# Decoupling Exploration and Exploitation for Meta-RL (DREAM):

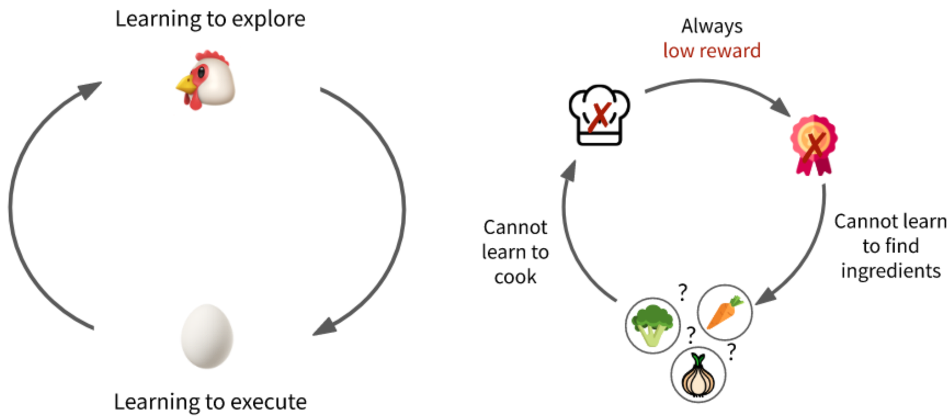
`https://arxiv.org/abs/2008.02790`

*Paper Summary Notes*

# 1 Main Ideas, High Level Assumptions

**Central Question of the Paper:** *How can we enable structured exploration in meta-rl, when adapting to a new task, without sacrificing policy quality?*

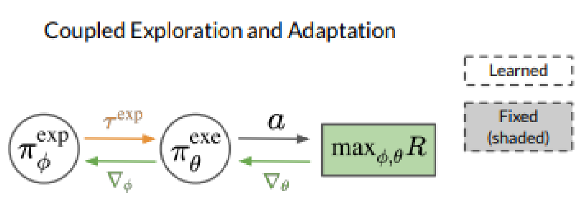
## 1.1 Exploration Challenge in Meta-RL




The coupling problem. What came first: the chicken (good exploration) or the egg (good execution)?

- common meta-rl approach trains RNN to maximize test set returns on each meta-training MDP, such that it can quickly adapt. Optimizing for exploration and exploitation jointly would work in principle, but in practice, this end-to-end optimization is difficult, and leads to local minima
- *chicken and egg problem*: learning what to explore requires knowing what information is crucial for solving the task, but learning to solve the task requires alrady agtherting this information via good exploation
  - ex: exploring to find cooking ingredients only helps robot prepare a meal if it already knows how to cook, but the robot can only learn to cook if it knows how to find the ingredients

### 1.1.1 Coupled Exploration and Exploitation

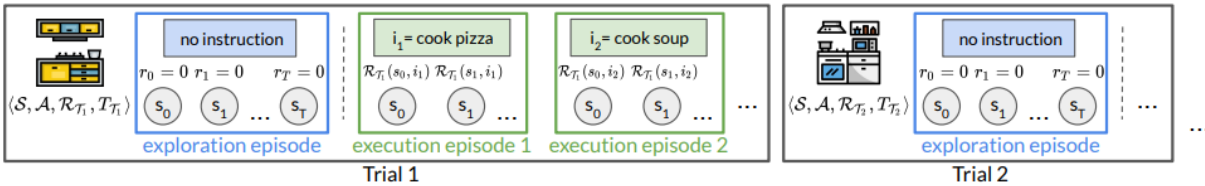



- $\pi^{exp}$ is the exploration policy
- $\pi^{exe}$ is the execution (exploitation) policy
- What's challenging about optimizing both simultaneously?
  - $\pi^{exe}$ relies on $\pi^{exp}$ for good exploration data
  - $\pi^{exp}$ relies on gradients passed through $\pi^{exe}$

*If $\pi^{exe}$ cannot effectively solve the task, then the gradients will be uninformative, and so even if our exploration is great, it will be penalized if we cannot execute well*

# 2 Approach

## 2.1 Meta RL Setup

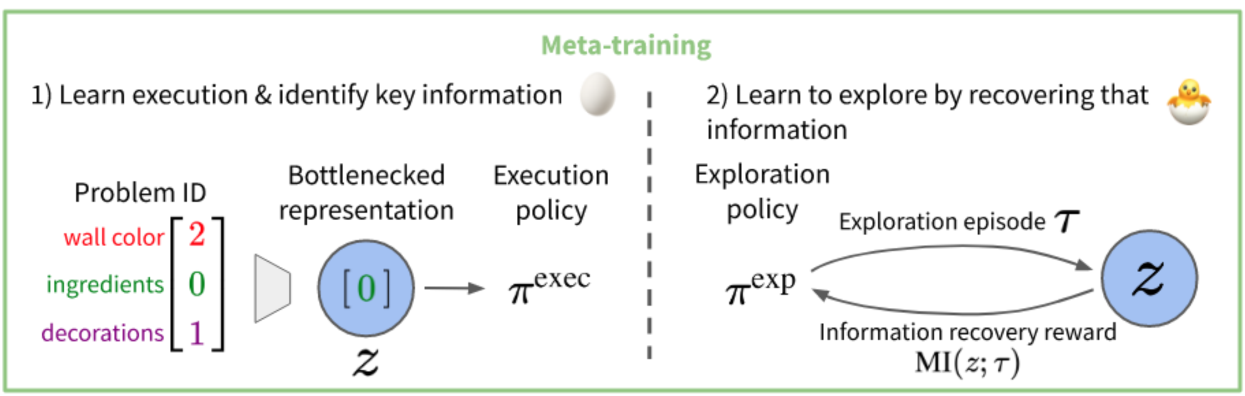



- each trail involves interaction with an MDP $M \sim p(\mathcal{M})$ which we assume can be labelled with a unique ID $i_M$ at meta-training time
- Each trail involves a single exploration episode, followed by $N$ exploitation episodes, and the goal of the agent is to maximize the rewards over the $N$ exploitation episodes
- We label the policy taken during the exploration phase as $\pi^{exp}$ and the policy taken during the exploitation phase as $\pi^{exe}$ which can be the same, or at least share parameters
- The goal can be formalized as:

$$\mathcal{J}[\pi^{exp}, \pi^{exe}] := \mathbb{E}_{M\sim p(\mathcal{M}, \tau^{exp} \sim pi^{exp})}[V^{\pi^{exe}}(\tau^{exp}; M)]$$

- intuitively: maximizing the expected returns acting according to the exploitation policy, conditioned on the exploration trajectory that the exploratory policy took in the first episode of the trial.

## 2.2 Intuition





- the high level approach, will be to decouple the exploration and exploitation, by explicitly defining an exploitation objective based on latent task relevant information (independent of $\pi^{exp}$), and train the exploration policy to identify the distinguishing characteristics of the environment, that can recover all the task relevant information for good execution

- derive a stochastic problem encoding $F(z|i_M)$ by training an execution policy $\pi^{exe}$ conditioned on the encoders outputs (with a bottleneck on $z$)
- then train an exploration policy $\pi^{exp}$ to produce trajectories that maximize the mutual information with $z$

## 2.3 Learning

### 2.3.1 Encoder

- Represent the encoder $F : \mathcal{I} \longrightarrow \mathcal{Z}$ which takes a unique task id, and embeds into a latent vector $z \in \mathcal{Z}$ that allows the execution policy to perform well as a gaussian centered function:

$$F(i_M) = \mathcal{N}(f_\psi(i_m), \rho I)$$

- the function is learned jointly with the execution policy as below

### 2.3.2 Execution Policy

- $\pi_\theta^{exe}$ represented by a Q-network
- Optimizes the following objective jointly with F:

$$max_{\psi,\theta}\mathbb{E}_{M\in p(\mathcal{M}),z\in F_\psi(z|i_M)}[V^{pi_\theta^{exe}}(i,z,M)] - \lambda \cdot \mathcal{I}(z,M)$$

- both terms are independent of the exploration policy
- first term maximizes execution rewards from environment, conditioned on the encoding produces by F
- second term is an information bottleneck that minimizes mutual information between task and encoding which with gaussian form of the encoder reduces to the regularizer:

$$\lambda \cdot \mathcal{I}(z,M) = \lambda \cdot |f_\psi(i_M)|_2^2$$

### 2.3.3 Exploration Policy

- $\pi_\phi^{exp}$ represented by another Q network
- Once we have an encoder $F_\psi$ to extract only the information necessary to optimally execute instructions, [1] we can optimize the exploration policy to produce trajectories that encode the same information;

$$max_\phi \mathcal{I}(\tau^{\pi_\phi^{exp}}, z)$$

which can be optimized using a variational bound given by

$$r_t^{exp}(a_t, s_{t+1}, \tau_{t-1}^{exp}, M) = \mathbb{E}_{z\sim F_\psi}[logq_w(z|s_{t+1}; a_t; \tau_{t-1}^{exp}) - logq_w(z|\tau_{t-1}^{exp})] - c$$

- intuitively, the reward for taking an action is high if the induced transition encodes more information about the problem than was already present in the trajectory
- constant $c$ is a small penalty that encourages efficient exploration
- the variational distribution $q_w$ approximates the true distributon $p(z|\tau^{exp})$ thus serving as a decoder that generates the encoding z from the exploration trajectory $\tau^{exp}$ (used at test time when we don't have MDP id)

[1] in practice, rather than training encoder and execution policy to completion, we train alongside the exploration policy using an expectation-maximization training procedure, which fixes the parameters from one net while optimizes the other, and vice-versa

### 2.3.4 Decoder

- the decoder $q_w(z|\tau^{exp})$ as explained above, is taken to be gaussian centered so around ${}_w(\tau^{exp})$ with variance $\rho^2 I$.
- this leads to simpler exploration reward formulation

## 2.4 PseudoCode

**Algorithm 1** DREAM DDQN

**Initialize** execution replay buffer $\mathcal{B}_{\text{exe}} = \{\}$ and exploration replay buffer $\mathcal{B}_{\text{exp}} = \{\}$<br>
**Initialize** execution Q-value $\hat{Q}^{\text{exe}}$ parameters $\theta$ and target network parameters $\theta'$<br>
**Initialize** exploration Q-value $\hat{Q}^{\text{exp}}$ parameters $\phi$ and target network parameters $\phi'$<br>
**Initialize** problem ID embedder $f_\psi$ parameters $\psi$ and target parameters $\psi'$<br>
**Initialize** trajectory embedder $g_\omega$ parameters $\omega$ and target parameters $\omega'$<br>
**for** trial $= 1$ **to** max trials **do**<br>
    Sample problem $\mathcal{T} \sim p(\mathcal{T})$, defining MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}_\mathcal{T}, T_\mathcal{T} \rangle$<br>
    Roll-out $\epsilon$-greedy exploration policy $\hat{Q}^{\text{exp}}(s_t, \tau^{\text{exp}}_{:t}, a_t; \phi)$, producing trajectory $\tau^{\text{exp}} = (s_0, a_0, \ldots, s_T)$.<br>
    Add tuples to the exploration replay buffer $\mathcal{B}_{\text{exp}} = \mathcal{B}_{\text{exp}} \cup \{(s_t, a_t, s_{t+1}, \mathcal{T}, \tau^{\text{exp}})\}_t$.

    Sample instruction $i \sim p(i)$.<br>
    Randomly select between embedding $z \sim \mathcal{N}(f_\psi(\mathcal{T}), \rho^2 I)$ and $z = g_\omega(\tau^{\text{exp}})$.<br>
    Roll-out $\epsilon$-greedy execution policy $\hat{Q}^{\text{exe}}(s_t, i, z, a_t; \theta)$, producing trajectory $(s_0, a_0, r_0, \ldots)$ with $r_t = \mathcal{R}_\mathcal{T}(s_{t+1}, i)$.<br>
    Add tuples to the execution replay buffer $\mathcal{B}_{\text{exe}} = \mathcal{B}_{\text{exe}} \cup \{(s_t, a_t, r_t, s_{t+1}, i, \mathcal{T}, \tau^{\text{exp}})\}_t$.

    Sample batches of $(s_t, a_t, s_{t+1}, \mathcal{T}, \tau^{\text{exp}}) \sim \mathcal{B}_{\text{exp}}$ from exploration replay buffer.<br>
    Compute reward $r^{\text{exp}}_t = \|f_\psi(\mathcal{T}) - g_\omega(\tau^{\text{exp}}_{:t})\|^2_2 - \|f_\psi(\mathcal{T}) - g_\omega(\tau^{\text{exp}}_{:t-1})\|^2_2 - c$ (Equation 5).<br>
    Optimize $\phi$ with DDQN update with tuple $(s_t, a_t, r^{\text{exp}}_t, s_{t+1})$

    Sample batches of $(s, a, r, s', i, \mathcal{T}, \tau^{\text{exp}}) \sim \mathcal{B}_{\text{exe}}$ from execution replay buffer.<br>
    Optimize $\theta$ and $\omega$ with DDQN update with tuple $((s, i, \tau^{\text{exp}}), a, r, (s', i, \tau^{\text{exp}}))$<br>
    Optimize $\theta$ and $\psi$ with DDQN update with tuple $((s, i, \mathcal{T}), a, r, (s', i, \mathcal{T}))$<br>
    Optimize $\psi$ on $\nabla_\psi \min(\|f_\psi(\mathcal{T})\|^2_2, K)$ (Equation 3)<br>
    Optimize $\omega$ on $\nabla_\omega \sum_t \|f_\psi(\mathcal{T}) - g_\omega(\tau^{\text{exp}}_{:t})\|^2_2$ (Equation 4)

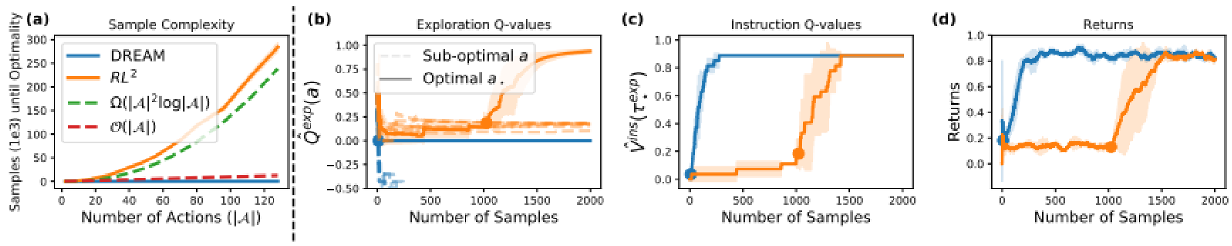
    **if** trial $\equiv 0 \pmod{\text{target freq}}$ **then**<br>
        Update target parameters $\phi' = \phi$, $\theta' = \theta$, $\psi' = \psi$, $\omega' = \omega$<br>
    **end if**<br>
**end for**

# 3 Results

The important question are:

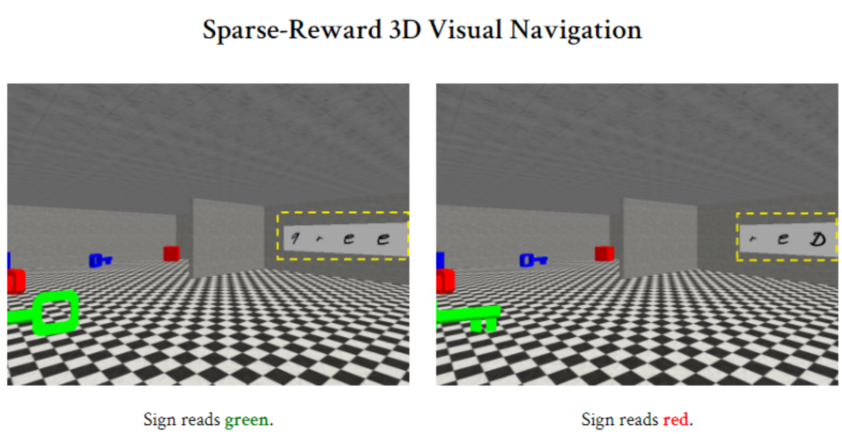
1. Can DREAM efficiently explore to discover only the information required to execute instructions?
2. Does DREAM improve over standard meta-RL algorithms in difficult exploration MDP's?

## 3.1 Bandit Task

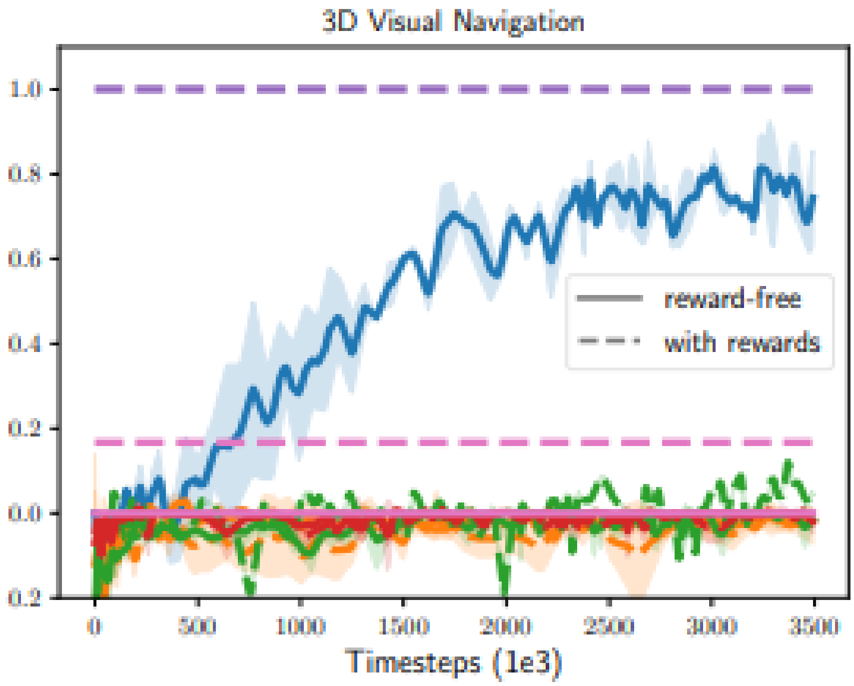



- Bandit task with sparse rewards, where each episode is a one-step bandit problem with action space $\mathcal{A}$
- Taking one of the actions reveals complete information about the problem, and all other actions give virtually no information
- Each mdp is queried with a one-hot-task id that specifies the correct action to explore
- in DREAM, the exploration Q values regress towards the decoder q, which learns fast since it does not depend on the execution actions. Thus, the exploration policy quickly becomes optimal, which allows for quickly learning the execution Q values achieving maximal returns

## 3.2 3D Navigation




- agent navigates a 3d world, involving a set of objects it must get to. Each MDP specifies, whether the agent should get a key, box or ball, but not what color to get to. The color is written on a wall that is *behind* where the agent spawned
- hence, the agent needs to have very good exploration, to move away from the objects, read the sign, and *then* reach the optimal color-object combo




- this difficult exploration problem cannot be solved by methods in meta-rl that couple exploration and exploitation