Self-supervised Reinforcement Learning

Jianye Hao
Jianye.hao@tju.edu.cn

NLPCC 2021 Tutorial
2021-10-14

Outline

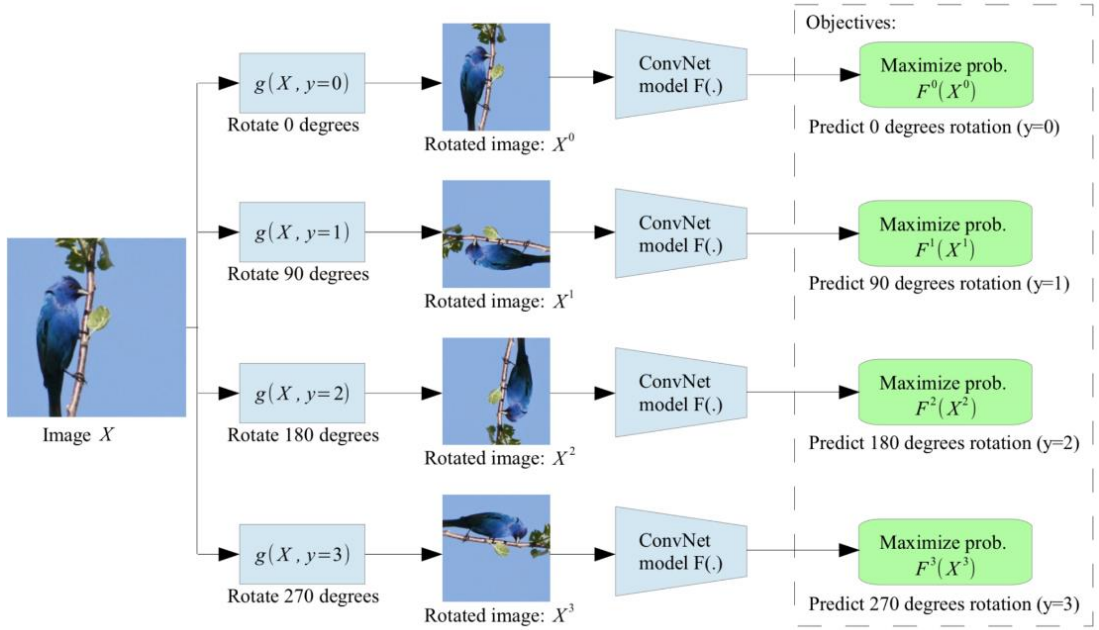
- **Self-supervised Learning**
  - Concepts and examples
  - Contrastive Learning

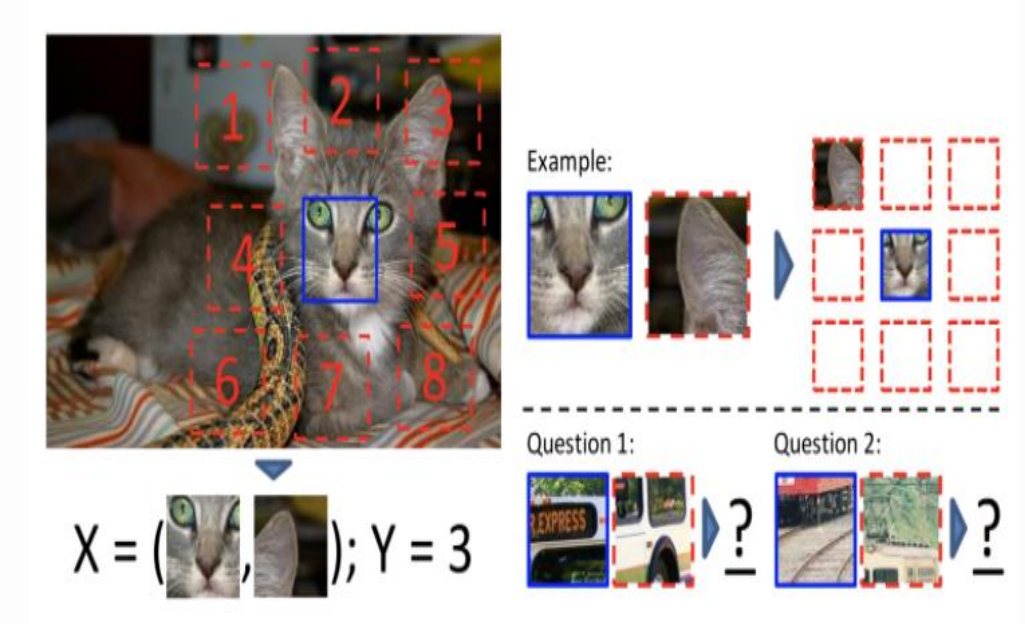
- Self-supervised Reinforcement Learning
  - Self-supervised Representation of States
  - Self-supervised Representation of Actions
  - Self-supervised Representation of Policies
  - Self-supervised Representation of Tasks/Environments

What is Self-supervised Learning?

- ✓ Labels are extracted from the samples
- ✓ The tasks require understanding

Examples in image-based tasks





`Rotation`: Illustration of self-supervised learning by rotating the entire input images. The model learns to predict which rotation is applied.

This Rotation pretext task drives the model to learn semantic concepts of objects: the model has to learn to recognize high-level object parts, such as heads, noses, and eyes, and the relative positions of these parts, rather than local patterns. (Image source: Gidaris et al. 2018)

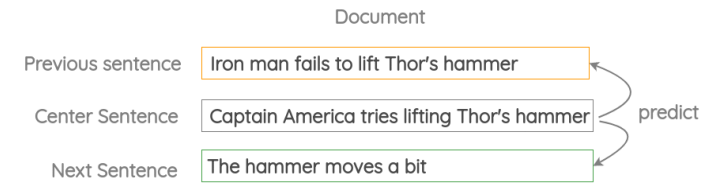
Patches: Illustration of self-supervised learning by predicting the relative position of two random patches.

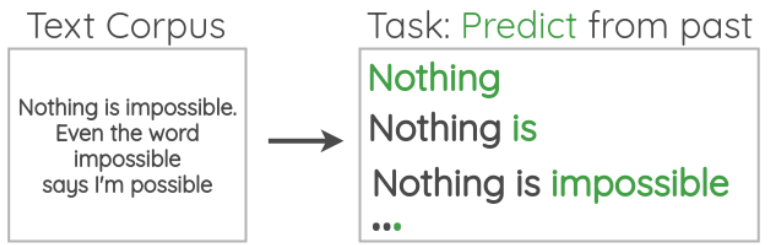
The Patches pretext task drives the model to learn to understand the spatial context of objects in order to tell the relative position between parts. (Image source: Doersch et al., 2015)

What is Self-supervised Learning?

Self-supervised learning in NLP tasks

- Center Word Prediction
- Neighbor Sentence Prediction
- Auto-regressive Language Modeling
- Masked Language Modeling
- Key techniques in GTP-2 and Bert





Example taken from https://amitness.com/2020/05/self-supervised-learning-nlp/

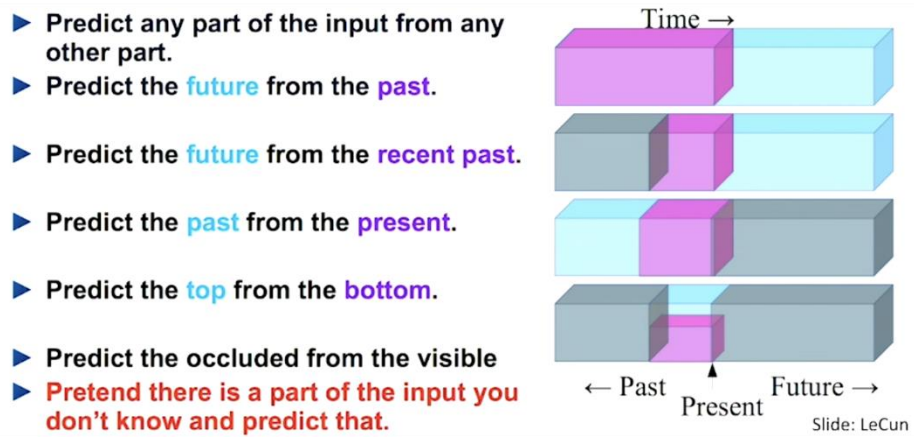
What is Self-supervised Learning?

- **Self-supervised training (SSL)**: a family of techniques for converting an **unsupervised learning problem** into a **supervised one** by creating surrogate labels from the unlabeled dataset.

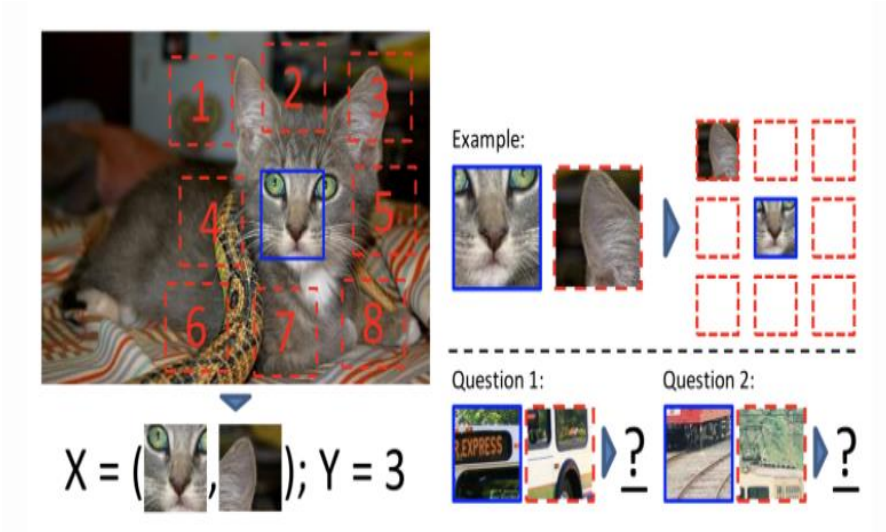
- We can achieve this by framing a supervised learning task in a special form to predict only a subset of information using the rest.

- SSL: learning **intermediate representation** with the expectation that this representation can carry **good semantic or structural meanings** and can be beneficial to a variety of practical downstream tasks.

- **Are Generative Models self-supervised?**



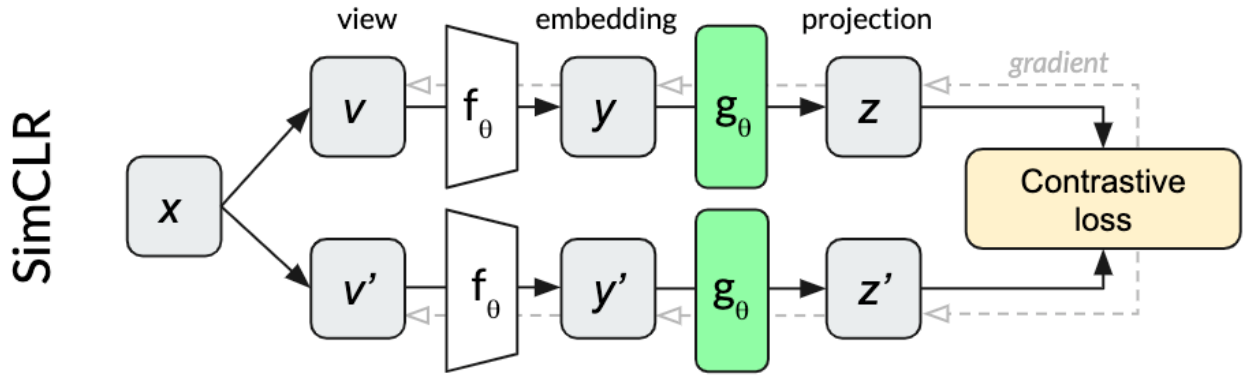
(Image source: LeCun's talk)

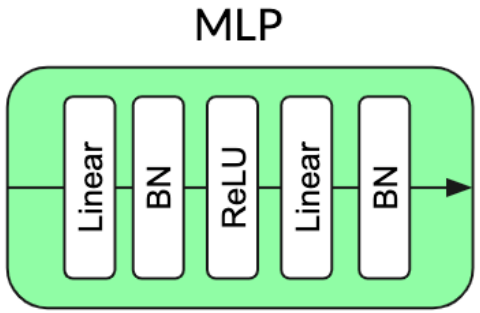


(Image source: Doersch et al., 2015)

Contrastive Learning: the SOTA Self-supervised Learning

- Representative Contrastive learning: SimCLR (Chen et al., 2020), MoCo (He et al. 2019), BYOL (Grill et al., 2020) …





$$\ell_{i,j} = -\log \frac{\exp\left(\text{sim}\left(\boldsymbol{z}_i, \boldsymbol{z}_j\right)/\tau\right)}{\sum_{k=1}^{2N} 1_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z}_i, \boldsymbol{z}_k)/\tau)}$$

https://generallyintelligent.ai/understanding-self-supervised-contrastive-learning.html

Outline

- Intro to Self-supervised Learning
  - Concepts and examples
  - Contrastive Learning

- **Self-supervised Reinforcement Learning**
  - Self-supervised Representation of States
  - Self-supervised Representation of Actions
  - Self-supervised Representation of Policies
  - Self-supervised Representation of Tasks/Environments

Self-supervised Reinforcement Learning

- How to learn better intermediate representation for better estimation of policies and Q-values?
  - **Representation of states**
    - Contrastive-learning based: task-agnostic representation (CURL)
    - Bisimulation-based: only encode task-relevant information (DBC, deepMDP)
  - Representation of policies
    - Policy-extended value function (generalization across value function estimation of different policies)
  - Representation of actions
    - Action representation in RL (action space reduction, generalization across policies and value functions)
  - Representation of tasks/environments
    - Task context representation learning for Meta-RL (generalization across new tasks)

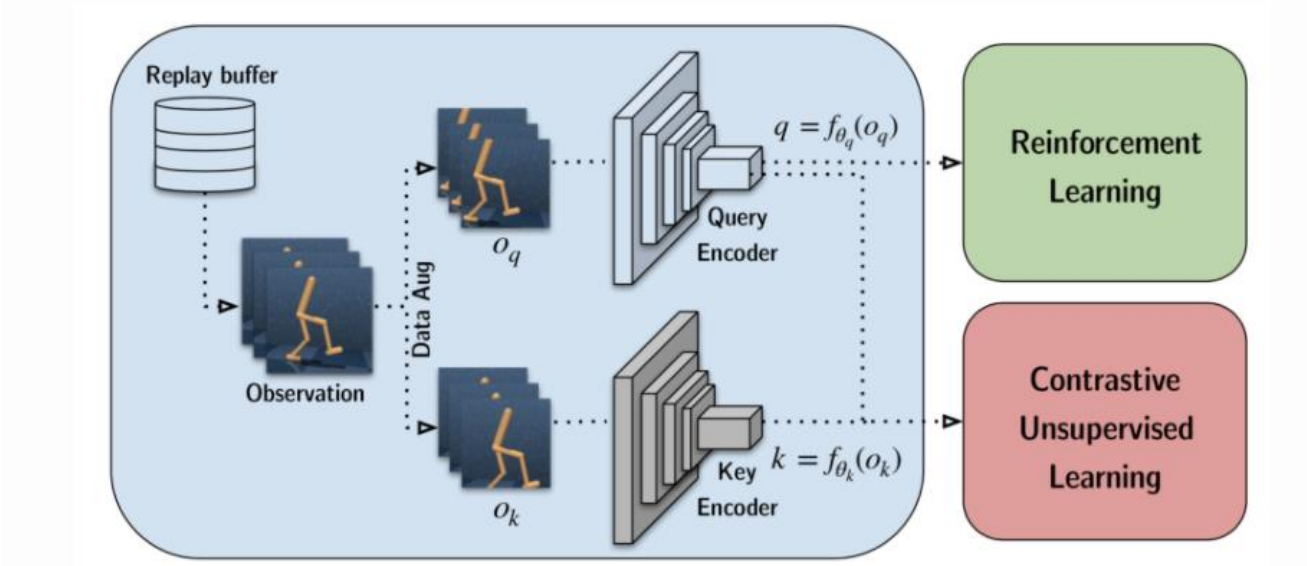
Self-supervised Reinforcement Learning

- **CV/NLP**: Learning useful semantic/structural representation for effective downstream tasks (object detection/segmentation, word/sentence generation)

- **RL**: Learning useful semantic/structural representation for effective downstream tasks (planning and control)
  - **Contrastive-based Representations**: the same way of constructing supervised loss for image-based deep learning tasks (reconstruction-based and **task-agnostic -** the models represent all dynamic elements they observe in the world, whether they are relevant to the task or not)
  - **Bisimulation and other metric representation**: The *bisimulation distance* between two states corresponds to how behaviorally different these two states are (e.g., DBC, deepMDP)

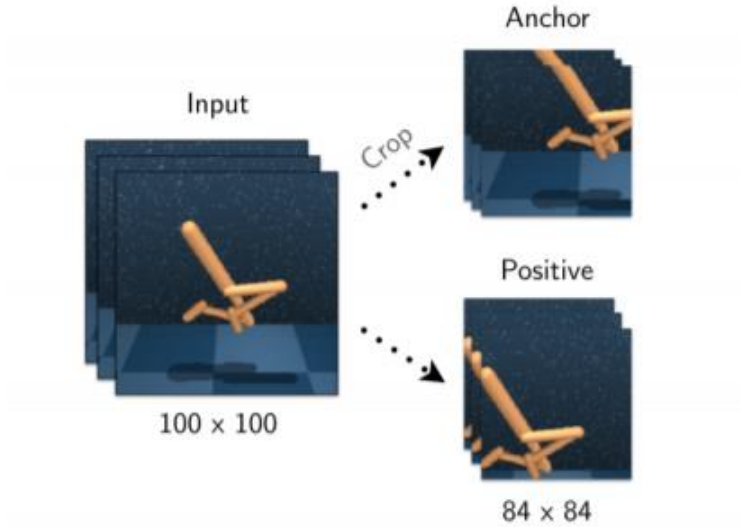
CURL – State Representation Learning for RL

- CURL extends the idea of MoCo to reinforcement learning by learning virtual/state representation for RL tasks.



The anchor and positive observations are two different augmentations (random crop operation) of the same image while negatives come from other images

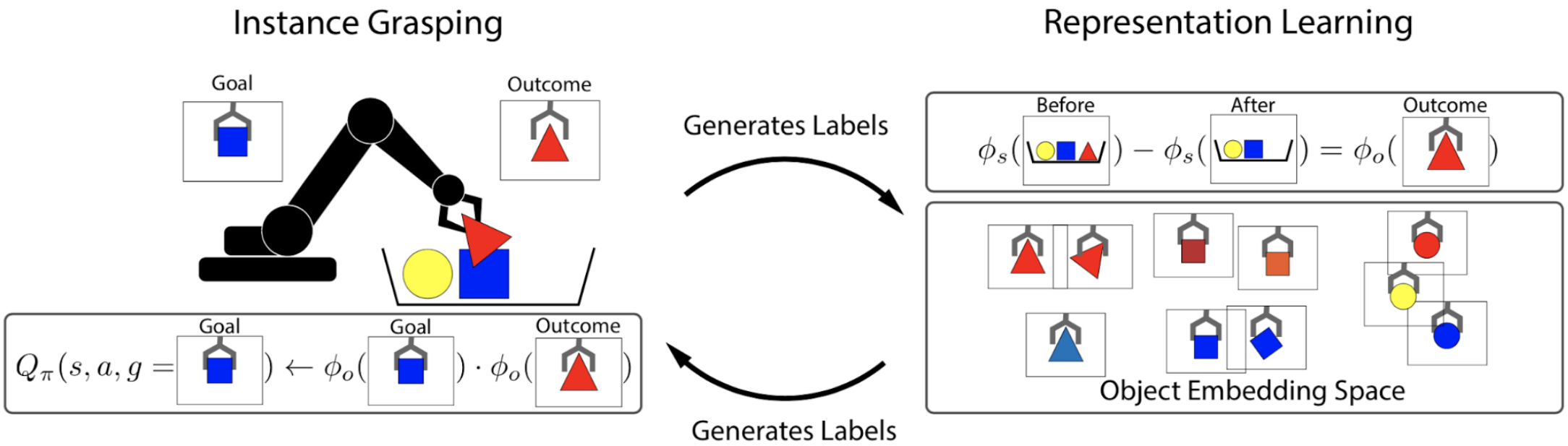
During the gradient update step, only the query encoder is updated. The key encoder weights are the moving average (EMA) of the query weights similar to MoCo



We crop a 84 × 84 image from a 100 × 100 simulation-rendered image. Applying the same random crop coordinates across all frames in the stack ensures time-consistency

(CURL: Srinivas & Laskin, et al., 2020)

Grasp2Vec: Object Representation Learning



Let ϕ_s and ϕ_o be the embedding functions for the scene and the object respectively. The model learns the representation by minimizing the distance between $\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}})$ and $\phi_o(o)$ using *n-pair* loss:

$$\mathcal{L}_{\text{grasp2vec}} = \text{NPair}\left(\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}), \phi_o(o)\right) + \text{NPair}\left(\phi_o(o), \phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}})\right)$$

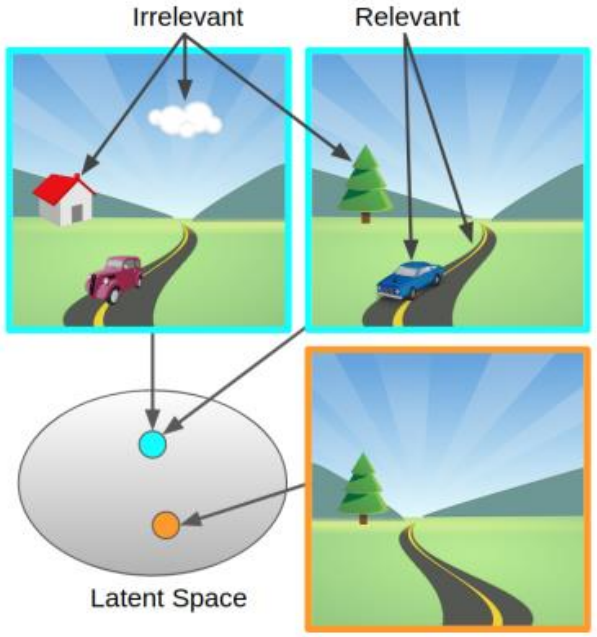
Where $\text{NPair}(a, p) = \sum_{i<B} -\log \frac{\exp(a_i^\top p_j)}{\sum_{j<B, i\neq j} \exp(a_i^\top p_j)} + \lambda\left(\|a_i\|_2^2 + \|p_i\|_2^2\right)$

Where B refers to a batch of (anchor, positive) sample pairs.

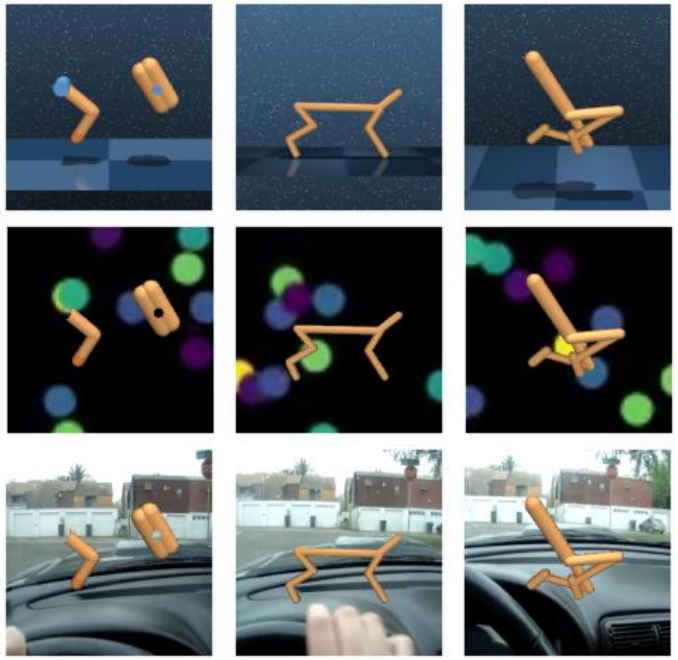
Graspa2Vec: Learning Object Representations from Self-Supervised Grasping
Jang & Devin et al., 2018

DBC: Bisimulation-based self-representation RL

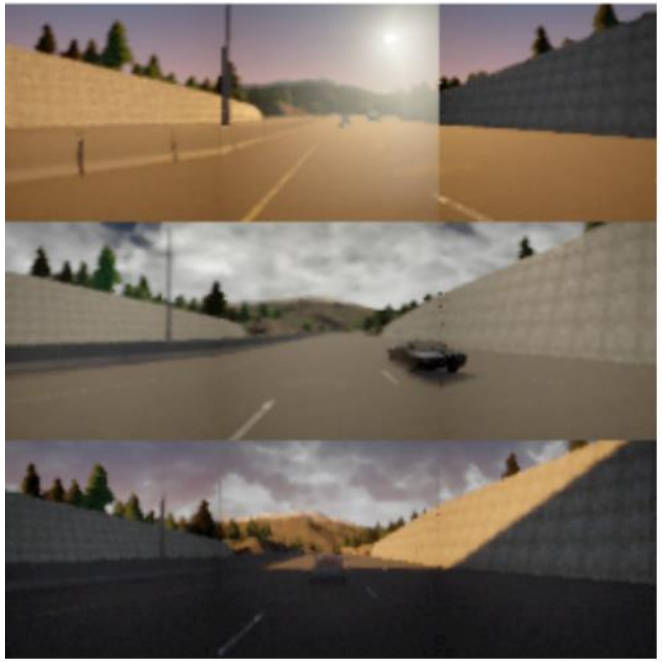
- Image-level self-supervised representation learning (e.g., CURL) can learn irrelevant information



Example 1. Robust representations of the visual scene should be insensitive to irrelevant objects (e.g., clouds) or details (e.g., car types), and encode two observations equivalently if their relevant details are equal (e.g., road direction and locations of other cars).



Example 2. Pixel observations in DMC in the default setting (top row) of the finger spin (left column), cheetah (middle column), and walker (right column), and natural video distractors (bottom row).

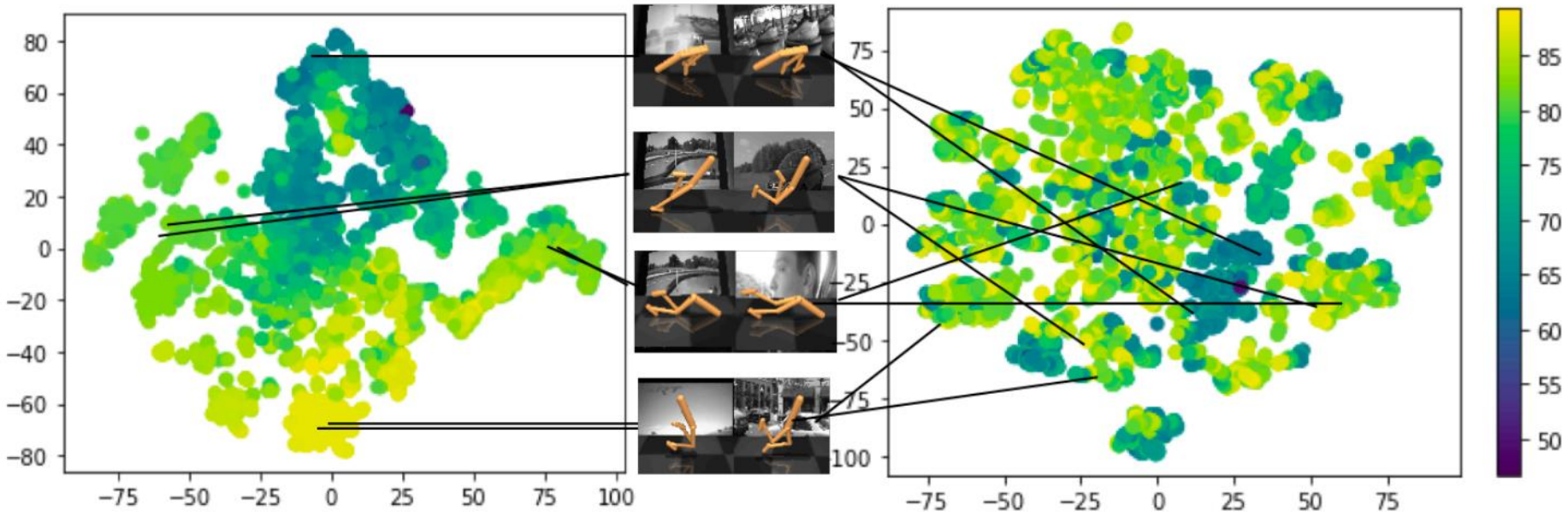


Example 3. Autonomous Driving, a clear highway driving scenario. Clouds and sun position are irrelevant.

Deep Bisimulation for Control: https://arxiv.org/pdf/2006.10742.pdf, ICLR21

DBC: Bisimulation-based self-representation RL

- DBC is effective in learning task-relevant state representation while VAE fails



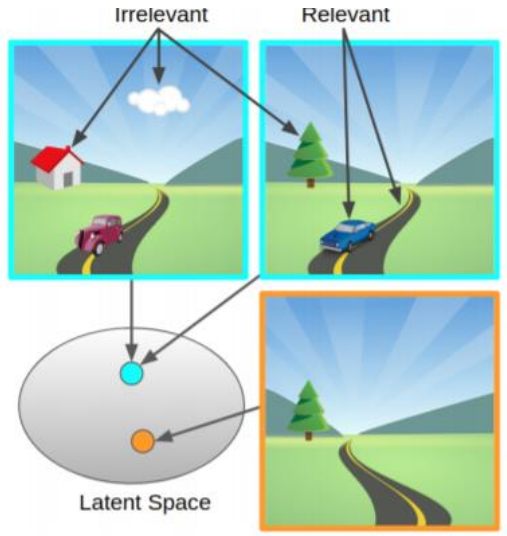
Representation visualization. t-SNE of latent spaces learned with a bisimulation metric (left t-SNE) and VAE (right t-SNE) after training has completed, color-coded with predicted state values (higher value yellow, lower value purple).

- Neighboring points in the embedding space learned with a bisimulation metric have similar states and correspond to observations with the same task-related information (depicted as pairs of images with their corresponding embeddings),
- Whereas no such structure is seen in the embedding space learned by VAE, where the same image pairs are mapped far away from each other.

Deep Bisimulation for Control: https://arxiv.org/pdf/2006.10742.pdf, ICLR21

DBC: Bisimulation-based self-representation RL

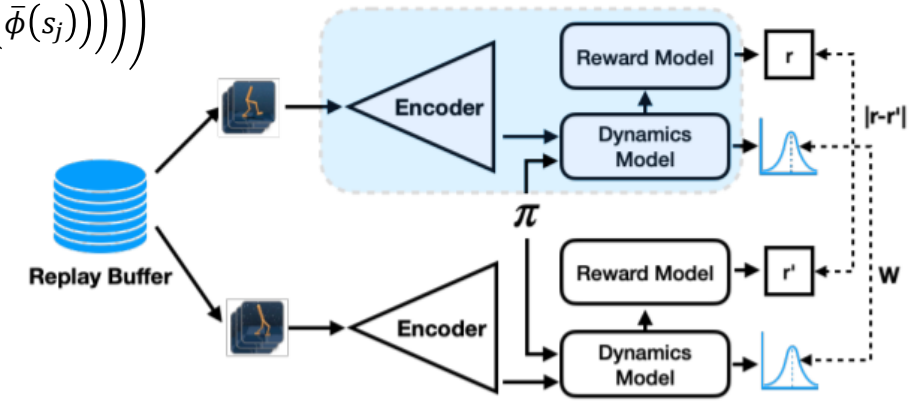
-learning behavioral similarity between states



- CURL- largely ignore the **sequential aspect of RL** when learning state representation
- DBC: capture representations of states that are suitable to control, while discarding any information that is irrelevant for control
- Robust representations of the visual scene should be insensitive to irrelevant objects (e.g., clouds) or details (e.g., car types), and encode two observations equivalently if their relevant details are equal (e.g., road direction and locations of other cars).
- The Deep Bisimulation for Control algorithm learns a bisimulation metric representation via learning a reward model and a dynamics model.

$$J(\phi) = \left(\|\phi(s_i) - \phi(s_j)\|_1 - \left|\hat{\mathcal{R}}\left(\bar{\phi}(s_i)\right) - \hat{\mathcal{R}}\left(\bar{\phi}(s_j)\right)\right| - \gamma W_2\left(\hat{\mathcal{P}}\left(\cdot \mid \bar{\phi}(s_i), \bar{\pi}\left(\bar{\phi}(s_i)\right)\right), \hat{\mathcal{P}}\left(\cdot \mid \bar{\phi}(s_j), \bar{\pi}\left(\bar{\phi}(s_j)\right)\right)\right)\right)^2$$

Where $\bar{\phi}(s)$ denotes $\phi(s)$ with stop gradient and $\bar{\pi}$ is the mean policy output. The learned reward model $\hat{\mathcal{R}}$ is deterministic and the learned forward dynamics model $\hat{\mathcal{P}}$ outputs a Gaussian distribution.



- Bisimulation metrics rely on reward information and may not provide a meaningful notion of behavioral similarity in certain environments.

Deep Bisimulation for Control: https://arxiv.org/pdf/2006.10742.pdf, ICLR21

DBC: Bisimulation-based self-representation RL

-learning behavioral similarity between states

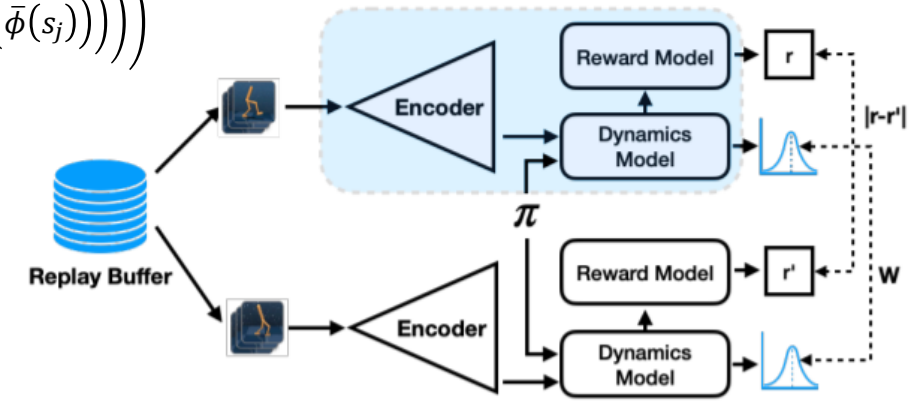
- CURL- largely ignore the **sequential aspect of RL** when learning state representation
- DBC: capture representations of states that are suitable to control, while discarding any information that is irrelevant for control
- Robust representations of the visual scene should be insensitive to irrelevant objects (e.g., clouds) or details (e.g., car types), and encode two observations equivalently if their relevant details are equal (e.g., road direction and locations of other cars).
- The Deep Bisimulation for Control algorithm learns a bisimulation metric representation via learning a reward model and a dynamics model.

Algorithm 1 Deep Bisimulation for Control (DBC)

for Time $t = 0$ to ∞ **do**
  Encode observation $\mathbf{z}_t = \phi(\mathbf{s}_t)$
  Execute action $\mathbf{a}_t \sim \pi(\mathbf{z}_t)$
  Record data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_{t+1}\}$
  Sample batch $B_i \sim \mathcal{D}$
  Permute batch randomly: $B_j = \text{permute}(B_i)$
  Train policy: $\mathbb{E}_{B_i}[J(\pi)]$ ▷ Algorithm 2
  Train encoder: $\mathbb{E}_{B_i,B_j}[J(\phi)]$ ▷ Equation (4)
  Train dynamics: $J(\hat{\mathcal{P}},\phi) = (\hat{\mathcal{P}}(\phi(\mathbf{s}_t), \mathbf{a}_t) - \bar{\mathbf{z}}_{t+1})^2$
  Train reward: $J(\hat{\mathcal{R}},\hat{\mathcal{P}},\phi) = (\hat{\mathcal{R}}(\hat{\mathcal{P}}(\phi(\mathbf{s}_t),\mathbf{a}_t)) - r_{t+1})^2$

$$J(\phi) = \left(\|\phi(s_i) - \phi(s_j)\|_1 - \left|\hat{\mathcal{R}}\left(\bar{\phi}(s_i)\right) - \hat{\mathcal{R}}\left(\bar{\phi}(s_j)\right)\right| - \gamma W_2\left(\hat{\mathcal{P}}\left(\cdot \mid \bar{\phi}(s_i), \bar{\pi}\left(\bar{\phi}(s_i)\right)\right), \hat{\mathcal{P}}\left(\cdot \mid \bar{\phi}(s_j), \bar{\pi}\left(\bar{\phi}(s_j)\right)\right)\right)\right)^2$$

Where $\bar{\phi}(s)$ denotes $\phi(s)$ with stop gradient and $\bar{\pi}$ is the mean policy output. The learned reward model $\hat{\mathcal{R}}$ is deterministic and the learned forward dynamics model $\hat{\mathcal{P}}$ outputs a Gaussian distribution.

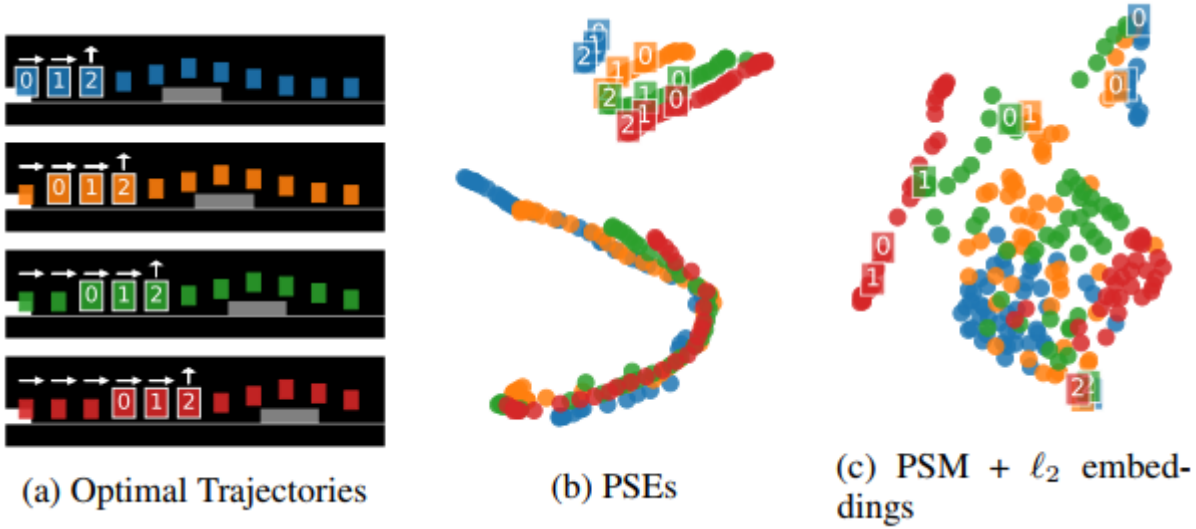


- Bisimulation metrics rely on reward information and may not provide a meaningful notion of behavioral similarity in certain environments.

Deep Bisimulation for Control: https://arxiv.org/pdf/2006.10742.pdf, ICLR21

Policy Similarity Measure (PSE)

-generalization across semantically equivalent tasks with similar underlying mechanics



(a) Optimal Trajectories (b) PSEs (c) PSM + ℓ_2 embeddings

Embedding visualization. (a) Optimal trajectories on original jumping task (visualized as coloured blocks) with different obstacle positions. The hidden representations are visualized using UMAP, where the color of points indicate the tasks of the corresponding observations. Points with the same number label correspond to same distance of the agent from the obstacle, the underlying optimal invariant feature across tasks.

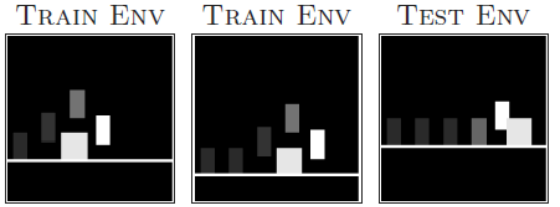
- The figure shows that PSEs partition the states into two sets: (1) states where a single suboptimal action leads to failure (all states before *jump*) and (2) states where actions do not affect the final outcome (states after *jump*). PSEs align the numbered states in the first set, which have the same distance from the obstacle, the invariant feature that generalizes across tasks.
- While l_2-embeddings (in DBC) with PSM do not align states with zero PSM except the state with the jump action – poor generalization.

Policy Similarity Measure

-generalization across semantically equivalent tasks with similar underlying mechanics

- Policy similarity metric (PSM): defines a notion of similarity between states originated from different environments by the proximity of the long-term optimal behavior from these states.
- the agent is optimized to learn an embedding in which states are close when the agent's optimal policies in these states and future states are similar.

$$L_\theta(\mathcal{M}_X, \mathcal{M}_Y) = \mathbb{E}_{y\sim\tau_y^*}\left[\ell_\theta(\tilde{x}_y, y; \tau_X^*)\right] \text{ where } \tilde{x}_y = \underset{x\in\tau_x^*}{\operatorname{argmax}}\, \Gamma(x, y)$$



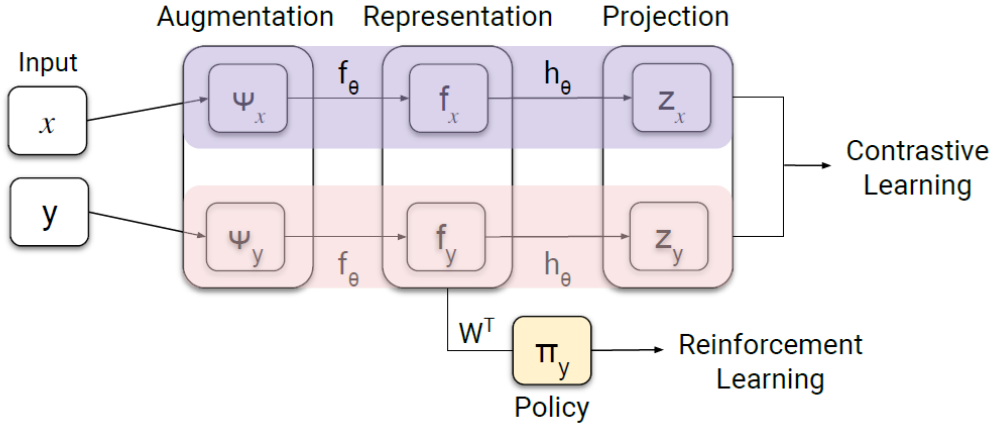
Different tasks consist in shifting the floor height and/or the obstacle position. To generalize, the agent needs to be invariant to the floor height while jump based on the obstacle position (26 locations*11 heights-286 tasks)
Deep RLagents trained on a few of these jumping tasks with different obstacle positions struggle to successfully jump in test tasks where obstacles are at previously unseen locations.

$$\ell_\theta(\tilde{x}_y, y; \mathcal{X}') = -\log \frac{\Gamma(\tilde{x}_y, y)\exp\left(\lambda s_\theta(\tilde{x}_y, y)\right)}{\Gamma(\tilde{x}_y, y)\exp\left(\lambda s_\theta(\tilde{x}_y, y)\right) + \sum_{x'\in\mathcal{X}'\setminus\{\tilde{x}_y\}} \left(1-\Gamma(x', y)\right)\exp\left(\lambda s_\theta(x', y)\right)}$$

soft version of the SimCLR contrastive loss $\quad \Gamma(x, y) = \exp(-d(x, y)/\beta)$

$$d^*(x, y) = \underbrace{\text{Dist}(\pi^*(x), \pi^*(y))}_{\text{(A)}} + \underbrace{\gamma \mathcal{W}_1(d^*)\left(P^{\pi^*}(\cdot|\, x), P^{\pi^*}(\cdot|\, y)\right)}_{\text{(B)}}$$

The Dist term captures the difference in local optimal behavior (A) while $\mathcal{W}_1$ captures long-term optimal behavior difference (B); the exact weights assigned to the two are given by the discount



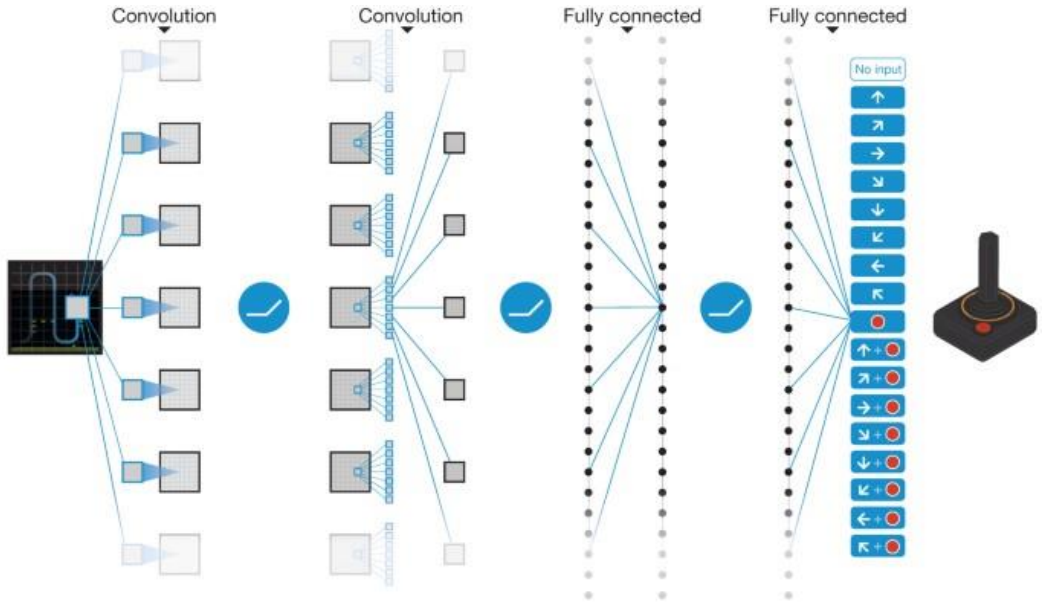
Framework for Learning Contrastive Metric Embeddings

Self-supervised Reinforcement Learning

- How to learn better intermediate representation for better estimation of policies and Q-values?
  - Representation of states
    - Contrastive-learning based: task-agnostic representation (CURL)
    - Bisimulation-based: only encode task-relevant information (DBC, deepMDP, PSM)
  - **Representation of policies**
    - Policy-extended value function (generalization across value function estimation of different policies)
  - Representation of actions
    - Action representation in RL (action space reduction, generalization across policies and value functions)
  - Representation of tasks/environments
    - Task context representation learning for Meta-RL (generalization across new tasks)

Policy Representation In RL

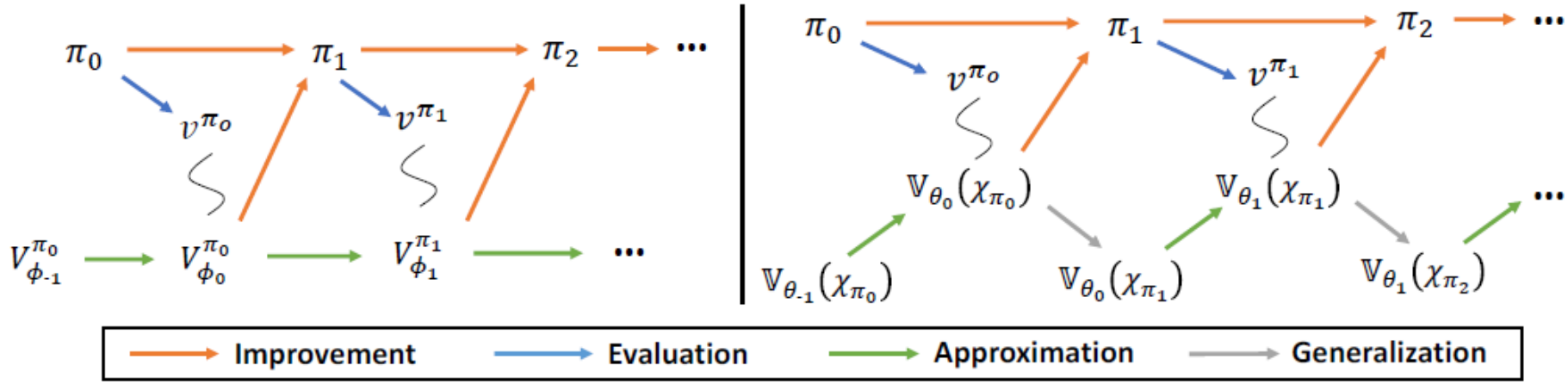
- Policy: mapping from state to action distribution
- Policy network = representation learning + policy learning
- Origin policy Representation: extract policy representation from the policy network parameters
- Surface policy representation: extract policy representation from state-action pairs (trajectories)



DQN: Human-level control through deep reinforcement learning, Nature 2015

Policy extended value function(PeVF)

- Generalized Policy Iteration (GPI) with function approximation

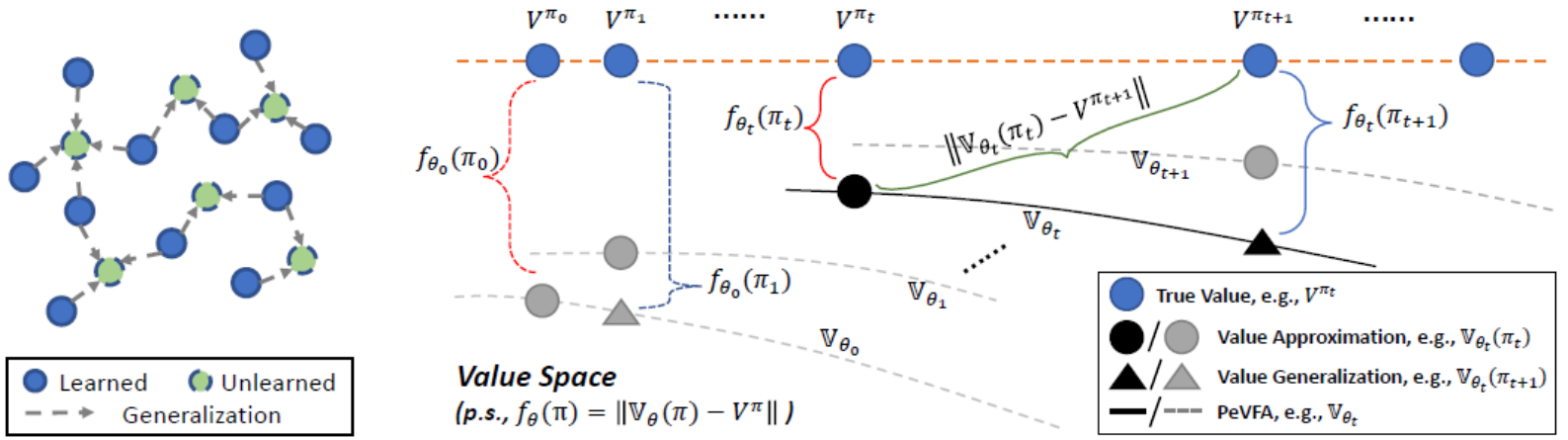


- Policy-extended Value Function Approximator (PeVFA)

$$\mathbb{V}(s,\pi) = v^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t} r_{t+1} | s_0 = s\right]$$

What About Taking Policy as Input of Value Function: Policy-extended Value Function Approximator. https://arxiv.org/abs/2010.09536

PeVF: Global vs. Local Generalization



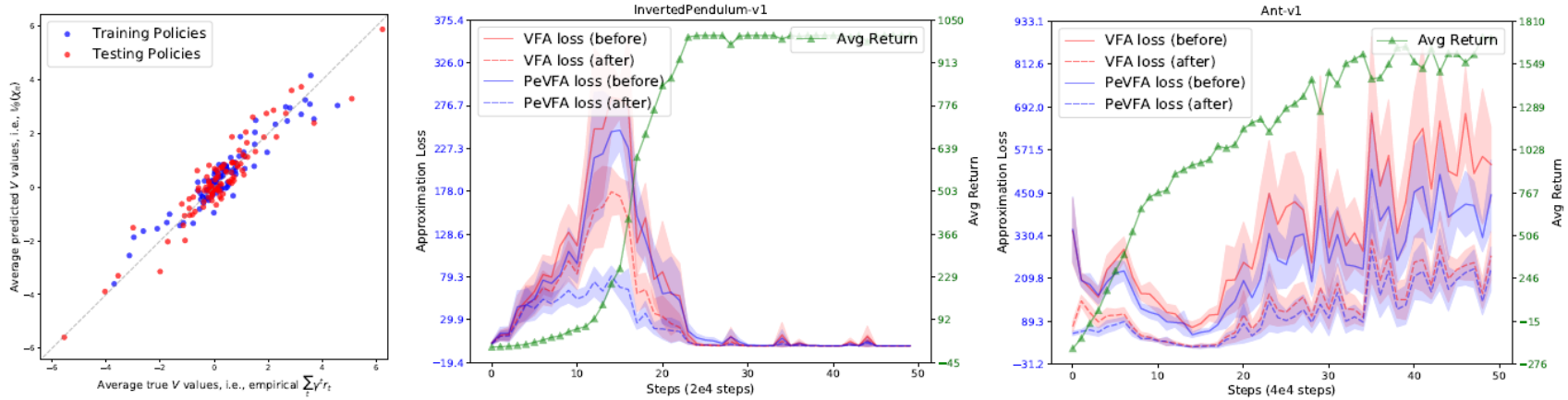
(a) Global Generalization (b) Local Generalization

Figure 2: Illustrations of value generalization among policies of PeVFA. Each circle denotes value function (estimate) of a policy. (a) *Global Generalization*: values learned from known policies can be generalized to unknown policies. (b) *Local Generalization*: values of previous policies (e.g., π_t) can be generalized to successive policies (e.g., π_{t+1}) along policy improvement path.

Theorem 1 *During* $\theta_{-1} \xrightarrow{\mathscr{P}_{\pi_0}} \theta_0 \xrightarrow{\mathscr{P}_{\pi_1}} \theta_1 \xrightarrow{\mathscr{P}_{\pi_2}} \ldots$, *for any* $t \geq 0$, *if* $f_{\theta_t}(\pi_t) + f_{\theta_t}(\pi_{t+1}) \leq \|V^{\pi_t} - V^{\pi_{t+1}}\|$, *then* $f_{\theta_t}(\pi_{t+1}) \leq \|\mathbb{V}_{\theta_t}(\pi_t) - V^{\pi_{t+1}}\|$.

What About Taking Policy as Input of Value Function: Policy-extended Value Function Approximator. https://arxiv.org/abs/2010.09536

PeVF: Global vs. Local Generalization

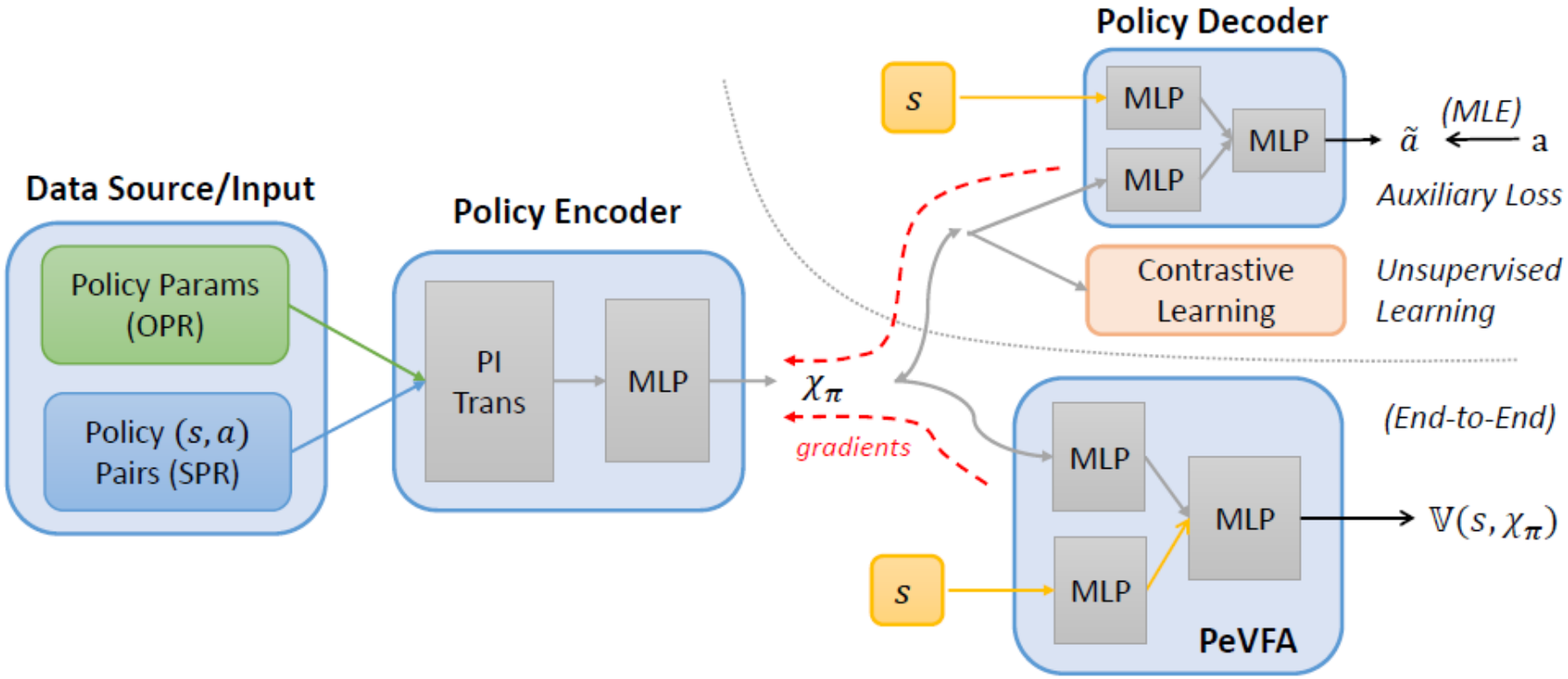


(a) Global Generalization (b) Local Generalization

Figure 3: Empirical evidences of two kinds of generalization of PeVFA. (a) *Global generalization*: PeVFA shows comparable value estimation performance on testing policy set (red) after learning on training policy set (blue). (b) *Local generalization*: PeVFA ($\mathbb{V}_\theta(\chi_\pi)$) shows lower losses than conventional VFA (V_ϕ) before and after the value approximation training for successive policies along policy improvement path. In (b), the left axis is for approximation loss (lower is better) and the right axis is for average return as a reference of the policy learning process (green curve).

What About Taking Policy as Input of Value Function: Policy-extended Value Function Approximator. https://arxiv.org/abs/2010.09536

Policy extended value function(PeVF)



Framework of policy representation training

What About Taking Policy as Input of Value Function: Policy-extended Value Function Approximator. https://arxiv.org/abs/2010.09536

Policy extended value function(PeVF)

- Origin policy Representation: extract policy representation from the policy network parameters
- Surface policy representation: extract policy representation from state-action pairs (trajectories)

	Benchmarks		Origin Policy Representation (Ours)			Surface Policy Representation (Ours)		
Environments	PPO	Ran PR	E2E	CL	AUX	E2E	CL	AUX
HalfCheetah-v1	2621	2470	3171 ± 427.63	**3725 ± 348.55**	3175 ± 517.52	2774 ± 233.39	**3349 ± 341.42**	3216 ± 506.39
Hopper-v1	1639	1226	2085 ± 310.91	2351 ± 231.11	2214 ± 360.78	2227 ± 297.35	**2392 ± 263.93**	**2577 ± 217.73**
Walker2d-v1	1505	1269	1856 ± 305.51	2038 ± 315.51	**2044 ± 316.32**	1930.57 ± 456.02	**2203 ± 381.95**	1980 ± 325.54
Ant-v1	2835	2742	3581 ± 185.43	**4019 ± 162.47**	**3784 ± 268.99**	3173 ± 184.75	3632 ± 134.27	3397 ± 200.03
InvDouPend-v1	9344	9355	**9357 ± 0.29**	9355 ± 0.64	9355 ± 0.68	9355 ± 0.89	**9356 ± 0.96**	9355 ± 1.42
LunarLander-v2	219	226	**238 ± 3.37**	**239 ± 3.70**	234 ± 3.47	236 ± 3.13	234 ± 3.13	235 ± 5.70

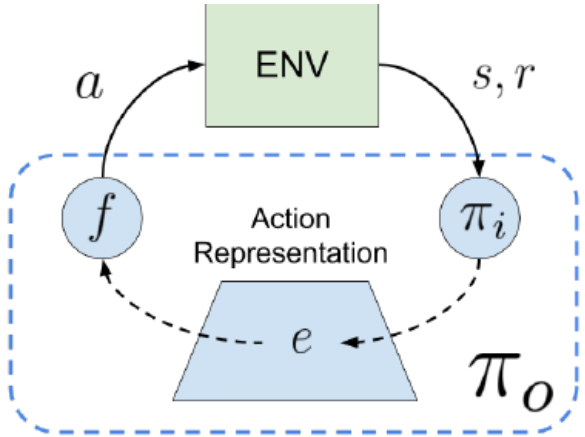
What About Taking Policy as Input of Value Function: Policy-extended Value Function Approximator. https://arxiv.org/abs/2010.09536

Self-supervised Reinforcement Learning

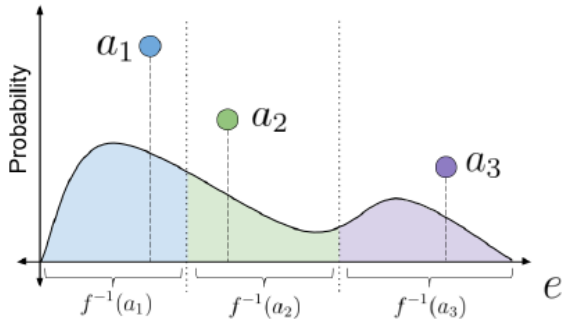
- How to learn better intermediate representation for better estimation of policies and Q-values?
  - Representation of states
    - Contrastive-learning based: task-agnostic representation (CURL)
    - Bisimulation-based: only encode task-relevant information (DBC, deepMDP, ourwork)
  - Representation of policies
    - Policy-extended value function (generalization across value function estimation of different policies)
  - **Representation of actions**
    - Action representation in RL (action space reduction, generalization across policies and value functions)
  - Representation of tasks/environments
    - Task context representation learning for Meta-RL (generalization across new tasks)

Action Representation in RL

- **Applications with large action space**: maximizing long-term portfolio value in finance using various trading strategies; Smart-grid power control by regulating voltage level of all the units; recommending millions of items in e-commence systems
- **Succinct Action Embedding** is important for reducing policy space and also providing generalization ability of policy and value function over action space, thus significantly speed up learning
- **Internal policy** is defined over action embedding space and function f maps action embedding back to the original action space



An RL agent interacting with environments with action embedding



Each color is associated with a specific action

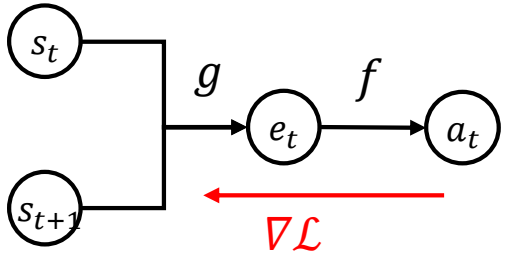
Learning Action Representations for Reinforcement Learning, ICML 2019

Action Representation in RL

- **How to construct Self-supervised Learning for action representation?**
  - exploits the structure in the action set by aligning actions based on the similarity of their impact on states.

- **Action Encoder g and Decoder f** are trained to maximize the prediction accuracy of the exact action that cause the transition between state s(t) and s(t+1).

- **The internal policy** is optimized over the action embedding space following policy gradient theorem (equivalent with the gradient of the performance function over the original action space), which is also relatively independent of the action embedding learning.



Assumption:

$$P(A_t \mid S_t, S_{t+1}) = \int_{\mathrm{e}} P(A_t \mid E_t = e) P(E_t = e \mid S_t, S_{t+1}) \mathrm{d}e$$

Approximation:

$$\widehat{P}(A_t \mid S_t, S_{t+1}) = \int_{\mathrm{e}} \hat{f}(A_t \mid E_t = e) \hat{g}(E_t = e \mid S_t, S_{t+1}) \mathrm{d}e$$

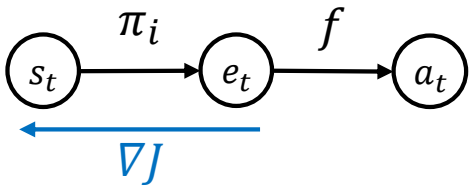
$$\boldsymbol{D_{kl}} = -E\left[\sum_{a \in \mathcal{A}} P(a \mid S_t, S_{t+1}) \ln\left(\frac{\hat{P}(a \mid S_t, S_{t+1})}{P(a \mid S_t, S_{t+1})}\right)\right]$$

$$= -E\left[\ln\left(\frac{\hat{P}(A_t \mid S_t, S_{t+1})}{P(A_t \mid S_t, S_{t+1})}\right)\right]$$

$$\mathcal{L}(\hat{f}, \hat{g}) = -\mathbf{E}\left[\ln\left(\hat{P}(A_t \mid S_t, S_{t+1})\right)\right]$$

on-policy sample estimation

Reward-independent - mitigating sparse and stochastic rewards problem



$$\frac{\partial J_i(\theta)}{\partial \theta} = \sum_{t=0}^{\infty} \mathbf{E}\left[\gamma^t \int_{\mathrm{e}} q^{\pi_i}(S_t, e) \frac{\partial}{\partial \theta} \pi_i(e \mid S_t) \mathrm{d}e\right]$$

$$\frac{\partial J_o(\theta, f)}{\partial \theta} = \frac{\partial J_i(\theta)}{\partial \theta}$$

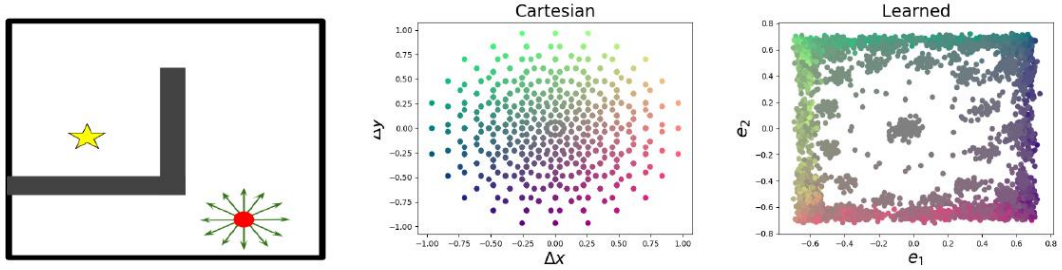
Gradient optimization over embedding action space is equivalent with original action space

Learning Action Representations for Reinforcement Learning, ICML 2019

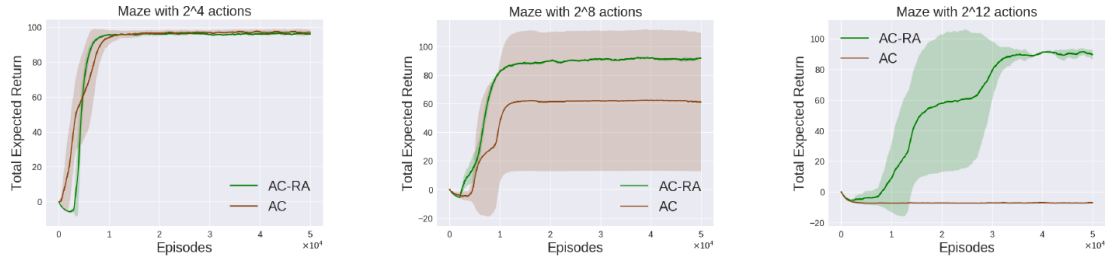
Action Representation in RL

Algorithm 3: Policy-Gradient with Representations for Action (PG-RA)

Initialize internal policy π_i, initialized critic (if any)
Initialize action representations $\hat{f}, \hat{g}$
Prepare replay buffer $\mathcal{D}$
repeat Stage ❶
  Update $\hat{f}, \hat{g}$ using samples in $\mathcal{D}$ to minimize $\mathcal{L}(\hat{f}, \hat{g}) = -\mathbf{E}\left[\ln\left(\hat{P}\left(A_t \mid S_t, S_{t+1}\right)\right)\right]$
until *reaching maximum warm-up steps*;
repeat Stage ❷
  for $t \leftarrow 1$ *to* T **do**
    Sample action embedding E_t, from $\pi_\omega\left(\cdot \mid S_t\right)$
    Decode action $A_t = \hat{f}\left(E_t\right)$
    Execute A_t, observe R_t and new state S_{t+1}
    Store $\{S_t, A_t, E_t, R_t, S_{t+1}\}$ in $\mathcal{D}$
    Update internal policy π_i with *any* policy gradient algorithm (e.g., A2C, PPO, DDPG)
    Update critic (if any) to minimize TD error
    Update $\hat{f}, \hat{g}$ using samples in $\mathcal{D}$ to minimize $\mathcal{L}(\hat{f}, \hat{g}) = -\mathbf{E}\left[\ln\left(\hat{P}\left(A_t \mid S_t, S_{t+1}\right)\right)\right]$
until *reaching maximum training steps*;



The size of the action set is exponential in the number of actuators (a); 2-D representations for the displacements in the Cartesian co-ordinates caused by each action (b) and the learned action embedding (c)



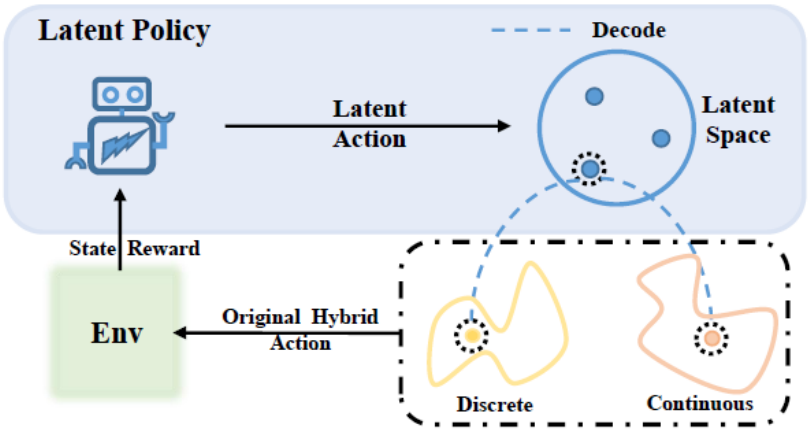
Learning Action Representations for Reinforcement Learning, ICML 2019

HyAR: Hybrid Action Representation in RL

- Applications with Discrete-continuous hybrid action space: Robotics control, Game AI, et al.
- **Solution 1:** Discretization/continualization: scalability issue or non-smooth piecewise-function action subspace (difficult to estimate)
- **Solution 2**: Learning directly over hybrid action spaces with **one policy network** (e.g., PADDPG ICLR16, HPPO IJCAI19)
- **Solution 3**: Learning directly over hybrid action space with **a hybrid structure of DQN and DDPG** (PDQN Arixv2018, HHQN IJCAI19)
- Three key properties should be well-captured: **Scalability** to large action space, **stationarity** between high and low-level learning, and **Correlation/dependence** between discrete and continuous actions
- **How to learn a coherent action embedding space to achieve this?**

Algorithm	Scalability	Stationarity	Dependence	Latent
PADDPG	✗	✓	✗	✗
HPPO	✗	✓	✗	✗
PDQN	✗	✓	✓	✗
HHQN	✓	✗	✓	✗
HyAR (Ours)	✓	✓	✓	✓

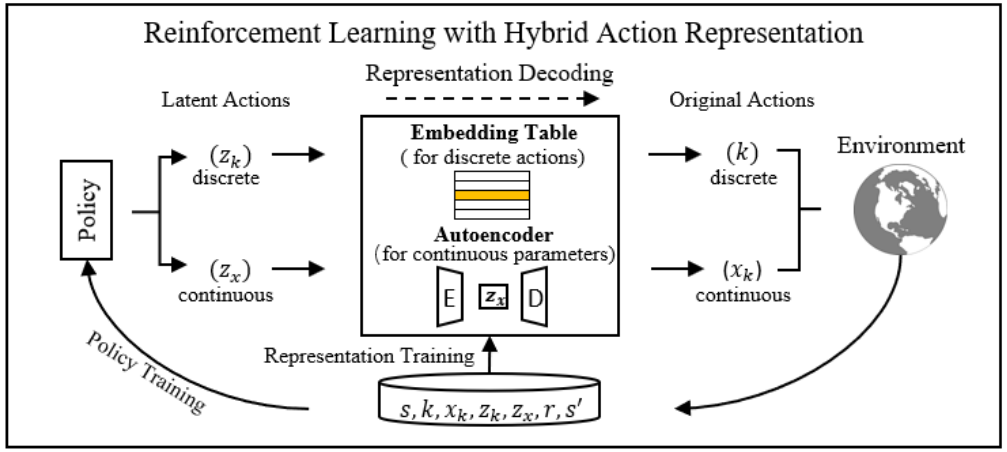
Desirable Properties of RL algorithm with hybrid action space



Learning with Hybrid Action space

HyAR: Addressing Discrete-Continuous Action Reinforcement Learning via Hybrid Action Representation, arxiv 2021

HyAR: Hybrid Action Representation in RL



Dependence-aware encoding and decoding

- A embedding table is used to map discrete actions while a VAE is used to learn embedding for continuous actions, while these two embedding are trained altogether to align them to the same space

$$L_{\mathrm{VAE}}(\phi,\psi,\zeta)=\mathbb{E}_{s,k,x_k}\left[\| x_k-\tilde{x}_k \|_2^2+D_{\mathrm{KL}}\left(q_\phi(\cdot|\, x_k,s,e_{\zeta,k}) \,\|\, \mathcal{N}(0,I)\right)\right]$$

- Hybrid action representations that are closer in the space reflects similar influence on environmental dynamics of their corresponding original hybrid actions.
- A subnetwork that cascaded after the main body of the conditional VAE decoder to produce the prediction of the state residual of transition dynamics

Prediction: $\tilde{\delta}_{s,s'}=p_\psi\left(z_x,s,e_{\zeta,k}\right)$ for s,k,x_k

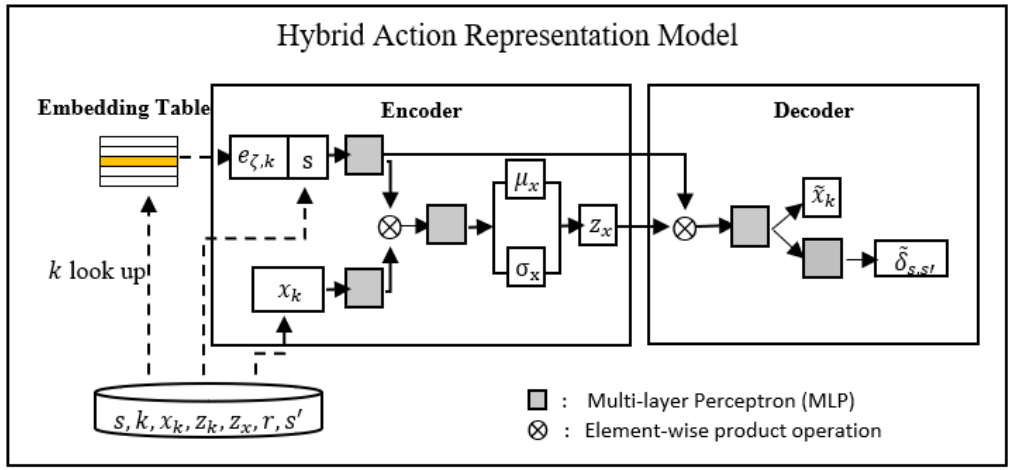
Then we minimize the L_2-norm square prediction error:

$$L_{\mathrm{Dyn}}(\phi,\psi,\zeta)=\mathbb{E}_{s,k,x_k,s'}\left[\left\|\tilde{\delta}_{s,s'}-\delta_{s,s'}\right\|_2^2\right]$$

Encode: $e_{\zeta,k}=E_\zeta(k)$ *and* $z_x\sim q_\phi\left(\cdot|\, x_k,s,e_{\zeta,k}\right)$ for s,k,x_k

Decode: $k=g_E(z_k)=\arg\min_{k\in\mathcal{K}}\left\|e_{\zeta,k}-z_k\right\|_2$,

$x_k=p_\psi\left(z_x,s,e_{\zeta,k}\right)$ for s,z_k,z_x



$$L_{\mathrm{HyAR}}(\phi,\psi,\zeta)=L_{\mathrm{VAE}}(\phi,\psi,\zeta)+\beta L_{\mathrm{Dyn}}(\phi,\psi,\zeta)$$

HyAR: Hybrid Action Representation in RL

Algorithm 2: HyAR-TD3

Initialize actor π_ω and critic networks $Q_{\theta_1}, Q_{\theta_2}$ with random parameters $\omega, \theta_1, \theta_2$
Initialize discrete action embedding table E_ζ and conditional VAE q_ϕ, p_ψ with random parameters ζ, ϕ, ψ
Prepare replay buffer $\mathcal{D}$
repeat Stage ❶
  Update ζ and ϕ, ψ using samples in $\mathcal{D}$ ▷ see Eq. 6
until *reaching maximum warm-up steps*;
repeat Stage ❷
  for $t \leftarrow 1$ *to* T **do**
    // select latent actions in representation space
    $z_k, z_x = \pi_\omega(s) + \epsilon_e$, with $\epsilon_e \sim \mathcal{N}(0, \sigma)$
    // decode into original hybrid actions
    $k = g_E(z_k), x_k = p_\psi(z_x, s, e_{\zeta,k})$ ▷ see Eq. 3
    Execute (k, x_k), observe r_t and new state s'
    Store $\{s, k, x_k, z_k, z_x, r, s'\}$ in $\mathcal{D}$
    Sample a mini-batch of N experience from $\mathcal{D}$
    Update $Q_{\theta_1}, Q_{\theta_2}$ ▷ see Eq. 7
    Update π_ω with policy gradient ▷ see Eq. 8
  repeat
    Update ζ and ϕ, ψ using samples in $\mathcal{D}$ ▷ see Eq. 6
  until *reaching maximum representation training steps*;
until *reaching maximum training steps*;

Eq. 3 : **Encode**: $e_{\zeta,k} = E_\zeta(k)$ *and* $z_x \sim q_\phi(\cdot \mid x_k, s, e_{\zeta,k})$ for s, k, x_k

Decode: $k = g_E(z_k) = \arg\min_{k \in \mathcal{K}} \|e_{\zeta,k} - z_k\|_2$,
$x_k = p_\psi(z_x, s, e_{\zeta,k})$ for s, z_k, z_x

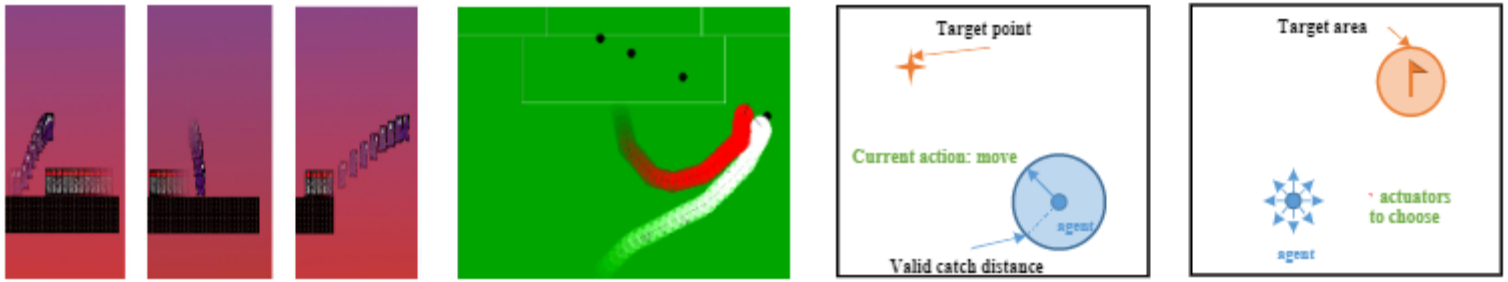
Eq. 6 : $L_{\text{HyAR}}(\phi, \psi, \zeta) = L_{\text{VAE}}(\phi, \psi, \zeta) + \beta L_{\text{Dyn}}(\phi, \psi, \zeta)$

Eq. 7 : $L_{\text{CDQ}}(\theta_i) = \mathbb{E}_{s,z_k,z_x,r,s'}\left[\left(y - Q_{\theta_i}(s, z_k, z_x)\right)^2\right]$,
where $y = r + \gamma \min_{j=1,2} Q_{\bar{\theta}_j}(s', \pi_{\bar{\omega}}(s'))$

Eq. 8 : $\nabla_\omega J(\omega) = \mathbb{E}_s\left[\nabla_{\pi_\omega(s)} Q_{\theta_1}(s, \pi_\omega(s)) \nabla_\omega \pi_\omega(s)\right]$

HyAR: Addressing Discrete-Continuous Action Reinforcement Learning via Hybrid Action Representation, arxiv 2021

HyAR: Hybrid Action Representation in RL



(a) Platform (b) Goal (c) Catch Point (d) Hard Move

Figure 4: Benchmarks with discrete-continuous actions: (a) the agent selects a discrete action (run, hop, leap) and the corresponding continuous parameter (horizontal displacement) to reach the goal; (b) The agent selects a discrete strategy (move, shoot) and the continuous 2-D coordinate to score; (c) The agent selects a discrete action (move, catch) and the continuous parameter (direction) to grab the target point; (d) The agent has n equally spaced actuators. It can choose whether each actuator should be on or off (thus 2^n combination in total) and determine the corresponding continuous parameter for each actuator (moving distance) to reach the target area.

ENV	HPPO	PADDPG	PDQN	HHQN	HyAR-DDPG	PATD3	PDQN-TD3	HHQN-TD3	HyAR-TD3
	PPO-based	DDPG-based				TD3-based			
Goal	0.0 ± 0.0	0.05 ± 0.10	0.70 ± 0.07	0.0±0.0	0.53±0.02	0.0±0.0	0.71±0.10	0.0±0.0	**0.78±0.03**
Hard Goal	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0±0.0	0.30±0.08	0.44±0.05	0.06±0.07	0.01±0.01	**0.60±0.07**
Platform	0.80 ± 0.02	0.36 ± 0.06	0.93 ± 0.05	0.46±0.25	0.87±0.06	0.94±0.10	0.93±0.03	0.62±0.23	**0.98±0.01**
Catch Point	0.69 ± 0.09	0.82 ± 0.06	0.77 ± 0.07	0.31±0.06	0.89±0.01	0.82±0.10	0.89±0.07	0.27±0.05	**0.90±0.03**
Hard Move (n = 4)	0.09 ± 0.02	0.03 ± 0.01	0.69 ± 0.07	0.39±0.14	0.91±0.03	0.66±0.13	0.85±0.10	0.52±0.17	**0.93±0.02**
Hard Move (n = 6)	0.05 ± 0.01	0.04 ± 0.01	0.41 ± 0.05	0.32±0.17	0.91±0.04	0.04±0.02	0.74±0.08	0.29±0.13	**0.92±0.04**
Hard Move (n = 8)	0.04 ± 0.01	0.06 ± 0.03	0.04 ± 0.01	0.05±0.02	0.85±0.06	0.06±0.02	0.05±0.01	0.05±0.02	**0.89±0.03**
Hard Move (n = 10)	0.05 ± 0.01	0.04 ± 0.01	0.06 ± 0.02	0.04±0.01	**0.82±0.06**	0.07±0.02	0.05±0.02	0.05±0.02	0.75±0.05

HyAR: Addressing Discrete-Continuous Action Reinforcement Learning via Hybrid Action Representation, arxiv 2021

Self-supervised Reinforcement Learning

- How to learn better intermediate representation for better estimation of policies and Q-values?
  - Representation of states
    - Contrastive-learning based: task-agnostic representation (CURL)
    - Bisimulation-based: only encode task-relevant information (DBC, deepMDP)
  - Representation of policies
    - Policy-extended value function (generalization across value function estimation of different policies)
  - Representation of actions
    - Action representation in RL (action space reduction, generalization across policies and value functions)
  - **Representation of tasks/environments**
    - Task context representation learning for Meta-RL (generalization across new tasks)

Task Representation in RL

- Meta-Learning RL

Meta-RL is a technique to learn an inductive bias that accelerates the learning of new task by training on a large of number of training tasks. Formally, meta-training on tasks from the meta-training set $\mathcal{D}_{\text{meta}} = \{D^k\}_{k=1,\ldots,n}$ involves learning a policy.

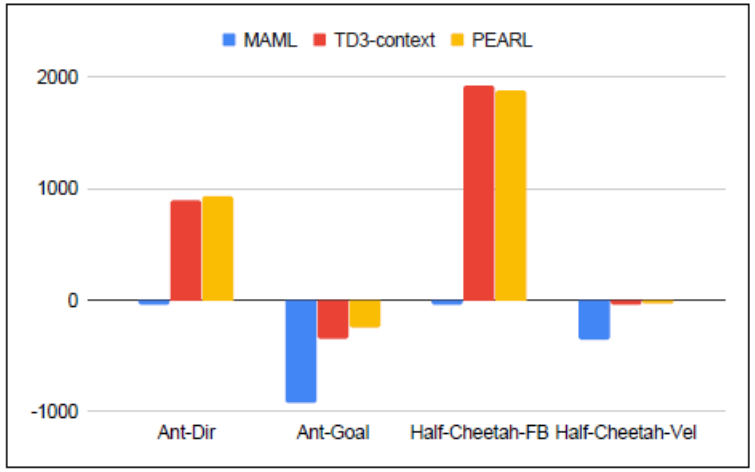
$$\hat{\theta}_{\text{meta}} = \arg\max_{\theta} \frac{1}{n} \sum_{k=1}^{n} \ell^k_{\text{meta}}(\theta)$$ General form of Meta-learning

$$\ell^k_{\text{meta}}(\theta) = \ell^k\left(\theta + \alpha \nabla_\theta \ell^k(\theta)\right)$$ MAML (on-policy meta-training, sample-inefficient)

$$\hat{\theta}_{\text{meta}} = \arg\min_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}_{\tau \sim D^k}[\text{TD}^2(\theta)]$$ Multi-task learning (off-policy, sample efficient)

Key idea: adapt to a new environment by inferring latent context from a small number of interactions with the environment.

Challenges: how to extract task-specific and remove noisy information?
how to efficiently explore environments to get distinct task context faced with a new environment?



Remark: MQL uses a deterministic context that is not permutation invariant. the direction of time affords crucial information about the dynamics of a task to the agent, e.g., a Half-Cheetah running forward versus backward has arguably the same state trajectory but in a different order. Further, the context in MQL is a deterministic function of the trajectory. Both these aspects are different than the context used by Rakelly et al. (2019) who design an inference network and sample a probabilistic context conditioned on a moving window. This result demonstrates that a simple context variable is enough is an important contribution.

Meta-Q: Fakoor et al., ICLR 2020

Task Representation in RL

- Meta-Learning RL

Meta-RL is a technique to learn an inductive bias that accelerates the learning of new task by training on a large of number of training tasks. Formally, meta-training on tasks from the meta-training set $\mathcal{D}_{\text{meta}} = \{D^k\}_{k=1,\ldots,n}$ involves learning a policy.

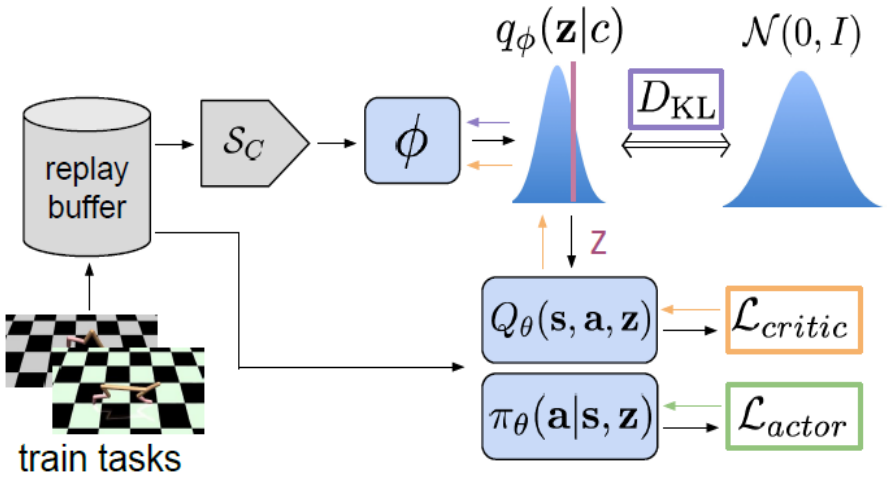
$$\hat{\theta}_{\text{meta}} = \arg\max_{\theta} \frac{1}{n} \sum_{k=1}^{n} \ell^k_{\text{meta}}(\theta)$$ General form of Meta-learning

$$\ell^k_{\text{meta}}(\theta) = \ell^k\left(\theta + \alpha \nabla_\theta \ell^k(\theta)\right)$$ MAML (on-policy meta-training, sample-inefficient)

$$\hat{\theta}_{\text{meta}} = \arg\min_{\theta} \frac{1}{n} \sum_{k=1}^{n} \mathbb{E}_{\tau \sim D^k}[\text{TD}^2(\theta)]$$ Multi-task learning (off-policy, sample efficient)

Key idea: adapt to a new environment by inferring latent context from a small number of interactions with the environment.

Challenges: how to extract task-specific and remove noisy information?
how to efficiently explore environments to get distinct task context faced with a new environment?



Meta-training procedure. The inference network q_ϕ uses context data to infer the posterior over the latent context variable Z, which conditions the actor and critic, and is optimized with gradients from the critic as well as from an information bottleneck on Z. De-coupling the data sampling strategies for context (S_C) and RL batches is important for off-policy learning.

PEARL: Rakelly et al., ICML 2019

Task Representation in RL with contrastive learning

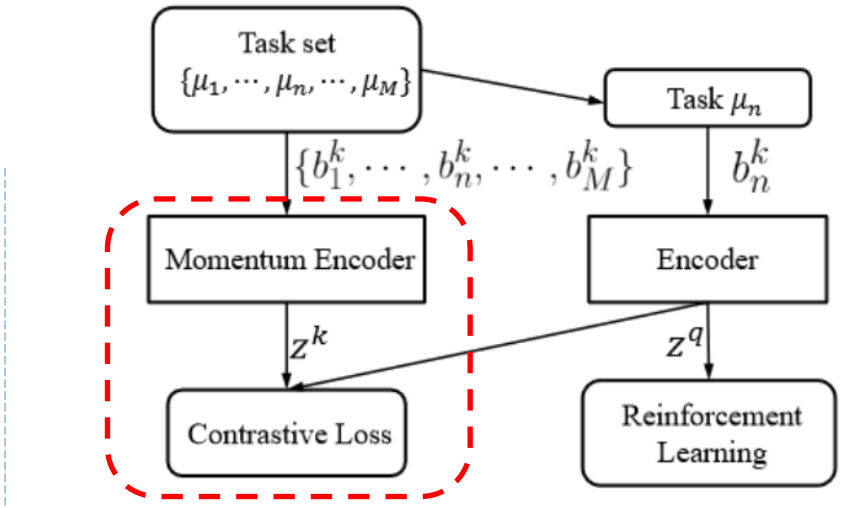
Task Representation with Contrastive Learning

- Contrastive context encoder: a general framework, can be integrated with any Meta-RL framework

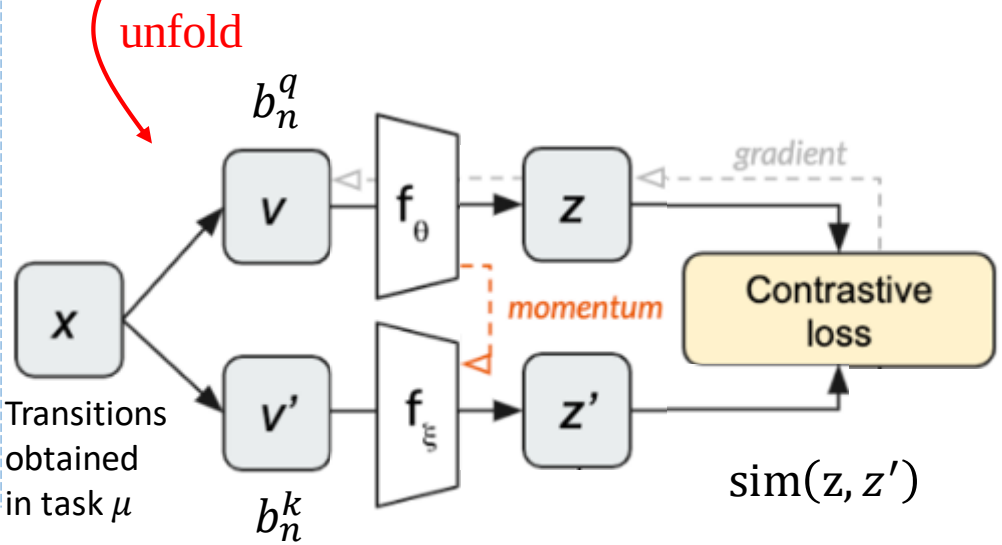
Concretely, assuming a training task set containing M different tasks from task distribution $p(\mu)$. We first generate trajectories with current policy for each task and store them in replay buffer. At each training step, we first sample a task μ_n from the training task set, and then randomly sample two batches of transitions b_n^q, b_n^k from task μ_n independently. b_n^q serves as a query in contrastive learning framework while b_n^k is the corresponding positive key. We also randomly sample $M-1$ batches of transitions from the other tasks as negative keys.

$$L_{contrastive} = -\mathbb{E}\left[\log \frac{\exp\left(\mathrm{sim}\left(z^q, z_n^k\right)\right)}{\sum_{k=1}^{M} \exp\left(\mathrm{sim}\left(z^q, z^k\right)\right)}\right] \quad \text{InfoNCE loss}$$

$$\mathrm{sim}\left(z^q, z^k\right) = z^{q\top} W z^k$$



Contrastive context encoder RL



Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning, AAAI21

Task Representation in RL with contrastive learning

Information-gain-based Exploration for Effective Context

- Exploration policy should collect trajectories as informative as possible
- We achieve this by introducing an additional exploration-encouraging external reward

$$r^e = r^{env} + \alpha r^{aux}, \text{where } r^{aux} = L_{upper} - L_{lower}$$

$$L_{upper} - L_{lower} = \mathbb{E}_{C_{pos}}\left[\log \frac{\exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_{pos}\right)\right)}{\sum_{c_j \in C} \exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_j\right)\right)}\right] - \mathbb{E}_{C_{pos}}\left[\log \frac{\exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_{pos}\right)\right)}{\sum_{c_j \in C_{neg}} \exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_j\right)\right)}\right]$$

$$= \mathbb{E}_{C_{pos}}\left[\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_{pos}\right) - \mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}-1}, c_{pos}\right)\right] - \mathbb{E}_{C_{pos}}\left[\log \frac{\sum_{c_j \in C} \exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_j\right)\right)}{\sum_{c_j \in C_{neg}} \exp\left(\mathrm{sim}\left(\mathrm{c}_{1:\mathrm{i}}, c_j\right)\right)}\right]$$

Here, we abuse the notion $\mathrm{sim}(c^1, c^2)$ to denote $\mathrm{sim}(z^1, z^2) = \mathrm{sim}(f(c^1), f(c^2))$ for convenience

- The **information gain** of collecting a new transition at time step i is equivalent with temporal difference of the mutual information between task belief and collected trajectories.

$$I(z|\tau_{1:i-1}; \tau_i) = H(z|\tau_{1:i-1}) - H(z|\tau_{1:i}) = H(z|\tau_{1:i-1}) + H(z) - H(z) - H(z|\tau_{1:i})$$
$$= I(z|\tau_{1:i}) - I(z|\tau_{1:i-1}) \qquad z \text{ is the task belief}$$

Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning, AAAI 21

Task Representation in RL with contrastive learning

Information-gain-based Exploration for Effective Context

- Exploration policy should collect trajectories as informative as possible
- We achieve this by introducing an additional exploration-encouraging external reward

$$r^e = r^{env} + \alpha r^{aux}, \text{where } r^{aux} = L_{upper} - L_{lower}$$

$$L_{upper} - L_{lower} = \mathbb{E}_{C_{pos}}\left[\log \frac{\exp(\mathrm{sim}(c_{1:i}, c_{pos}))}{\sum_{c_j \in C} \exp(\mathrm{sim}(c_{1:i}, c_j))}\right] - \mathbb{E}_{C_{pos}}\left[\log \frac{\exp(\mathrm{sim}(c_{1:i}, c_{pos}))}{\sum_{c_j \in C_{neg}} \exp(\mathrm{sim}(c_{1:i}, c_j))}\right]$$

$$= \mathbb{E}_{C_{pos}}[\mathrm{sim}(c_{1:i}, c_{pos}) - \mathrm{sim}(c_{1:i-1}, c_{pos})] - \mathbb{E}_{C_{pos}}\left[\log \frac{\sum_{c_j \in C} \exp(\mathrm{sim}(c_{1:i}, c_j))}{\sum_{c_j \in C_{neg}} \exp(\mathrm{sim}(c_{1:i}, c_j))}\right]$$

estimating how much the similarity score for positive pairs has improved after collecting new transition

Regularization term: limit the policy not visiting places where the negative score enhances as well

- The information gain of collecting a new transition at time step i is equivalent with temporal difference of the mutual information between task belief and collected trajectories.

$$I(z|\tau_{1:i-1}; \tau_i) = H(z|\tau_{1:i-1}) - H(z|\tau_{1:i}) = H(z|\tau_{1:i-1}) + H(z) - H(z) - H(z|\tau_{1:i})$$
$$= I(z|\tau_{1:i}) - I(z|\tau_{1:i-1})$$

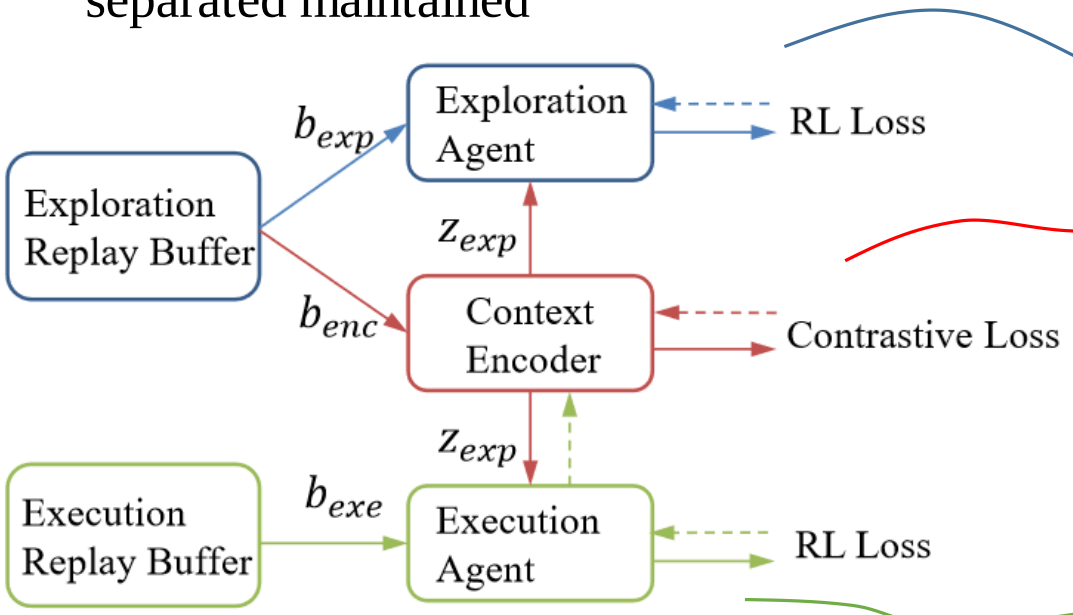
z is the task belief

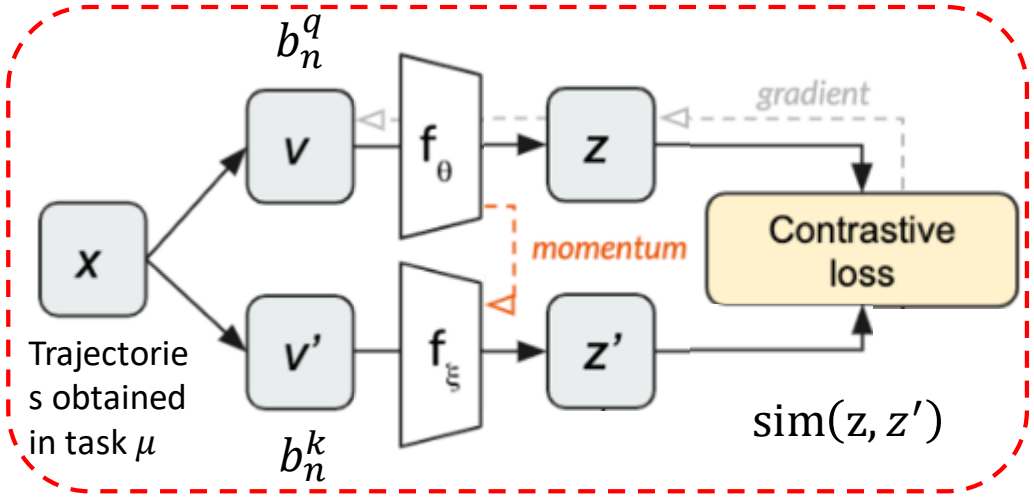
Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning, AAAI21

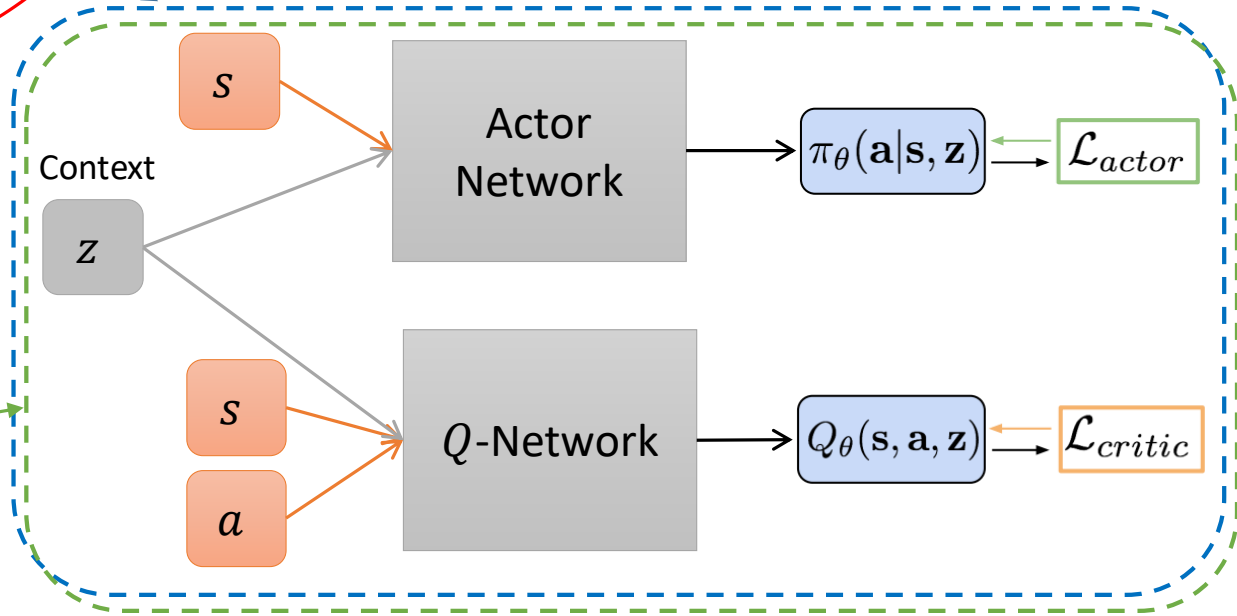
Task Representation in RL with Contrastive Learning

The Overall Workflow

- Exploration replay buffer and execution replay buffer are separated maintained







Note: To avoid the intrinsic reward being to noisy, we pretrain the context encoder for several episodes and set the exploration replay buffer to only contain recently collected data

Task Representation in RL with Contrastive Learning

Algorithm 1 CCM Meta-training

Require: Batch of training tasks $\{\mu_n\}_{n=1,...,M}$ from $p(\mu)$,

Initialize execution replay buffer $\mathcal{B}^i_{exe}$, exploration replay buffer $\mathcal{B}^i_{exp}$ for each training task
Initialize parameters θ_{exp} and θ_{exe} for the off-policy method employed by exploration and execution respectively, e.g., SAC
Initialize parameters θ_{enc} for context encoder network
while not done **do**
  for each task μ_n **do**
    Roll-out exploration policy π_{exp}, producing transitions $\{(s_j, a_j, s'_j, r^{env}_j, r^{exp}_j)\}_{j:1\cdots N}$, where $r^{exp}_j = r^{env}_j + \alpha r^{aux}_j$
    Add tuples to exploration replay buffer $\mathcal{B}^i_{exp}$ and execution replay buffer $\mathcal{B}^i_{exe}$
    Roll-out execution policy π_{exe}, producing transitions $\{(s_k, a_k, s'_k, r^{env}_k)\}_{k:1\cdots K}$
    Add tuples to execution replay buffer $\mathcal{B}^i_{exe}$
  end for
  for each training step **do**
    for each task μ_n **do**
      Sample transitions for encoder $b^n_{enc} = \{(s_i, a_i, s'_i, r^{env}_i)\}_{i:1\cdots T} \sim \mathcal{B}^n_{exp}$ and RL batch $b^n_{exp} = \{(s_i, a_i, s'_i, r^{exp}_i)\}_{i:1\cdots N'} \sim \mathcal{B}^n_{exp}$, $b^n_{exe} = \{(s_i, a_i, s'_i, r^{env}_i)\}_{i:1\cdots K'} \sim \mathcal{B}^n_{exe}$
      Update θ_{enc} and θ_{exe} with RL loss and contrastive loss using b^n_{enc}, b^n_{exe}
      Update θ_{exp} with RL loss using b^n_{exp}
    end for
  end for
end while

Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning, AAAI21

Task Representation in RL with Contrastive Learning

Algorithm 2 CCM Meta-testing

Require: Test task $\mu \sim p(\mu)$,

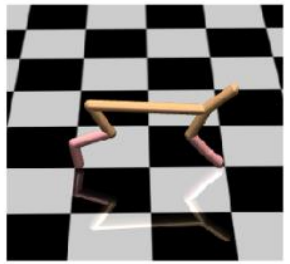
Initialize $b^{\mu} = \{\}$
for each episode **do**
  for $k = 1, \cdots, K$ **do**
    Roll-out exploration policy π_{exp}, producing transition $D^{\mu}_{exp} = \{(s_j, a_j, s'_j, r^{env}_j)\}$
    Accumulate transitions $b^{\mu} = b^{\mu} \cap D^{\mu}_{exp}$
  end for
end for
for $i = 1, \cdots, K$ **do**
  Roll-out execution policy π_{exe} conditioned on b^{μ}, producing transition $D^{\mu}_{exe} = \{(s_j, a_j, s'_j, r^{env}_j)\}$
  Accumulate transitions $b^{\mu} = b^{\mu} \cap D^{\mu}_{exe}$
end for

Towards Effective Context for Meta-Reinforcement Learning: an Approach based on Contrastive Learning, AAAI21

Task Representation in RL with Contrastive Learning

- humanoid-dir: The target direction of running changes across tasks.
- ant-mass: The mass of ant changes across tasks to change transition dynamics
- OOD-tasks: CCM+RV (recover value function) outperforms CCM+DP (dynamic prediction) -> contrastive loss combined with loss from recovering value function obtains better generalization for different tasks

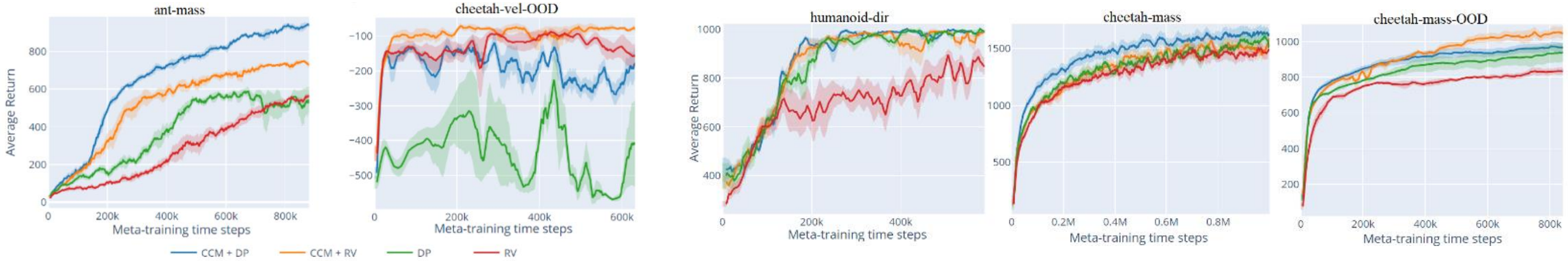
ant-mass



cheetah-vel-OOD

humanoid-dir

cheetah-mass & cheetah-mass-OOD



Overall Conclusion:our methods CCM + DP and CCM + RV achieve consistently better performance compared to existing methods.

Task Representation in RL with Contrastive Learning

- Comparison of Overall Adaptation Performance on Complex Environments

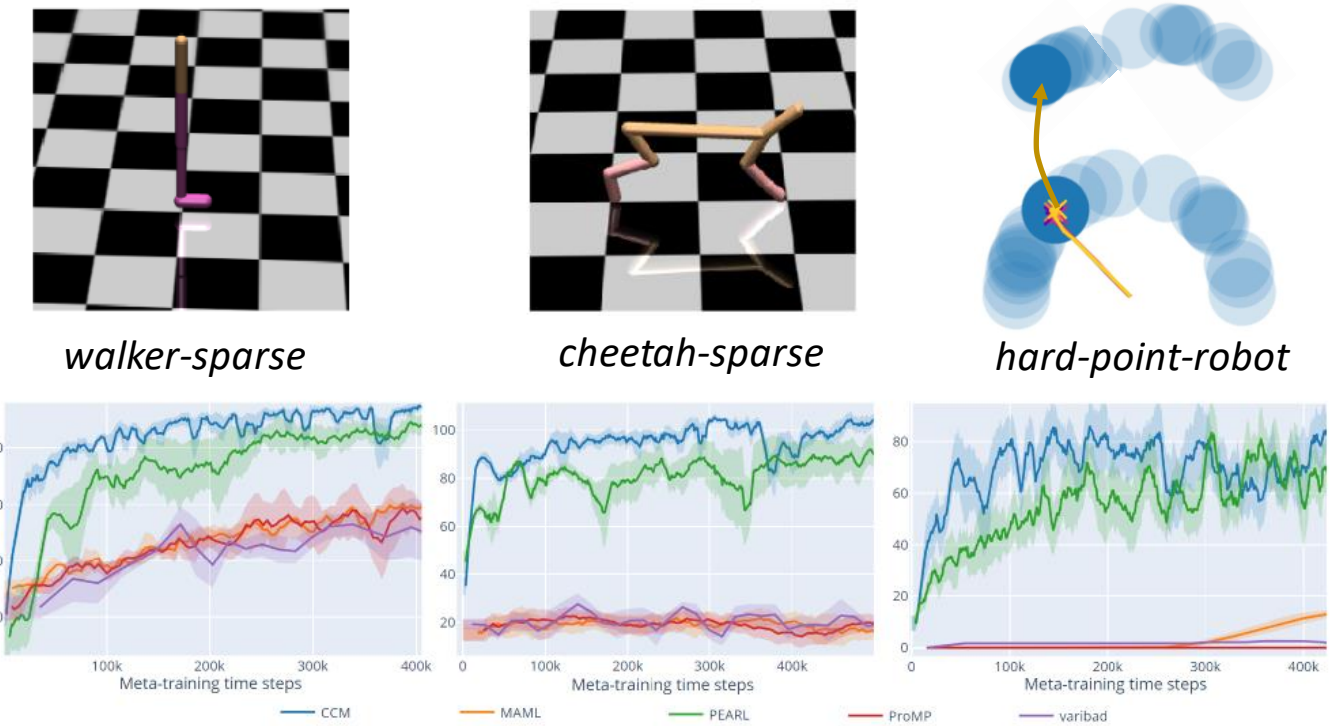


Figure 4: CCM's overall performance compared with state-of-the-art Meta-RL methods on complex sparse-reward environments. From left to right: *walker-sparse*,*cheetah-sparse*, *hard-point-robot*.The error bar shows 1 standard deviation.

PEARL-CL - a contrastive context encoder for a fair comparison

Conclusions and Future Directions

- From Perception to Planning and Control: self-representation learning is the key to bridge the gap.
- Self-representation learning in RL: state, policy, action and task level representation learning can improve the sample efficiency and policy generalization ability of RL across different tasks.

- Policy and environment/task dynamics are entangled: how to disentangle environment representation with policy representation?
- The representation of different RL elements are learned separately: how to combine them altogether?
- Representation learning in Model-based RL and Opponent modeling in MASs

Thanks
Q&A