

# Reinforcement Learning

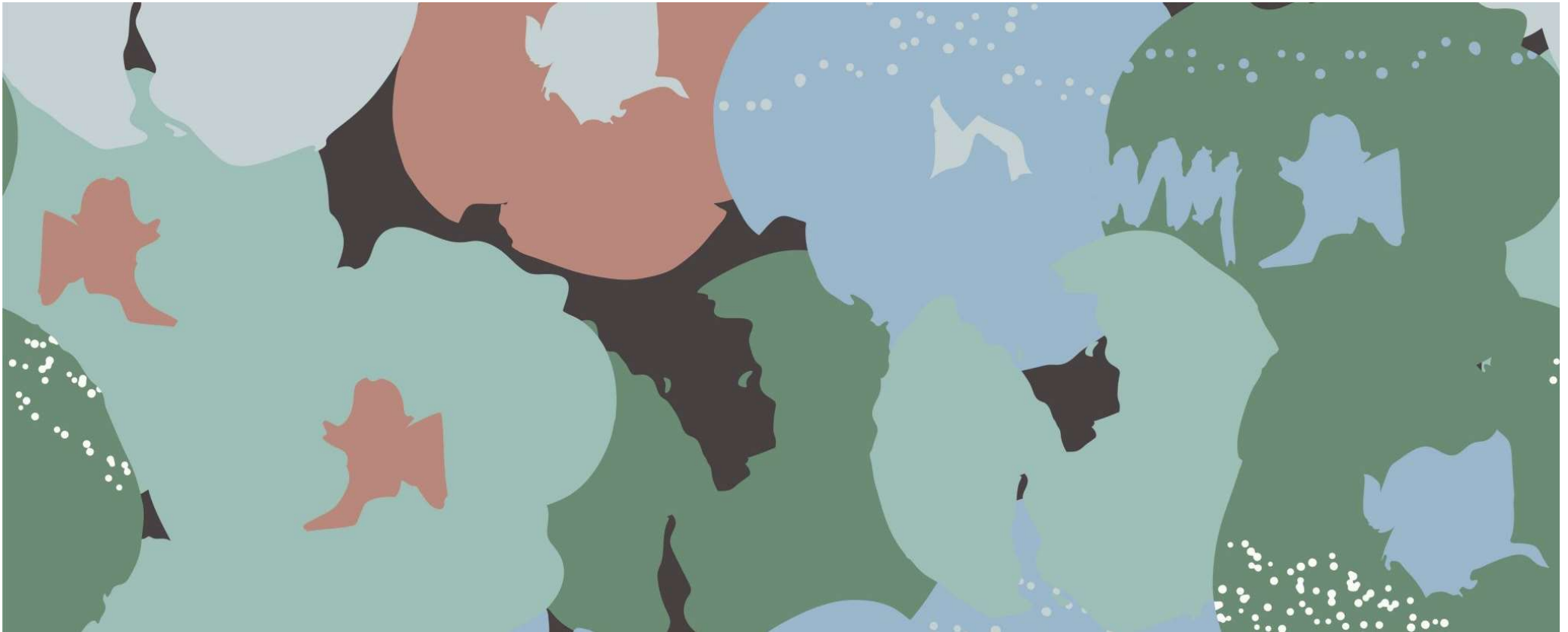
---

## Exercise 10: RND/ICM

15.07.2025

Alexander Mattick

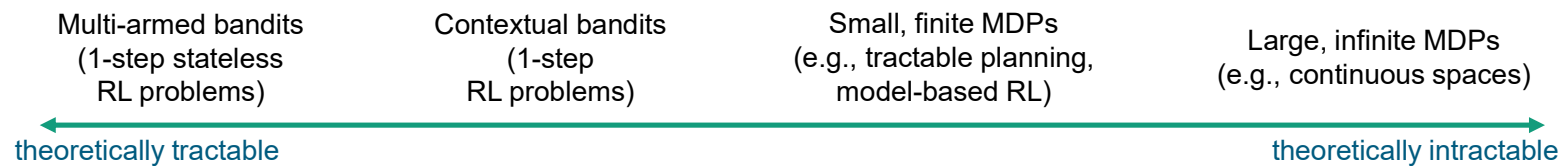
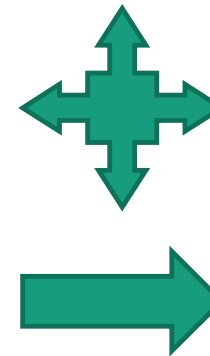
# Exploration vs. Exploitation



# Exploration vs. Exploitation

## Why and How

- Both definitions stem from the same problem:
  - Exploration:** do things you haven't done before (in the hopes of getting even higher reward)  
→ increase knowledge
  - Exploitation:** do what you know to yield highest reward  
→ maximize performance based on knowledge

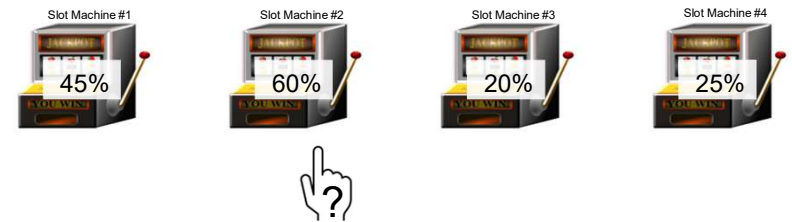


(illustration adapted from Sergey Levine's CS285 class from UC Berkeley)

# Exploration vs. Exploitation

## Multi-Armed Bandits and Regret

- The multi-armed-bandit problem is a classic problem used to study the exploration vs. exploitation dilemma
- Imagine you are in a casino with multiple slot machines, each configured with an unknown reward probability:



- Under the assumption of an infinite number of trials:  
 → **What is the best strategy to achieve highest long-term rewards?**

- Our loss function is the total regret we might have by not select the optimal action up to the time step  $T$ :

$$\mathcal{L}_T = \mathbb{E} \left[ \sum_{t=1}^T (\theta^* - Q(a_t)) \right] = \sum_{a \in \mathcal{A}} N_T(a) \Delta_a$$

Diagram annotations for the equation:
 

- An upward arrow from  $\theta^*$  points to the text "what we should have been doing".
- A downward arrow from  $Q(a_t)$  points to the text "what we did".
- An orange arrow from  $\Delta_a$  points to the text "per-action regret".
- A blue arrow from  $N_T(a)$  points to the text "action-selection counter".

## Exploration vs. Exploitation

Straightforward but usually bad: Greedy or  $\epsilon$ -greedy

- Greedy may select a suboptimal action forever  
→ Greedy has hence linear expected total regret
- $\epsilon$ -greedy continues to explore forever
  - with probability  $1 - \epsilon$  it selects  $a = \arg \max_{a \in \mathcal{A}} Q_T(a)$
  - with probability  $\epsilon$  it selects a random action
- Will hence continue to select all suboptimal actions with (at least) a probability of  $\frac{\epsilon}{|\mathcal{A}|}$   
→  $\epsilon$ -greedy, with a constant  $\epsilon$  has a linear expected total regret
- **Option #1: decrease  $\epsilon$  over course of training might work**
  - It is not easy to tune the parameters
- **Option #2: be optimistic with options of high uncertainty**
  - Prefer actions for which you do not have a confident value estimation yet  
→ Those have a great potential to be high-rewarding!
  - This idea is called **Upper Confidence Bounds**

# Exploration vs. Exploitation

## Upper Confidence Bounds (UCB1)

- Idea: estimate an upper confidence  $U_t(a)$  for each action value, such that with a high probability we satisfy

$$Q(a) \leq \hat{Q}_t(a) + U_t(a)$$

Large  $N_t(a) \rightarrow$  small bound  $U_t(a)$  (estimated value is *certain/accurate*)

- Next, we select the action that maximizes the upper confidence bound:

$$a_t^{UCB} = \arg \max_{a \in \mathcal{A}} [Q_t(a) + U_t(a)]$$

Small  $N_t(a) \rightarrow$  large bound  $U_t(a)$  (estimated value is *uncertain*)

- The vanilla **UCB1** algorithm uses  $p = t^{-4}$ :

$$U_t(a) = \sqrt{\frac{2 \log t}{N_t(a)}} \quad \text{and} \quad a_t^{UCB} = \arg \max_{a \in \mathcal{A}} Q(a) + \sqrt{\frac{2 \log t}{N_t(a)}}$$

Derived from Hoeffding's Inequality:  
 $P(\mathbb{E}[X] \geq \bar{X}_t + u) \leq e^{-2tu^2}$

- This ensures that we always keep exploring
- But we select the optimal action much more often as  $t \rightarrow \infty$

# Exploration vs. Exploitation

## Probability Matching via Thompson Sampling

We can also try the idea of directly sampling the action

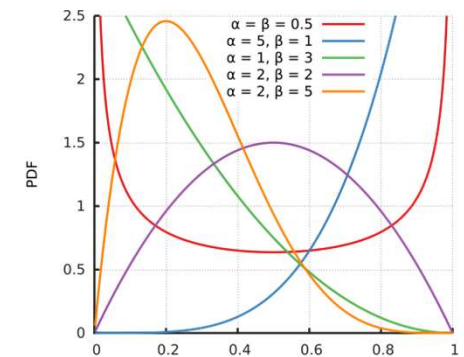
- Select action  $a$  according to probability that  $a$  is the optimal action (given the history of everything we observed so far):

$$\begin{aligned}\pi_t(a|h_t) &= P[Q(a) > Q(a'), \forall a' \neq a | h_t] \\ &= \mathbb{E}_{r|h_t} \left[ \mathbb{I} \left( a = \arg \max_{a \in \mathcal{A}} Q(a) \right) \right]\end{aligned}$$

Probability matching via Thompson Sampling:

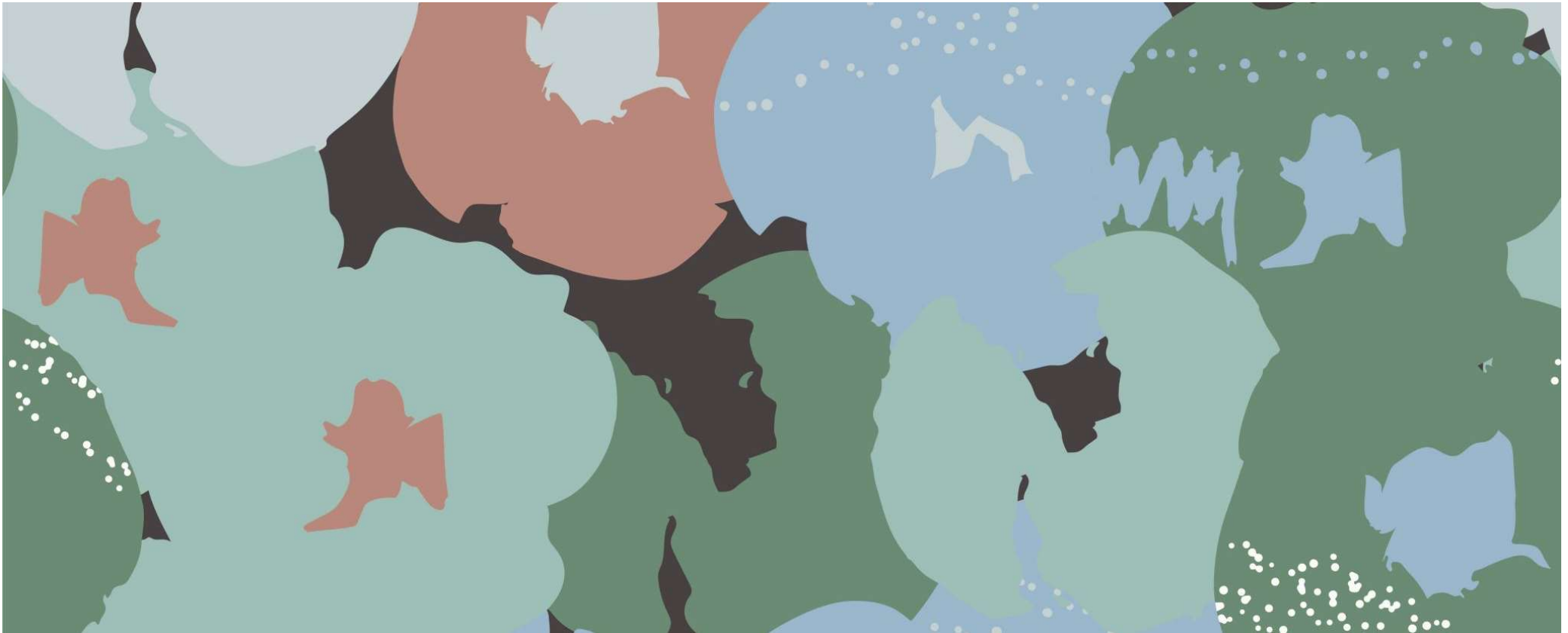
- Assume  $Q(a)$  follows a Beta distribution for the Bernoulli bandit
  - As  $Q(a)$  is the success probability of  $\theta$
  - Beta( $\alpha, \beta$ ) is within  $[0,1]$ , and  $\alpha$  and  $\beta$  relate to the counts of success/failure
- Initialize prior (e.g.,  $\alpha = \beta = 1$  or something different/what we think it is)
- At each time step  $t$  we sample an expected reward  $\hat{Q}(a)$  from the prior Beta( $\alpha_i, \beta_i$ ) for every action
  - We select and execute the best action among the samples:  $a_t^{TS} = \arg \max_{a \in \mathcal{A}} \hat{Q}(a)$
- With the newly observed experience we update the Beta distribution:

$$\begin{aligned}\alpha_i &\leftarrow \alpha_i + r_i \mathbb{I}[a_t^{TS} = a_i] \\ \beta_i &\leftarrow \beta_i + (1 - r_i) \mathbb{I}[a_t^{TS} = a_i]\end{aligned}$$



# Exercise Sheet 10

## Bandits



# Prediction-based Exploration in Deep RL

ICM and RND



# Exploration in Deep RL

## Intrinsic Rewards as Exploration Bonuses

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

↑  
decreases with  $N(s)$

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

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

- $\beta$  is a hyperparameter that adjusts the balance between exploitation and exploration
  - $r_t^e$  is called the extrinsic reward from 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<sup>1</sup> and we can transfer those findings to RL too:
    1. Discovery of novel states
    2. Improvement of the agent's knowledge about the environment

<sup>1</sup> Pierre-Yves Oudeyer and Frederic Kaplan: How can we define intrinsic motivation? 8th Intl. Conf. Epigenetic Robotics

# Prediction-based Exploration

## Predicting Models: Forward Dynamics

---

- Idea of the **forward dynamics prediction model**:

- The agent learns a parameterized function  $f_\theta$  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 Forward Dynamics

### Deep Predictive Models<sup>1</sup>

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

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

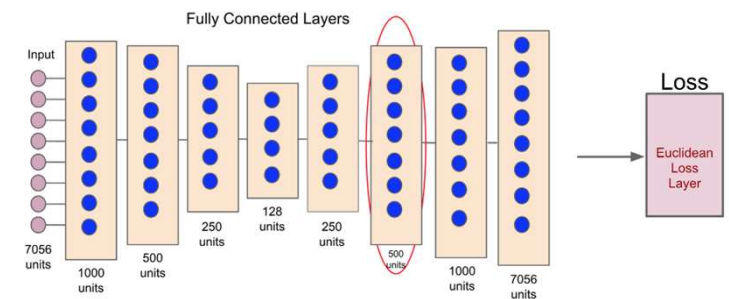
- Normalize the prediction error at time  $T$  by the maximum error so far:

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

- 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.



<sup>1</sup> Stadie, Levine, Abbeel: Incentivizing Exploration in Reinforcement Learning with Deep Predictive Models. 2015.

# Prediction-based Exploration

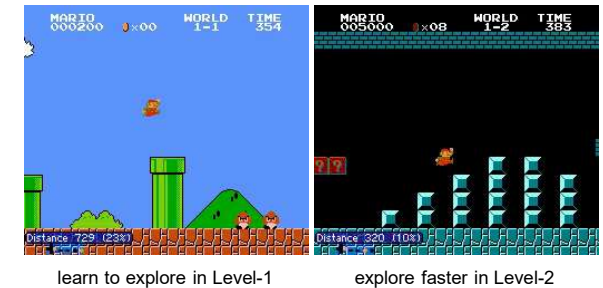
## Predicting Forward Dynamics

### Intrinsic Curiosity Module (ICM)<sup>1</sup>

- 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  $g$ :

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

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



<sup>1</sup> Deepak Pathak et al.: Curiosity-driven Exploration by Self-Supervised Prediction. ICML 2017.

# Prediction-based Exploration

## Predicting Forward Dynamics

### Intrinsic Curiosity Module (ICM)<sup>1</sup>, given

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

- The policy is jointly optimized as a whole:



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

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

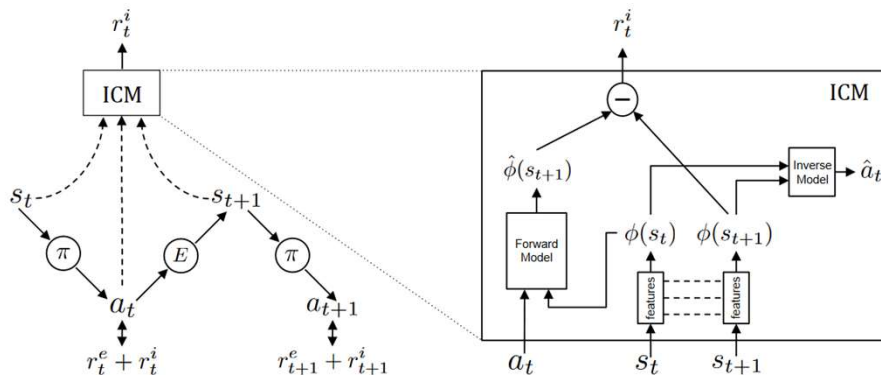
$$r_t^i = \frac{\eta}{2} \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$

if actions are discrete: softmax ML  
under multinomial distribution

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

policy gradient loss

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



<sup>1</sup> Deepak Pathak et al.: Curiosity-driven Exploration by Self-Supervised Prediction. ICML 2017.

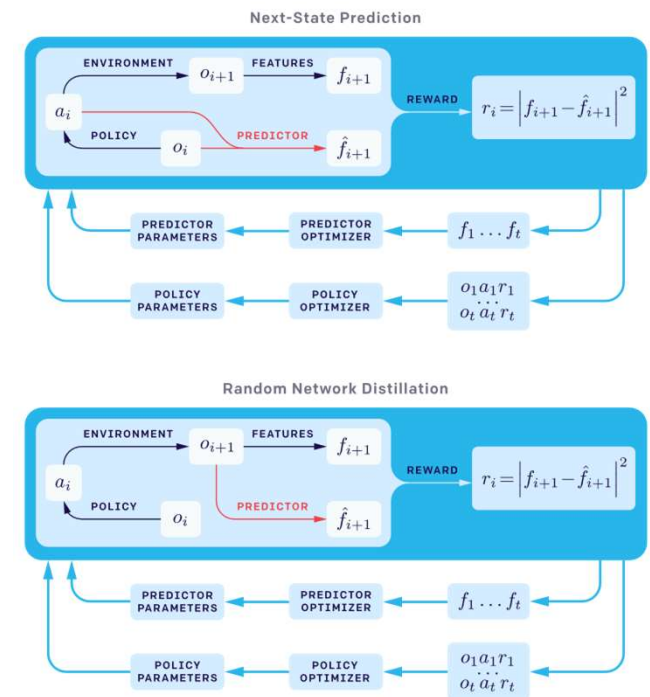
# Prediction-based Exploration

## Prediction Models: Random Networks

### Random Network Distillation (RND)<sup>1,2</sup>

- 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:  $r^i(s_t) = \|\hat{f}(s_t; \theta) - f(s_t)\|_2^2$

Comparison of Next-State Prediction with RND



<sup>1</sup> Yuri Burda et al.: *Exploration by Random Network Distillation*. ICLR 2019.

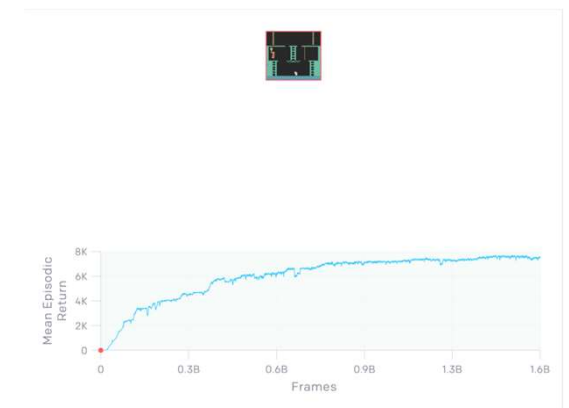
<sup>2</sup> <https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/>

# 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)



<sup>1</sup> <https://openai.com/blog/reinforcement-learning-with-prediction-based-rewards/>

# Exercise Sheet 11

## ICM and RND

---





Fraunhofer-Institut für Integrierte  
Schaltungen IIS

**Thank you for your attention!**