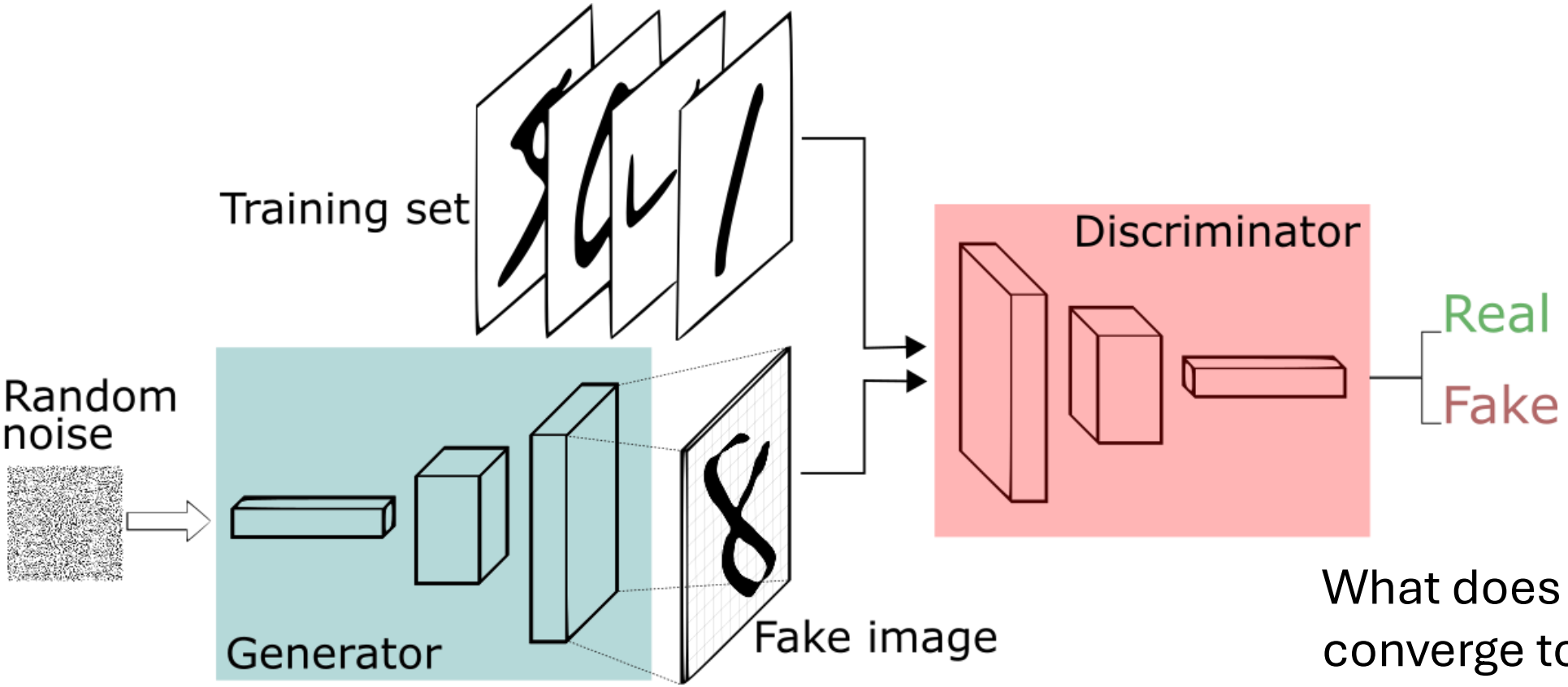
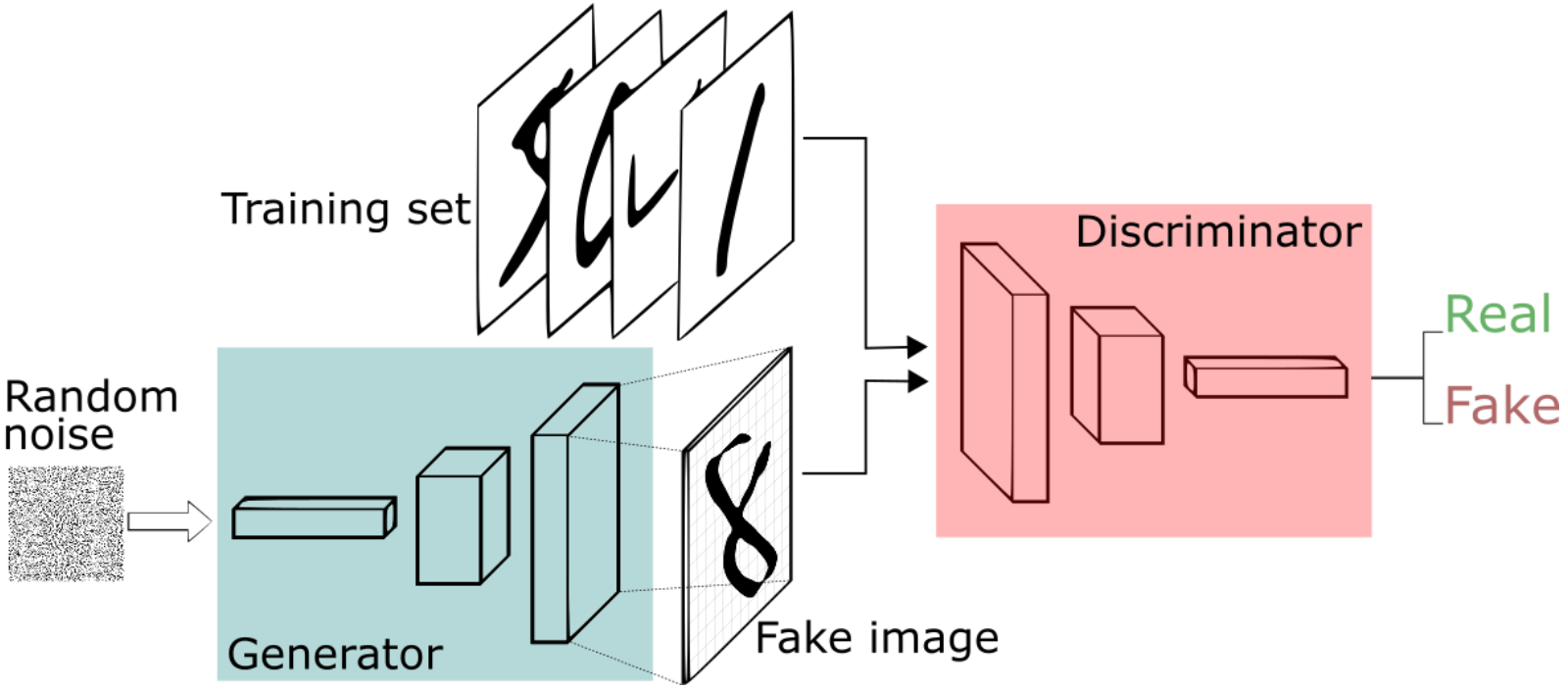
GANs, Adversarial IRL, and Bayesian Reward Learning

Generative Adversarial Networks (GANs)



What does the discriminator converge to at optimality?

$$D^*(x) = \frac{p_{\text{expert}}(x)}{p_{\text{expert}}(x) + p_\pi(x)}$$



$$L_D = -\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] - \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

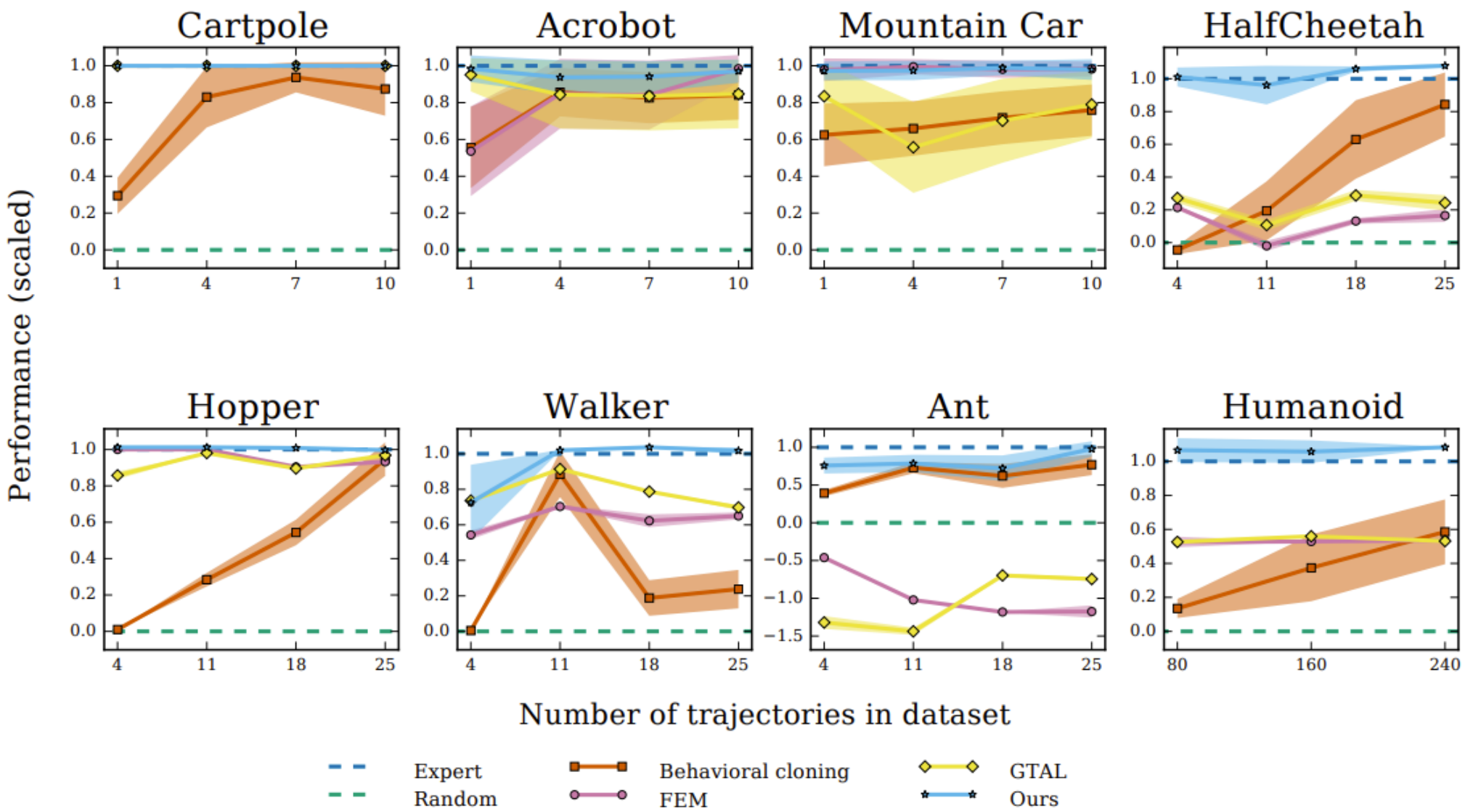
$$L_G = -\mathbb{E}_{z \sim p_z(z)}[\log D(G(z))]$$

Generative Adversarial Imitation Learning

Jonathan Ho
Stanford University
hoj@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Generative Adversarial Imitation Learning (GAIL)


Cartpole
Acrobot
Mountain Car
HalfCheetah
Hopper
Walker
Ant
Humanoid
Performance (scaled)
Number of trajectories in dataset
Expert
Random
Behavioral cloning
FEM
GTAL
Ours

Why learn rewards rather than imitate?

- Infer agent intentions and goals
- Optimize for policy in new action space or under new dynamics
- Generalize to new states

LEARNING ROBUST REWARDS WITH ADVERSARIAL INVERSE REINFORCEMENT LEARNING

Justin Fu, Katie Luo, Sergey Levine
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA 94720, USA
justinjfu@eecs.berkeley.edu, katieluo@berkeley.edu,
svlevine@eecs.berkeley.edu

Reward ambiguity

- Potential-Based Reward Shaping

Policy invariance under reward transformations: Theory and application to reward shaping

Andrew Y. Ng, Daishi Harada, Stuart Russell
Computer Science Division
University of California, Berkeley

$$\hat{r}(s, a, s') = r(s, a, s') + \gamma\Phi(s') - \Phi(s)$$

Disentangled Rewards

- Goal: We want the reward to transfer across multiple transition functions.
- Conclusion: The reward needs to be a function only of state.

Adversarial IRL

- Discriminator in our GAN has a special form:

$$D_{\theta,\phi}(s,a,s') = \frac{\exp\{f_{\theta,\phi}(s,a,s')\}}{\exp\{f_{\theta,\phi}(s,a,s')\} + \pi(a|s)}$$

Where $f_{\theta,\phi}(s,a,s') = g_\theta(s,a) + \gamma h_\phi(s') - h_\phi(s)$

Reward approximator

Shaping/value function

Looks familiar....

$$A^*(s,a) = r(s,a,s') + \gamma V^*(s') - V^*(s)$$

Making sense of this

- Under maximum entropy RL we have

$$\pi^*(a|s) \propto \exp(Q^*(s,a) - V^*(s))$$

$$D^*(x) = \frac{p_{\text{expert}}(x)}{p_{\text{expert}}(x) + p_\pi(x)}$$

$$D(s,a,s') \approx P(\text{expert} \mid s,a,s')$$

$$D_{\theta,\phi}(s,a,s') = \frac{\exp\{f_{\theta,\phi}(s,a,s')\}}{\exp\{f_{\theta,\phi}(s,a,s')\} + \pi(a|s)}$$

Adversarial IRL

- Discriminator in our GAN has a special form:

$$D_{\theta,\phi}(s,a,s') = \frac{\exp\{f_{\theta,\phi}(s,a,s')\}}{\exp\{f_{\theta,\phi}(s,a,s')\} + \pi(a|s)}$$

Where $f_{\theta,\phi}(s,a,s') = g_\theta(s,a) + \gamma h_\phi(s') - h_\phi(s)$

Reward approximator

Shaping/value function

In practice to avoid entangled rewards we make g just a function of s!

Algorithm

Algorithm 1 Adversarial inverse reinforcement learning

Obtain expert trajectories τ_i^E
Initialize policy π and discriminator $D_{\theta,\phi}$.
for step t in $\{1, \ldots, \mathrm{N}\}$ **do**
    Collect trajectories $\tau_i = (s_0, a_0, ..., s_T, a_T)$ by executing π.
    Train $D_{\theta,\phi}$ via binary logistic regression to classify expert data τ_i^E from samples τ_i.
    Update reward $r_{\theta,\phi}(s, a, s') \leftarrow \log D_{\theta,\phi}(s, a, s') - \log(1 - D_{\theta,\phi}(s, a, s'))$
    Update π with respect to $r_{\theta,\phi}$ using any policy optimization method.
end for

Why does the reward make sense

$$\hat{r} = \log D - \log(1 - D)$$

- Plug in D

Experiments

- Can AIRL learn disentangled rewards?
- Does it scale to high-dimensional tasks?
- Can it compete with imitation learning?

- https://sites.google.com/view/adversarial-irl

Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences

Daniel S. Brown [1] **Russell Coleman** [1 2] **Ravi Srinivasan** [2] **Scott Niekum** [1]

Abstract

Bayesian reward learning from demonstrations enables rigorous safety and uncertainty analysis

demonstrations, it is important for an agent to be able to provide high-confidence bounds on its performance with respect to the demonstrator; however, while there exists much

Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzman rational
  - Demonstrator follows a softmax policy with inverse temperature c

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

$Q^*(s,a,R) =$ How much reward will I expect to see if I take action a in state s and act optimally thereafter.

Ramachandran and Amir, "Bayesian Inverse Reinforcement Learning." IJCAI, 2007.

Bayesian Inverse Reinforcement Learning

(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzman rational
  - Demonstrator follows a softmax policy with inverse temperature c

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

How good is what the demonstrator did under R?

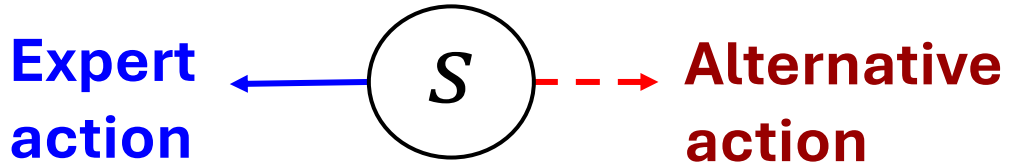
How good are the alternernatives under R?

Reward functions that make the demonstrations look optimal result in higher likelihood scores.

Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzman rational
  - Demonstrator follows a softmax policy with inverse temperature c

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$



$$P((s,\leftarrow)|R) = \frac{e^{Q^*(s,\leftarrow,R)}}{e^{Q^*(s,\leftarrow,R)} + e^{Q^*(s,\dashrightarrow,R)}}$$

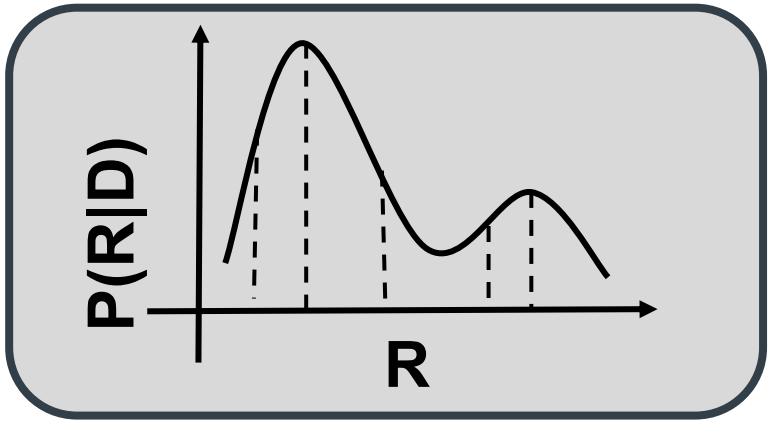
Bayesian Inverse Reinforcement Learning
(Ramachandran and Amir 2007)

- Assume demonstrator is Boltzman rational
  - Demonstrator follows a softmax policy with inverse temperature c

$$P(D|R) = \prod_{(s,a)\in D} \frac{e^{cQ^*(s,a,R)}}{\sum_{b\in A} e^{cQ^*(s,b,R)}}$$

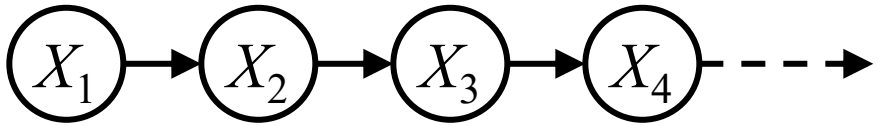
  - Perform Bayesian inference (MCMC) to sample from posterior distribution

$$P(R|D) \propto P(D|R)P(R)$$



Markov Chain Monte Carlo (MCMC)

Markov chain:



$$P(X_1) \qquad P(X_t|X_{t-1})$$

Stationary Distribution: $$P_\infty(X) = P_{\infty+1}(X) = \sum_x P(X|x)P_\infty(x)$$

MCMC is a sampling approach for Bayesian inference where we construct a Markov chain such that the stationary distribution is the posterior distribution we care about.

Markov Chain Monte Carlo (MCMC) sampling

- Remember Bayes' rule

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

- Computing P(D) is hard
- We want to get a sample estimate of the posterior.

Markov Chain Monte Carlo (MCMC) sampling

- Start with random θ_0
- Loop
  - Propose a value for θ_{t+1}
  - Compare θ_t and θ_{t+1}…
  - Pick “best” one …

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

Markov Chain Monte Carlo (MCMC) sampling

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

- Start with random θ_0
- For _ in range 0:T-1
  - Propose a new value θ'
  - Compute p_accept = $\frac{P(\theta'|D)}{P(\theta_t|D)}$
  - If np.random.rand() < p_accept :
    - $\theta_{t+1} = \theta'$
  - Else:
    - $\theta_{t+1} = \theta_t$
- Return $\theta_{0:T}$

Can we combine the best of both worlds?

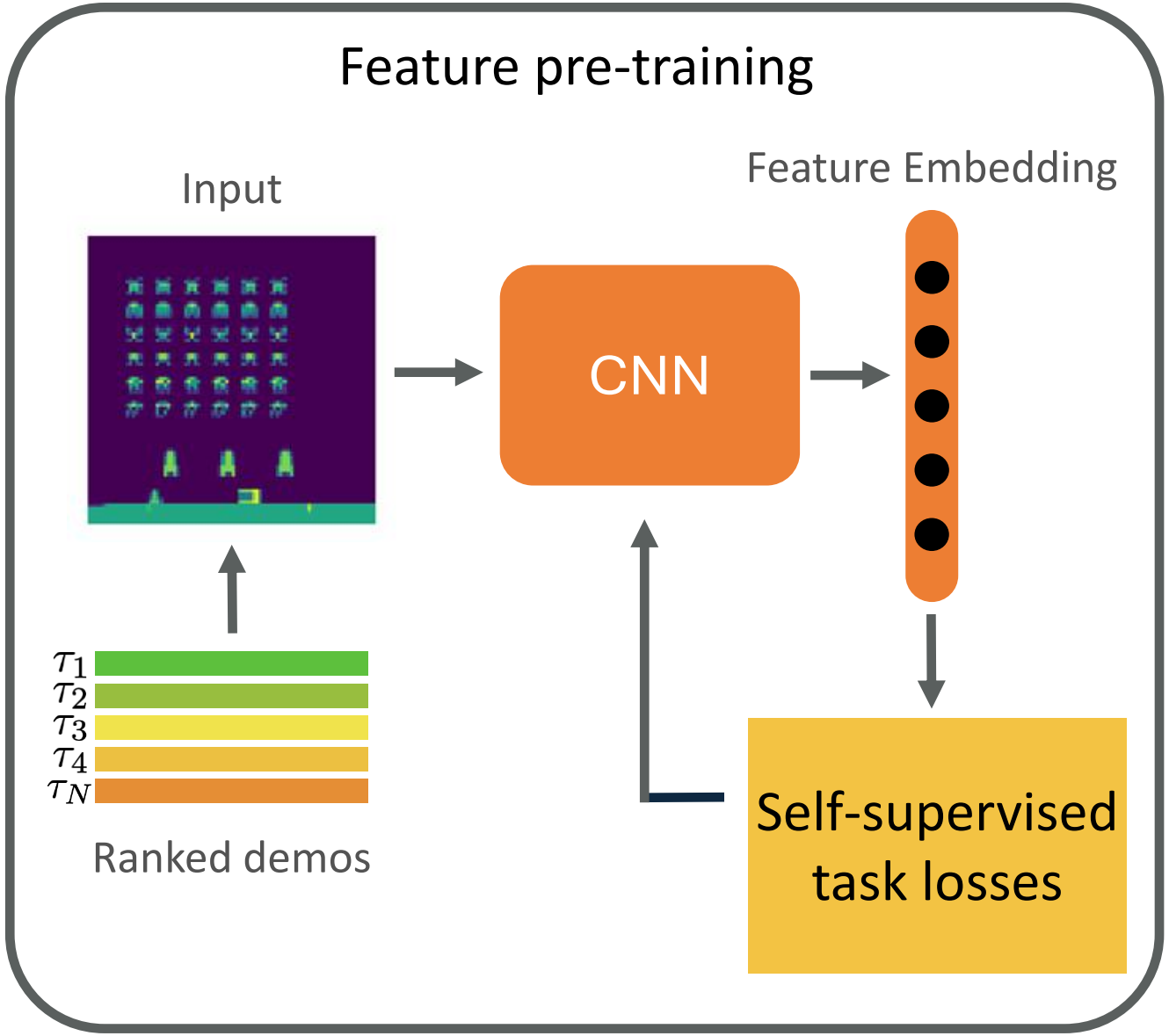
T-REX / D-REX

- Computationally efficient.
- Better than demonstrator performance.
- MLE reward function. No representation of uncertainty.

Bayesian IRL

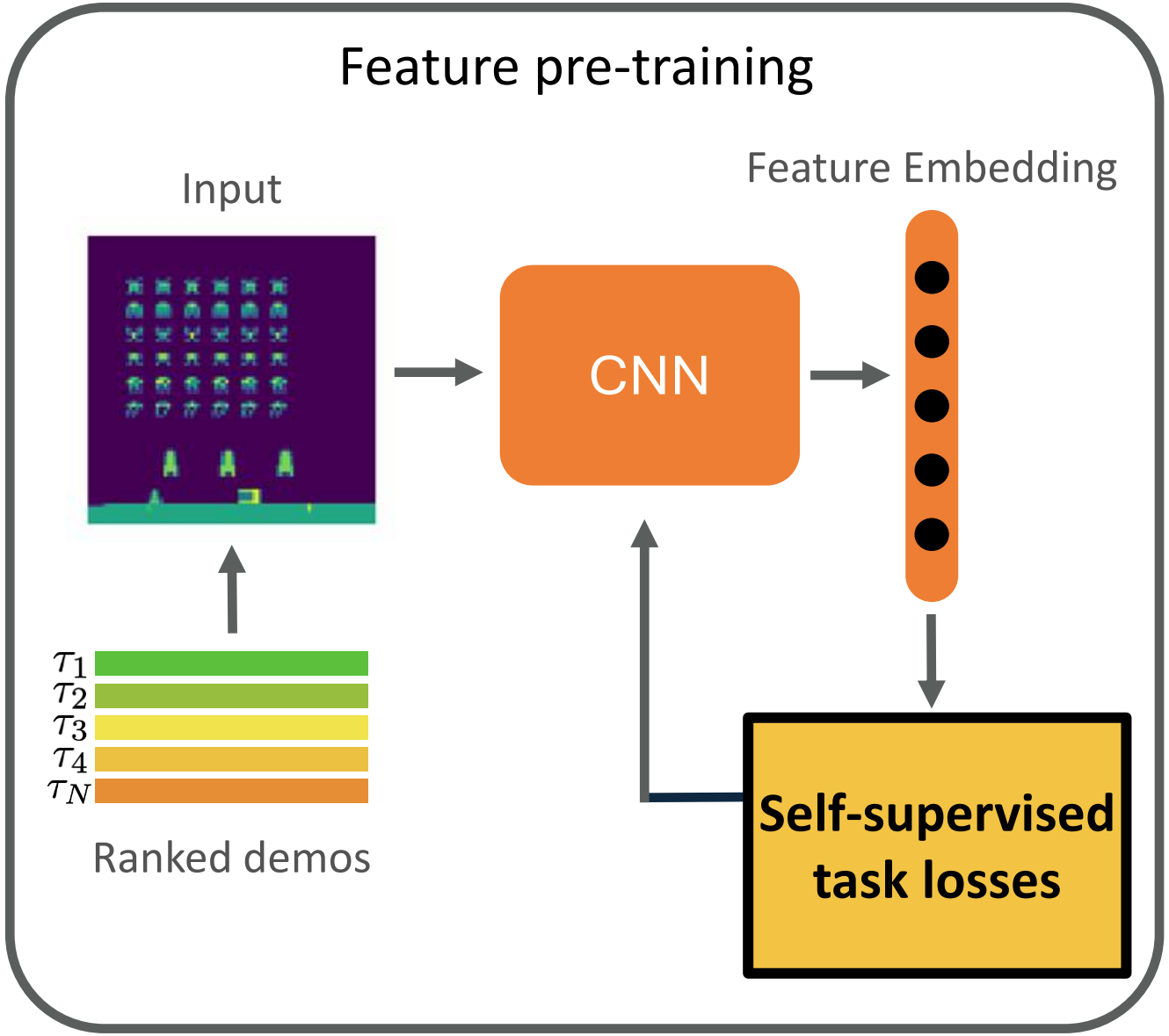
- Computationally inefficient.
- Seeks a reward function that makes the demonstrator optimal.
- Returns a posterior distribution which enables risk and uncertainty analysis.

Bayesian Reward Extrapolation (Bayesian REX)



Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Bayesian Reward Extrapolation (Bayesian REX)

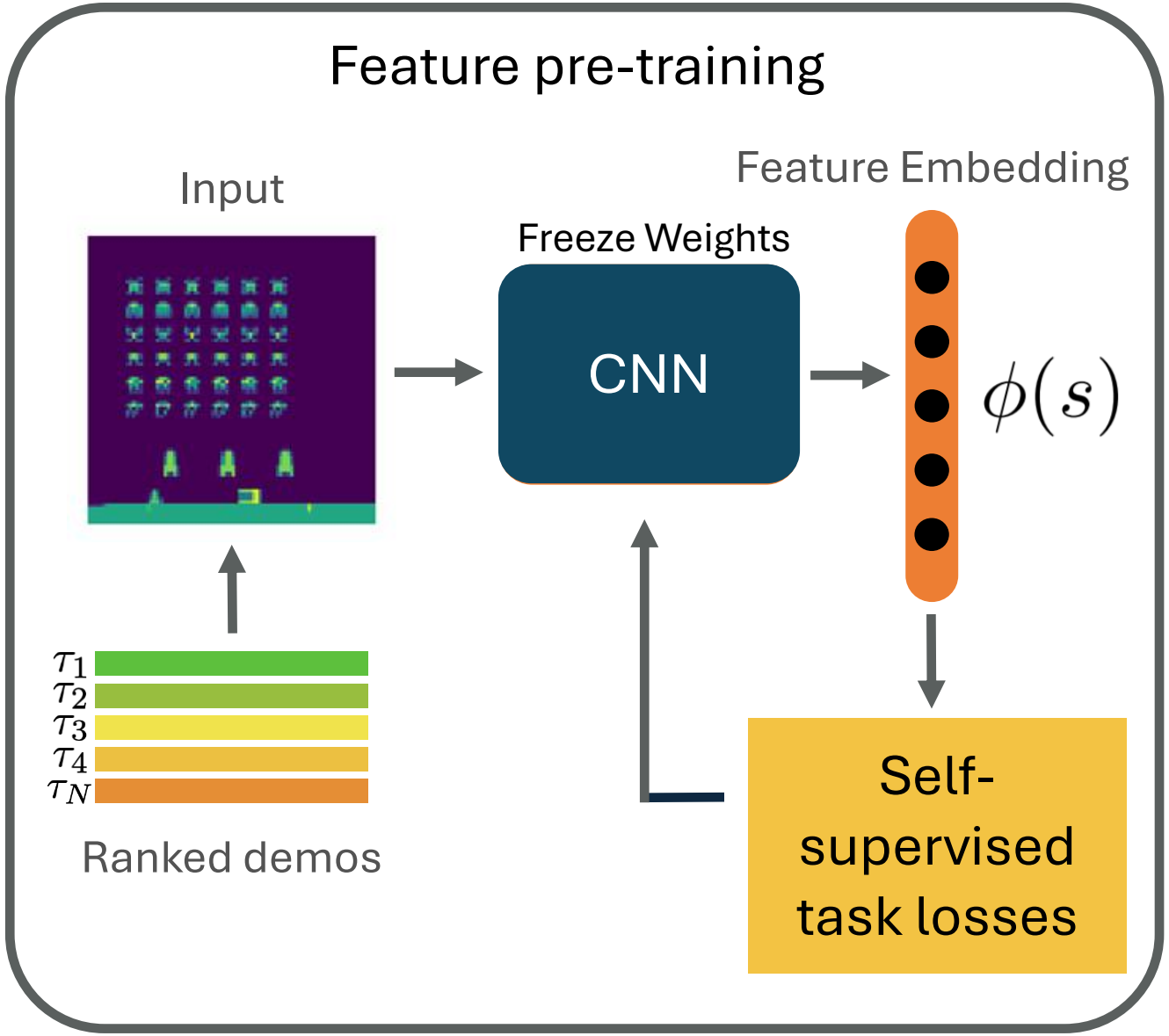


1. Variational Autoencoder
2. Temporal Distance
3. Inverse Dynamics
4. Forward Dynamics

\+ T-REX ranking loss

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Bayesian Reward Extrapolation (Bayesian REX)



Linear Reward Function

$$R_\theta(s) = w^T \phi(s)$$

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Why Linear Reward Functions?

- Fast Bayesian Inference

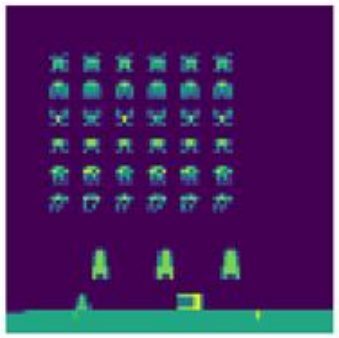
$$R_\theta(s) = w^T \phi(s)$$

$$R_\theta(\tau) = \sum_{s \in \tau} w^T \phi(s) = w^T \sum_{s \in \tau} \phi(s) = w^T \Phi_\tau$$

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input



Feature Embedding

CNN

τ_1
τ_2
τ_3
τ_4
τ_N

Ranked demos

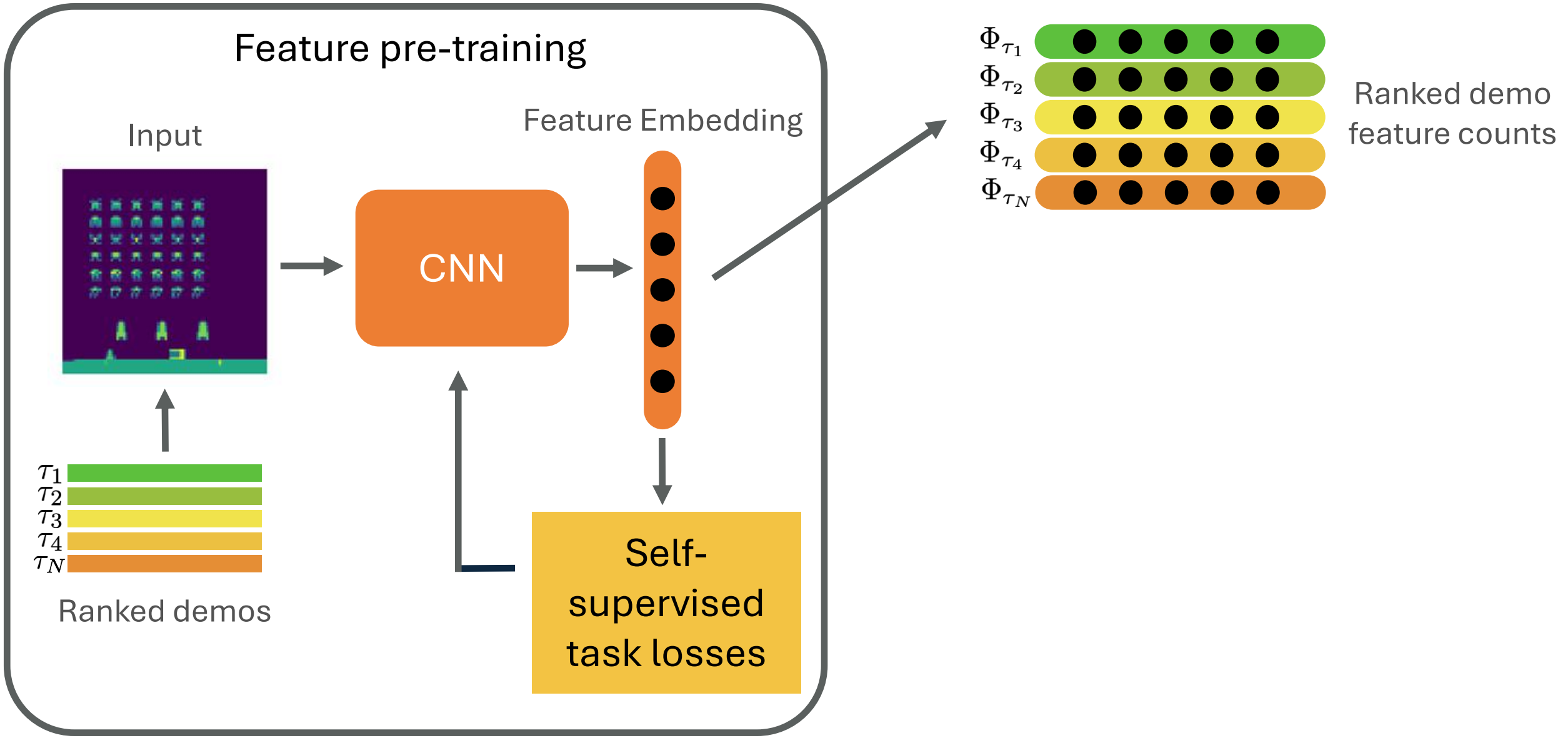
Self-supervised task losses

Φ_{τ_1}
Φ_{τ_2}
Φ_{τ_3}
Φ_{τ_4}
Φ_{τ_N}

Ranked demo feature counts

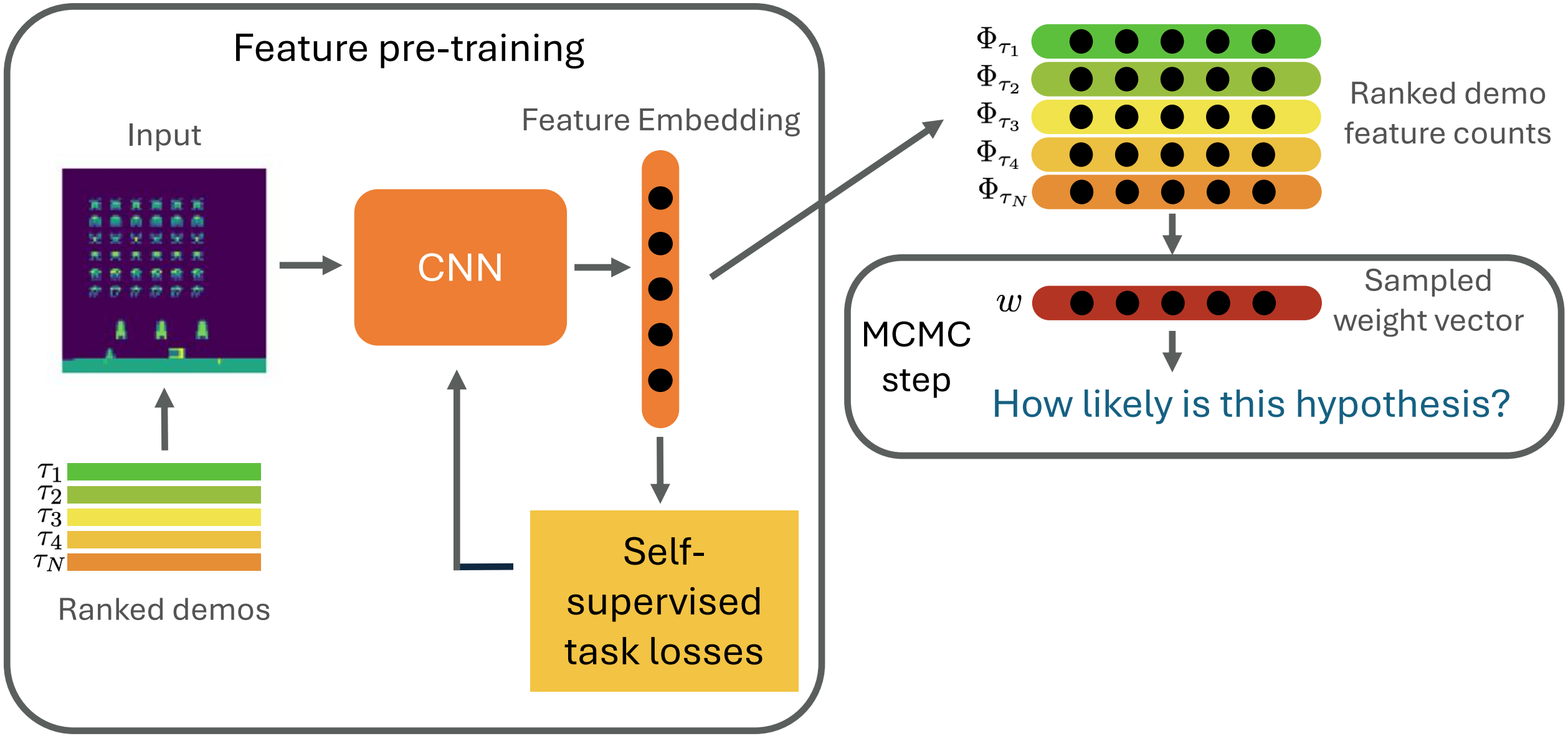
Embed demonstrations in latent feature space.

Bayesian Reward Extrapolation (Bayesian REX)



Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Bayesian Reward Extrapolation (Bayesian REX)



Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Why Linear Reward Functions?

- Fast Bayesian Inference given preferences over demonstrations:

$$P(\mathcal{P}, D \mid R_\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta R_\theta(\tau_j)}}{e^{\beta R_\theta(\tau_i)} + e^{\beta R_\theta(\tau_j)}}$$

If we change the entire network for each proposal, we must run a lot of states through a deep neural network each time we evaluate a proposal.

Why Linear Reward Functions?

- Fast Bayesian Inference given preferences over demonstrations:

$$P(\mathcal{P}, D \mid R_\theta) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta R_\theta(\tau_j)}}{e^{\beta R_\theta(\tau_i)} + e^{\beta R_\theta(\tau_j)}}$$

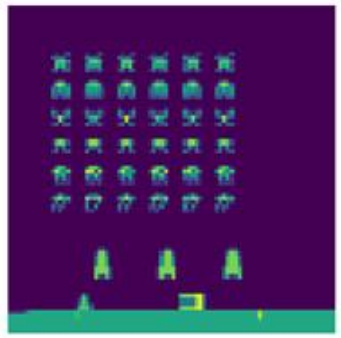
$$P(\mathcal{P}, D \mid w) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta w^T \Phi_{\tau_j}}}{e^{\beta w^T \Phi_{\tau_j}} + e^{\beta w^T \Phi_{\tau_i}}}$$

No MDP model/solver required!

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input



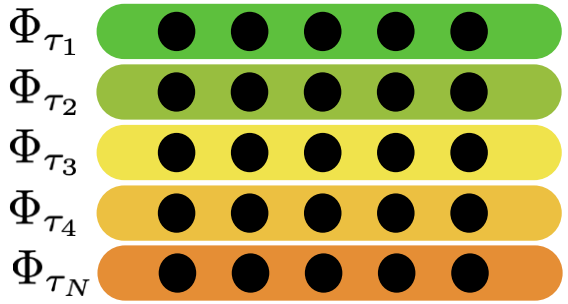
Feature Embedding

CNN

τ_1
τ_2
τ_3
τ_4
τ_N

Ranked demos

Self-supervised task losses



Ranked demo feature counts

MCMC step w

Sampled weight vector

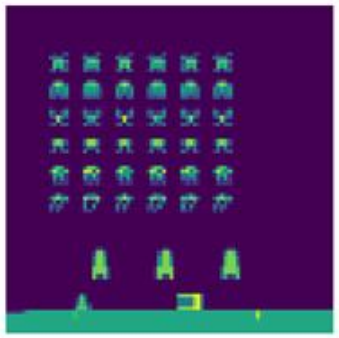
$$P(\mathcal{P}, D \mid w) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta w^T \Phi_{\tau_j}}}{e^{\beta w^T \Phi_{\tau_j}} + e^{\beta w^T \Phi_{\tau_i}}}$$

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input



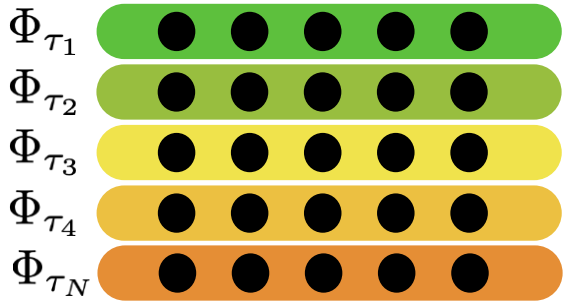
Feature Embedding

CNN

Self-supervised task losses

τ_1
τ_2
τ_3
τ_4
τ_N

Ranked demos



Ranked demo feature counts

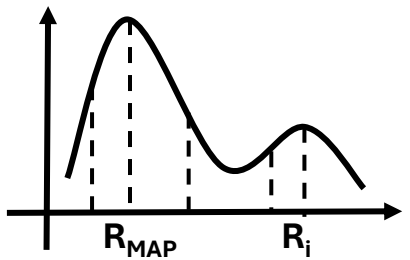
MCMC step w

Sampled weight vector

$$P(\mathcal{P}, D \mid w) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta w^T \Phi_{\tau_j}}}{e^{\beta w^T \Phi_{\tau_j}} + e^{\beta w^T \Phi_{\tau_i}}}$$

Repeat N times to sample from posterior

$$P(R_\theta | D, \mathcal{P})$$

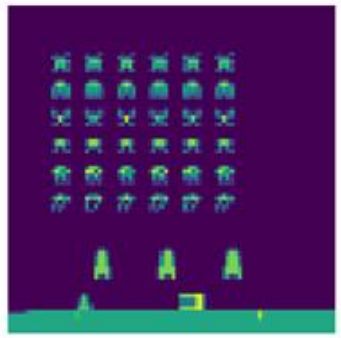


Generate 100,000 samples in 5 minutes!

Bayesian Reward Extrapolation (Bayesian REX)

Feature pre-training

Input



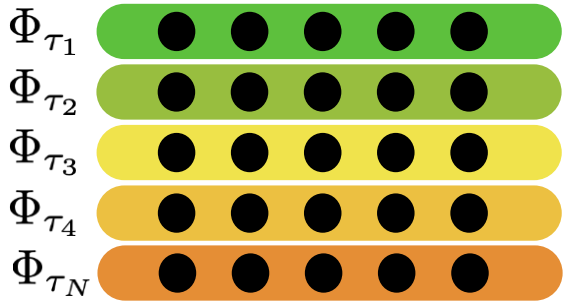
Feature Embedding

CNN

Self-supervised task losses

τ_1
τ_2
τ_3
τ_4
τ_N

Ranked demos



Ranked demo feature counts

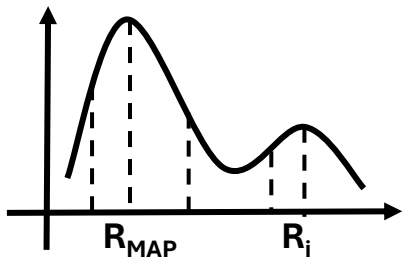
MCMC step w

Sampled weight vector

$$P(\mathcal{P}, D \mid w) = \prod_{(i,j)\in\mathcal{P}} \frac{e^{\beta w^T \Phi_{\tau_j}}}{e^{\beta w^T \Phi_{\tau_j}} + e^{\beta w^T \Phi_{\tau_i}}}$$

Repeat N times to sample from posterior

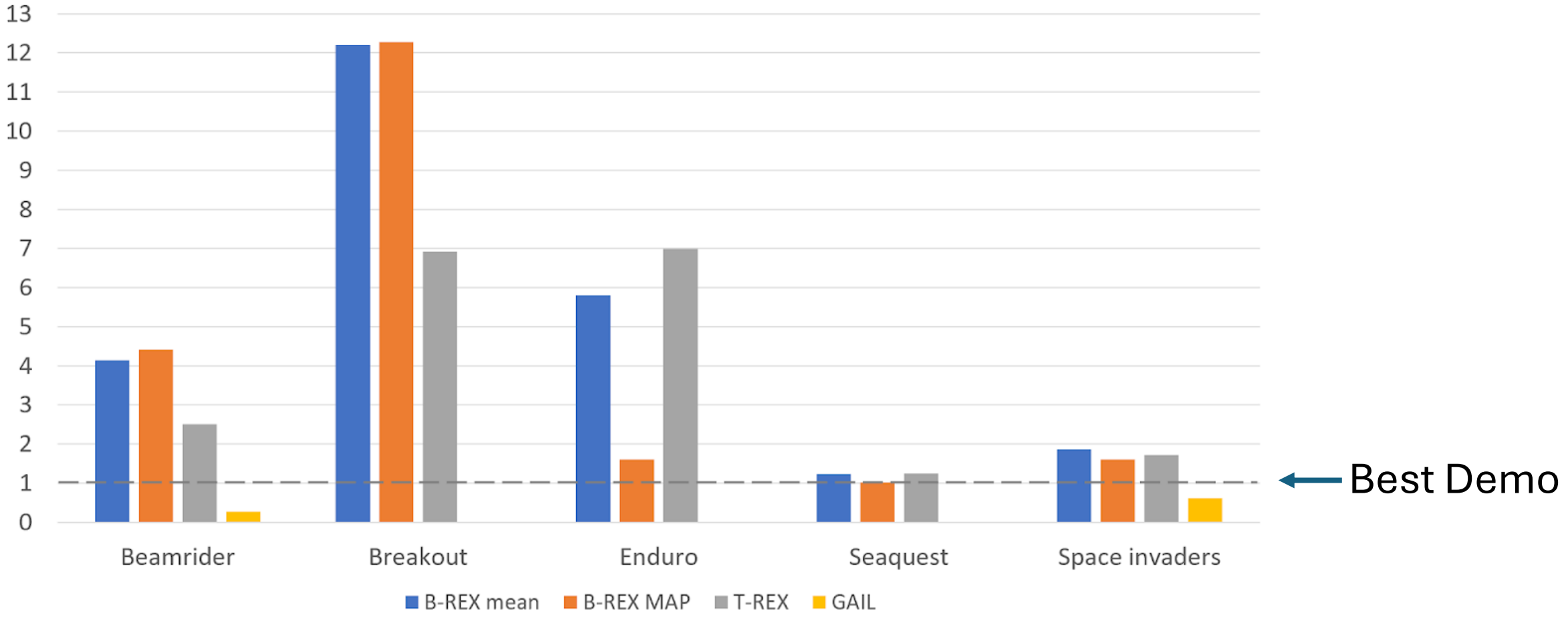
$$P(R_\theta | D, \mathcal{P})$$



10+ hours for Bayesian IRL to generate one sample.

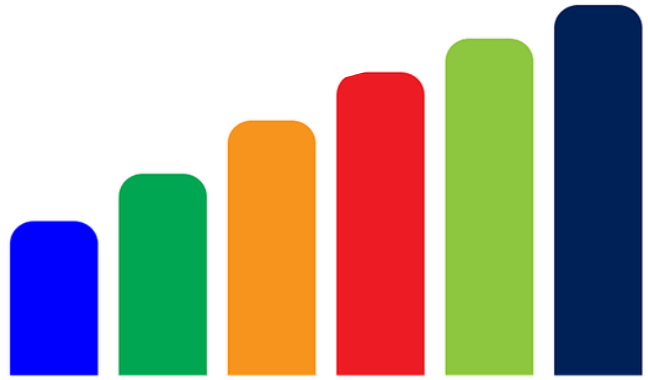
Bayesian REX Performance on Atari

Policy optimized via PPO on mean or MAP reward function from posterior.



Utilizing a Reward Function Posterior

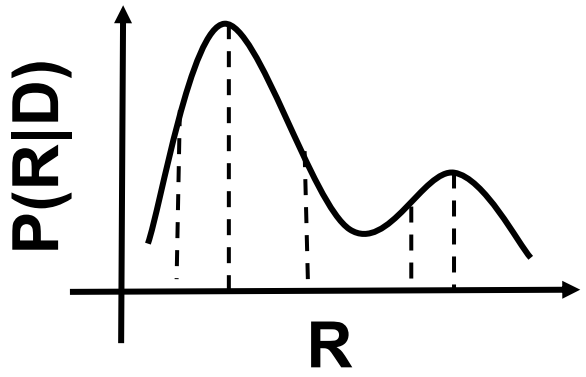
- High-confidence performance bounds for ranking a variety of evaluation policies.



$$\pi_1 \prec \pi_2 \prec \cdots \prec \pi_n$$

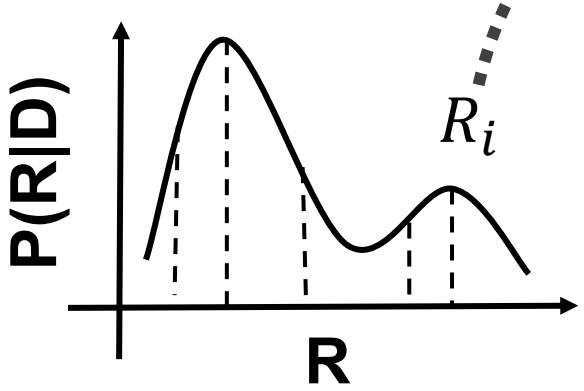
Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

High-Confidence Policy Evaluation



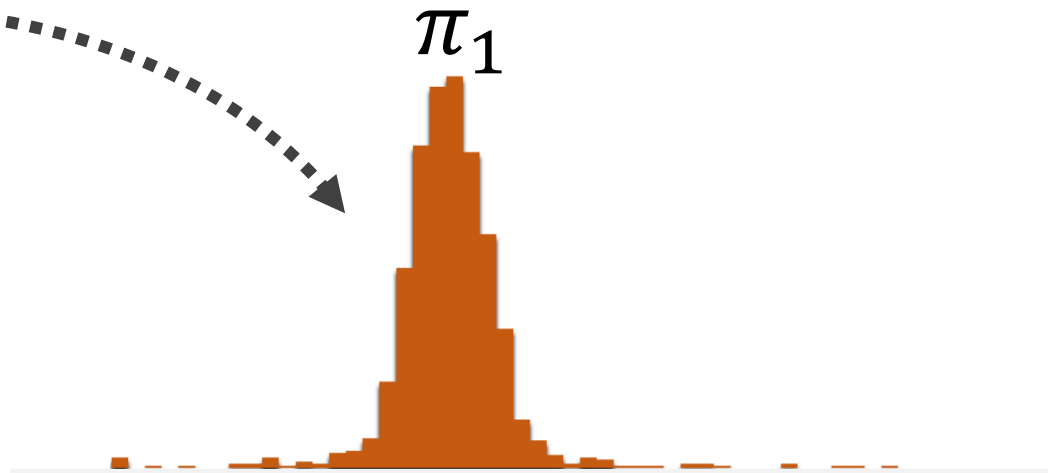
$$\pi_1 \succ \pi_2 ??$$

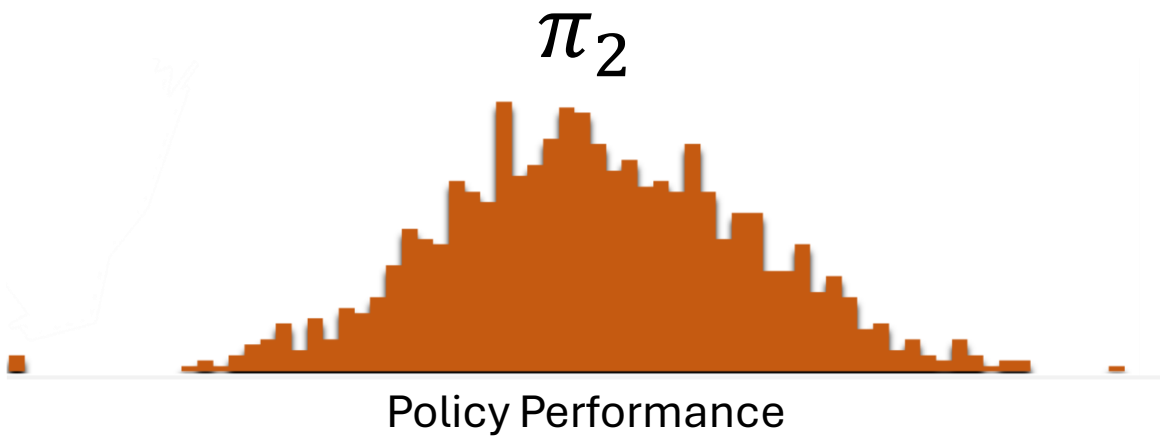
High-Confidence Policy Evaluation



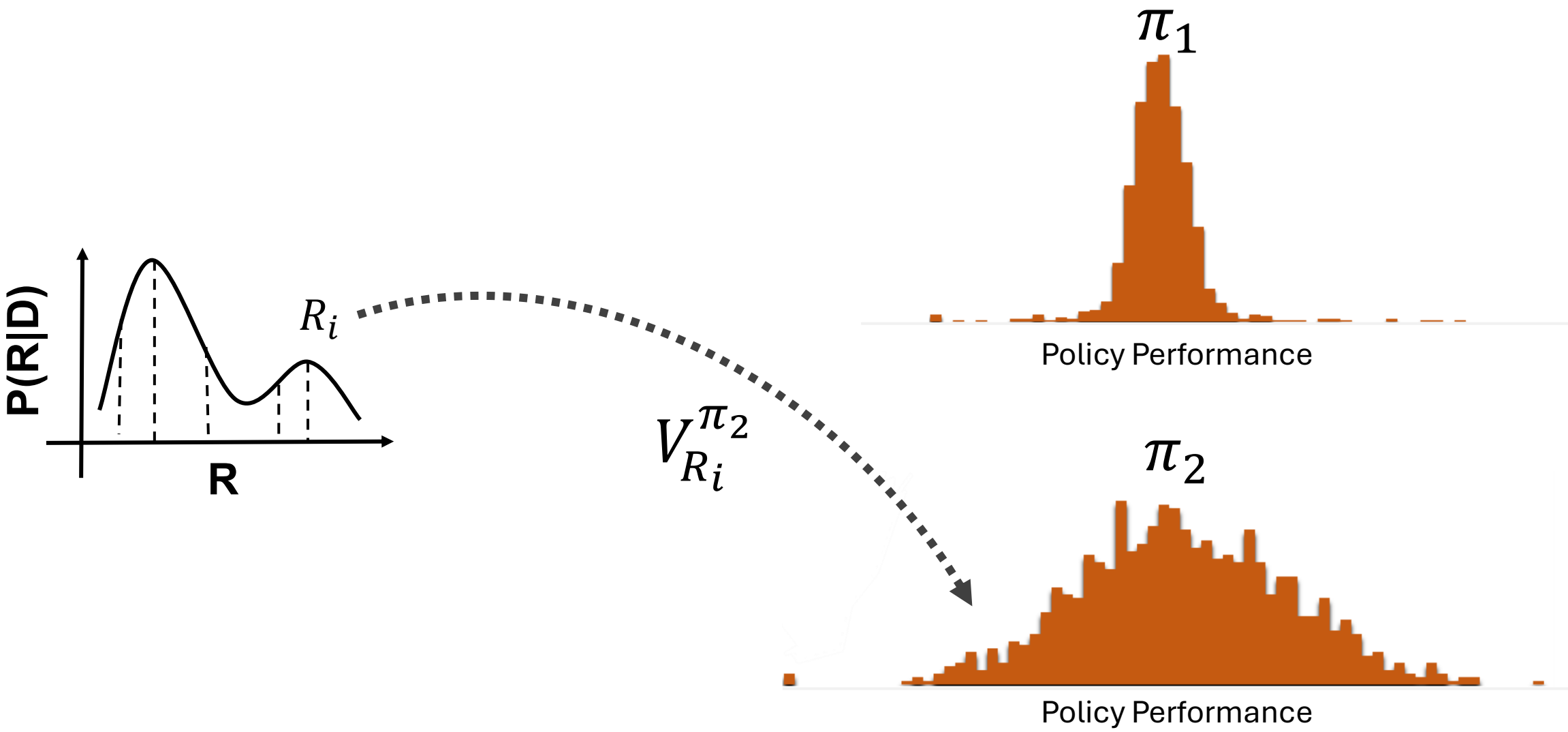
R_i

$V_{R_i}^{\pi_1}$

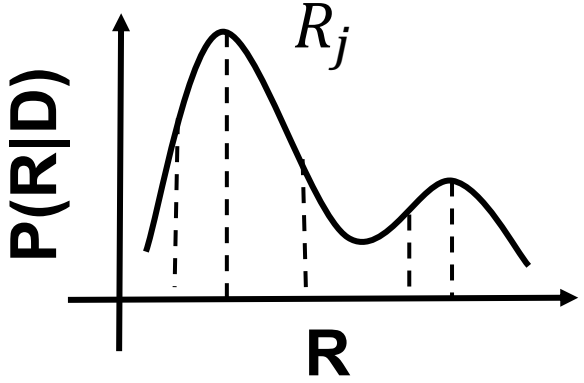




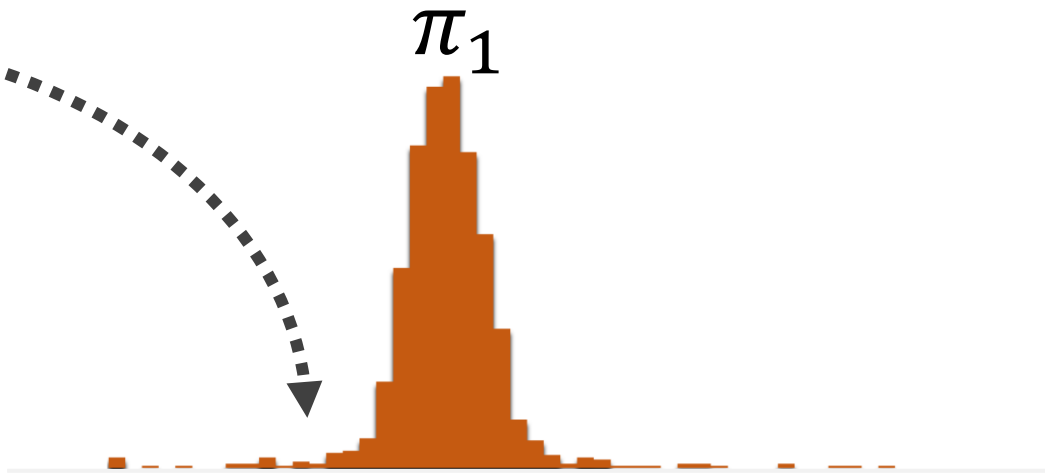
High-Confidence Policy Evaluation

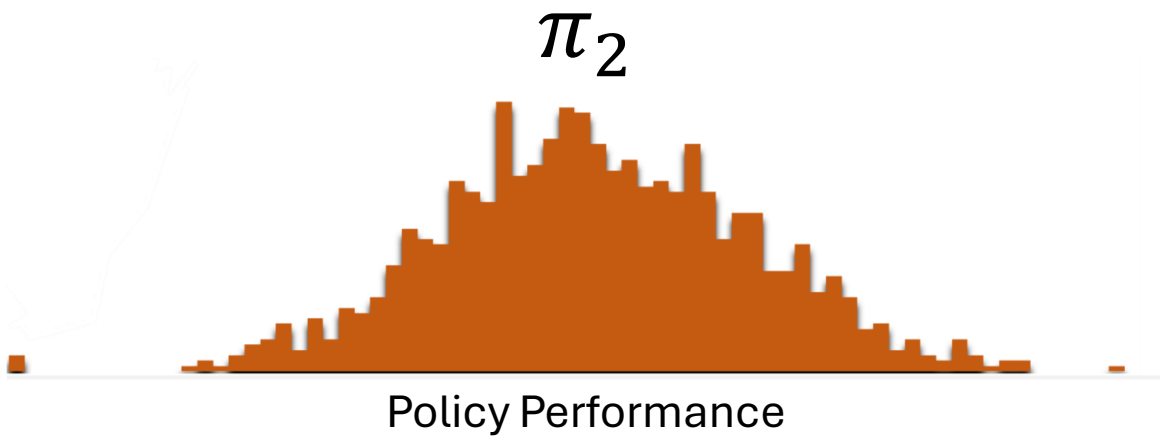


High-Confidence Policy Evaluation

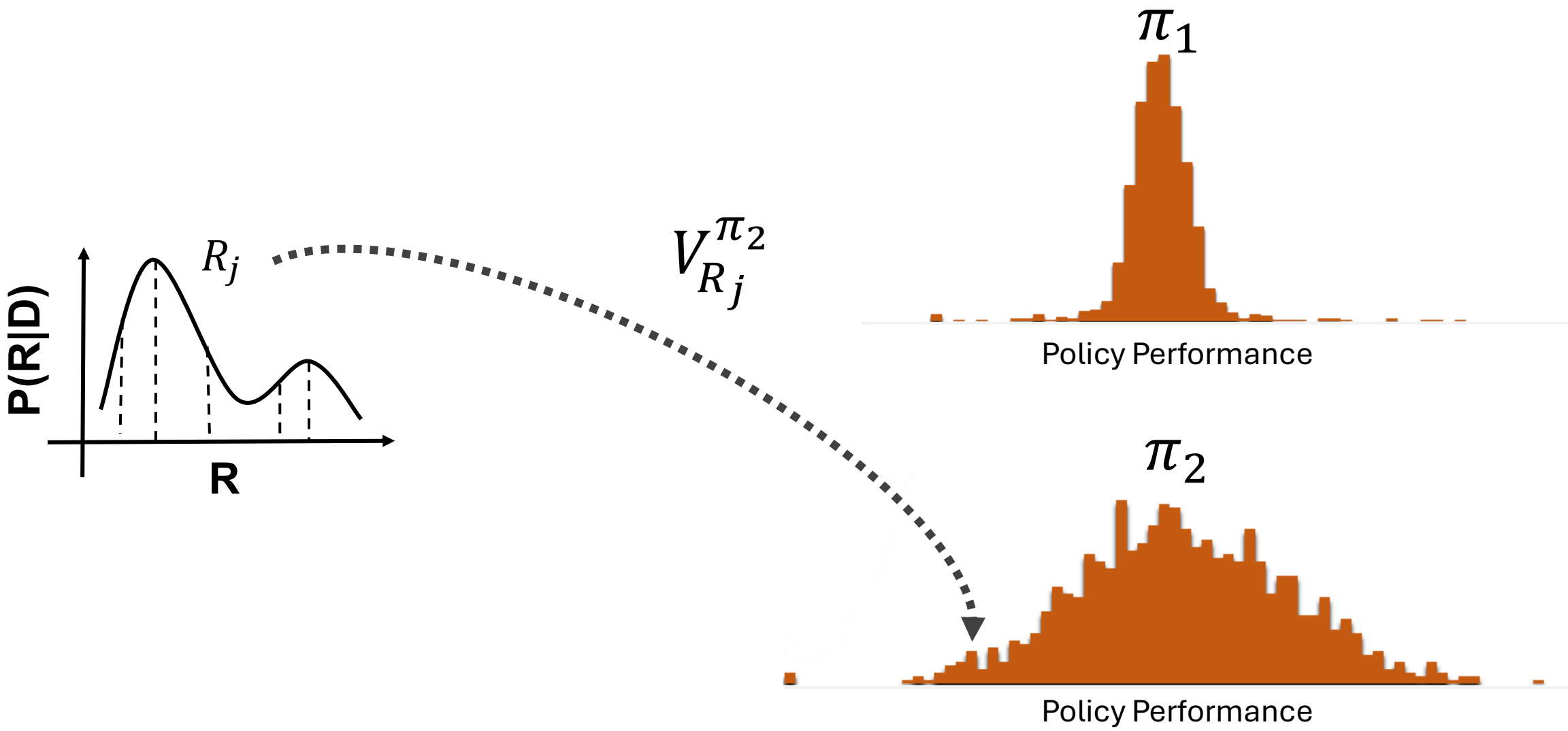


$V_{R_j}^{\pi_1}$

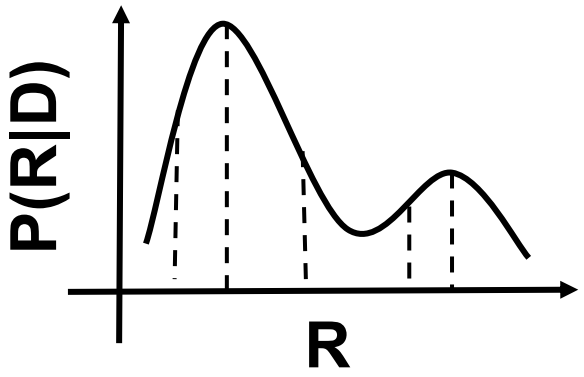




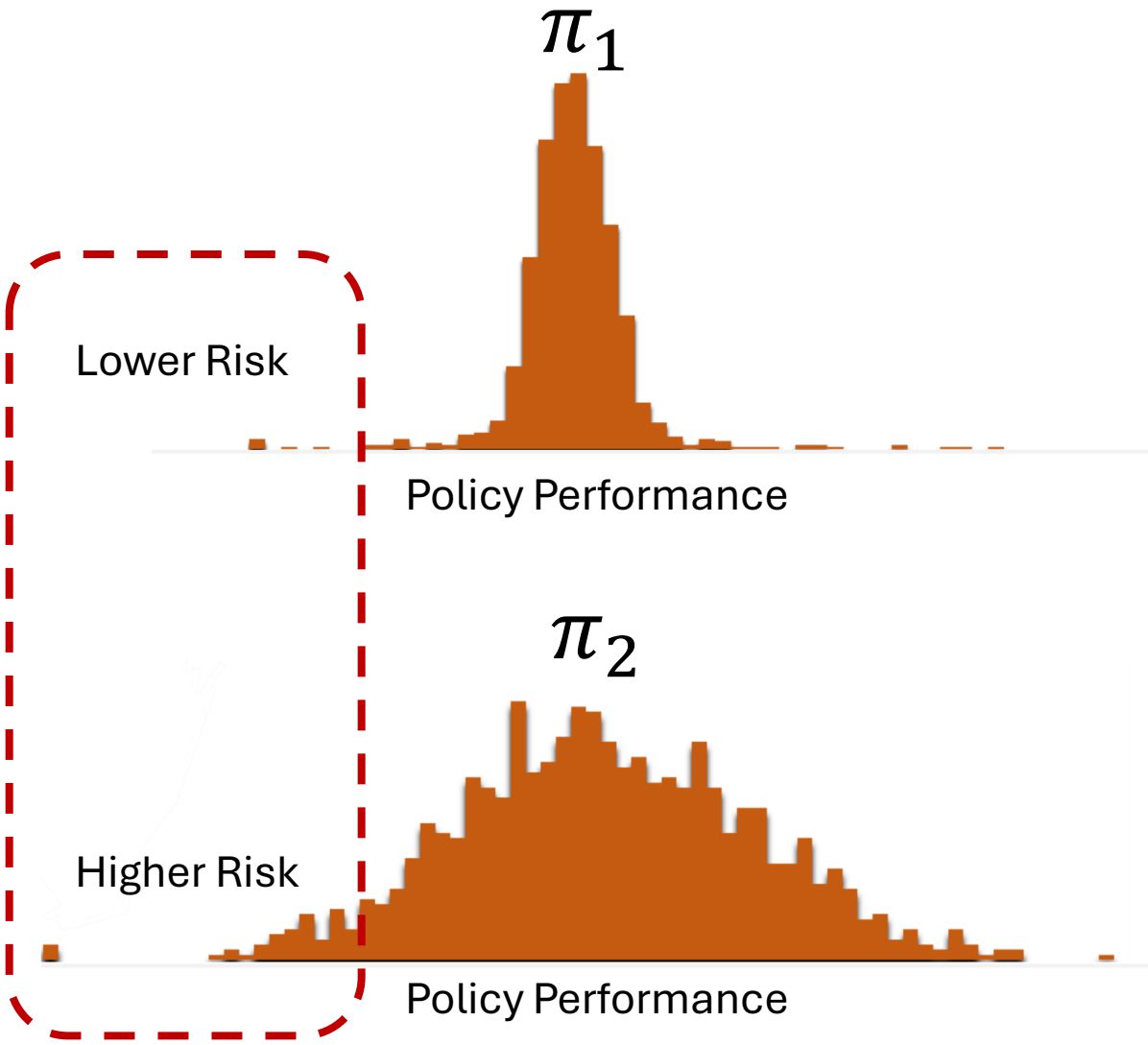
High-Confidence Policy Evaluation



High-Confidence Policy Evaluation



Policy Evaluation
over Posterior

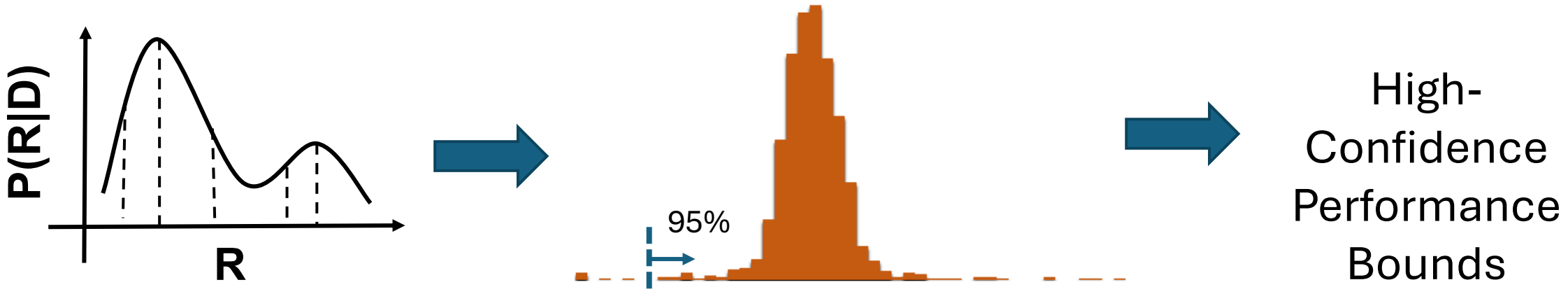


Why Linear Reward Function?

- Fast policy evaluation under many different reward functions.

$$V_R^\pi = w^T \mathbb{E}_\pi [\sum_{t=0}^{\infty} \gamma^t \phi(s_t)] = w^T \Phi_\pi$$

- Perform a single policy evaluation to obtain feature expectations.
- Then policy evaluations are just dot products.



Detecting reward hacking

- Beam Rider

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score
A	17.1	7.9	480.6
B	22.7	11.9	703.4
C	45.5	24.9	1,828.5
D	57.6	31.5	2,586.7

Detecting reward hacking

- Beam Rider

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score	Avg. Game Length
A	17.1	7.9	480.6	1,372.6
B	22.7	11.9	703.4	1,412.8
C	45.5	24.9	1,828.5	2,389.9
D	57.6	31.5	2,586.7	2,965.0

Detecting reward hacking

- Beam Rider

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score	Avg. Game Length
A	17.1	7.9	480.6	1,372.6
B	22.7	11.9	703.4	1,412.8
C	45.5	24.9	1,828.5	2,389.9
D	57.6	31.5	2,586.7	2,965.0
No-Op	102.5	-1557.1	0.0	99,994

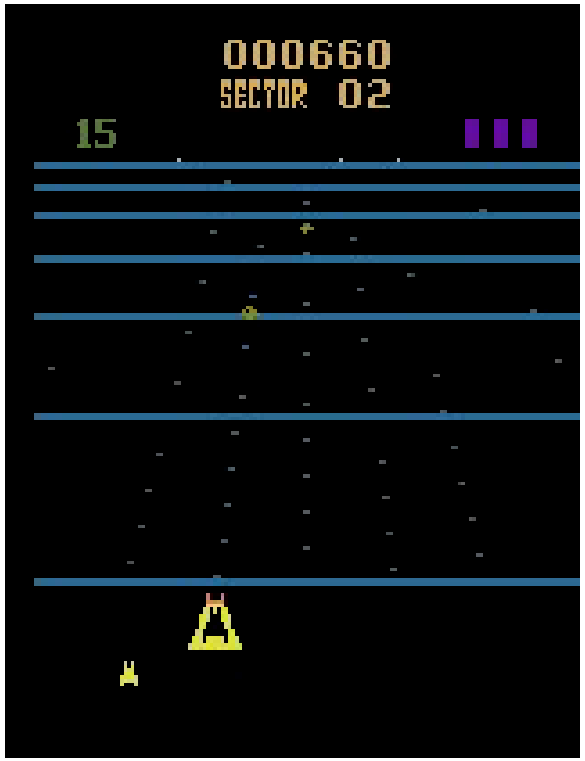
Detecting reward hacking

- Beam Rider

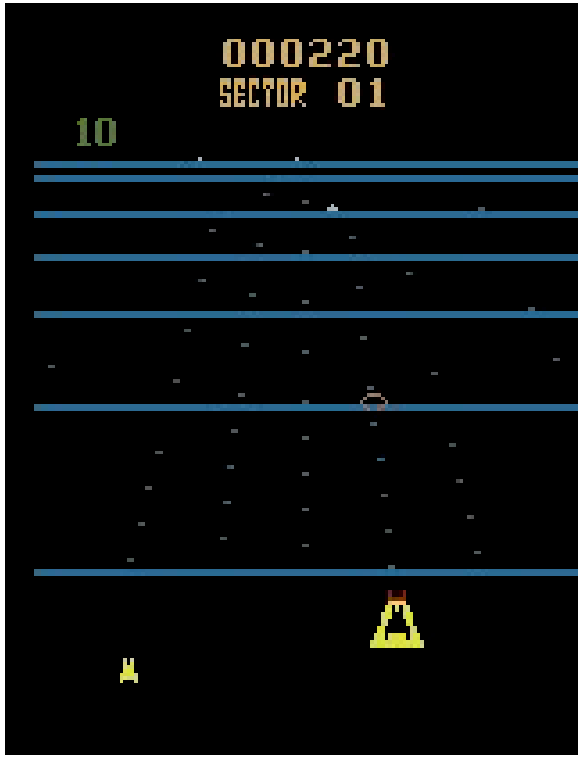
Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score	Avg. Game Length
A	17.1	7.9	480.6	1,372.6
B	22.7	11.9	703.4	1,412.8
C	45.5	24.9	1,828.5	2,389.9
D	57.6	31.5	2,586.7	2,965.0
No-Op	102.5	-1557.1	0.0	99,994

Human Demos

Good

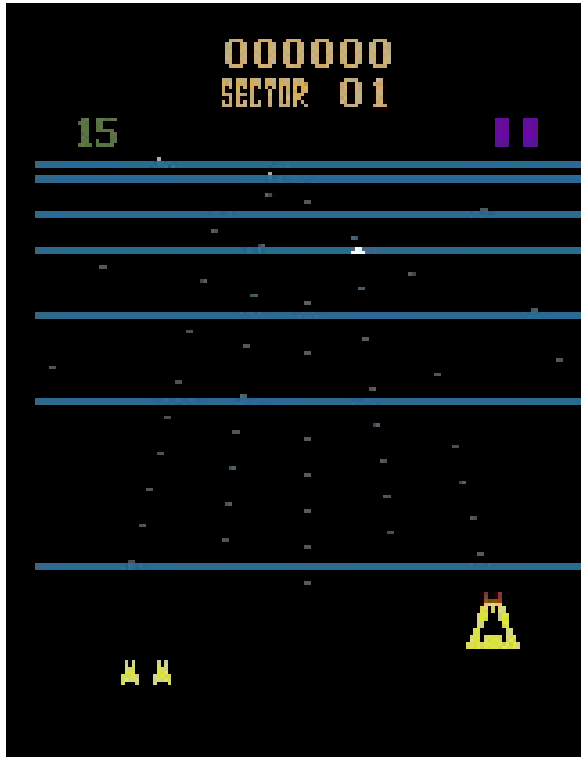


Bad



Deceptive

Pessimal



Ranking Based on High-Confidence Bounds

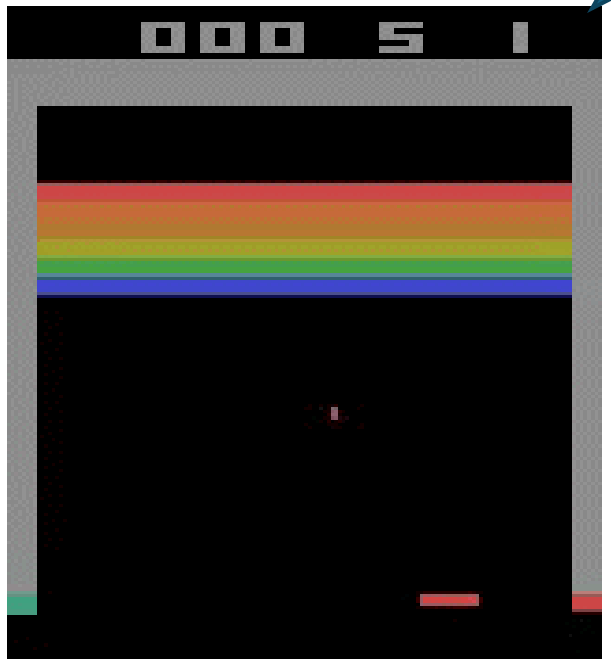
1 (best) **2** **3** **4 (worst)**

What if demonstrations are extremely suboptimal?

Best Demo

Bayesian REX

No-Op



Still “playing”

What if demonstrations are extremely suboptimal?

High uncertainty about true performance!

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score
A	1.5	0.5	1.9
B	6.3	37	15.8
C	10.6	5.8	27.7
D	13.9	6.2	41.2
Bayesian REX	98.2	-370.2	401
No-Op	41.2	1.0	0.0

Integrate Additional Preference Queries

- Ask demonstrator to rank MAP and No-Op versus original demonstrations. Recompute posterior distribution and high-confidence bounds.

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score
A	1.5	0.5	1.9
B	6.3	37	15.8
C	10.6	5.8	27.7
D	13.9	6.2	41.2
Bayesian REX	98.2	-370.2	401
No-Op	41.2	1.0	0.0

Integrate Additional Preference Queries

- Ask demonstrator to rank MAP and No-Op versus original demonstrations. Recompute posterior distribution and high-confidence bounds.

Evaluation Policy	Expectation	95%-confidence lower bound	Avg. Score
A	0.7	0.3	1.9
B	8.7	5.5	15.8
C	18.3	12.1	27.7
D	26.3	17.1	41.2
Bayesian REX	606.8	289.1	401
No-Op	-5.0	-13.5	0.0

Only required quickly rerunning MCMC. Pretrained features stayed the same.

Low-Dimensional MDPs with Known Features

1. Preferences over suboptimal demos vs. Optimal demos
   - ➢Bayesian REX performs on par or better than Bayesian IRL

2. Preferences over suboptimal demos only
   - ➢Bayesian REX always performs better than Bayesian IRL

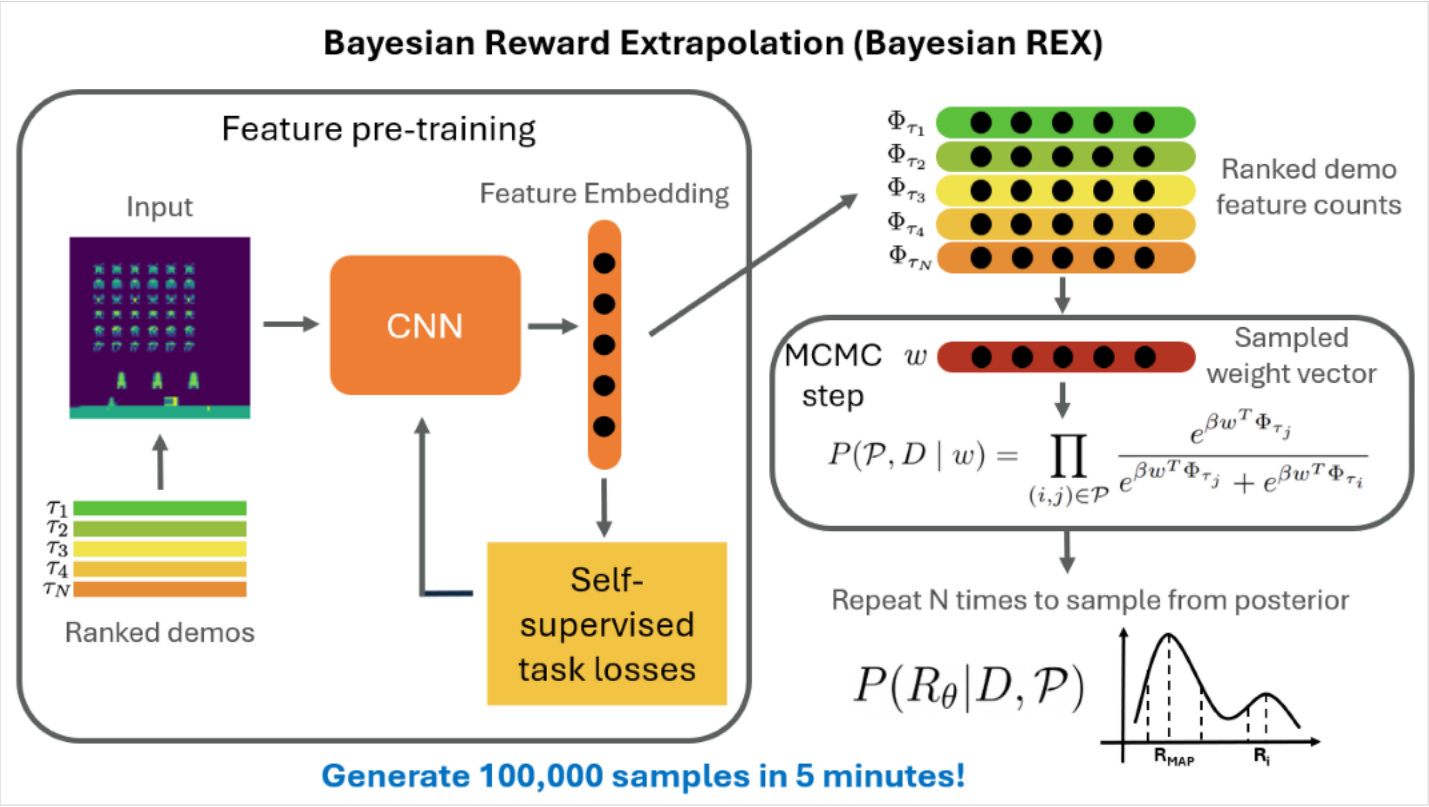
3. Optimal demos only
   - ➢Bayesian IRL performs better than Bayesian REX.

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

When should you choose which method?

	Bayesian REX	Bayesian IRL
Preferences available over suboptimal demos	✓	
Optimal demos and fast MDP solver		✓
No model of MDP	✓	
Slow MDP solver	✓	

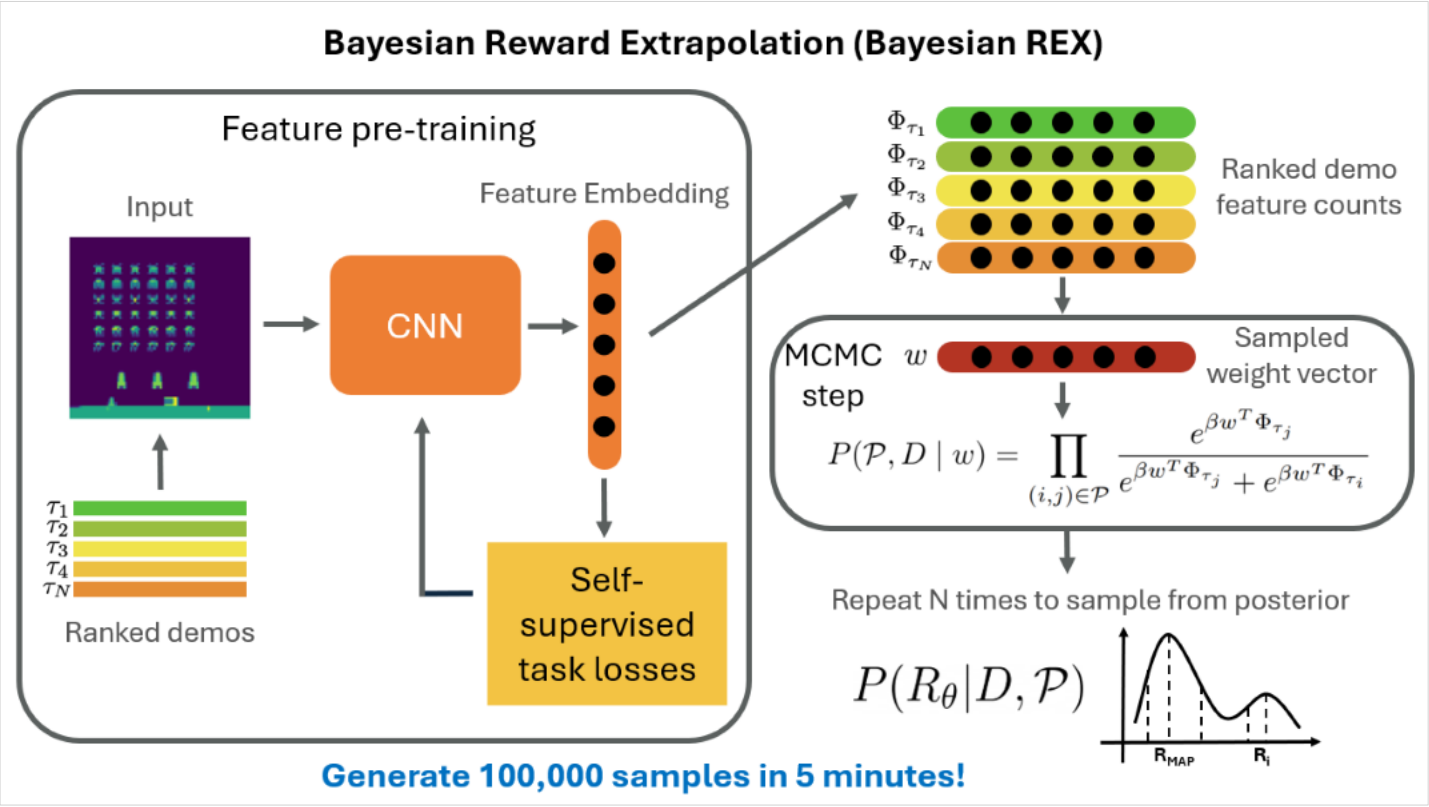
Bayesian REX



- First Bayesian reward inference algorithm to scale to visual imitation learning tasks.
- Achieves state-of-the-art imitation learning performance on complex Atari benchmarks.
- Enables scalable safe imitation learning via high-confidence bounds on performance.

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2

Bayesian REX



- First Bayesian reward inference algorithm to scale to visual imitation learning tasks.
- Achieves state-of-the-art imitation learning performance on complex Atari benchmarks.
- Enables scalable safe imitation learning via high-confidence bounds on performance.

Brown et al. “Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences." ICML 2