capture semantic similarity
  - Idea: learn a compression using an autoencoder
  - A special dense layer uses k sigmoid functions in the latent space to generate a binary activation map



[1] Haoran Tang et al.: Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. NIPS 2017.

Exploration in Deep RL

Counting after Hashing

- Alternative idea:
  - Map high-dimensional states into a k-bit hash code via $\phi(s)$ and count $N(\phi(s))$ instead of $N(s)$
  - → Shorter codes = more hash collisions
  - → Similar states = similar hashes?

- **Learning Hash-Codes[1]**
  - SimHash works poorly on high-dimensional input with complex structure (such as images) as measuring the similarity on pixel-level fails to capture semantic similarity
  - Idea: learn a compression using an autoencoder
  - A special dense layer uses k sigmoid functions in the latent space to generate a binary activation map

$$L\left(\{s_n\}_{n=1}^N\right) = -\frac{1}{N}\sum_{n=1}^{N}\Big[\underbrace{\log p(s_n)}_{\text{reconstruction loss}} - \underbrace{\tfrac{\lambda}{K}\textstyle\sum_{i=1}^{D}\min\left\{(1-b_i(s_n))^2, b_i(s_n)^2\right\}}_{\text{Sigmoid activation being closer to binary}}\Big],$$

[1] *Haoran Tang et al.: Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. NIPS 2017.*

Exploration in Deep RL

Prediction-based Exploration

- 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

Exploration in Deep RL

Prediction-based Exploration

- So far, we derived the bonus with respect to the *novelty* of states we encounter
  - The bonus correlates with the state visitation
  - We encourage the agent to look for states it did not see that often
- However, we also could interpret intrinsic motivation more widely in terms of *curiosity*
  - Obtaining knowledge about the environment
  - Familiarity with the environment dynamics, reward structure, …
- In RL, the idea of using a prediction model actually dates back to 1991[1]

https://www.youtube.com/watch?v=8vNxjwt2AqY

A Possibility for Implementing Curiosity and Boredom in Model-Building Neural Controllers

Jürgen Schmidhuber*
TUM

In J. A. Meyer and S. W. Wilson, editors, Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pages 222-227. MIT Press/Bradford Books, 1991.

Abstract

This paper introduces a framework for 'curious neural controllers' which employ an adaptive world model for goal directed on-line learning.

First an on-line reinforcement learning algorithm for autonomous 'animats' is described. The algorithm is based on two fully recurrent 'self-supervised' continually running networks which learn in parallel. One of the networks learns to represent a complete model of the environmental dynamics and is called the 'model network'. It provides complete 'credit assignment paths' into the past for the second network which controls the animats physical actions in a possibly reactive environment. The animats goal is to maximize cumulative reinforcement and minimize cumulative 'pain'.

The algorithm has properties which allow to implement something like *the desire to improve the model network's knowledge about the world.* This is related to *curiosity.* It is described how the particular algorithm (as well as similar model-building algorithms) may be augmented by dynamic *curiosity* and *boredom* in a natural manner. This may be done by introducing (delayed) reinforcement for actions that increase the model network's knowledge about the world. This in turn requires the model network *to model its own ignorance*, thus showing a rudimentary form of *self-introspective* behavior.

[1] *Jürgen Schmidhuber: A Possibility for Implementing Curiosity and Boredom in Model-Builing Neural Controllers. Intl. Conf. Simulation of Adaptive Behavior. 1991.*

Prediction-based Exploration

Predicting Models: Forward Dynamics

- Idea of the **forward dynamics prediction model**:
  - The agent learns a parameterized function f_θ such that:

$$f_\theta : (s_t, a_t) \rightarrow s_{t+1}$$

  - Derive a reward bonus based on the prediction error of the dynamics model

$$e(s_t, a_t) = \| f(s_t, a_t) - s_{t+1} \|_2^2$$

    - Large prediction error: high bonus (as we encountered something unusual/unknown)
    - Low prediction error: low bonus (as we have seen this coming)

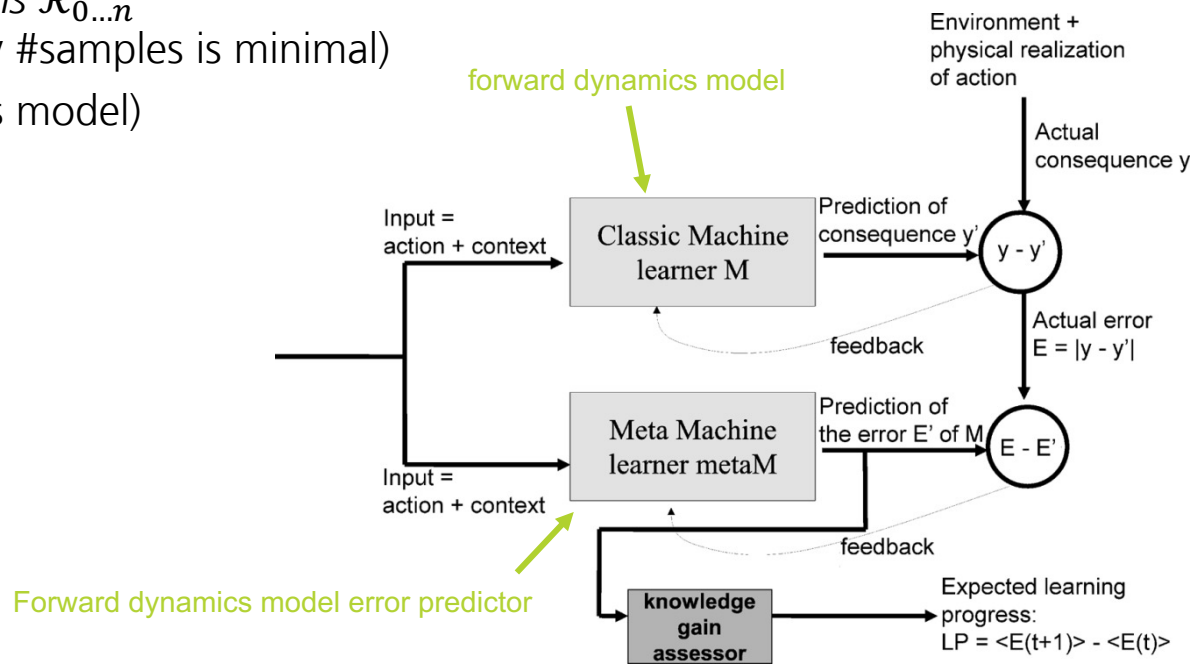
  - Our agent uses all the experience samples (s_t, a_t, s_{t+1}) collected so far and retrains its prediction model as it interacts with the environment

Prediction-based Exploration

Predicting Models: Forward Dynamics

Intelligent Adaptive Curiosity (IAC)[1]

- A memory M stores all the experiences encounted so far: $M = \{(s_t, a_t, s_{t+1})\}$
- Idea:
  - IAC (incrementally) splits the sensorimotor space into *regions* $\mathcal{R}_{0\ldots n}$ (criterion: the sum of variances of the two sets weighted by #samples is minimal)
  - Each region $\mathcal{R}_n$ has an associated *expert* (forward dynamics model) that is trained using the data from its region
- Learning:
  - With each new action the prediction error of the forward dynamics model is calculated using the MSE and put in a sliding window associated to that region
  - The decrease in the mean error rate is given as a reward to the agent



[1] *Pierre-Yves Oudeyer et al.: Intrinsic Motivation Systems for Autonomous Mental Development. Trans. Evol. Computation. 11(2). 2007.*

Prediction-based Exploration

Predicting Forward Dynamics

Deep Predictive Models[1]

- Predicting high-dimensional state spaces (images) can become very difficult
- Train a forward dynamics model in an encoding space ϕ (train an autoencoder):

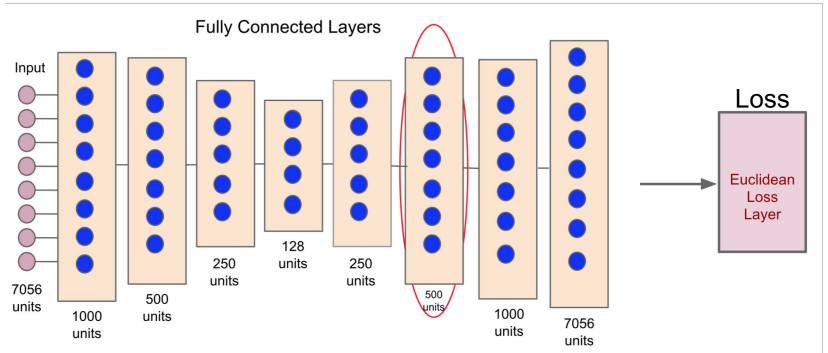
$$f_\phi: (\phi(s_t), a_t) \rightarrow \phi(s_{t+1})$$

- Normalize the prediction error at time T by the maximum error so far:

$$\bar{e}_t = \frac{e_t}{\max\limits_{i \leq t} e_i}$$

- Define the extrinsic reward accordingly (C is a decay parameter):

$$r_t^i = \left(\frac{e_t(s_t, a_t)}{t \cdot C} \right)$$



- The autoencoder can be trained upfront using images collected randomly or trained along with the policy and being updated steadily.

[1] *Stadie, Levine, Abbeel: Incentivizing Exploration in Reinforcement Learning with Deep Predictive Models. 2015.*

Prediction-based Exploration

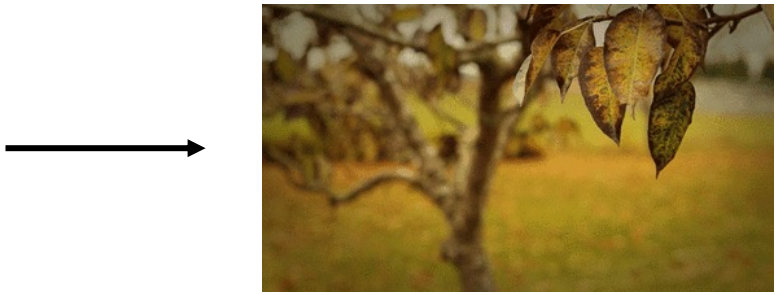
Predicting Forward Dynamics

Intrinsic Curiosity Module (ICM)[1]

- Instead of an autoencoder ICM trains the state space encoding $\phi(s_t)$ with a self-supervised *inverse dynamics* model
- Motivation:
  - Predicting s_{t+1} given (s_t, a_t) is not always easy as many factors in the environment cannot be controlled/affected by the agent
  - Popular example: imagine this tree with leaves
  - Such factors should not be part of the encoded state space as the agent should not base its decision based on these factors
- Solution: Learn an inverse dynamics model g:

$$g: \big(\phi(s_t), \phi(s_{t+1})\big) \rightarrow a_t$$

learn to explore in Level-1 explore faster in Level-2



- The feature space then only captures those changes in the environment related to actions that the agent takes, and ignores the rest

[1] *Deepak Pathak et al.: Curiosity-driven Exploration by Self-Supervised Prediction. ICML 2017.*

Prediction-based Exploration

Predicting Forward Dynamics

Intrinsic Curiosity Module (ICM)[1], given

- a forward model f with parameters θ_F
- an inverse dynamics model g with parameters θ_I
- and an observation (s_t, a_t, s_{t+1})

$$\hat{a}_t = g(\phi(s_t), \phi(s_{t+1}); \theta_I)$$

$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), a_t; \theta_F)$$

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

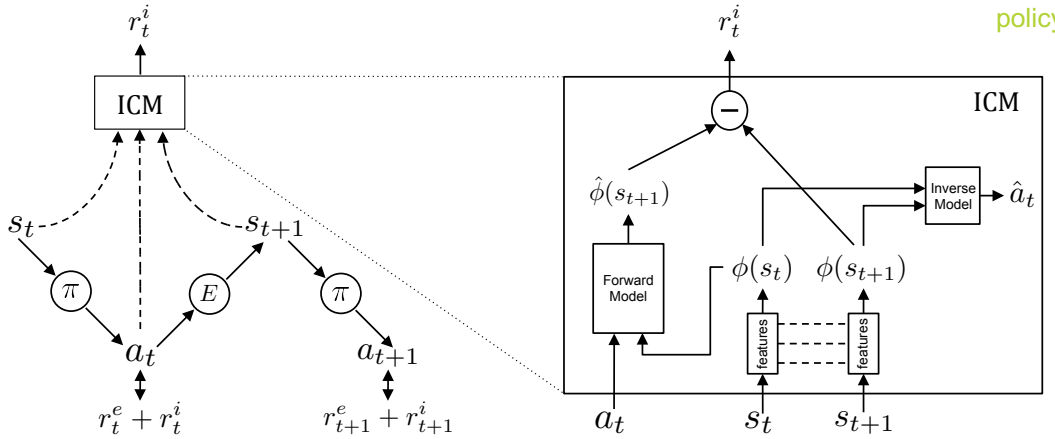
- The policy is jointly optimized as a whole:

if actions are discrete: softmax ML under multinomial distribution

$$\min_{\theta_P, \theta_I, \theta_F}\left[-\lambda \mathbb{E}_{\pi(s_t;\theta_P)}\left[\sum_t r_t\right] + (1-\beta)L_I + \beta L_F\right]$$

policy gradient loss

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



[1] *Deepak Pathak et al.: Curiosity-driven Exploration by Self-Supervised Prediction. ICML 2017.*

Prediction-based Exploration

Predicting Forward Dynamics

Intrinsic Curiosity Module (ICM)[1]: Large-Scale Study

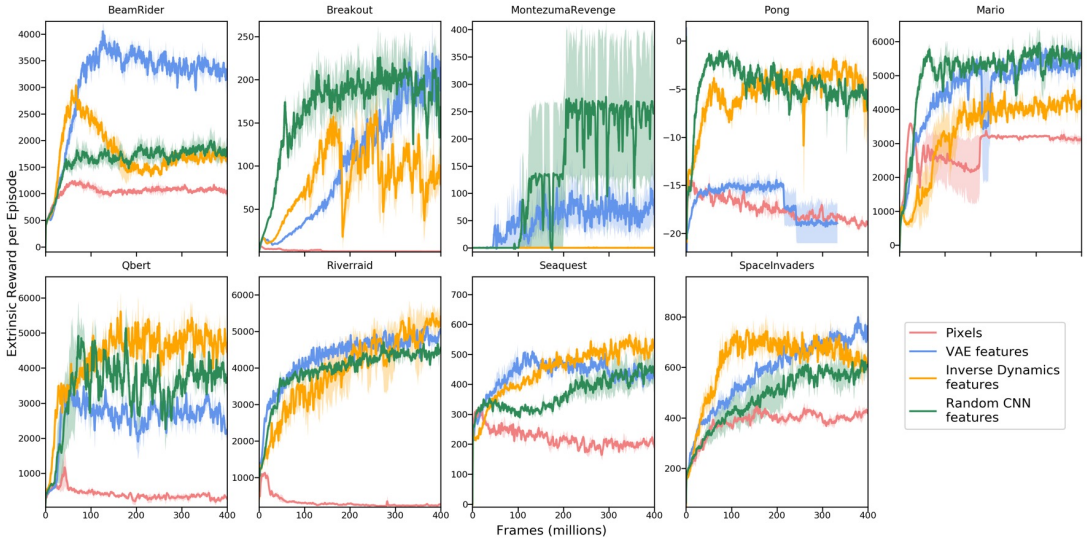
- Analyze the influence of curiosity by purely using the intrinsic reward:

$$r_t = r_t^i = \|f(s_t, a_t) - \phi(s_{t+1})\|_2^2$$

- What requirements must ϕ satisfy? When does it work best?

compact, sufficient, stable

	VAE	IDF	RF	Pixels
stable	✗	✗	✓	?
compact	✓	✓	?	✗
sufficient	✓	?	?	✓



→ Random CNN features are simple yet surprisingly strong!

→ However, IDF generalizes better (transfer knowledge from Super Mario Bros. Level 1 → Level 2)

Details on experiments:
- Robust learning: PPO
- Reward, Advantage, Observation Normalization
- 128 parallel actors
- Feature normalization
- Infinite horizon (avoid done flag)

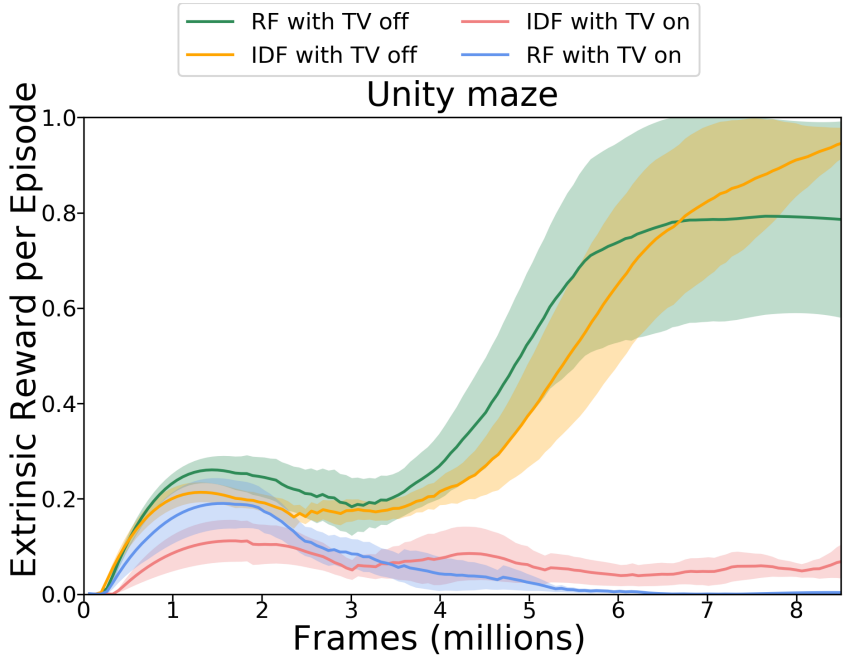
[1] *Yuri Burda et al.: Large-Scale Study of Curiosity-Driven Learning. ICLR 2019.*

Prediction-based Exploration

Predicting Forward Dynamics

Intrinsic Curiosity Module (ICM)[1]: Noisy TV

- Noisy TV drastically slows down the learning as extrinsic rewards are considerably lower in time
  → stochasticity of the environment poses problems
- Stochasticity is not always a problem, sometimes the agent escapes



[1] *Yuri Burda et al.: Large-Scale Study of Curiosity-Driven Learning. ICLR 2019.*

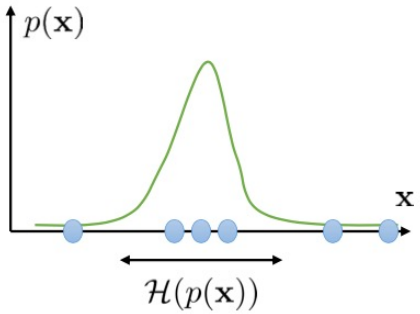
Prediction-based Exploration

Predicting Forward Dynamics

only for reference

Variational Information Maximizing Exploration (VIME)[1]:

- Some useful theoretic quantities first[2]:
  - $p(x)$: distribution (e.g., over observations x)
  - $\mathcal{H}\big(p(x)\big) = -\mathbb{E}_{x\sim p(x)}[\log p(x)]$: entropy (how broad is $p(x)$)
  - $\pi(s)$: state marginal distribution of policy π
  - $\mathcal{H}(\pi(s))$: state marginal entropy of policy π (quantifies *coverage*)
  - $\mathcal{I}(s_{t+1}; a_t) = \mathcal{H}(s_{t+1}) - \mathcal{H}(s_{t+1}|a_t)$: mutual information (empowerment)



[2]

- Idea of VINE:
  - Given: environment transition function $\mathcal{P}$ and forward prediction model $p(s_{t+1}|s_t, a_t; \theta), \theta \in \Theta$
  - We see a trajectory $\xi_t = \{s_1, a_1, \dots s_t\}$
  - We want to reduce entropy whenever we acquire new knowledge (see new states), i.e., maximize:

$$\sum_t H(\Theta|\xi_t, a_t) - H(\Theta|s_{t+1}, \xi_t, a_t) = \cdots = \mathbb{E}_{s_{t+1}\sim\mathcal{P}(\cdot|\xi_t, a_t)}[D_{KL}(p(\theta|\xi_t, a_t, s_{t+1})\|p(\theta|\xi_t))]$$

[2] See also Sergey Levine CS285, Lecture 14

[1] Rein Houthooft et al.: VIME: Variational Information Maximization Exploration. NIPS 2016.

Prediction-based Exploration

Predicting Forward Dynamics

Variational Information Maximizing Exploration (VIME)[1]:

- Unfortunately, computing the posterior $p(\theta|\xi_t, a_t, s_{t+1})$ is intractable:

$$p(\theta|\xi_t, a_t, s_{t+1}) = \frac{p(\theta|\xi_t, a_t)p(s_{t+1}|\xi_t, a_t; \theta)}{p(s_{t+1}|\xi_t, a_t)}$$

$$= \frac{p(\theta|\xi_t)p(s_{t+1}|\xi_t, a_t; \theta)}{p(s_{t+1}|\xi_t, a_t)}$$

action does not affect the belief

$$= \frac{p(\theta|\xi_t)p(s_{t+1}|\xi_t, a_t; \theta)}{\int_\Theta p(s_{t+1}|\xi_t, a_t; \theta)p(\theta|\xi_t)d\theta}$$

hard to compute directly

- As it is difficult to compute $p(\theta|\xi_t)$, we approximate it with an alternative distribution $q_\phi(\theta)$
- Variational Inference: using the variational lower bound: maximizing $q_\phi(\theta)$ is equivalent to
  - Maximizing $p(\xi_t|\theta)$ and minimizing $D_{KL}\left[q_{\phi_{t+1}}(\theta)\|p(\theta)\right]$

[1] *Rein Houthooft et al.: VIME: Variational Information Maximization Exploration. NIPS 2016.*

Prediction-based Exploration

Predicting Forward Dynamics

Variational Information Maximizing Exploration (VIME)[1]:

- Using the approximated distribution q, the intrinsic reward is

$$r_t^i = D_{KL}\left[q_{\phi_{t+1}}(\theta)||q_{\phi_t}(\theta)\right],$$

  where ϕ_{t+1} are the parameters of q after seeing a_t and s_{t+1}

- Normalize by division by moving median to KL-divs when using as a reward

- VIME uses a Bayesian neural network (BNN) to maintain a distribution over weights
  - The weights are modeled as Gaussians, and we can sample $\theta \sim q_\phi(\cdot)$
  - KL-Divergence is estimated from the 2nd-order Taylor-expansion using the FIM (which is easy to compute as the Gaussians result in a diagonal covariance matrix)

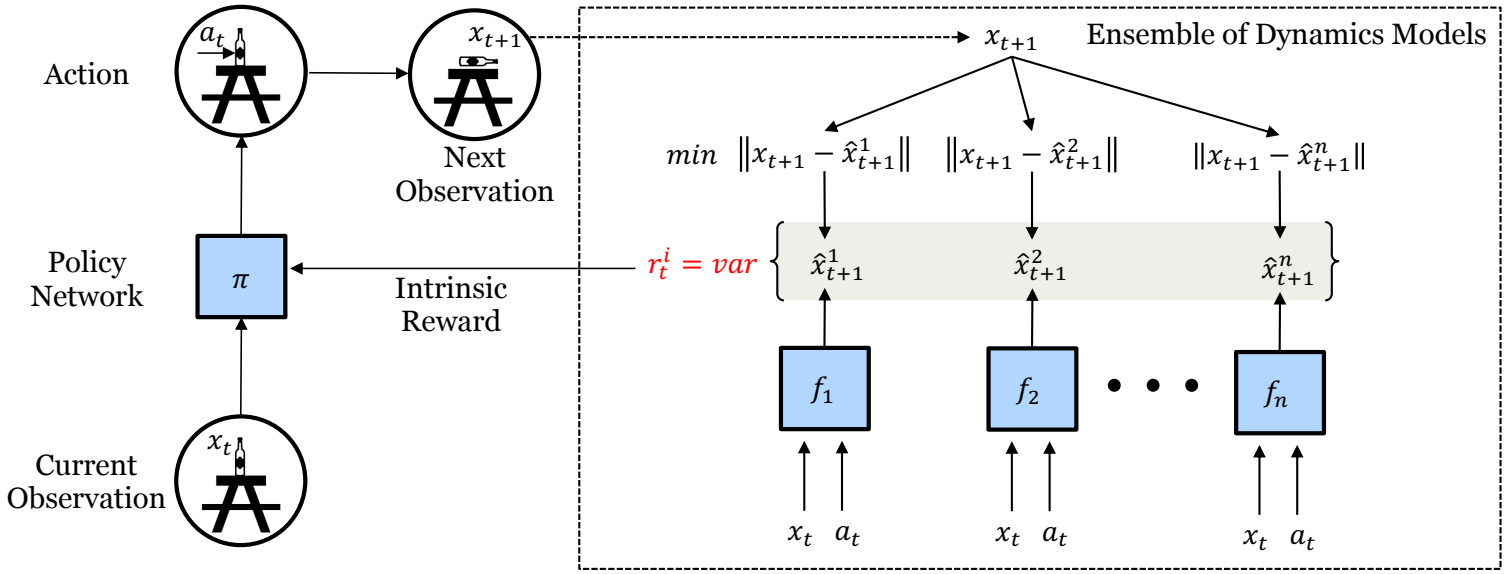
[1] *Rein Houthooft et al.: VIME: Variational Information Maximization Exploration. NIPS 2016.*

Prediction-based Exploration

Predicting Forward Dynamics

Self-Supervised Exploration via Disagreement[1]:

- Use an ensemble of prediction models and use their disagreement as bonus
- High disagreement → low confidence → needs more exploration
- r_t^i is differentiable → intrinsic reward can be directly optimized
- very efficient differentiable approach



[1] *Deepal Pathak et al.: Self-Supervised Exploration via Disagreement. ICML 2019.*

Prediction-based Exploration

Prediction Models: Random Networks

- From the noisy TV we can identify sources of prediction errors[1]

- The prediction error is high
  1. … where the predictor fails to generalize from previously seen examples.
     Novel experience then corresponds to high prediction error
  2. … because the prediction target is stochastic
  3. … because information necessary for the prediction is missing, or
     the model class of predictors is too limited to fit the complexity of the target function.

But this is our basic assumption – we need this

- How can we avoid the issues raised by (2) and (3)?

- What if the focus of our prediction task is not on environment dynamics at all?

- **Don’t care about the dynamics? Sounds crazy? Just wait…**

[1] *Yuri Burda et al.: Exploration by Random Network Distillation. ICLR 2019.*

Prediction-based Exploration

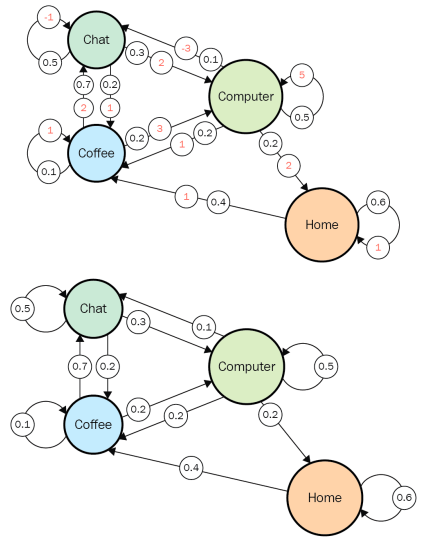
Prediction Models: Random Networks

Directed Outreaching Reinforcement Action-Selection (DORA)[1]

"If there's a place you gotta go - I'm the one you need to know."
(Map, Dora The Explorer)

- We use MDPs:
  - (1) the original MDP
  - (2) a copy of (1) with $r(s,a) = 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
- We learn Q on (2), and we call the entries "E"-values
  - So, we learn $E(s,a)$
  - "E"-values are initialized with 1
  - The model should predict all the E-values to be 0
- State-action pairs with *high* E-values lack information
- This is very similar to *visit counters*
- Given the predicted E-value $E(s_t, a_t)$, the exploration bonus is

$$r_i = \frac{1}{\sqrt{-\log E(s_t, a_t)}}$$



[1] *Leshem Choshen et al.: DORA The Explorer: Directed Outreaching Reinforcement Action-Selection. ICLR 2018.*

Prediction-based Exploration

Prediction Models: Random Networks

Directed Outreaching Reinforcement Action-Selection (DORA)[1]

Input: Stochastic action-selection rule f, learning rate α, Exploration discount factor γ_E
initialize $Q(s,a) = 0$, $E(s,a) = 1$;
foreach *episode* **do**
  init s;
  while *not terminated* **do**
    Choose $a = \arg\max_x \log f_Q(x|s) - \log\log_{1-\alpha} E(s,x)$;
    Observe transitions (s, a, r, s', a');
    $Q(s,a) \leftarrow (1-\alpha)\, Q(s,a) + \alpha\,(r + \gamma \max_x Q(s',x))$;
    $E(s,a) \leftarrow (1-\alpha)\, E(s,a) + \alpha\gamma_E E(s',a')$;
  end
end

[1] *Leshem Choshen et al.: DORA The Explorer: Directed Outreaching Reinforcement Action-Selection. ICLR 2018.*

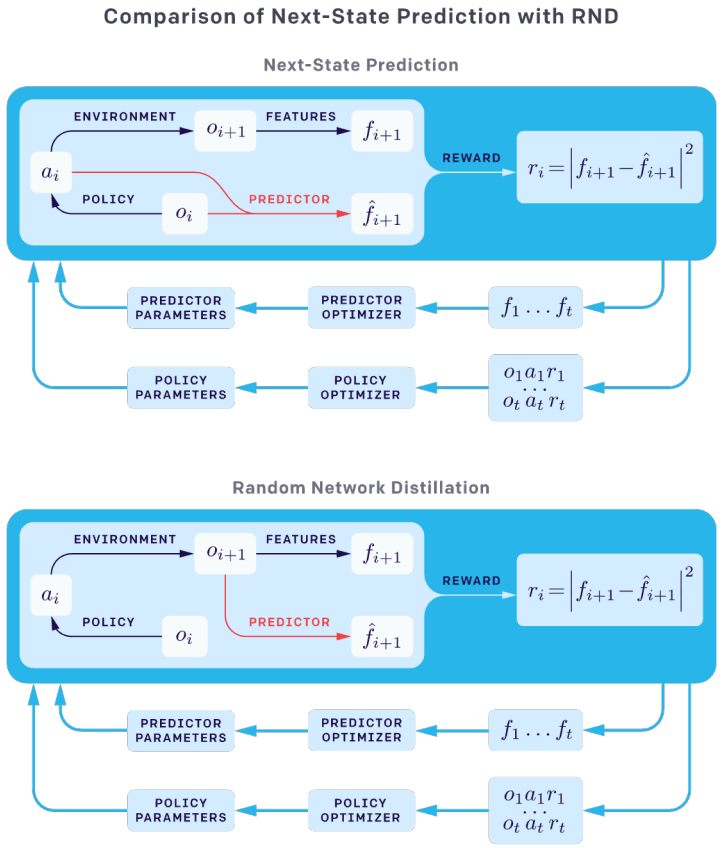
Prediction-based Exploration

Prediction Models: Random Networks

Random Network Distillation (RND)[1,2]

- Similar idea: predict something that is independent from the main task
- We use two neural networks:
  1. A randomly initialized but **fixed** neural network to transform a state into a feature space: $f(s_t)$
  2. A network $\hat{f}(s_t;\theta)$ that we train to predict the same features as the fixed network

  → We want $\hat{f}(s_t;\theta) = f(s_t)$
- Intuition: Similar states have similar features
  - And if we have already seen them, we should also have a lower error on predicting them!
- We use an exploration bonus: $r^i(s_t) = \left\|\hat{f}(s_t;\theta) - f(s_t)\right\|_2^2$



[1] *Yuri Burda et al.: Exploration by Random Network Distillation. ICLR 2019.*
[2] *https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/*

Prediction-based Exploration

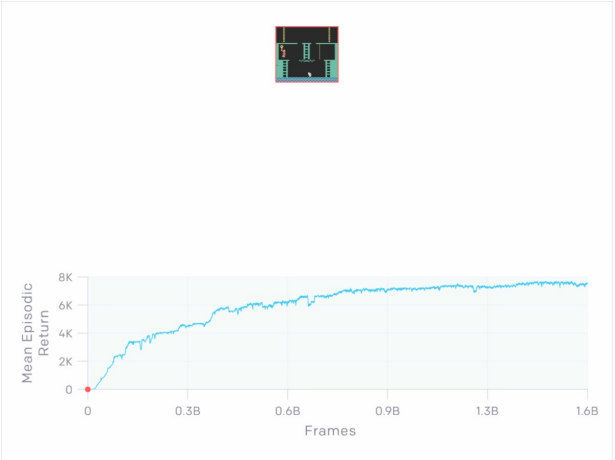
Random Network Distillation

Random Network Distillation (RND)

- Advantage of synthetic prediction problem:
  - The fixed network makes the prediction target deterministic (bypassing issue #2)
  - It is inside the class of functions that the predictor can represent (bypassing issue #3) if the predictor and the target network have the same architecture.

- Results:
  - RND works well for hard-exploration problems
    → maximizing RND bonus finds half of the rooms in Montezuma's Revenge
  - Normalization is important! The scale of the rewards is tricky to adjust given a random network as prediction target
    → Normalize by a running estimate of standard deviations of intrinsic return

  - Non-episodic settings work better, especially in cases without extrinsic rewards (the return is not truncated at *game over* and intrinsic return can spread across multiple episodes)



[1] *https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/*

Prediction-based Exploration

Random Network Distillation

- RND problem in episodic tasks[1]

2.3 Combining intrinsic and extrinsic returns

In preliminary experiments that used only intrinsic rewards, treating the problem as non-episodic resulted in better exploration. In that setting the return is not truncated at "game over". We argue that this is a natural way to do exploration in simulated environments, since the agent's intrinsic return should be related to all the novel states that it could find in the future, regardless of whether they all occur in one episode or are spread over several. It is also argued in (Burda et al., 2018) that using episodic intrinsic rewards can leak information about the task to the agent.

We also argue that this is closer to how humans explore games. For example let's say Alice is playing a videogame and is attempting a tricky maneuver to reach a suspected secret room. Because the maneuver is tricky the chance of a game over is high, but the payoff to Alice's curiosity will be high if she succeeds. If Alice is modelled as an episodic reinforcement learning agent, then her future return will be exactly zero if she gets a game over, which might make her overly risk averse. The real cost of a game over to Alice is the opportunity cost incurred by having to play through the game from the beginning (which is presumably less interesting to Alice having played the game for some time).

However using non-episodic returns for extrinsic rewards could be exploited by a strategy that finds a reward close to the beginning of the game, deliberately restarts the game by getting a game over, and repeats this in an endless cycle.

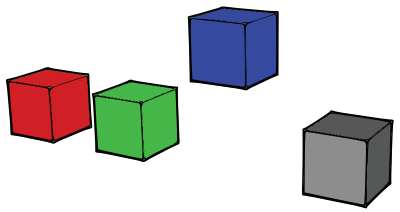
It is not obvious how to estimate the combined value of the non-episodic stream of intrinsic rewards i_t and the episodic stream of extrinsic rewards e_t. Our solution is to observe that the return is linear in

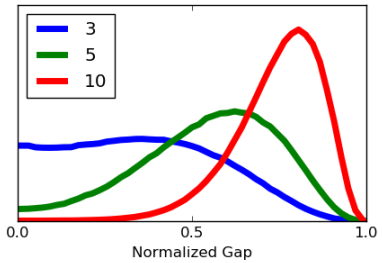
[1] *Yuri Burda et al.: Exploration by Random Network Distillation. ICLR 2019.*

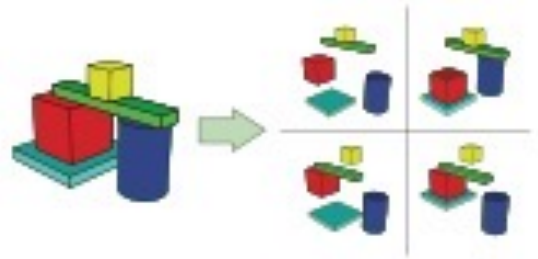
Prediction-based Exploration

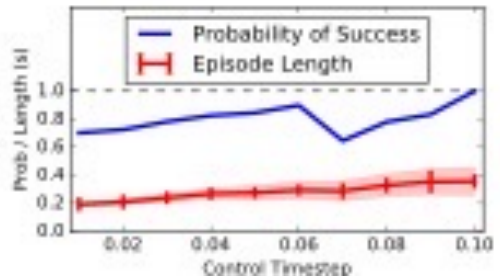
Physical Properties

- Motivation:
  - In many application (such as robotics) it helps the RL agent to explicitly understand and infer physical properties (such as mass, friction, etc.)
- Idea: Let an RL agent learn such properties by
  1. Exploration phase: letting the agent interact with the environment (without any specific task)
  2. Ask a question and give reward based on a labeling action (e.g.: *Which of the boxes is heaviest?*)
  - The agent must efficiently play around to figure out the physics and provide the correct answer
  - Exploration happens implicitly









[1] *Misha Denil et al.: Learning to Perform Physics Experiments via Deep Reinforcement Learning. ICLR 2017.*

Memory-based Exploration

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

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

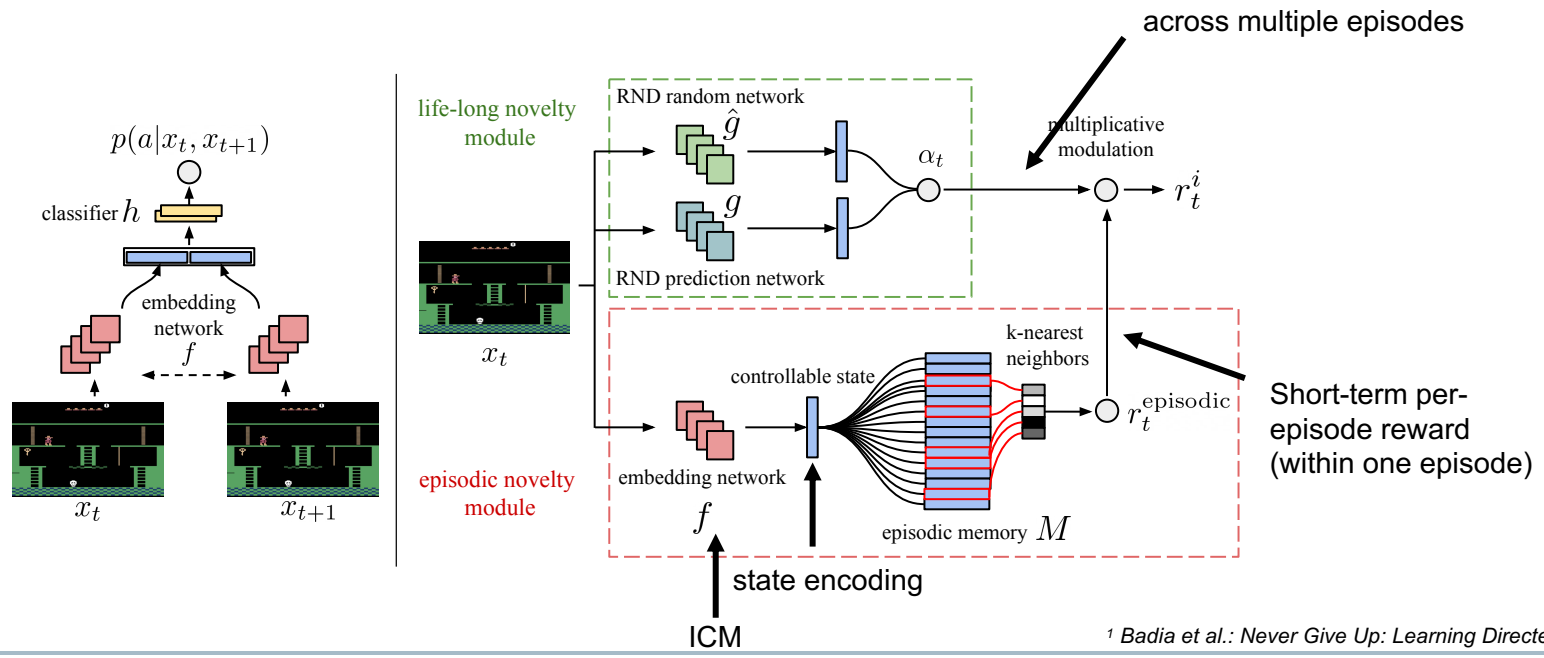
Idea of memory-based exploration:

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

Memory-based Exploration

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.*

Memory-based Exploration

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

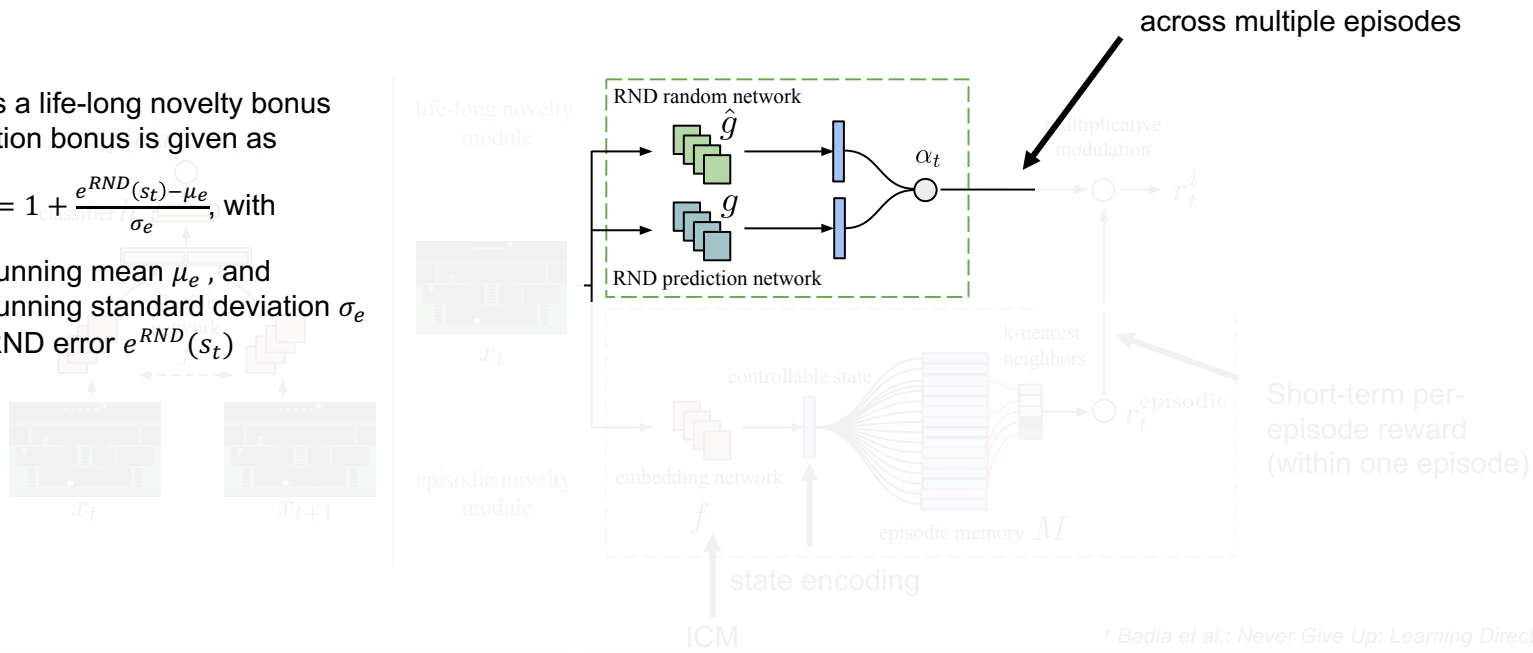
- So far: RND works great but suffers from episodic settings
- Idea: use two modules:
  1. RND as a lifelong novelty module, and
  2. an episodic novelty module for rapid in-episode adaptation

- 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.

Memory-based Exploration

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:

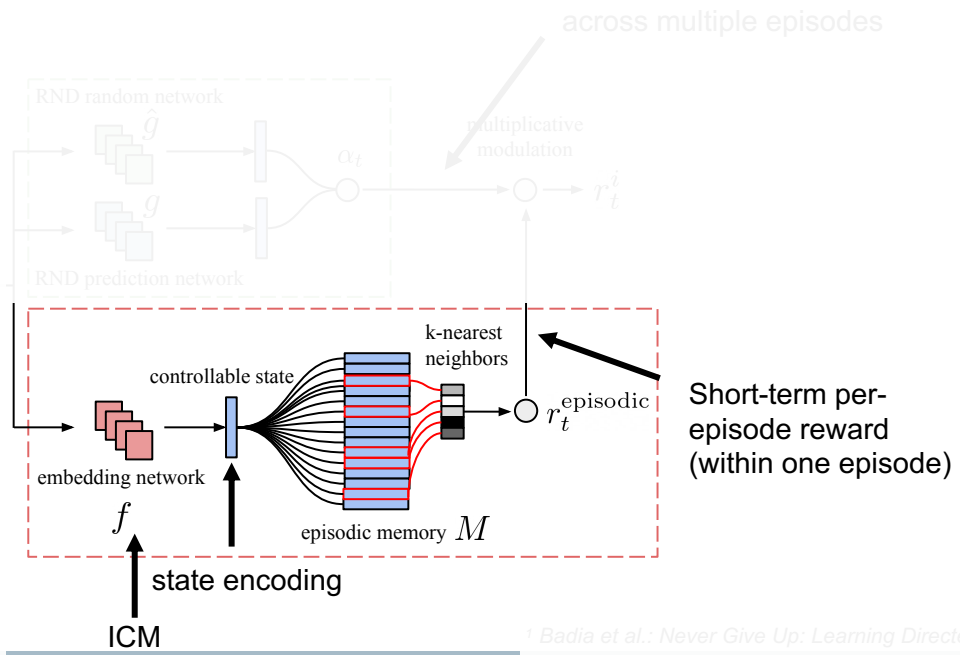
$$r_t^e = \frac{1}{\sqrt{\sum_{\phi_i \in N_k} K(\phi(x_t), \phi_i)} + c}, \text{ with}$$

  - 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)

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

Memory-based Exploration

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.

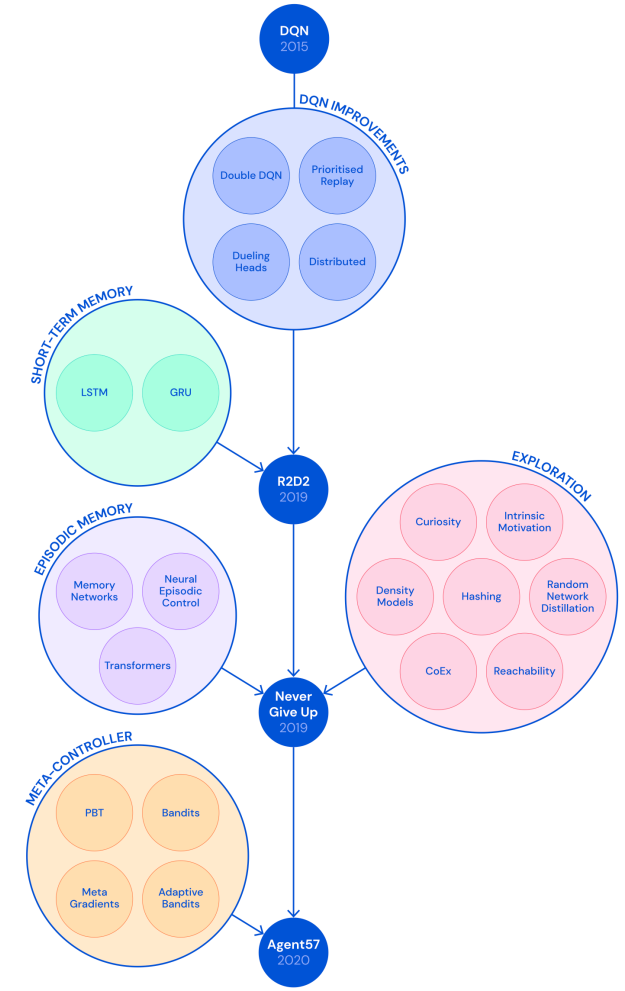
Memory-based Exploration

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



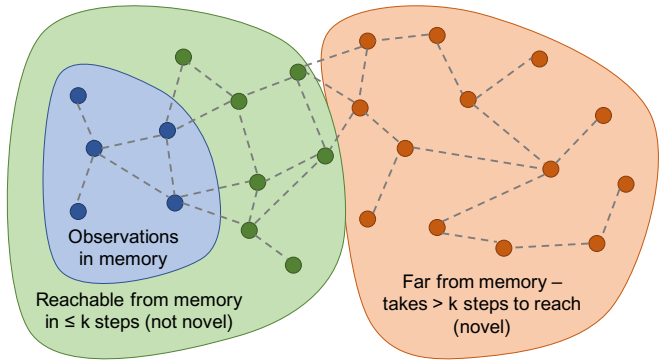
https://deepmind.com/blog/article/Agent57-Outperforming-the-human-Atari-benchmark

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

Memory-based Exploration

Episodic Curiosity through Reachability[1]

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



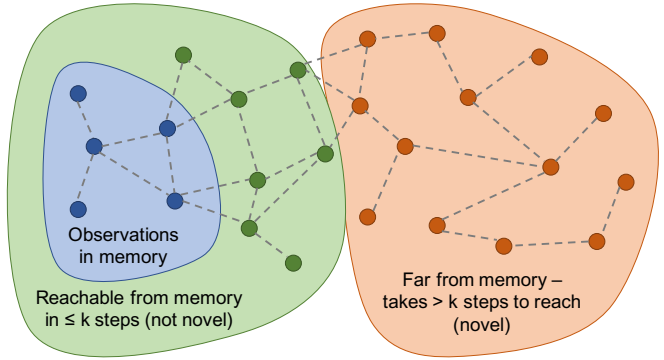
- There is a better thing than using the Euclidean distance*
- Episodic Curiosity (EC) module:
  - Measure the number of steps needed to transit between two states
  - The novelty than depends on the *reachability* between states
- General idea/steps:
  1. Clear episodic memory M on environment reset
  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*?**

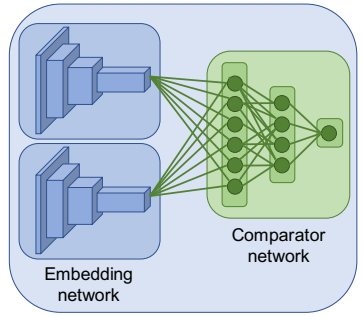
[1] Savinov et al.: Episodic Curiosity through Reachability, ICLR 2019.

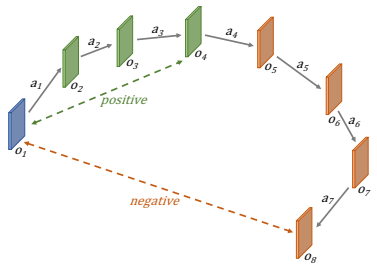
Memory-based Exploration

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: $R(o_i, o_j) = C\left(E(o_i), E(o_j)\right)$





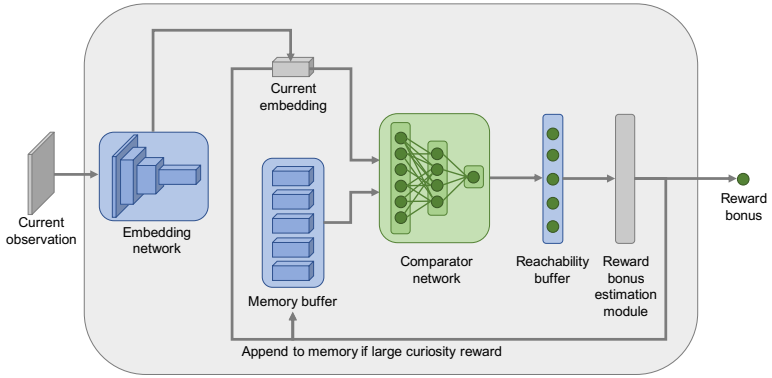


[1] Savinov et al.: Episodic Curiosity through Reachability. ICLR 2019.

Memory-based Exploration

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 $M = \langle e_1, \dots, e_{|M|} \rangle$ via C
   → fills the reachability buffer with values
   $$c_i = C(e_i, e), \quad i = 1, \dots, M.$$
3. Compute the similarity score between e and the memory buffer M as $C(M, e) = F(c_1, \dots, c_{|M|}) \in [0,1]$, 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 $b = B(M, e) = \alpha(\beta - C(M, e))$
   - α tunes scale of task rewards
   - β defines the sign of the reward (0.5 works well for fixed-duration episodes)

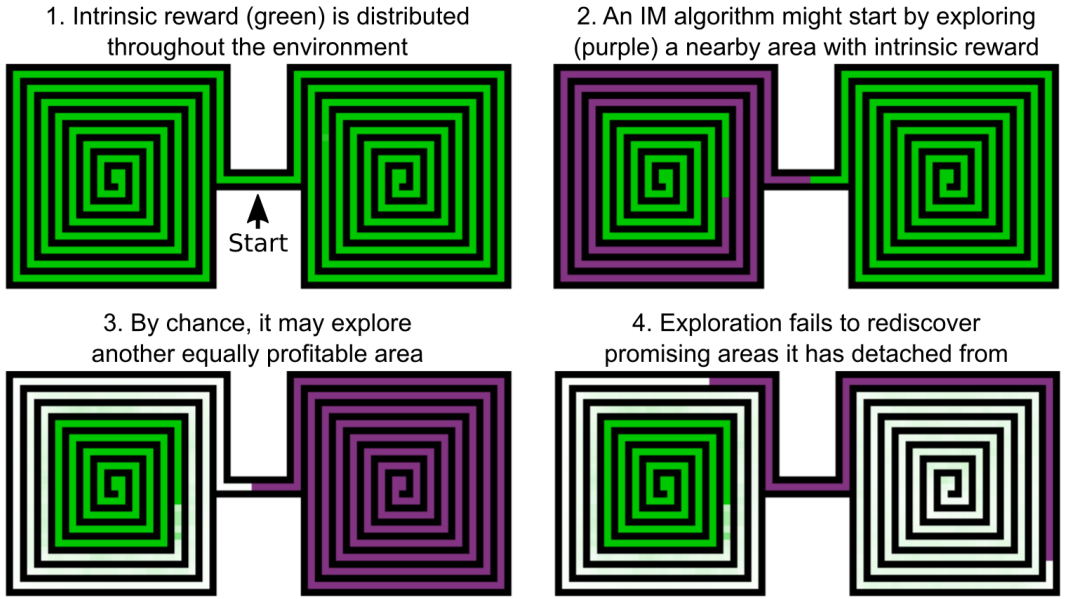
[1] Savinov et al.: Episodic Curiosity through Reachability, ICLR 2019.

Memory-based Exploration

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



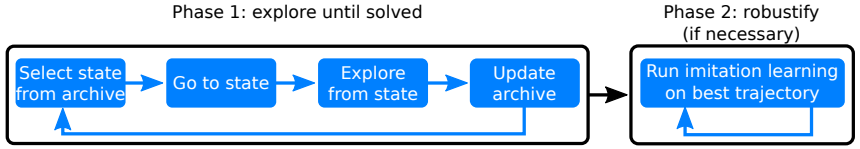
→ *with naïve perturbations in complex environments this becomes highly unlikely*

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.*

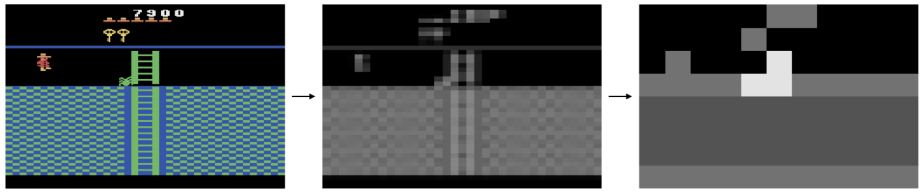
Memory-based Exploration

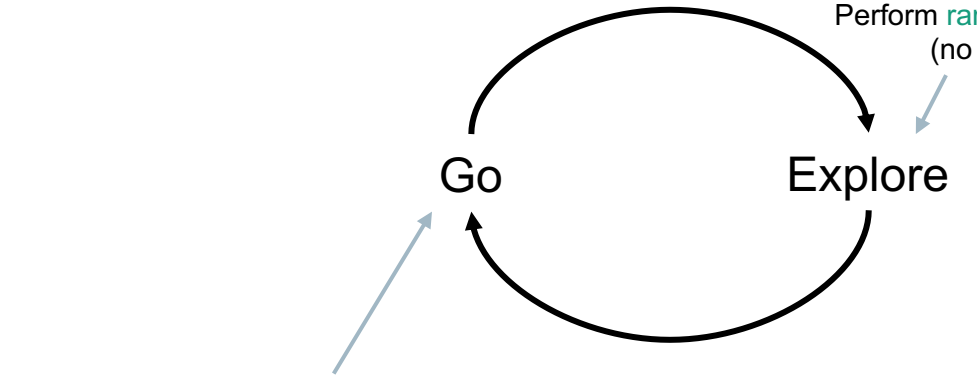
Direct Exploration: Go-Explore



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)

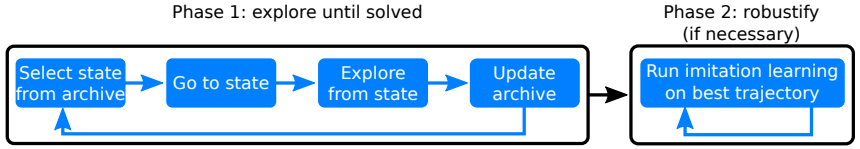




[1] *Adrian Ecoffet et al.: Go-Explore: a new Approach for Hard-Exploration Problems. 2020.*

Memory-based Exploration

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.*

Memory-based Exploration

Direct Exploration: Go-Explore

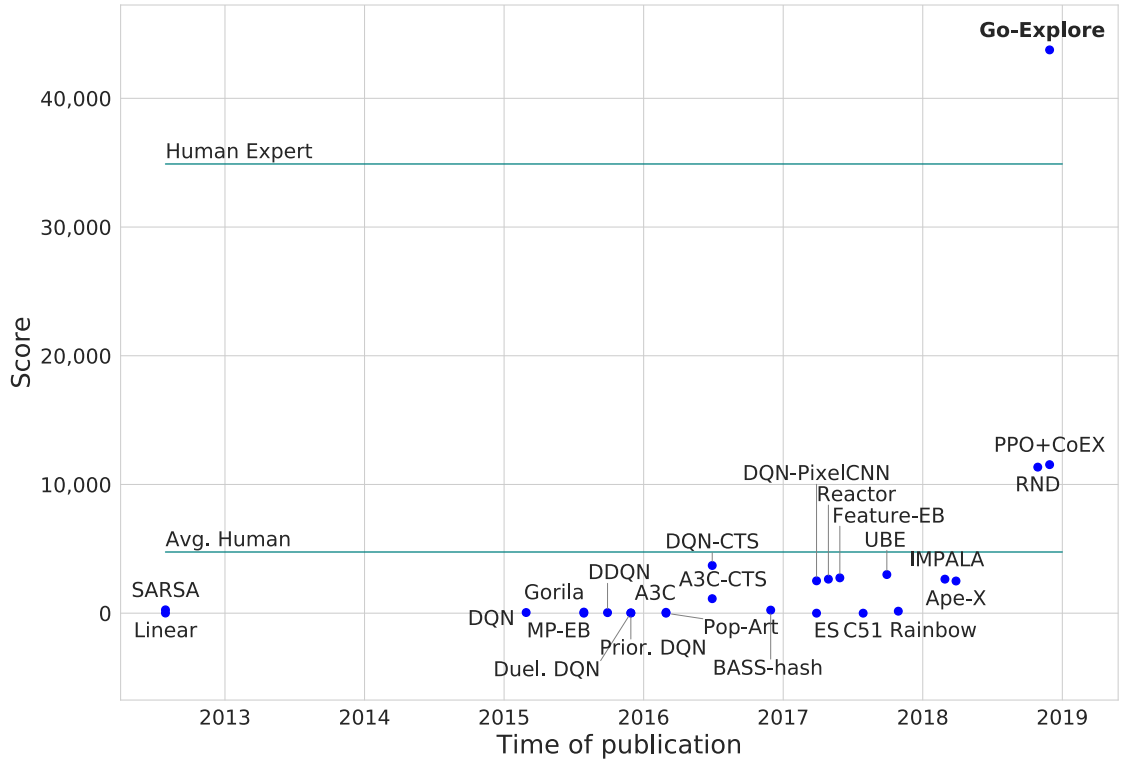


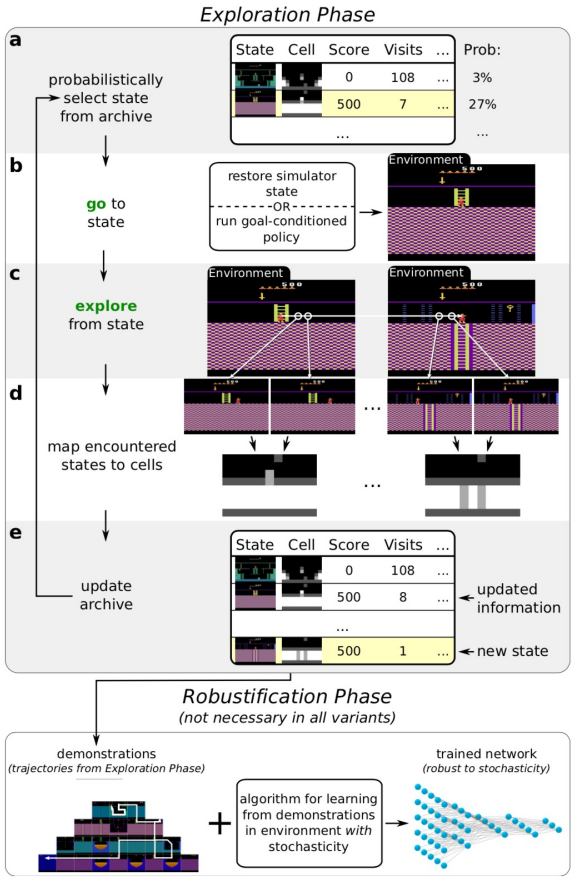
Figure 6: **History of progress on Montezuma's Revenge vs. the version of Go-Explore that does not harness domain knowledge.** Go-Explore significantly improves on the prior state of the art.

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

Memory-based Exploration

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 Rewards"[2]:
  - 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.*

Memory-based Exploration

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**

Exploration in Deep RL

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

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/