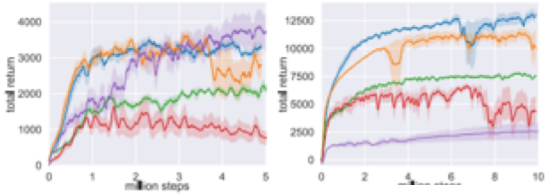
Hierarchical Reinforcement Learning

Ivy

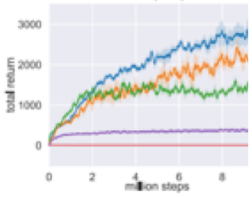
Outline

- **Motivation and problem Recap (30min)**
- Method (30min)
  - Option-Critic Architecture
  - Feudal Network
- Method in more detail: Latent Space Policies (1h)
    - Maximum Entropy RL
    - Hierarchical training
    - Invertible transformation between levels

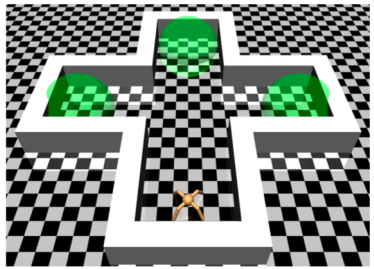
Recap: Motivating problems

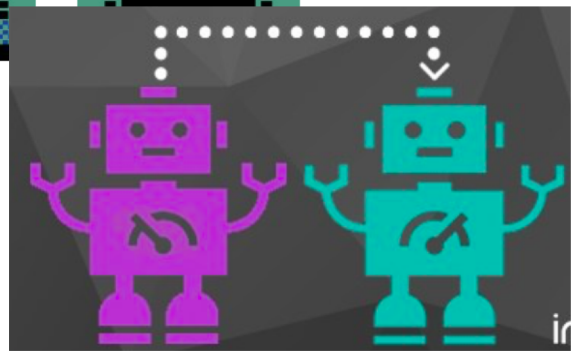


(c) Walker2d-v1



(e) Ant (rllab)



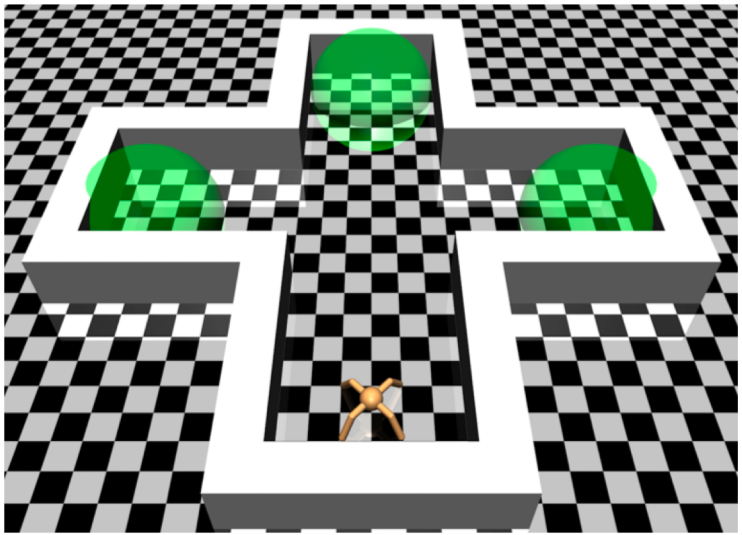


Recap: Sparse reward problem

Challenges:

- Sparse reward signals
- Long term credit assignment

Temporal abstraction encourages long term credit assignment

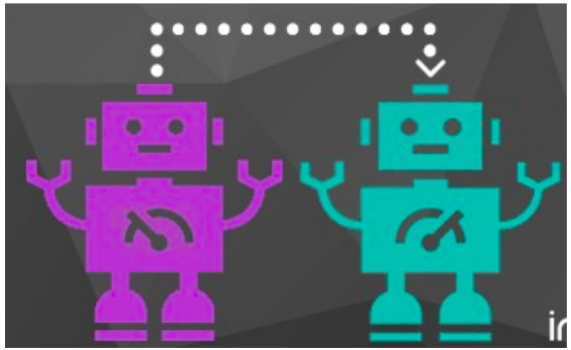


Recap: Transfer learning problem

Challenges:

- Disentangle representations

Different levels of the hierarchies can be disentangled.



Recap: Hierarchy

Hierarchical representation: enable reasoning and decision making at different levels of abstraction

Recap: Variations of hierarchy

Hierarchy is defined differently:

- Could be algorithm being trained, i.e, hierarchical DQN
- Could be architecture that has high-level and low-level, i.e., feudal neural network, option-critic
- Could be subtask design, i.e., learning and transfer of modulated locomotor controllers
- Could be reward design, i.e., value function decomposition
- Could be training procedure, i.e., latent space policies for RL
- Could be imitation…, i.e., hierarchical imitation for RL

For this talk, we focus on generalizable architectures and follow the line of how they solve the sparse reward problem.

Recap: Value function, Policy gradient, Actor-critic

Value function

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s; \pi]$$

Policy gradient

$$\nabla_\theta J(\theta) \approx \sum_i \left(\sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_t^i | \mathbf{s}_t^i) \right) \left(\sum_t r(\mathbf{s}_t^i, \mathbf{a}_t^i) \right)$$

Actor-critic method

$$\hat{A}^{\pi}(\mathbf{s}_i, \mathbf{a}_i) = r(\mathbf{s}_i, \mathbf{a}_i) + \gamma \hat{V}_\phi^{\pi}(\mathbf{s}_i') - \hat{V}_\phi^{\pi}(\mathbf{s}_i)$$

$$\nabla_\theta J(\theta) \approx \sum_i \nabla_\theta \log \pi_\theta(\mathbf{a}_i | \mathbf{s}_i) \hat{A}^{\pi}(\mathbf{s}_i, \mathbf{a}_i)$$

Outline

- Motivation and problem Recap (30min)
- Method (30min)
  - **Option-Critic Architecture**
  - Feudal Network
- Method in more detail: Latent Space Policies (1h)
    - Maximum Entropy RL
    - Hierarchical training
    - Invertible transformation between levels

Option: Intuition

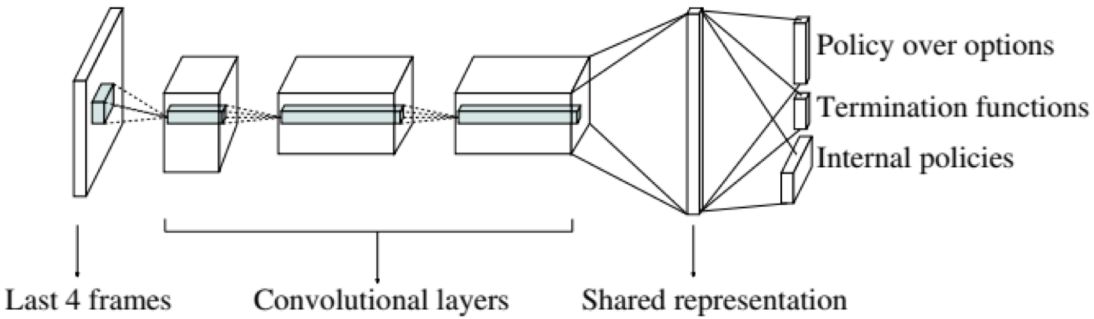
Motivation: Option extended action

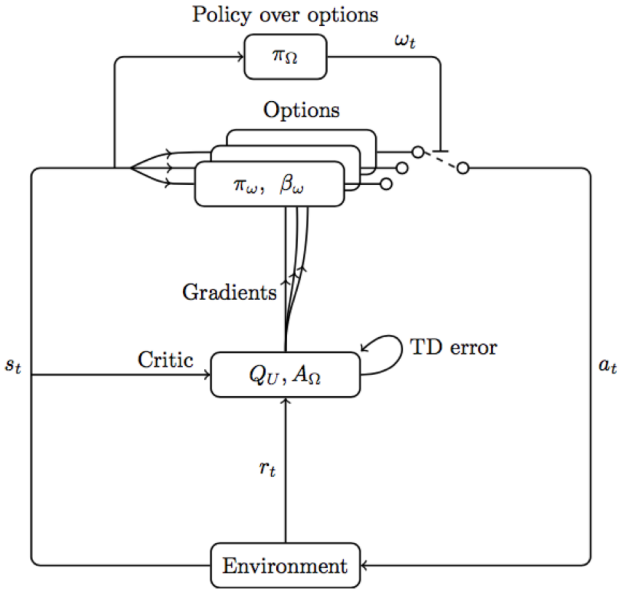
Definition: Option is defined as a tuple $w = \langle \mathcal{I}, \pi, \beta \rangle$

- A policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$
- A termination condition $\beta : \mathcal{S}^+ \to [0, 1]$
- An initiation set $\mathcal{I} \subset \mathcal{S}$

Example: Open-the-door

Option-Critic: Architecture





Option-Critic: update

Policy gradient:

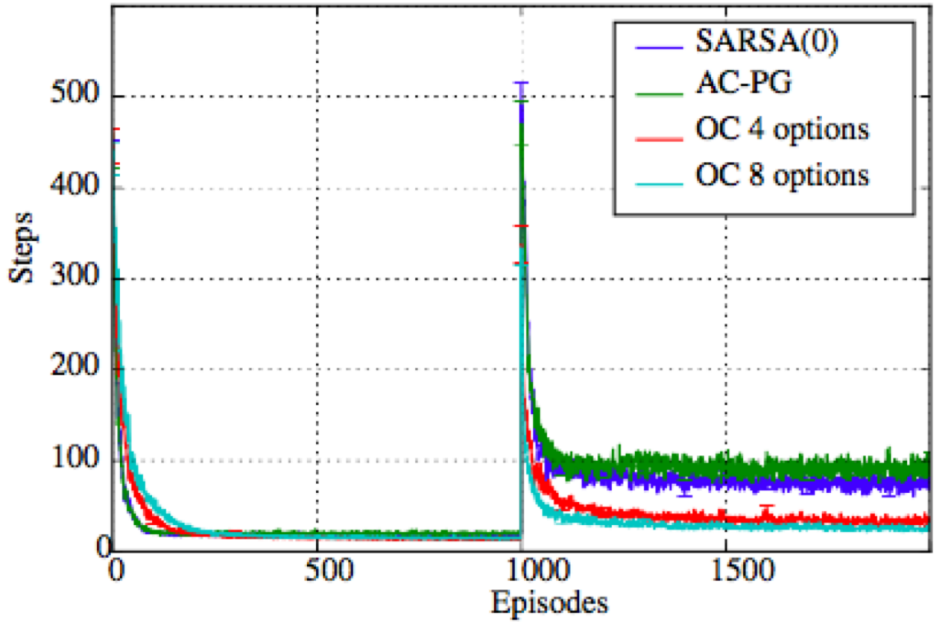
$$\nabla_\theta J(\theta) = \Sigma_s d_{\pi_\theta}(s) \Sigma_a \frac{\partial \pi_\theta(a|s)}{\partial \theta} Q_{\pi_\theta}(s,a)$$
$$d_{\pi_\theta}(s) = \Sigma_t \gamma^t P(s_t = s | s_0)$$

Intra option policy gradient:

$$\nabla_\theta J(\theta, w) = \Sigma_{s,w} d_{\pi_\theta}(s,w) \Sigma_a \frac{\partial \pi_{w,\theta}(a|s)}{\partial \theta} Q_{\pi_{w,\theta}}(s,w,a)$$
$$d_{\pi_\theta}(s,w) = \Sigma_t \gamma^t P(s_t = s, w_t = w | s_0, w_0)$$

Extended reading: Sutton, Richard S., et al. "Policy gradient methods for reinforcement learning with function approximation." Advances in neural information processing systems. 2000.

Option-Critic: Experiment



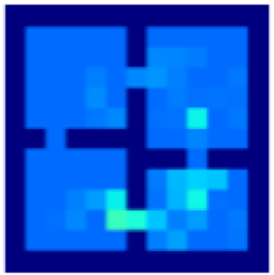


Fig1. Four-room maze task where the target moves after every 1000 episodes. A good algorithm takes the fewest steps to recover the changed target. Option-critic (“OC”) recovers faster than actor-critic and SARSA(0).

Option-Critic: Experiment

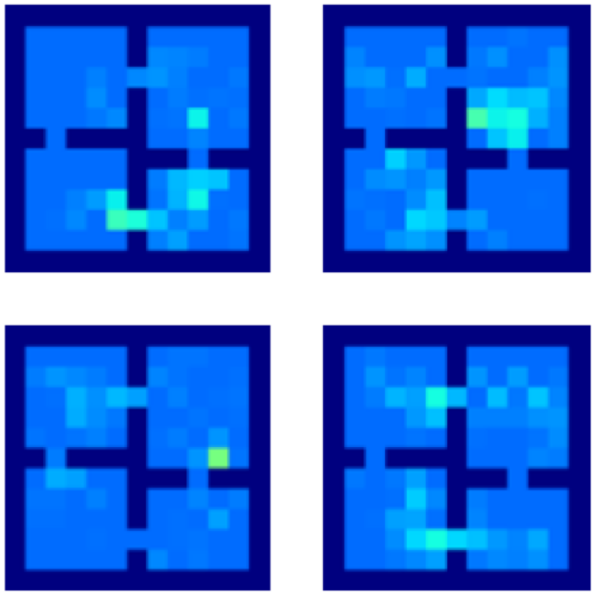


Fig 2. Four-room maze task, Termination probabilities for the option-critic agent learning with 4 options. The black color is the walls of the maze. Lighter colors encode higher termination probabilities.

Option-Critic Intuition for Solving Sparse Reward problem

- Temporal abstraction: option (extended action).

Option-Critic: Critique

Benefits:

- End-to-end
- Learns at speed comparable or better than using just primitive actions

Drawbacks:

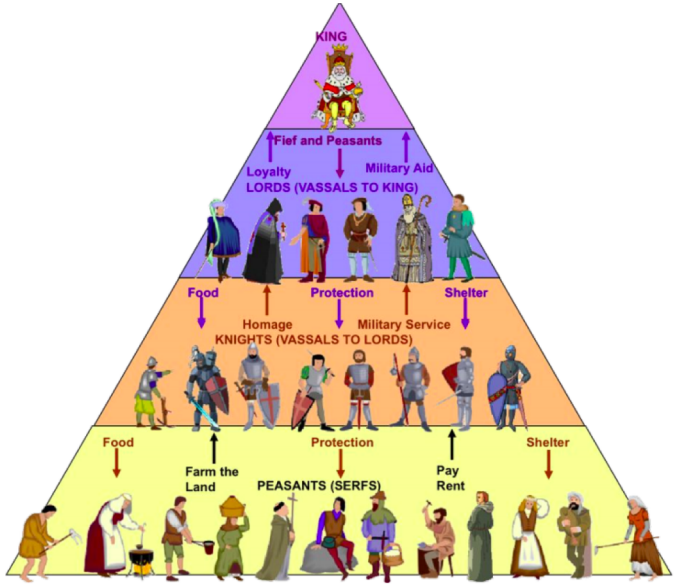
- Tend to converge to trivial solutions as all policy gradient algorithms, need entropy regularization

Outline

- Motivation and problem Recap (30min)
- Method (30min)
  - Option-Critic Architecture
  - **Feudal Network**
- Method in more detail: Latent Space Policies (1h)
    - Maximum Entropy RL
    - Hierarchical training
    - Invertible transformation between levels

FuN: Intuition

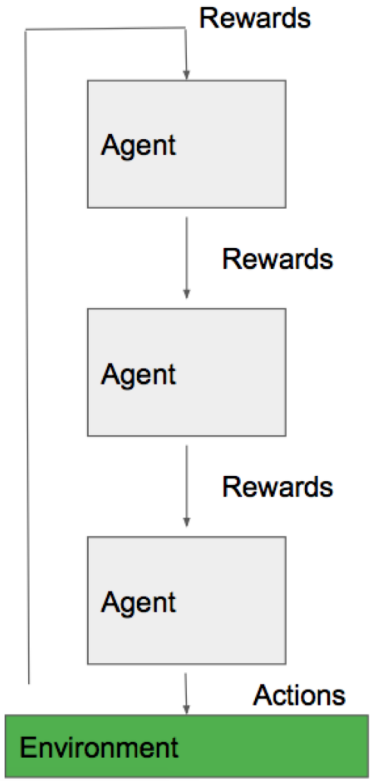
- Governance system in Europe in middle ages
- Extremely-hierarchical based on ownership of property
- Higher level people have control of the lower levels, but not over people many layers lower.



Borrowed slides from https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring18/slides/cs885-lecture16b.pdf

FuN: Intuition (Feudal RL paper)

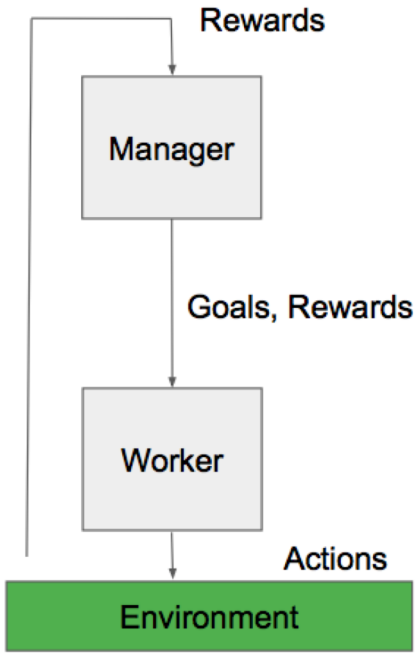
- Reward Hiding:
  - Managers reward sub-managers for satisfying their commands, not through an external reward
  - Managers have absolute control
- Information Hiding
  - Observe world at different resolutions
  - Managers don’t know what happens at a other levels of the hierarchy



Borrowed slides from https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring18/slides/cs885-lecture16b.pdf

FuN: Intuition

- Manager
  - Sets directional goals for the worker
  - Rewarded by environment
  - Does not directly act in environment
- Worker
  - Higher temporal resolution
  - Reward for achieving manager's goals
  - Produces primitive actions in environment

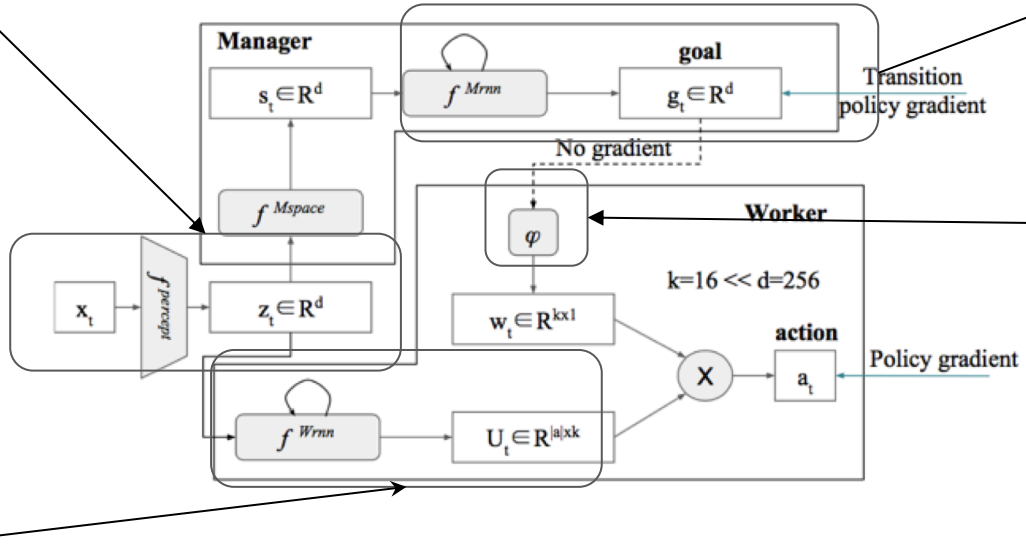


Borrowed slides from https://cs.uwaterloo.ca/~ppoupart/teaching/cs885-spring18/slides/cs885-lecture16b.pdf

FuN: Architecture

Manager's Goal embedding
(goals summed over last 10 time steps)

Shared visual
input embedding



Goal embedding to
Worker:
Compress manager's
goal to dim k

Worker's Action embedding

FuN: Manager's update rule

- Directional goals

$$\nabla g_t = A_t^M \nabla_\theta d_{\cos}(s_{t+c} - s_t, g_t(\theta))$$

$$A_t^M = R_t - V_t^M(x_t, \theta)$$

M: manager
Rt: environment reward

FuN: Worker's update rule

- Update policy

$$\nabla \pi_t = A_t^D \nabla_\theta \log \pi(a_t | x_t; \theta)$$

$$A_t^D = (R_t + \alpha R_t^I - V_t^D(x_t; \theta))$$

D: worker
RtI: intrinsic reward

FuN: Experiments

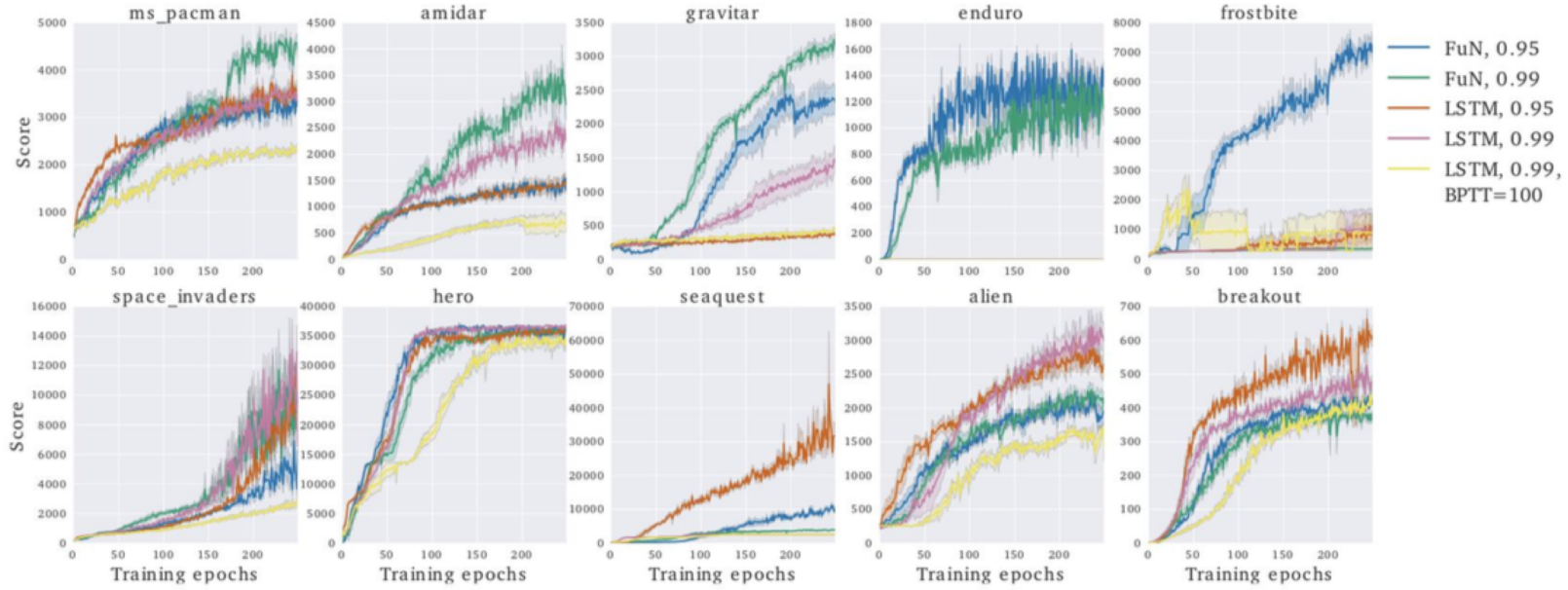


Fig 3, Atari games, compared with baselines.

FuN: Experiments

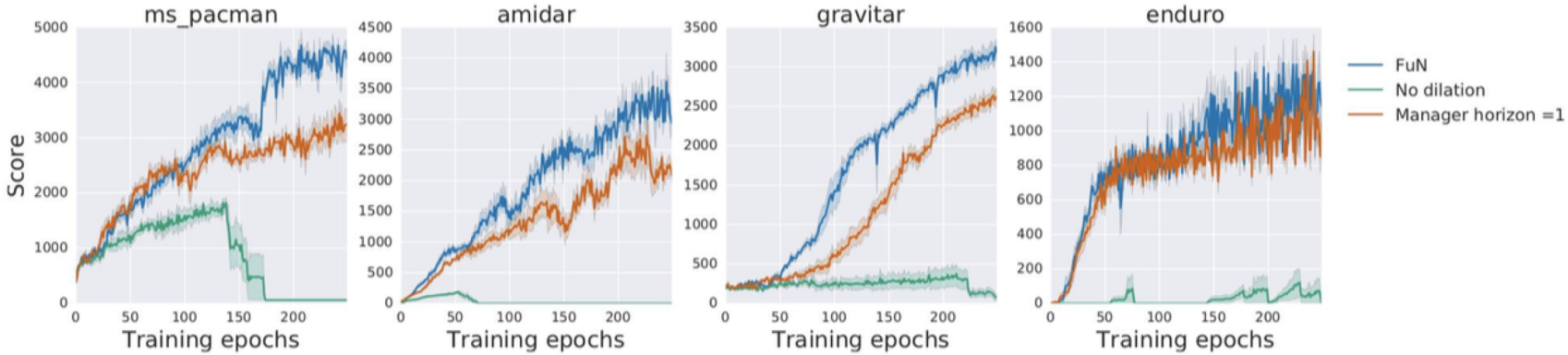


Fig 3, Ablation on temporal resolution ablations

FuN intuition for solving the sparse reward problem

- Temporal abstraction: enforce lower temporal resolution through dilated LSTM and summing goal over 10 timesteps

- Decouple goal setting and goal achievement: Manually separated manager and worker to make sure they are not learning the same thing.

- Subgoal setting: directional goals rather than absolute goals

FuN: Comparison with Option-Critic

Option-Critic

- Adding entropy regularization to avoid trivial policies
- Process-over-options and options are trained together
- Based on option (extended action)

FuN

- Subgoals as directional goals are naturally diverse.
- Manager and Worker are trained separately with different goals
- Based on intrinsic reward from the manager

FuN: Critique

Directional goal is a subgoal design

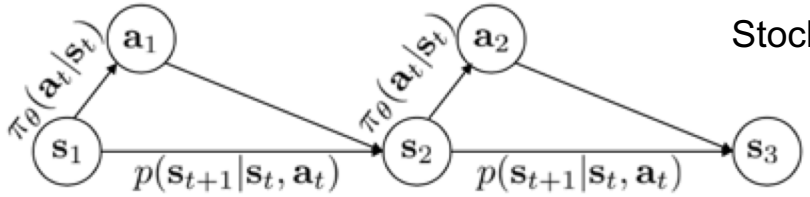
In problems such as locomotion, good subgoal design is hard to come by.

For locomotion task, can we learn good subgoals?

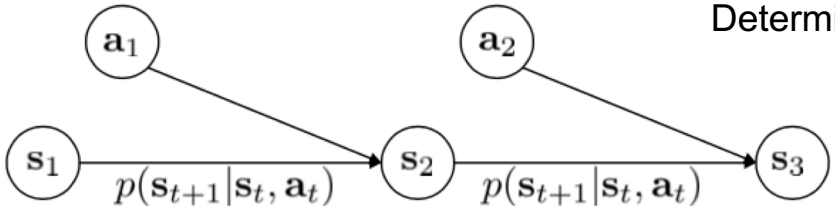
Outline

- Motivation and problem Recap (30min)
- Method (30min)
  - Option-Critic Architecture
  - Feudal Network
- **Method in more detail: Latent Space Policies (1h)**
    - Maximum Entropy RL
    - Hierarchical training
    - Invertible transformation between levels

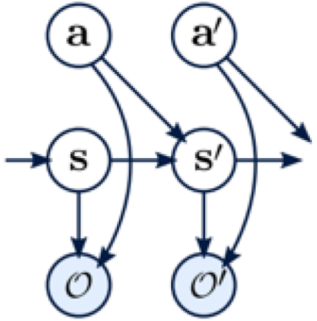
Maximum Entropy RL and optimal trajectory



Stochastic MDP



Deterministic MDP



Optimality variable

Set $p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$

Maximum Entropy RL and optimal trajectory

$$p(\tau) = p(\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_T, \mathbf{a}_T|\theta) = p(\mathbf{s}_1)\prod_{t=1}^{T} p(\mathbf{a}_t|\mathbf{s}_t, \theta)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t).$$

$$p(\mathcal{O}_t|\mathbf{s}_t, \mathbf{a}_t) = \exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

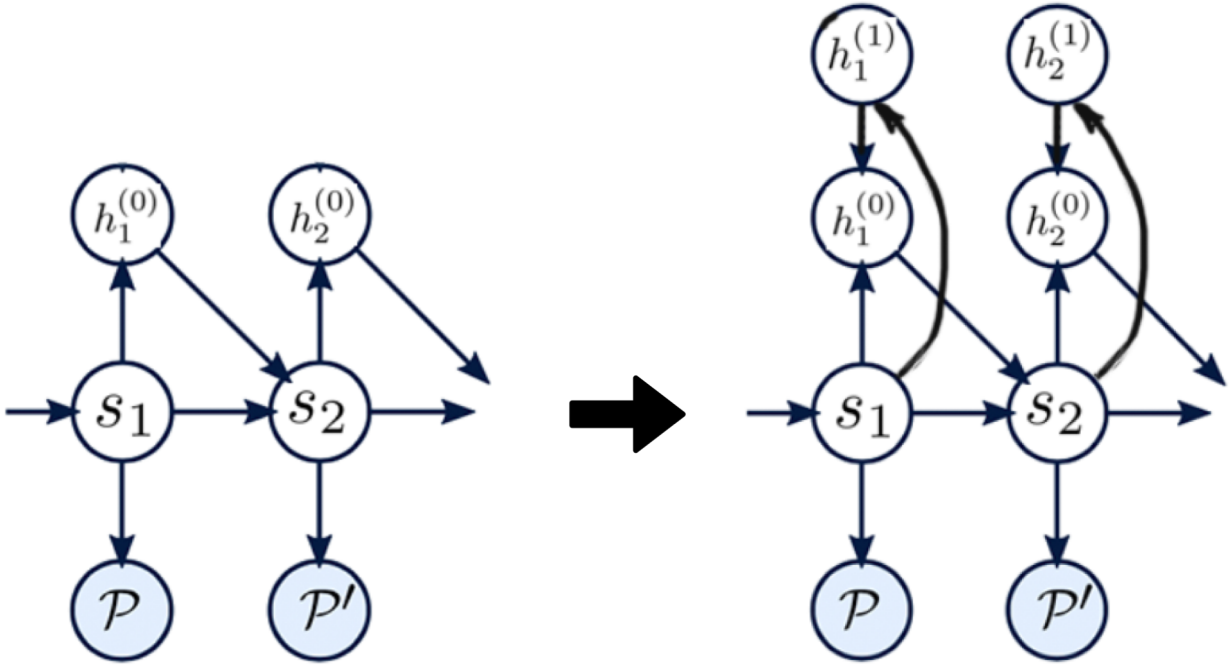
$$p(\tau|\mathcal{O}_{0:T}) \propto p(\mathbf{s}_0)\prod_{t=0}^{T} p(\mathbf{a}_t)p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)\exp(r(\mathbf{s}_t, \mathbf{a}_t))$$

Thus, to approximate the optimal trajectory with $\hat{p}(\tau)$, minimize $D_{KL}(\hat{p}(\tau)||p(\tau|\mathcal{O}_{0:T}))$

And since $-D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t, \mathbf{a}_t) + \mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))]$

⇒ The optimal trajectory maximizes the maximum entropy objective

LSP: Training - Intuition



$$\mathbf{h}_t^{(i)} = f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t)$$

LSP: Training a Multi layer policy

Input: True environment $p(\mathbf{s}_{t+1}|, \mathbf{s}_t, \mathbf{h}_t^{(0)})$, where $\mathbf{h}_t^{(0)}$ corresponds to the physical actions $\mathbf{a}_t$.

Input: Reward $\mathcal{R}$

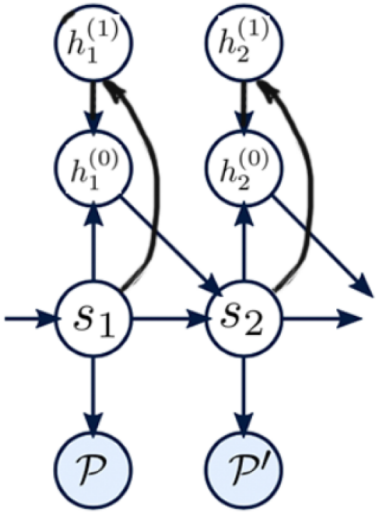
for $i = 0$ to $K - 1$ **do**

Learn the weights of f_i so that $\mathbf{h}_t^{(i)} = f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t)$, where $\mathbf{h}_t^{(i+1)} \sim p(\mathbf{h}_t^{(i+1)})$, optimizes $\mathcal{R}$ on $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{h}_t^{(i)})$.

Embed the new layer f_i into the environment: $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{h}_t^{(i+1)}) \leftarrow p(\mathbf{s}_{t+1}|\mathbf{s}_t, f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t))$.

end for

Output: A hierarchical policy $f = f_0 \circ f_1 \circ ... \circ f_{K-1}$.

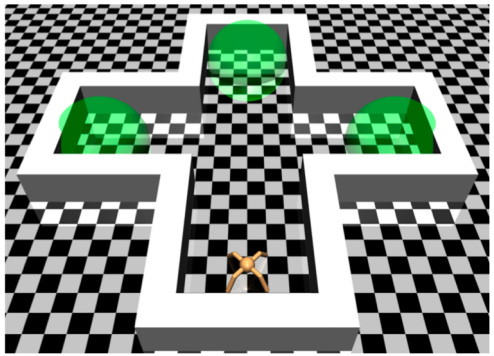


LSP: Training a Multi layer policy w/ Reward Design

Example (spider navigation):

R1: Low-level reward as motion in any direction, still aim to maximum entropy ⇒ motion in many different directions

R2: final reward, i.e., navigation to target



LSP: Invertible transformation between levels - Intuition

$$\mathbf{h}_t^{(i)} = f_i(\mathbf{h}_t^{(i+1)}; \mathbf{s}_t)$$

- The higher layer can undo any behavior in the lower layer if it is detrimental to task success.
- Both higher layer and lower layer retain full expressivity

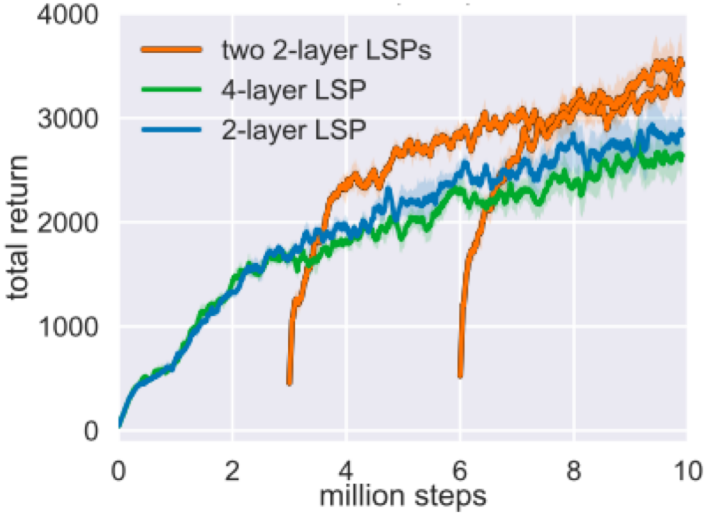
LSP: Invertible transformation between levels

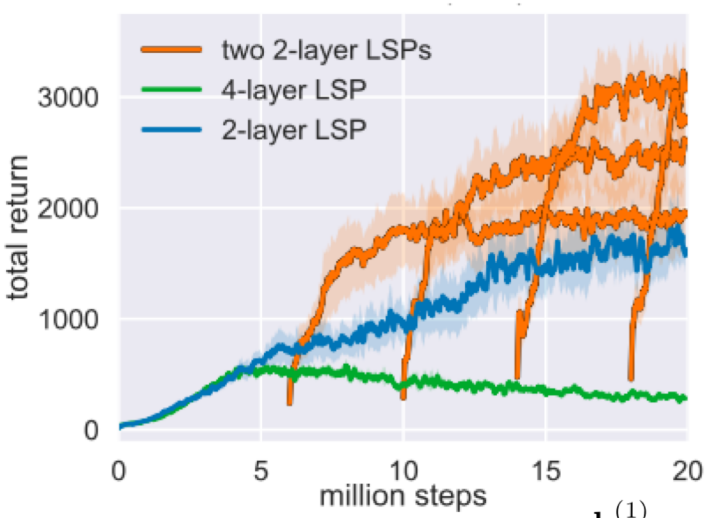
- Representation of sub-policies:
  - Each subpolicy should be tractable
  - The subpolicies should be expressive, i.e., not cripple the ability of the higher level to solve the task.
  - Conditional factor of subpolicies should be deterministic.

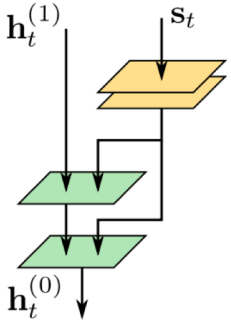
⇐ Real NVP neural network

Extended reading: Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estima- tion using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

LSP: Experiments







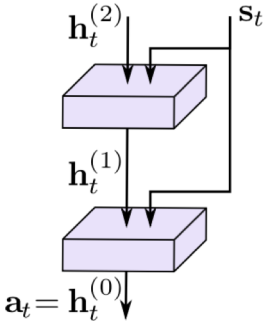


Fig 3. Ant and Humanoid tasks, Single policy with 2 layers, single policy with 4 layers, and two-level policy. Training of two-level policies yields the best performance.

LSP: Experiments

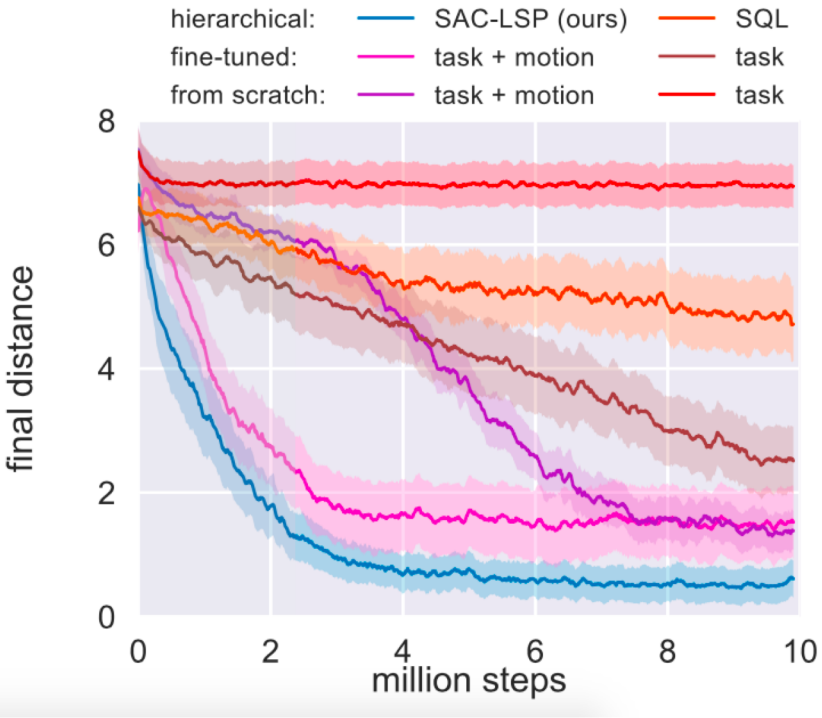


Fig 4, Spider navigation task. We first trained a low-level policy with a motion reward in a pretraining environment not including the walls, then fixed the policy and trained another policy level with a target reward

LSP Intuition for Solving Sparse Reward problem

- Each layer directly attempts to solve the maximum entropy objective and thus help the layer above it to solve the same maximum entropy objective.
- Bijective transformation - high level and low level are not constrained, but the high level can always reverse what low level did.

LSP: Comparison with FuN

Feudal NN:

- Diverse set of goals by learning directional goals.
- High level action disintegrated from low level action.
- Trained with different rewards

Option-Critic:

- Entropy regularization
- Extended action.
- Trained together

LSP:

- Maximum entropy objective
- Latent action space.
- Trained level by level

Thank you!

Control as probabilistic inference: deterministic

$$p(\tau) = p(\mathbf{s}_1, \mathbf{a}_t, \ldots, \mathbf{s}_T, \mathbf{a}_T|\theta) = p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{a}_t|\mathbf{s}_t, \theta) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t).$$

$$p(\tau|\mathbf{o}_{1:T}) \propto p(\tau, \mathbf{o}_{1:T}) = \left[p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \right] \exp\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}
$$

$$p(\tau|\mathbf{o}_{1:T}) \propto \mathbb{1}[p(\tau) \neq 0] \exp\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right)$$

Control as probabilistic inference: deterministic

In case of exact inference

$$D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = 0$$

$$D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = -E_{\tau\sim\hat{p}(\tau)}[\log p(\tau) - \log \hat{p}(\tau)]$$

Control as probabilistic inference: deterministic

$$
\begin{aligned}
-D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) &= E_{\tau\sim\hat{p}(\tau)}\Bigg[\log p(\mathbf{s}_1) + \sum_{t=1}^{T}(\log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) + r(\mathbf{s}_t,\mathbf{a}_t)) - \\
&\qquad\qquad \log p(\mathbf{s}_1) - \sum_{t=1}^{T}(\log p(\mathbf{s}_{t+1}|\mathbf{s}_t,\mathbf{a}_t) + \log\pi(\mathbf{a}_t|\mathbf{s}_t))\Bigg] \\
&= E_{\tau\sim\hat{p}(\tau)}\left[\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t) - \log\pi(\mathbf{a}_t|\mathbf{s}_t)\right] \\
&= \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t,\mathbf{a}_t) - \log\pi(\mathbf{a}_t|\mathbf{s}_t)] \\
&= \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t,\mathbf{a}_t)] + E_{\mathbf{s}_t\sim\hat{p}(\mathbf{s}_t)}[\mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))].
\end{aligned}
$$

$$-D_{\mathrm{KL}}(\hat{p}(\tau)\|p(\tau)) = \sum_{t=1}^{T} E_{(\mathbf{s}_t,\mathbf{a}_t)\sim\hat{p}(\mathbf{s}_t,\mathbf{a}_t))}[r(\mathbf{s}_t,\mathbf{a}_t) + \mathcal{H}(\pi(\mathbf{a}_t|\mathbf{s}_t))]$$

Control as probabilistic inference: stochastic

$$p(\tau) = \left[p(\mathbf{s}_1) \prod_{t=1}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \right] \exp\left(\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) \right)$$

$$\begin{aligned}
\log p(\mathcal{O}_{1:T}) &= \log \int \int p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) d\mathbf{s}_{1:T} d\mathbf{a}_{1:T} \\
&= \log \int \int p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \frac{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})}{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} d\mathbf{s}_{1:T} d\mathbf{a}_{1:T} \\
&= \log E_{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \sim q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \left[\frac{p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T})}{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \right] \\
&\geq E_{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \sim q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \left[\log p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) - \log q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \right]
\end{aligned}$$

$$
\begin{aligned}
\log p(\mathcal{O}_{1:T}) &= \log \int \int p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) d\mathbf{s}_{1:T} d\mathbf{a}_{1:T} \\
&= \log \int \int p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \frac{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})}{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} d\mathbf{s}_{1:T} d\mathbf{a}_{1:T} \\
&= \log E_{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \sim q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \left[\frac{p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T})}{q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \right] \\
&\geq E_{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \sim q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \left[\log p(\mathcal{O}_{1:T}, \mathbf{s}_{1:T}, \mathbf{a}_{1:T}) - \log q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \right]
\end{aligned}
$$

$$
\log p(\mathcal{O}_{1:T}) \geq E_{(\mathbf{s}_{1:T}, \mathbf{a}_{1:T}) \sim q(\mathbf{s}_{1:T}, \mathbf{a}_{1:T})} \left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t) - \log q(\mathbf{a}_t | \mathbf{s}_t) \right]
$$