

## 459 Appendix:

### 460 A Bound Value Difference in Policy Transfer

461 In this section, we provide detailed theoretical ground for our policy transfer approach, as a supplement to Sec. 3. We first define a binary relation for actions to describe the correspondent actions  
 462 behaving equivalently on two MDPs (Definition 1). Building upon the notion of action equivalence,  
 463 we derive the upper bound of value difference between policies on two MDPs (Theorem 1). Finally,  
 464 we reach a proposition for the upper bound of value difference (Proposition 1) to explain that  
 465 minimizing our objective function results in bounding the value difference between the source and  
 466 transferred policy.  
 467

468 **Definition 1.** Given two MDPs  $\mathcal{T}^{(i)} = \{\mathcal{S}, \mathcal{A}, p^{(i)}, r^{(i)}, \gamma, \rho_0\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}, \mathcal{A}, p^{(j)}, r^{(j)}, \gamma, \rho_0\}$   
 469 with the same state space and action space, for each state  $s \in \mathcal{S}$ , we define a binary relation  
 470  $B_s \in \mathcal{A} \times \mathcal{A}$  called **action equivalence relation**. For any action  $a^{(i)} \in \mathcal{A}$ ,  $a^{(j)} \in \mathcal{A}$ , if  $(a^{(i)}, a^{(j)}) \in$   
 471  $B_s$  (i.e.  $a^{(i)} B_s a^{(j)}$ ), the following conditions hold:

$$r^{(i)}(s, a^{(i)}) = r^{(j)}(s, a^{(j)}) \text{ and } p^{(i)}(\cdot | s, a^{(i)}) = p^{(j)}(\cdot | s, a^{(j)}) \quad (5)$$

472 Based on Definition 1, at state  $s$ , action  $a^{(i)}$  on  $\mathcal{T}^{(i)}$  is equivalent to action  $a^{(j)}$  on  $\mathcal{T}^{(j)}$  if  $a^{(i)} B_s a^{(j)}$ .  
 473 Note that the binary relation  $B_s$  is defined for each  $s$  separately. The action equivalence relation  
 474 might change on varied states. On two MDPs with the same dynamic and reward functions, it is  
 475 trivial to get the equivalent action with identity mapping. However, we are interested in more complex  
 476 cases where the reward and dynamic functions are not identical on two MDPs.

477 Ideally, the equivalent action always exists on the target MDP  $\mathcal{T}^{(i)}$  for any state-action pair on the  
 478 source MDP  $\mathcal{T}^{(j)}$  and there exists an action translator function  $H : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{A}$  to identify the exact  
 479 equivalent action. Starting from state  $s$ , the translated action  $\tilde{a} = H(s, a)$  on the task  $\mathcal{T}^{(i)}$  generates  
 480 reward and next state distribution the same as action  $a$  on the task  $\mathcal{T}^{(j)}$  (i.e.  $\tilde{a} B_s a$ ). Then any  
 481 deterministic policy  $\pi^{(j)}$  on the source task  $\mathcal{T}^{(j)}$  can be perfectly transferred to the target task  $\mathcal{T}^{(i)}$   
 482 with  $\pi^{(i)}(s) = H(s, \pi^{(j)}(s))$ . The value of the policy  $\pi^{(j)}$  on the source task  $\mathcal{T}^{(j)}$  is equal to the  
 483 value of transferred policy  $\pi^{(i)}$  on the target task  $\mathcal{T}^{(i)}$ .

484 Without the assumption of existence of a perfect correspondence for each action, given any  
 485 two deterministic policies  $\pi^{(j)}$  and  $\pi^{(i)}$ , we prove that the difference in the policy value  
 486 is upper bounded by a scalar  $\frac{d}{1-\gamma}$  depending on L1-distance between reward functions  
 487  $|r^{(i)}(s, \pi^{(i)}(s)) - r^{(j)}(s, \pi^{(j)}(s))|$  and total-variation distance between next state distributions  
 488  $D_{TV}(p^{(i)}(\cdot | s, \pi^{(i)}(s)), p^{(j)}(\cdot | s, \pi^{(j)}(s)))$ .

489 **Theorem 1.** Let  $\mathcal{T}^{(i)} = \{\mathcal{S}, \mathcal{A}, p^{(i)}, r^{(i)}, \gamma, \rho_0\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}, \mathcal{A}, p^{(j)}, r^{(j)}, \gamma, \rho_0\}$  be  
 490 two MDPs sampled from the distribution of tasks  $p(\mathcal{T})$ .  $\pi^{(i)}$  is a deterministic policy on  
 491  $\mathcal{T}^{(i)}$  and  $\pi^{(j)}$  is a deterministic policy on  $\mathcal{T}^{(j)}$ . Let  $M = \sup_{s \in \mathcal{S}} |V^{\pi^{(i)}}(s, \mathcal{T}^{(i)})|$ ,  $d =$   
 492  $\sup_{s \in \mathcal{S}} [|r^{(i)}(s, \pi^{(i)}(s)) - r^{(j)}(s, \pi^{(j)}(s))| + 2\gamma M D_{TV}(p^{(i)}(\cdot | s, \pi^{(i)}(s)), p^{(j)}(\cdot | s, \pi^{(j)}(s)))]$ . For  
 493  $\forall s \in \mathcal{S}$ , we have

$$|V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)})| \leq \frac{d}{1-\gamma} \quad (6)$$

494 *Proof.* Let  $a^{(i)} = \pi^{(i)}(s)$  and  $a^{(j)} = \pi^{(j)}(s)$ .  $s'$  denotes the next state following state  $s$ .  $s''$  denotes  
 495 the next state following  $s'$ .

496 We rewrite the value difference as:

$$\begin{aligned}
V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)}) &= r^{(i)}(s, a^{(i)}) + \gamma \sum_{s'} p^{(i)}(s'|s, a^{(i)}) V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \\
&- r^{(j)}(s, a^{(j)}) - \gamma \sum_{s'} p^{(j)}(s'|s, a^{(j)}) V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \\
&= (r^{(i)}(s, a^{(i)}) - r^{(j)}(s, a^{(j)})) \\
&+ \gamma \left[ \sum_{s'} p^{(i)}(s'|s, a^{(i)}) V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - \sum_{s'} p^{(j)}(s'|s, a^{(j)}) V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \\
&\quad \text{*minus and plus } \gamma \sum_{s'} p^{(j)}(s'|s, a^{(j)}) V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \\
&= (r^{(i)}(s, a^{(i)}) - r^{(j)}(s, a^{(j)})) \\
&+ \gamma \sum_{s'} \left[ p^{(i)}(s'|s, a^{(i)}) - p^{(j)}(s'|s, a^{(j)}) \right] V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \\
&+ \gamma \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right]
\end{aligned}$$

497 Then we consider the absolute value of the value difference:

$$\begin{aligned}
\left| V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)}) \right| &\leq \left| r^{(i)}(s, a^{(i)}) - r^{(j)}(s, a^{(j)}) \right| \\
&+ \gamma \left| \sum_{s'} \left[ p^{(i)}(s'|s, a^{(i)}) - p^{(j)}(s'|s, a^{(j)}) \right] V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \right| \\
&+ \gamma \left| \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \right| \\
&\quad \text{*property of total variation distance when the set is countable} \\
&= \left| r^{(i)}(s, a^{(i)}) - r^{(j)}(s, a^{(j)}) \right| + 2\gamma MD_{TV}(p^{(i)}(\cdot|s, a^{(i)}), p^{(j)}(\cdot|s, a^{(j)})) \\
&+ \gamma \left| \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \right| \\
&\leq d + \gamma \left| \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \right| \\
&\leq d + \gamma \sup_{s'} \left| V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right| \\
&\quad \text{*by induction} \\
&\leq d + \gamma \left[ d + \gamma \sup_{s''} \left| V^{\pi^{(i)}}(s'', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'', \mathcal{T}^{(j)}) \right| \right] \\
&\leq d + \gamma d + \gamma^2 \sup_{s''} \left| V^{\pi^{(i)}}(s'', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'', \mathcal{T}^{(j)}) \right| \\
&\leq \dots \\
&\leq d + \gamma d + \gamma^2 d + \gamma^3 d + \dots = \frac{d}{1 - \gamma}
\end{aligned}$$

498

□

499 For a special case where reward function  $r(s, a, s')$  only depends on the current state  $s$  and next state  
500  $s'$ , the upper bound of policy value difference is only related to the distance in next state distributions.  
501

502 **Proposition 1.** Let  $\mathcal{T}^{(i)} = \{\mathcal{S}, \mathcal{A}, p^{(i)}, r^{(i)}, \gamma, \rho_0\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}, \mathcal{A}, p^{(j)}, r^{(j)}, \gamma, \rho_0\}$  be two MDPs  
503 sampled from the distribution of tasks  $p(\mathcal{T})$ .  $\pi^{(i)}$  is a deterministic policy on  $\mathcal{T}^{(i)}$  and  $\pi^{(j)}$  is a  
504 deterministic policy on  $\mathcal{T}^{(j)}$ . Assume the reward function only depends on the state and next state  
505  $r^{(i)}(s, a^{(i)}, s') = r^{(j)}(s, a^{(j)}, s') = r(s, s')$ . Let  $M = \sup_{s \in \mathcal{S}, s' \in \mathcal{S}} |r(s, s') + \gamma V^{\pi^{(i)}}(s', \mathcal{T}^{(i)})|$   
506 and  $d = \sup_{s \in \mathcal{S}} 2MD_{TV}(p^{(i)}(\cdot|s, \pi^{(i)}(s)), p^{(j)}(\cdot|s, \pi^{(j)}(s)))$ .  $\forall s \in \mathcal{S}$ , we have

$$\left| V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)}) \right| \leq \frac{d}{1 - \gamma} \quad (4)$$

507 *Proof.* Let  $a^{(i)} = \pi^{(i)}(s)$  and  $a^{(j)} = \pi^{(j)}(s)$ .  $s'$  denotes the next state following state  $s$ .  $s''$  denotes  
508 the next state following  $s'$ . In the special case of  $r^{(i)}(s, a^{(i)}, s') = r(s, s')$ , the value of policy is:

$$\begin{aligned} V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) &= r^{(i)}(s, a^{(i)}) + \gamma \sum_{s'} p^{(i)}(s'|s, a^{(i)}) V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \\ &= \sum_{s'} p^{(i)}(s'|s, a^{(i)}) r(s, s') + \gamma \sum_{s'} p^{(i)}(s'|s, a^{(i)}) V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \\ &= \sum_{s'} p^{(i)}(s'|s, a^{(i)}) \left[ r(s, s') + \gamma V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \right] \end{aligned}$$

509 We can derive the value difference:

$$\begin{aligned} &V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)}) \\ &= \sum_{s'} p^{(i)}(s'|s, a^{(i)}) \left[ r(s, s') + \gamma V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \right] - \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ r(s, s') + \gamma V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \\ &\quad \text{*minus and plus } \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ r(s, s') + \gamma V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \right] \\ &\quad \text{**combine the first two terms, combine the last two terms} \\ &= \sum_{s'} \left[ p^{(i)}(s'|s, a^{(i)}) - p^{(j)}(s'|s, a^{(j)}) \right] \left[ r(s, s') + \gamma V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) \right] \\ &+ \gamma \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \end{aligned}$$

510 Then we take absolute value of the value difference:

$$\begin{aligned} \left| V^{\pi^{(i)}}(s, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s, \mathcal{T}^{(j)}) \right| &\leq 2MD_{TV}(p^{(i)}(\cdot|s, a^{(i)}), p^{(j)}(\cdot|s, a^{(j)})) \\ &+ \gamma \left| \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \right| \\ &\leq d + \gamma \left| \sum_{s'} p^{(j)}(s'|s, a^{(j)}) \left[ V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right] \right| \\ &\leq d + \gamma \sup_{s'} \left| V^{\pi^{(i)}}(s', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s', \mathcal{T}^{(j)}) \right| \\ &\leq d + \gamma \left[ d + \gamma \sup_{s''} \left| V^{\pi^{(i)}}(s'', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'', \mathcal{T}^{(j)}) \right| \right] \\ &\leq d + \gamma d + \gamma^2 \sup_{s''} \left| V^{\pi^{(i)}}(s'', \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'', \mathcal{T}^{(j)}) \right| \\ &\leq \dots \\ &\leq d + \gamma d + \gamma^2 d + \gamma^3 d + \dots = \frac{d}{1 - \gamma} \end{aligned}$$

511

□

## B Algorithm of MCAT

---

**Algorithm 1** MCAT combining context-based meta-RL algorithm with policy transfer
 

---

```

1: Initialize critic networks  $Q_{\theta_1}, Q_{\theta_2}$  and actor network  $\pi_\phi$  with random parameters  $\theta_1, \theta_2, \phi$ 
2: Initialize target networks  $\theta'_1 \leftarrow \theta_1, \theta'_2 \leftarrow \theta_2, \phi' \leftarrow \phi$ 
3: Initialize replay buffer  $\mathcal{B} = \mathcal{B}^{(1)} \cup \mathcal{B}^{(2)} \cup \dots \cup \mathcal{B}^{(|\mathcal{T}|)}$  and  $\mathcal{B}^{(i)} \leftarrow \emptyset$  for each  $i$ .
4: Initialize SIL replay buffer  $\mathcal{D} \leftarrow \emptyset$ 
5: Initialize context encoder  $C_{\psi_C}$ , forward model  $F_{\psi_F}$ , action translator  $H_{\psi_H}$ 
6: Initialize set of trajectory rewards for shared policy on each task in recent timesteps as  $R^{(i)} = \emptyset$ , set of trajectory rewards for transferred policy from  $\mathcal{T}^{(j)}$  to  $\mathcal{T}^{(i)}$  in recent timesteps as  $R^{(j) \rightarrow (i)} = \emptyset$ .  $\bar{R}$  denotes average episode rewards in the set.
7: for each iteration do
8:   // Collect training samples
9:   for each task  $\mathcal{T}^{(i)}$  do
10:    if  $R^{(i)} = \emptyset$  then
11:      use the shared policy in this episode
12:    else if there exist  $j \in 1, 2, \dots, |\mathcal{T}|$  such that  $R^{(j) \rightarrow (i)} = \emptyset$  and  $\bar{R}^{(j)} > \bar{R}^{(i)}$  then
13:      use transferred policy from source task  $\mathcal{T}^{(j)}$  to target task  $\mathcal{T}^{(i)}$  in this episode
14:    else if there exist  $j \in 1, 2, \dots, |\mathcal{T}|$ , such that  $j = \arg \max_{j'} \bar{R}^{(j') \rightarrow (i)}$  and  $\bar{R}^{(j) \rightarrow (i)} > \bar{R}^{(i)}$  then
15:      use transferred policy from source task  $\mathcal{T}^{(j)}$  to target task  $\mathcal{T}^{(i)}$  in this episode
16:    else
17:      use the shared policy in this episode
18:    end if
19:    for  $t = 1$  to TaskHorizon do
20:      Get context latent variable  $z_t = C_{\psi_C}(\tau_{t,K})$ 
21:      Select the action  $a$  based on the transferred policy or shared policy, take the action with noise  $a_t = a + \epsilon$  where  $\epsilon \sim \mathcal{N}(0, \sigma)$ , observe reward  $r_t$  and new state  $s_{t+1}$ .
22:      Update  $\mathcal{B}^{(i)} \leftarrow \mathcal{B}^{(i)} \cup \{s_t, a_t, r_t, s_{t+1}, \tau_{t,K}\}$ 
23:    end for
24:    Compute returns  $R_t = \sum_{k=t}^{\infty} \gamma^{k-t} r_k$  and update  $\mathcal{D} \leftarrow \mathcal{D} \cup \{s_t, a_t, r_t, s_{t+1}, \tau_{t,K}, R_t\}$  for every step  $t$  in this episode.
25:    Update the average reward of shared policy on task  $\mathcal{T}^{(i)}$  (i.e.  $R^{(i)}$ ) if we took shared policy in this episode, or update the average reward of the transferred policy from  $\mathcal{T}^{(j)}$  to  $\mathcal{T}^{(i)}$  (i.e.  $R^{(j) \rightarrow (i)}$ ) if we took the transferred policy.
26:  end for
27:  // Update the context encoder  $C_{\psi_C}$  and forward model  $F_{\psi_F}$  with  $\mathcal{L}_{forw}$  and  $\mathcal{L}_{cont}$ 
28:  // Update the action translator  $H_{\psi_H}$  with  $\mathcal{L}_{trans}$ 
29:  // Update the critic network  $Q_{\theta_1}, Q_{\theta_2}$  and actor network  $\pi_\phi$  with TD3 and SIL objective function
30:  for step in training steps do
31:    Update  $\theta_1, \theta_2$  for the critic networks to minimize  $\mathcal{L}_{td3} + \mathcal{L}_{sil}$  (see Algorithm 2)
32:    Update  $\phi$  for the actor network with deterministic policy gradient
33:    Update the  $\theta'_1, \theta'_2, \phi'$  for target networks with soft assignment
34:  end for
35:  // Update the trajectory reward for shared policy and transferred policy if necessary
36:  for each task  $\mathcal{T}^{(i)}$  do
37:    pop out trajectory rewards in  $R^{(i)}$  which were stored before the last G timesteps
38:    pop out trajectory rewards in  $R^{(j) \rightarrow (i)} (\forall j)$  which were stored before the last G timesteps
39:  end for
40: end for

```

---



---

**Algorithm 2** Compute critic loss based on TD3 algorithm and SIL algorithm

---

- 1: Sample batch data of transitions  $(s_t, a_t, r_t, s_{t+1}, \tau_{t,K}) \in \mathcal{B}$
  - 2: Get context variable  $z_t = C_{\psi_C}(\tau_{t,K})$ .
  - 3: Get next action  $a_{t+1} \sim \pi_{\phi'}(z_t, s_{t+1}) + \epsilon, \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  - 4: Get target value for critic network  $y = r_t + \gamma \min_{l=1,2} Q_{\theta'_l}(z_t, s_{t+1}, a_{t+1})$ .
  - 5: Compute TD error  $\mathcal{L}_{td3} = \min_{l=1,2} (y - Q_{\theta_l}(z_t, s_{t+1}, a_{t+1}))^2$
  - 6: Sample batch data of transitions  $(s_t, a_t, \tau_{t,K}, R_t) \in \mathcal{D}$
  - 7: Get context variable  $z_t = C_{\psi_C}(\tau_{t,K})$ .
  - 8: Compute SIL loss  $\mathcal{L}_{sil} = \sum_{l=1,2} \max(R_t - Q_{\theta_l}(z_t, s_t, a_t), 0)^2$
-

## C Experiment Details

In this section, we explain more details for Section 5 and show additional experimental results.

### C.1 Environment

**MuJoCo** We use Hopper, HalfCheetah and Ant environments from OpenAI Gym [3] based on the MuJoCo physics engine [31]. The goal is to move forward while keeping the control cost minimal.

- **Hopper** Hopper agent consists of 5 rigid links with 3 joints. Observation  $s_t$  is an 11-dimension vector consisting of root joint’s position (except for x-coordinate) and velocity angular position and velocity of all 3 joints. Action  $a_t$  lies in the space  $[-1.0, 1.0]^3$ , which corresponds to the torques applied to 3 joints. Reward  $r_t = v_{\text{torso},t} - 0.001\|a_t\|^2 + 1.0$  means the forward velocity of the torso  $v_{\text{torso},t}$  minus the control cost for action  $0.001\|a_t\|^2$  and plus the survival bonus 1.0 at each step. We modify the size of each rigid part to enlarge/contract the body of the agent, so we can create tasks with various dynamics.
- **HalfCheetah** Half-cheetah agent consists of 7 rigid links (1 for torso, 3 for forelimb, and 3 for hindlimb), connected by 6 joints. State  $s_t$  is a 17-dimension vector consisting of root joint’s position (except for x-coordinate) and velocity, angular position and velocity of all 6 joints. Action  $a_t$  is sampled from the space  $[-1.0, 1.0]^6$ , representing the torques applied to each of the 6 joints. Reward  $r_t = v_{\text{torso},t} - 0.1\|a_t\|^2$  is the forward velocity of the torso minus the control cost for action. In order to design multiple tasks with varying dynamics on HalfCheetah, we modify the armature value (similarly to [41]) or scale the mass of each rigid link by a fixed scale factor (similarly to [12]).
- **Ant** Ant agent consists of 13 rigid links connected by 8 joints. Observation  $s_t$  is a 27-dimension vector including information about the root joint’s position and velocity, angular position and velocity of all 8 joints, and frame orientations. Action  $a_t \in [-1.0, 1.0]^8$  is the torques applied to each of 8 joints. Reward is  $r_t = v_{\text{torso},t} + 0.05$ , meaning the velocity of moving forward plus the survival bonus 0.05 for each step. To change the environment dynamics, we modify the damping of every leg. Specifically, given a scale factor  $d$ , we modify two legs to have damping multiplied by  $d$ , and the other two legs to have damping multiplied by  $1/d$  (similarly to [12]). Alternatively, we can cripple one of the agent’s four legs to change the dynamics function. The torques applied to two joints on the crippled leg (i.e. two correspondent elements in actions) are set as 0. (similarly to [25]).

**MetaWorld** Additionally, we consider the tasks of pushing Cylinder, Coffee Mug and Cube. They are named as push-v2, coffee-push-v2, and sweep-into-goal-v2 on MetaWorld benchmark [37] respectively. The goal is to move the objects from a random initial location to a random goal location. The observation is of dimension 14, consisting of the location of the robot hand, the distance between two gripper fingers, the location and position of the object, and the target location. The action  $a \in [-1.0, 1.0]^4$  controls the movement of the robot hand and opening/closing of the gripper. The reward is 1.0 when the object is close to the target location (i.e. distance less than 0.05). Otherwise, the environment reward is 0.0. The length of an episode is 500 steps. The tasks of manipulating different objects have different dynamics. We change the physics parameters armature and damping across tasks to make the policy transfer more challenging.

### C.2 Implementation Details for Policy Transfer with Fixed Dataset & Source Policy

In Section 5.1, we study the performance of policy transfer with our action translator with a fixed dataset and source policy. In this experiment, we demonstrate our proposed policy transfer approach trained with fixed datasets and source policy outperforms the baselines. We provide the experimental details as follows.

#### Source Policy and Dataset

- **MuJoCo** On HalfCheetah, the armature value on the source and target task is 0.1 and 0.5 respectively. On Ant, the leg 0 is crippled on the source task while the leg 3 is crippled on the target task. We train well-performing policies on the source tasks as source policies, and we also train mediocre policies on both source tasks and target tasks to obtain training data.

Parameter name	Value
Start Timesteps	2.5e4
Gaussian exploration noise $\sigma$	0.1
Batch Size	256
Discount $\gamma$	0.99
Target network update rate	5e-3
Policy noise $\tilde{\sigma}$	0.2
Noise clip $c$	0.5
Policy update frequency	2
Replay buffer size	1e6
Actor learning rate	3e-4
Critic learning rate	3e-4
Optimizer	Adam
Actor layers	3
Hidden dimension	256

Table 6: The hyperparameters for TD3 algorithm.

We apply the TD3 algorithm[8] and dense rewards to learn policies. The hyperparameters for the TD3 algorithm are listed in Table 6. Specifically, during the start 25K timesteps, the TD3 agent collects data by randomly sampling from the action space. After the first 25K timesteps, the agent learns an deterministic policy based on the data collected in the replay buffer. During training, the agent collects data with actions following the learned policy with Gaussian noise, and updates the replay buffer as well. On HalfCheetah environment, we use the learned policy at 300K timesteps as good policy, and use the learned policy at 80K timesteps as mediocre policy. On Ant environment, the learned policy at 400K timesteps and 20K timesteps are used as good policy and mediocre policy respectively.

With the mediocre policies, we collect 100K transition samples on the source and target tasks respectively. During data collection, at each step, we record the following information: (a) current state; (b) current action drawn from the mediocre policies; (c) next state; (d) historical observations in the past 10 steps; (e) historical actions in the past 10 steps. The historical transition information are employed to learn the context model for forward dynamics prediction.

- **MetaWorld** On source tasks, we keep the default physics parameters. However, on the target task, the value of armature and damping for the gripper joints is 0.1 multiplying the default. We get the manually designed good policies from official public code<sup>2</sup>. The performance of the good source policy is shown in Tab. 7. By adding Gaussian noise following  $\mathcal{N}(0, 1.0)$  to action drawn from the good policies, we collect 100K transition samples on the source and target tasks respectively.

With the fixed datasets on both source and target tasks, we can train action translator to transfer the fixed source policy. First, we learn the forward dynamics model. Then we learn the action translator based on the well-trained forward dynamics model. For fair comparison, we train the baseline [41] and our action translator with the same dataset and source policy. The hyperparameters and network structures applied in the baseline and our approach are introduced as follows

**Transferred Policy [41]** This baseline is implemented using the code provided by Zhang et al. [41]<sup>3</sup>. The forward dynamics model first encodes the state and action as 128-dimensional vectors respectively via a linear layer with ReLU activation. The state embedding and action embedding is then concatenated to predict the next state with an MLP with 2 hidden layers of 256 units and ReLU activation. We train the forward dynamics model with batch size 32 and decaying learning rate from 0.001, 0.0003 to 0.0001. In order to optimize the forward dynamics model, the objective function is L1-loss between the predicted next state and the actual next state. With these hyper-parameters settings, we train the forward model  $F$  and the context model  $C$  for 30 epochs, each epoch with 10K steps.

<sup>2</sup><https://github.com/rlworkgroup/metaworld/tree/master/metaworld/policies>

<sup>3</sup>[https://github.com/sjtuzq/Cycle\\_Dynamics](https://github.com/sjtuzq/Cycle_Dynamics)

The action translator first encodes the state and action as 128-dimensional vectors respectively via a linear layer with ReLU activation. The state embedding and action embedding are then concatenated to generate the translated action via an MLP with 2 hidden layers of 256 units and ReLU activation. As for the objective function with three terms: adversarial loss, domain cycle-consistency loss, and dynamics cycle-consistency loss, we tune three weights. We train the action translator for 30 epochs. After each epoch, the performance of transferred policy with the action translator is evaluated on the target task. We average episode rewards in 100 episodes as the epoch performance. Finally, we report the best epoch performance over the 30 epochs.

Setting	Source policy on source task	Source policy on target task	Transferred policy [41] on target task	Transferred policy (Ours) on target task
HalfCheetah	5121.4	2355.0	<b>3017.1</b> ( $\pm 44.2$ )	2937.2( $\pm 9.5$ )
Ant	476.8	55.8	97.2( $\pm 2.5$ )	<b>208.1</b> ( $\pm 8.2$ )
Cylinder-Mug	317.3	0.0	308.1( $\pm 75.3$ )	<b>395.6</b> ( $\pm 19.4$ )
Cylinder-Cube	439.7	0.0	262.4( $\pm 48.1$ )	<b>446.1</b> ( $\pm 1.1$ )

Table 7: Performance of source and transferred policy on target task. This is expanding Tab. 1 in the main text.

**Transferred Policy (Ours)** We encode the context features with  $K = 10$  past transitions. The historical state information is postprocessed as state differences between two consecutive states. The historical transition at one step is concatenation of past 10 actions and past 10 postprocessed states. The historical transition data are fed into an MLP with 3 hidden layers with [256, 128, 64] hidden units and Swish activation. The context vector is of dimension 10. The forward dynamics model is an MLP with 4 hidden layers of 200 hidden units and ReLU activation, predicting the state difference between two consecutive states in the future  $M=10$  steps. The learning rate is 0.001 and the batch size is 1024. The objective function is simply  $\mathcal{L}_{forw} + \mathcal{L}_{cont}$  (Equation 1 and Equation 2). With these hyper-parameters settings, we train the forward model  $F$  and the context model  $C$  for 30 epochs, each epoch with 10K steps.

The action translator  $H$  first encodes state and action as 128-dimensional vectors respectively. Then, the state embedding and action embedding is concatenated and fed into an MLP with 3 hidden layers of 256 units and ReLU activations. We train the action translator with a decaying learning rate from  $3e-4$ ,  $5e-5$  to  $1e-5$ , and the batch size is also 1024. With these hyper-parameters settings, we train the action translator for 30 epochs, each epoch with 3,000 steps. The objective function is simply  $\mathcal{L}_{trans}$  (Equation 3). After each epoch, the performance of the action translator is also evaluated on the target task via averaging the episode rewards in 100 episodes. Finally, the best epoch performance over the 30 epochs is reported.

**Context-conditioned Action Translator** We also demonstrate the performance of policy transfer on more than two tasks as heatmaps in Fig. 4. The heatmaps demonstrate performance gain when comparing our transferred policy against the source policy on the target task. We calculate the improvement in the average episode rewards for every pair of source-target tasks sampled from the training task set. The tasks in the HalfCheetah environment are  $\mathcal{T}^{(1)} \dots \mathcal{T}^{(5)}$  with different armature values, namely {0.1, 0.2, 0.3, 0.4, 0.5}. The tasks in the Ant environment are  $\mathcal{T}^{(1)} \dots \mathcal{T}^{(4)}$  with different leg crippled, namely {0, 1, 2, 3}. As mentioned above, we apply the TD3 algorithm[8] and dense rewards to learn source policies and mediocre policies for each task in training set. Then we collect 100K transition data on each training tasks with the corresponding mediocre policies.

The architecture of context model  $C$  and the forward model  $F$  remains the same as above, while the learning rate is kept as  $5e-4$  instead. The architecture of action translator  $H$  is expanded to condition on the source task embeddings and target task embeddings. As mentioned in Sec. 2.2, in order to get the representative task feature for any arbitrary training task, we sample 1024 historical transition samples on this task, calculate the their context embedding through context model  $C$  and average the 1024 context embedding to get the task feature as an 10-dimensional context vector. The source target feature and target task feature are then encoded as 128-dimensional vectors respectively via a linear layer with ReLU activation. Then the state embedding, action embedding, source task embedding and target task embedding are concatenated to produce the translated action via an MLP with 3 linear layers of 256 hidden units and ReLU activation. The learning rate and batch size for  $H$  are  $3e-4$  and 1024. With these hyper-parameters settings, we train the action translator with 100 epochs, each with

1,000 steps. We report the percentage gain comparing well-trained transferred policies with source policies on each pair of source-target tasks.

### C.3 Policy transfer on tasks sharing a general reward function, differing in dynamics

As explained in Sec. 3, many real-world sparse-reward tasks fall under the umbrella of Proposition 1. Thus, we are mainly interested in policy transfer across tasks with the same reward function  $r(s, s')$  but different dynamics. To solve policy transfer across these tasks, our objective function  $\mathcal{L}_{trans}$  can be applied so that the transferred policy achieves a value on the target task similar to the source policy on the source task. Experiments in Sec. 5 validate the efficacy of  $\mathcal{L}_{trans}$  for learning policy transfer.

As for a more general case, we further consider tasks with different dynamics that have **the same state space, action space and reward function, where the general reward function  $r(s, a, s')$  cannot be expressed as  $r(s, s')$** . Theorem 1 in Appendix A covers this scenario. For source task  $\mathcal{T}^{(j)} = \{\mathcal{S}, \mathcal{A}, p^{(j)}, r, \gamma, \rho_0\}$  and target task  $\mathcal{T}^{(i)} = \{\mathcal{S}, \mathcal{A}, p^{(i)}, r, \gamma, \rho_0\}$ , we can bound the value difference between source policy  $\pi^{(j)}$  and transferred policy  $\pi^{(i)}$  by minimizing both reward difference  $|r(s, \pi^{(i)}(s)) - r(s, \pi^{(j)}(s))|$  and total-variation difference in next state distribution  $D_{TV}(p^{(i)}(\cdot|s, \pi^{(i)}(s)), p^{(j)}(\cdot|s, \pi^{(j)}(s)))$ . Accordingly, we modify transfer loss  $\mathcal{L}_{trans}$  (Equation 3) with an additional term of reward difference.

Formally,  $\mathcal{L}_{trans,r} = |r_t^{(j)} - R(s_t^{(j)}, \tilde{a}^{(i)}, z^{(i)})| - \lambda \log F(s_{t+1}^{(j)}|s_t^{(j)}, \tilde{a}^{(i)}, z^{(i)})$ , where  $R$  is a learned reward prediction model,  $\lambda$  is a hyper-parameter weight of next state distribution loss, and  $\tilde{a}^{(i)} = H(s_t^{(j)}, a_t^{(j)}, z^{(j)}, z^{(i)})$  is the translated action. This objective function drives the action translator  $H$  to find an action on the target task leading to a reward and next state, similarly to the source action on the source task.

As explained in Appendix C.1.1, MuJoCo environments award the agent considering its velocity of moving forward  $v_{torso}$  and the control cost  $\|a\|^2$ , i.e.  $r = v_{torso} - c\|a\|^2$ . If the coefficient  $c = 0$ , we can simplify this reward function as  $r(s, s')$  because  $v_{torso}$  is calculated only based on the current state  $s$  and next state  $s'$ . If  $c > 0$ ,  $r$  becomes a general reward function  $r(s, a, s')$ . We evaluate our action translator trained with  $\mathcal{L}_{trans}$  and  $\mathcal{L}_{trans,r}$  for this general case of reward function. We search the hyper-parameter value of  $\lambda$  in  $\mathcal{L}_{trans,r}$  and  $\lambda = 10$  performs well across settings.

Control cost coefficient	Source policy on source task	Source policy on target task	Transferred policy [41] on target task	Transferred policy (ours with $\mathcal{L}_{trans}$ ) on target task	Transferred policy (ours with $\mathcal{L}_{trans,r}$ ) on target task
c=0.001	511.1	54.7	133.27	193.7	<b>203.1</b>
c=0.002	488.4	53.7	129.86	179.3	<b>195.4</b>
c=0.005	475.8	38.9	112.36	148.5	<b>171.8</b>

Table 8: Average episode rewards on Ant environments. We consider the settings with different coefficients for control cost.

Our action translator with either  $\mathcal{L}_{trans}$  or  $\mathcal{L}_{trans,r}$  performs well for policy transfer. When the rewards depend on the action more heavily (i.e.  $c$  becomes larger), the advantage of  $\mathcal{L}_{trans,r}$  becomes more obvious. However, ours with  $\mathcal{L}_{trans,r}$  requires the extra complexity of learning a reward prediction model  $R$ . When the reward function is mostly determined by the states and can be approximately simplified as  $r(s, s')$ , we recommend  $\mathcal{L}_{trans}$  because it is simpler and achieves a competitive performance.

On Hopper and HalfCheetah, the control cost coefficient is  $c > 0$  by default. Our proposed policy transfer and MCAT achieve performance superior to the baselines on these environments (Sec. 5). This verifies the merits of our objective function  $\mathcal{L}_{trans}$  on tasks with a general reward function  $r(s, a, s')$ .

### C.4 Implementation Details for Comparison with Context-based Meta-RL Algorithms

#### C.4.1 Environment

We modify the physics parameters in the environments to get multiple tasks with varying dynamics functions. We delay the environment rewards to make sparse-reward tasks so that the baseline

685 methods may struggle in these environments. The episode length is set as 1000 steps. The details of  
the training task set and test task set are shown in Table 9.

Environment	Reward Delay Steps	Physics Parameter	Training Tasks	Test Tasks
Hopper	100	Size	{0.02, 0.03, 0.04, 0.05, 0.06}	{0.01, 0.07}
HalfCheetah	500	Armature	{0.2, 0.3, 0.4, 0.5, 0.6}	{0.05, 0.1, 0.7, 0.75}
		Mass	{0.5, 1.0, 1.5, 2.0, 2.5}	{0.2, 0.3, 2.7, 2.8}
Ant	500	Damping	{1.0, 10.0, 20.0, 30.0}	{0.5, 35.0}
		Crippled Leg	{ No crippled leg, crippled leg 0, 1, 2 }	{ crippled leg 3 }

Table 9: Modified physics parameters used in the experiments.

686

#### 687 C.4.2 Implementation Details

688 In Section 5.2, we compare our proposed method with other context-based meta-RL algorithms on  
689 environments with sparse rewards. Below we describe the implementation details of each method.

690 **PEARL[21]** We use the implementation provided by the authors<sup>4</sup>. The PEARL agent consists of  
691 the context encoder model and the policy model. Following the default setup, the context encoder  
692 model is an MLP encoder with 3 hidden layers of 200 units each and ReLU activation. We model the  
693 policy as Gaussian, where the mean and log variance is also parameterized by MLP with 3 hidden  
694 layers of 300 units and ReLU activation. Same to the default setting, the log variance is clamped  
695 to [-2, 20]. We mostly use the default hyper-parameters and search the dimension of the context  
696 vector in {5, 10, 20}. We report the performance of the best hyper-parameter, which achieves highest  
697 average score on training tasks.

698 **MQL[5]** We use the implementation provided by the authors<sup>5</sup>. The context encoder is a Gated  
699 Recurrent Unit model compressing the information in recent historical transitions. The actor network  
700 conditioning on the context features is an MLP with 2 hidden layers of 300 units each and a ReLU  
701 activation function. The critic network is of the same architecture as the actor network. We search the  
702 hyper-parameters: learning rate in {0.0003, 0.0005, 0.001}, history length in {10, 20}, GRU hidden  
703 units in {20, 30}, TD3 policy noise in {0.1, 0.2}, TD3 exploration noise in {0.1, 0.2}. We report the  
704 performance of the best set of hyper-parameters, which achieves highest score on training tasks.

705 **Distral[30]** We use the implementation in the MTRL repository<sup>6</sup>. The Distral framework consists  
706 of a central policy and several task-specific policies. The actor network of the central policy is  
707 an MLP with 3 hidden layers of 400 units each and a ReLU activation function. The actor and  
708 critic networks of the task-specific policies are of the same architecture as the actor network of the  
709 central policy. As for the hyperparameters, we set  $\alpha$  to 0.5 and search  $\beta$  in {1, 10, 100}, where  $\frac{\alpha}{\beta}$   
710 controls the divergence between central policy and task-specific policies, and  $\frac{1}{\beta}$  controls the entropy  
711 of task-specific policies. When optimizing the actor and critic networks, the learning rates are 1e-3.  
712 We report the performance of the best hyper-parameter, which achieves highest average score on  
713 training tasks.

714 **HiP-BMDP[40]** We use the implementation in the MTRL repository (same as the Distral baseline  
715 above). The actor and critic networks are also the same as the ones in Distral above. When optimizing  
716 the actor and critic network, the learning rates for both of them are at 1e-3. The log variance of the  
717 policy is bound to [-20, 2]. We search the  $\Theta$  learning error weight  $\alpha_\psi$  in {0.01, 0.1, 1}, which scales  
718 their task bisimulation metric loss. We report the performance of the best hyper-parameter, which  
719 achieves highest average score on training tasks.

720 **MCAT (Ours)** The architectures of the context model  $C$ , forward dynamics model  $F$  and the  
721 action translator  $H$  are the same as introduced in Appendix C.2. The actor network and critic network  
722 are both MLPs with 2 hidden layers of 256 units and ReLU activations. As described in Algorithm 1,

<sup>4</sup><https://github.com/katerakelly/oyster>

<sup>5</sup><https://github.com/amazon-research/meta-q-learning>

<sup>6</sup><https://github.com/facebookresearch/mtrl>



at each iteration, we collect 5K transition data from training tasks. Then we train the context model  $C$  and forward dynamics model  $F$  for 10K training steps. We train the action translator  $H$  for 1K training steps. The actor and critic networks are updated for 5K training steps. In order to monitor the performance of transferred and learned policy in recent episodes, we clear the information about episode reward in  $R^{(i)}$  and  $R^{(j) \rightarrow (i)}$  before the last  $G = 20000$  steps.

The learning rate and batch size of training  $C$ ,  $F$  and  $H$  are the same as introduced in “Context-conditioned Action Translator” in Appendix C.2. The hyper-parameters of learning the actor and critic are the same as listed in Table 6. Besides, we adapt the official implementation<sup>7</sup> to maintain SIL replay buffer with their default hyper-parameters on MuJoCo environments.

Even though there are a couple of components, they are trained alternatively not jointly. The dynamics model is learned with  $\mathcal{L}_{forw}$  to accurately predict the next state. The learned context embeddings for different tasks can separate well due to the regularization term  $\mathcal{L}_{const}$ . With the fixed context encoder and dynamics model, the action translator can be optimized. Then, with the fixed context encoder, the context-conditioned policy learns good behavior from data collected by the transferred policy. These components are not moving simultaneously and this fact facilitates the learning process. To run our approach on MuJoCo environments, for each job, we need to use one GPU card (NVIDIA GeForce GTX TITAN X) for around 4 days. Fig. 6 show the performance of our approach and baselines on various environments.

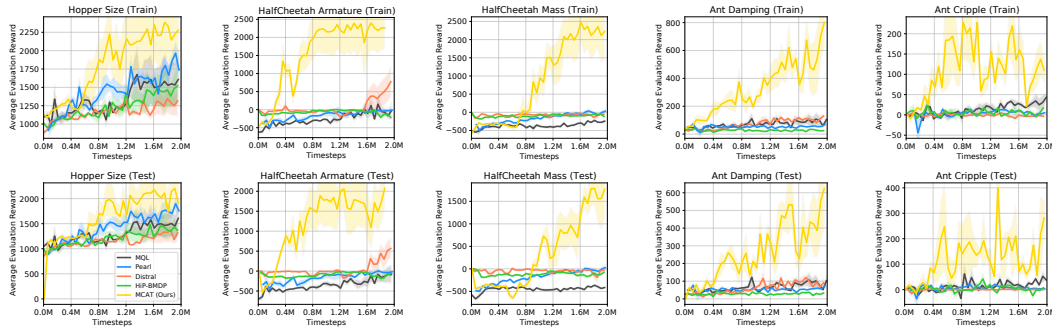


Figure 6: Learning curves of episode rewards on both training and test tasks, averaged over 3 runs. Shadow areas indicate standard error. This adds the performance on training tasks in comparison to Fig. 5

Furthermore, we present additional experimental results on MetaWorld environment. In Section 5.1, we introduced the tasks of moving objects to target locations and the reward is positive only when the object is close to the goal. We combine context-based TD3 with policy transfer to learn a policy operating multiple objects: drawer, coffee mug, soccer, cube, plate. Then we test whether the policy could generalize to moving a large cylinder. In Tab. 10, MCAT agent earns higher success rate than the baselines on both training and test tasks after 2M timesteps in the sparse-reward tasks.

	MQL [5]	PEARL [21]	PCGrad [36]	MCAT
Training tasks (reward)	164.8(±23.6)	161.2(±25.3)	44.8(±31.7)	<b>204.1(±43.1)</b>
Test tasks (reward)	0.0(±0.0)	0.0(±0.0)	0.0(±0.0)	<b>10.2(±8.3)</b>
Training tasks (success rate)	40.0%(±0.0%)	33.3%(±5.4%)	10.0%(±7.1%)	<b>53.3%(±5.4%)</b>
Test tasks (success rate)	0.0%(±0.0%)	0.0%(±0.0%)	0.0%(±0.0%)	<b>16.7%(±13.6%)</b>

Table 10: Performance of learned policies at 2M timesteps, averaged over 3 runs.

<sup>7</sup><https://github.com/junhyukoh/self-imitation-learning>

## D Ablative Study

### D.1 Effect of Policy Transfer

In Section 5.3, we investigate the effect of policy transfer (PT). In Figure 7 we provide the learning curves of MCAT and MCAT w/o PT on both training tasks and test tasks.

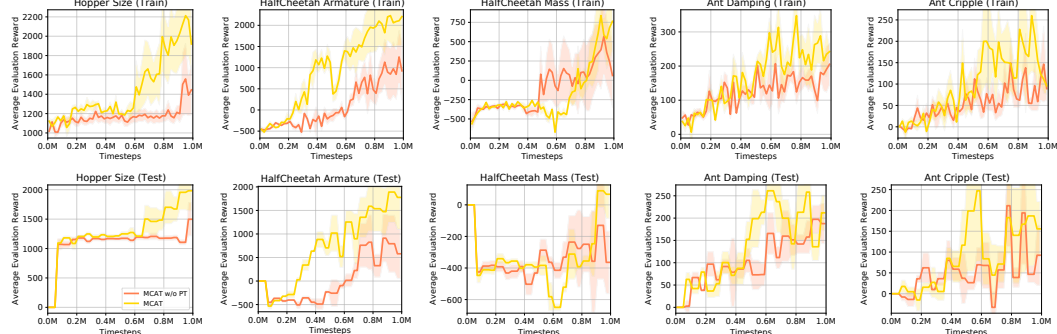


Figure 7: Learning curves of the average episode reward, averaged over 3 runs. The average episode reward and standard error are reported on training tasks and test tasks respectively. This repeats Figure 7 with addition of learning curves on training tasks.

### D.2 More Sparse Rewards

In Section 5.3, we report the effect of policy transfer when the rewards become more sparse in the environments. On HalfCheetah, we delay the environment rewards for different number of steps 200, 350, 500. In Figure 8, we show the learning curves on training and test tasks. In Table 4, we report the average episode rewards and standard error over 3 runs at 1M timesteps.

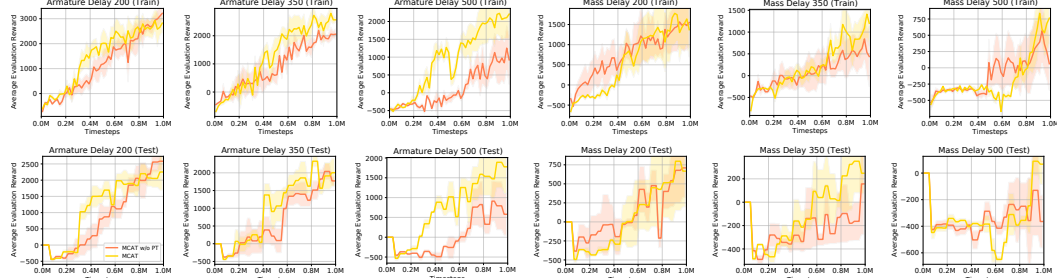


Figure 8: Learning curves of the average episode reward, averaged over 3 runs. The average episode reward and standard error are reported on training tasks and test tasks respectively.

### D.3 More Diverse Tasks

We include more settings of training and test tasks where the discrepancy among training tasks varies. On HalfCheetah, the environment rewards are delayed for 500 steps. In Table 11, we list the details of the settings.

Physics Parameter	Setting	Train	Test
Armature	Set 1	{0.2, 0.25, 0.3, 0.35, 0.4}	{0.05, 0.1, 0.5, 0.55}
	Set 2	{0.2, 0.3, 0.4, 0.5, 0.6}	{0.2, 0.3, 0.7, 0.75}
	Set 3	{0.2, 0.35, 0.5, 0.65, 0.8}	{0.2, 0.3, 0.9, 0.95}
Mass	Set 1	{0.5, 0.75, 1.0, 1.25, 1.5}	{0.2, 0.3, 1.7, 1.8}
	Set 2	{0.5, 1.0, 1.5, 2.0, 2.5}	{0.2, 0.3, 2.7, 2.8}
	Set 3	{0.5, 1.25, 2.0, 2.75, 3.5}	{0.2, 0.3, 3.7, 3.8}

Table 11: Modified physics parameters used in the experiments.

We consider baseline MQL because it performs reasonably well on HalfCheetah among all the baselines (Figure 5). Table 12 demonstrates that policy transfer (PT) is generally and consistently



effective. In Figure 9, we show the learning curves on training and test tasks. In Table 12, we report the average episode rewards and standard error over 3 runs at 1M timesteps.

Setting	Armature Set 1		Armature Set 2		Armature Set 3		Mass Set 1		Mass Set 2		Mass Set 3	
Task	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
MQL	-129.3 ( $\pm 46.7$ )	-248.0 ( $\pm 32.0$ )	-277.2 ( $\pm 25.2$ )	-335.0 ( $\pm 20.8$ )	-85.0 ( $\pm 33.5$ )	-214.7 ( $\pm 28.9$ )	-100.8 ( $\pm 37.8$ )	-291.3 ( $\pm 25.8$ )	-403.7 ( $\pm 16.1$ )	-467.8 ( $\pm 6.5$ )	-175.3 ( $\pm 6.2$ )	-287.9 ( $\pm 11.7$ )
MCAT w/o PT	837.6 ( $\pm 646.5$ )	785.3 ( $\pm 733.1$ )	924.0 ( $\pm 690.1$ )	579.1 ( $\pm 527.1$ )	452.8 ( $\pm 386.6$ )	616.5 ( $\pm 305.0$ )	-60.5 ( $\pm 313.4$ )	-258.2 ( $\pm 151.1$ )	62.5 ( $\pm 411.0$ )	-364.3 ( $\pm 198.5$ )	-328.1 ( $\pm 55.8$ )	-412.4 ( $\pm 7.7$ )
MCAT	3372.1 ( $\pm 186.4$ )	2821.9 ( $\pm 137.7$ )	2207.3 ( $\pm 697.7$ )	1776.8 ( $\pm 680.8$ )	1622.2 ( $\pm 402.2$ )	918.3 ( $\pm 142.5$ )	1222.2 ( $\pm 754.9$ )	482.4 ( $\pm 624.2$ )	763.4 ( $\pm 377.7$ )	67.1 ( $\pm 152.9$ )	705.7 ( $\pm 503.4$ )	-86.2 ( $\pm 111.8$ )
Improvement(%)	302.6	259.3	133.9	206.8	258.3	49.0	2120.2	286.8	1121.4	118.4	315.1	79.1

Table 12: The performance of learned policy on various task settings. We modify *armature* and *mass* to get 5 training tasks and 4 test tasks in each setting. We compute the improvement of MCAT over MCAT w/o PT.

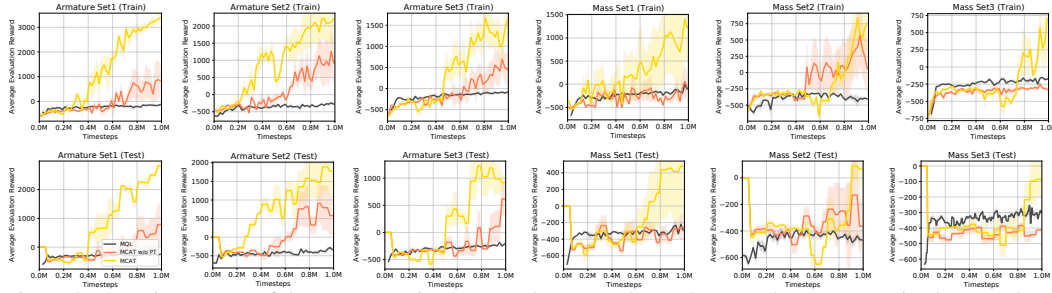


Figure 9: Learning curves of the average episode reward, averaged over 3 runs. The average episode reward and standard error are reported on training tasks and test tasks respectively.

#### D.4 Effect of Self-Imitation Learning

We run experiments combining baseline methods with self-imitation learning (SIL) [18]. SIL brings improvement to baselines but still ours shows significant advantages. In Tab. 13, MCAT w/o SIL compares favorably with the baseline methods. MCAT further improves the performance of MCAT w/o SIL, and MCAT outperform the variants of baseline methods with SIL.

Setting	Hopper Size		HalfCheetah Armature		HalfCheetah Mass		Ant Damping		Ant Cripple	
Task	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
MQL[5]	1586.1 ( $\pm 321.4$ )	1607.5 ( $\pm 327.5$ )	-31.4 ( $\pm 243.5$ )	-77.9 ( $\pm 214.3$ )	-243.1 ( $\pm 69.8$ )	-413.9 ( $\pm 11.1$ )	93.8 ( $\pm 24.5$ )	103.1 ( $\pm 35.7$ )	17.4 ( $\pm 4.3$ )	38.2 ( $\pm 4.0$ )
Distal[30]	1364.0 ( $\pm 216.3$ )	1319.8 ( $\pm 162.2$ )	774.7 ( $\pm 405.9$ )	566.9 ( $\pm 246.7$ )	-54.3 ( $\pm 14.8$ )	-29.5 ( $\pm 3.0$ )	123.0 ( $\pm 20.0$ )	90.5 ( $\pm 28.4$ )	-2.5 ( $\pm 1.7$ )	-0.1 ( $\pm 0.7$ )
HiP-BMDP[40]	1590.3 ( $\pm 238.7$ )	1368.3 ( $\pm 150.7$ )	-212.4 ( $\pm 52.2$ )	-102.4 ( $\pm 24.9$ )	-81.3 ( $\pm 8.31$ )	-101.8 ( $\pm 29.6$ )	15.0 ( $\pm 5.7$ )	33.1 ( $\pm 6.0$ )	12.7 ( $\pm 5.3$ )	7.3 ( $\pm 2.6$ )
MCAT w/o SIL	1261.6 ( $\pm 55.2$ )	1165.1 ( $\pm 8.6$ )	1548.8 ( $\pm 418.4$ )	883.8 ( $\pm 267.2$ )	610.6 ( $\pm 482.3$ )	119.0 ( $\pm 210.0$ )	123.3 ( $\pm 25.8$ )	123.8 ( $\pm 26.9$ )	97.3 ( $\pm 3.6$ )	163.1 ( $\pm 26.1$ )
MQL+SIL	1395.5 ( $\pm 60.8$ )	1398.9 ( $\pm 85.9$ )	1399.7 ( $\pm 350.2$ )	743.5 ( $\pm 246.1$ )	617.8 ( $\pm 133.1$ )	-63.3 ( $\pm 158.3$ )	153.0 ( $\pm 28.3$ )	144.3 ( $\pm 28.1$ )	13.9 ( $\pm 19.8$ )	10.2 ( $\pm 2.3$ )
Distal+SIL	1090.2 ( $\pm 18.7$ )	1090.9 ( $\pm 7.8$ )	1014.1 ( $\pm 121.4$ )	970.3 ( $\pm 164.2$ )	809.7 ( $\pm 294.2$ )	746.7 ( $\pm 120.5$ )	174.3 ( $\pm 66.1$ )	122.2 ( $\pm 44.5$ )	107.7 ( $\pm 57.7$ )	9.1 ( $\pm 5.0$ )
HiP-BMDP+SIL	1573.3 ( $\pm 32.4$ )	1589.5 ( $\pm 110.3$ )	954.8 ( $\pm 192.3$ )	713.3 ( $\pm 85.4$ )	953.5 ( $\pm 61.2$ )	506.6 ( $\pm 99.0$ )	653.9 ( $\pm 262.6$ )	523.6 ( $\pm 300.8$ )	<b>170.9</b> ( $\pm 68.7$ )	215.4 ( $\pm 130.3$ )
MCAT (Ours)	<b>2278.8</b> ( $\pm 426.2$ )	<b>1914.8</b> ( $\pm 373.2$ )	<b>2267.2</b> ( $\pm 579.2$ )	<b>2071.5</b> ( $\pm 447.4$ )	<b>2226.3</b> ( $\pm 762.6$ )	<b>1771.1</b> ( $\pm 617.7$ )	<b>1322.7</b> ( $\pm 57.4$ )	<b>1014.0</b> ( $\pm 69.9$ )	110.4 ( $\pm 30.5$ )	<b>281.6</b> ( $\pm 65.6$ )

Table 13: Mean ( $\pm$  standard error) of episode rewards on the training and test tasks, at 2M timesteps.

On one task, SIL boosts the performance by exploiting the successful past experiences. But on multiple tasks, enhancing performance on one task with luckily collected good experiences may not benefit the exploration on other tasks. If other tasks have never seen the good performance

before, SIL might even prevent the exploration on these tasks because the shared policy is trained to overfit highly-rewarding transitions on the one task with good past trajectories. We observe that after combining with SIL, the baselines show even more severe performance imbalance among multiple training tasks. Therefore, we believe the idea of policy transfer is complementary to SIL, in that it makes each task benefit from good policies on any other tasks.

## D.5 Effect of Contrastive Loss

To show the contrastive loss indeed helps policy transfer, we compare our method with and without the contrastive loss  $\mathcal{L}_{cont}$  (Equation 2). In Fig. 10, one can observe that  $\mathcal{L}_{cont}$  helps cluster the embeddings of samples from the same task and separate the embeddings from different tasks. We note that the tasks  $\mathcal{T}^{(1)}, \mathcal{T}^{(2)}, \mathcal{T}^{(3)}, \mathcal{T}^{(4)}, \mathcal{T}^{(5)}$  have different values of the physics parameter armature 0.2, 0.3, 0.4, 0.5, 0.6. As mentioned in Sec. 2.2, the learned context embeddings maintain the similarity between tasks. In Fig. 10, the context embeddings of two tasks are closer if their values of armature is closer.

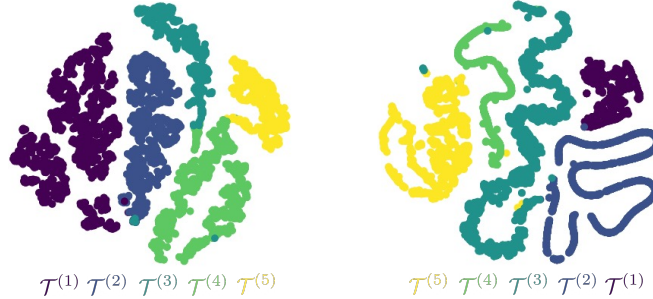


Figure 10: t-SNE visualization[32] of the context embeddings learned via our method with and without contrastive loss. Different colors correspond to different training tasks.

MCAT shows superior performance to the variant without the contrastive loss. Here we show the learning curves on training and test tasks separately(Fig. 11).

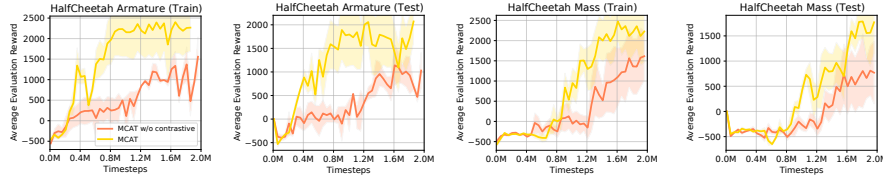


Figure 11: Learning curves of the average episode reward, averaged over 3 runs. The average episode reward and standard error are reported on training tasks and test tasks respectively.

## D.6 Design Choice of Action Translator

We add this experimental comparison with the action translator by [41]. To learn a shared policy solving multiple tasks, we combine the context-based TD3 algorithm, self-imitation learning, and policy transfer with their action translator. Using their action translator underperforms ours. The main reason is that, with changing datasets and policies, their action translator may be harder to tune because there are more moving components (i.e. another action translator, a discriminator) and more loss terms to be balanced (i.e. domain cycle-consistency loss, adversarial loss).

Setting	HalfCheetah Armature		HalfCheetah Mass	
Tasks	Training	Test	Training	Test
MCAT	<b>2267.2</b> ( $\pm 579.2$ )	<b>2071.5</b> ( $\pm 447.4$ )	<b>2226.3</b> ( $\pm 762.6$ )	<b>1771.1</b> ( $\pm 617.7$ )
MCAT with [41] action translator	2255.2 ( $\pm 644.4$ )	1664.8 ( $\pm 660.8$ )	1185.8 ( $\pm 798.0$ )	684.7( $\pm 759.0$ )

Table 14: Mean ( $\pm$  standard error) of episode rewards on training and test tasks at 2M timesteps.

## E Extension of Policy Transfer

As clarified in Sec. 2.1, in this work, we mainly focus on tasks with the same state space, action space, reward function but varying dynamics. However, we note that our proposed method of learning action translator may be extended to tackle the challenge of policy transfer in more general cases, such as (1) Tasks differing in reward function, (2) Tasks differing in state space and action space. In this section, we establish the theory and method in details to extend our policy transfer approach, as a supplement to Sec. 6.

### E.1 Theoretical Analysis

Intuitively, on two general tasks, we aim to discover correspondent state-action pairs achieving the same reward and transiting to correspondent next states. With the state and action correspondence, the behavior of good source policy can be “replicated” in the target task and the high value of the good source policy can be maintained by the transferred policy on the target task. Inspired by this idea, we extend our theory in Sec. 3 and Appendix A.

We first define a binary relation for states to describe the equivalent states on two MDPs (Definition 2) and define an invertible function to capture the state equivalence relation (Definition 3). Building upon the notion of state equivalence, we derive the upper bound of value difference between policies on two MDPs (Theorem 2). Finally, we reach a proposition for the upper bound of value difference (Proposition 2) to explain that our objective function in learning action translator can be extended to bound the value difference between the source and transferred policy.

**Definition 2.** Given two MDPs  $\mathcal{T}^{(i)} = \{\mathcal{S}^{(i)}, \mathcal{A}^{(i)}, p^{(i)}, r^{(i)}, \gamma, \rho_0^{(i)}\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}^{(j)}, \mathcal{A}^{(j)}, p^{(j)}, r^{(j)}, \gamma, \rho_0^{(j)}\}$ , we define a binary relation  $B \in \mathcal{S}^{(i)} \times \mathcal{S}^{(j)}$  called **state equivalence relation**. Let  $s'^{(i)}$  denote the next state following state  $s^{(i)}$ , and  $s'^{(j)}$  denote the next state following state  $s^{(j)}$ . For states  $s^{(i)} \in \mathcal{S}^{(i)}$ ,  $s^{(j)} \in \mathcal{S}^{(j)}$ , we have  $(s^{(i)}, s^{(j)}) \in B$  (i.e.  $s^{(i)}Bs^{(j)}$ ) if for any  $a^{(i)} \in \mathcal{A}^{(i)}$  there exists  $a^{(j)} \in \mathcal{A}^{(j)}$  satisfying the following conditions:

$$r^{(i)}(s^{(i)}, a^{(i)}) = r^{(j)}(s^{(j)}, a^{(j)})$$

$$\forall s'^{(i)} \in \mathcal{S}^{(i)}, \exists s'^{(j)} \in \mathcal{S}^{(j)} \text{ s.t. } p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) = p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \text{ and } s'^{(i)}Bs'^{(j)}$$

We call the state  $s^{(i)}$  and  $s^{(j)}$  are correspondent/equivalent when  $(s^{(i)}, s^{(j)}) \in B$ . Also, in this case, the action  $a^{(i)}$  for state  $s^{(i)}$  on the MDP  $\mathcal{T}^{(i)}$  is equivalent to the action  $a^{(j)}$  for state  $s^{(j)}$  on the MDP  $\mathcal{T}^{(j)}$ .

This definition is related to stochastic bisimulation relation in [6, 39, 40]. Unlike these prior works about state bisimulation, we allow the equivalent actions  $a^{(j)} \neq a^{(i)}$ . So action  $a$  on the task  $\mathcal{T}^{(i)}$  might not be equivalent to  $a$  on the task  $\mathcal{T}^{(j)}$ , and hence we need to involve action translator in learning of both the state correspondence and action correspondence.

Drawing upon Definition 2, we define a one-to-one mapping to identify the equivalent state across two spaces  $\mathcal{S}^{(i)}$  and  $\mathcal{S}^{(j)}$ .

**Definition 3.** Given two MDPs  $\mathcal{T}^{(i)} = \{\mathcal{S}^{(i)}, \mathcal{A}^{(i)}, p^{(i)}, r^{(i)}, \gamma, \rho_0^{(i)}\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}^{(j)}, \mathcal{A}^{(j)}, p^{(j)}, r^{(j)}, \gamma, \rho_0^{(j)}\}$  with state equivalence relation  $B$ , we consider subsets  $\mathcal{S}_B^{(i)} \subset \mathcal{S}^{(i)}$  and  $\mathcal{S}_B^{(j)} \subset \mathcal{S}^{(j)}$  satisfying:  $\forall s^{(i)} \in \mathcal{S}_B^{(i)}, \exists s^{(j)} \in \mathcal{S}_B^{(j)} \text{ s.t. } (s^{(i)}, s^{(j)}) \in B$ . We define an invertible function  $G : \mathcal{S}_B^{(i)} \rightarrow \mathcal{S}_B^{(j)}$  called **state translator function**, satisfying:  $(s^{(i)}, G(s^{(i)})) \in B$ .

Based on Definition 2 and 3, given two correspondent states  $s^{(i)} \in \mathcal{S}_B^{(i)}$  and  $s^{(j)} \in \mathcal{S}_B^{(j)}$ , we can derive the upper bound for the value difference between  $V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)})$  and  $V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)})$ .

**Theorem 2.**  $\mathcal{T}^{(i)} = \{\mathcal{S}^{(i)}, \mathcal{A}^{(i)}, p^{(i)}, r^{(i)}, \gamma, \rho_0^{(i)}\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}^{(j)}, \mathcal{A}^{(j)}, p^{(j)}, r^{(j)}, \gamma, \rho_0^{(j)}\}$  are two MDPs sampled from the distribution of tasks  $p(\mathcal{T})$ .  $\pi^{(i)}$  is a deterministic policy on  $\mathcal{T}^{(i)}$  and  $\pi^{(j)}$  is a deterministic policy on  $\mathcal{T}^{(j)}$ . We assume there exist state equivalence relation  $B \in \mathcal{S}^{(i)} \times \mathcal{S}^{(j)}$  and a state translator function  $G$  defining a one-to-one mapping from  $\mathcal{S}_B^{(i)}$  to  $\mathcal{S}_B^{(j)}$ . Let  $M = \sup_{s^{(i)} \in \mathcal{S}^{(i)}} |V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)})|$  and  $d =$

838  $\sup_{s^{(i)} \in \mathcal{S}_B^{(i)}} [|r^{(i)}(s^{(i)}, \pi^{(i)}(s)) - r^{(j)}(s^{(j)}, \pi^{(j)}(s))| + 2\gamma MD_{TV}(p^{(i)}(\cdot|s^{(i)}, \pi^{(i)}(s^{(i)})), p^{(j)}(G(\cdot)|s^{(j)}, \pi^{(j)}(s^{(j)})))].$   
839 Then  $\forall s^{(i)} \in \mathcal{S}_B^{(i)}, s^{(j)} = G(s^{(i)})$ , we have

$$|V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)})| \leq \frac{d}{1-\gamma}$$

840 *Proof.* Let  $a^{(i)} = \pi^{(i)}(s^{(i)})$  and  $a^{(j)} = \pi^{(j)}(s^{(j)})$ . We rewrite the value difference.

$$\begin{aligned} & V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)}) \\ = & r^{(i)}(s^{(i)}, a^{(i)}) + \gamma \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\ & - r^{(j)}(s^{(j)}, a^{(j)}) - \gamma \sum_{s'^{(j)} \in \mathcal{S}^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \\ = & (r^{(i)}(s^{(i)}, a^{(i)}) - r^{(j)}(s^{(j)}, a^{(j)})) \\ & + \gamma \left( \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - \sum_{s'^{(j)} \in \mathcal{S}^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right) \end{aligned}$$

841 According to Definition 2, since  $s^{(i)} \in \mathcal{S}_B^{(i)}$ , we have  $s'^{(i)} \in \mathcal{S}_B^{(i)}$ . Similarly,  $s'^{(j)} \in \mathcal{S}_B^{(j)}$ .

842 Then we derive the second term in the right side of the equation above:

$$\begin{aligned} & \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - \sum_{s'^{(j)} \in \mathcal{S}^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \\ & \quad \text{*replace } \mathcal{S}^{(i)} \text{ by } \mathcal{S}_B^{(i)} \text{ because } s'^{(i)} \in \mathcal{S}_B^{(i)}, \text{ replace } \mathcal{S}^{(j)} \text{ by } \mathcal{S}_B^{(j)} \text{ because } s'^{(j)} \in \mathcal{S}_B^{(j)} \\ & \quad \text{**minus and plus } \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\ = & \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\ & + \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - \sum_{s'^{(j)} \in \mathcal{S}_B^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \\ & \quad \text{*combine the first two terms, rewrite the third term because } G \text{ is invertible function} \\ = & \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} \left[ p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) - p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \right] V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\ & + \sum_{s'^{(j)} \in \mathcal{S}_B^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - \sum_{s'^{(j)} \in \mathcal{S}_B^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \\ & \quad \text{*combine the last two terms} \\ = & \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} \left[ p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) - p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \right] V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\ & + \sum_{s'^{(j)} \in \mathcal{S}_B^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right] \end{aligned}$$

843 Therefore, we can bound the absolute value of the value difference according to the two equation  
844 arrays above:

$$\begin{aligned}
& \left| V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)}) \right| \\
& \leq \left| r^{(i)}(s^{(i)}, a^{(i)}) - r^{(j)}(s^{(j)}, a^{(j)}) \right| \\
& + \gamma \sum_{s'^{(i)} \in \mathcal{S}_B^{(i)}} \left[ p^{(i)}(s'^{(i)} | s^{(i)}, a^{(i)}) - p^{(j)}(G(s'^{(i)}) | s^{(j)}, a^{(j)}) \right] V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\
& + \gamma \sum_{s'^{(j)} \in \mathcal{S}_B^{(j)}} p^{(j)}(s'^{(j)} | s^{(j)}, a^{(j)}) \left[ V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right] \\
& \leq \left| r^{(i)}(s^{(i)}, a^{(i)}) - r^{(j)}(s^{(j)}, a^{(j)}) \right| + 2\gamma MD_{TV}(p^{(i)}(\cdot | s^{(i)}, a^{(i)}), p^{(j)}(G(\cdot) | s^{(j)}, a^{(j)})) \\
& + \gamma \sup_{s'^{(j)} \in \mathcal{S}_B^{(j)}} \left| V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right| \\
& \leq d + \gamma \sup_{s'^{(i)} \in \mathcal{S}_B^{(i)}} \left| V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(G(s'^{(i)}), \mathcal{T}^{(j)}) \right| \leq \frac{d}{1-\gamma}
\end{aligned}$$

845

□

846 Theorem 2 proves the value difference is upper bounded by a scalar  $d$ , depending on the reward differ-  
847 ence  $|r^{(i)}(s^{(i)}, \pi^{(i)}(s^{(j)})) - r^{(j)}(s^{(j)}, \pi^{(j)}(s^{(j)}))|$  and  $D_{TV}(p^{(i)}(\cdot | s^{(i)}, a^{(i)}), p^{(j)}(G(\cdot) | s^{(j)}, a^{(j)}))$ ,  
848 i.e. the total-variation distance between probability distribution of next state on  $\mathcal{T}^{(i)}$  and probability  
849 distribution of correspondent next state on  $\mathcal{T}^{(j)}$ . Indeed, if the state equivalence relation is only true  
850 for identical states (i.e.  $G$  is an identity mapping,  $s^{(i)} B s^{(j)}$  if and only if  $s^{(i)} = s^{(j)}$ ), then Theorem  
851 2 degenerates into Theorem 1. We note the proof of Theorem 2 is similar to proof of Theorem 1 in  
852 Appendix A.

853 For a special case, where the reward only depends on the current state and next state, we can formulate  
854 a simpler definition of scalar  $d$ . The following Proposition 2 is analogous to Proposition 1 in the  
855 assumption about reward function.

856 **Proposition 2.**  $\mathcal{T}^{(i)} = \{\mathcal{S}^{(i)}, \mathcal{A}^{(i)}, p^{(i)}, r^{(i)}, \gamma, \rho_0^{(i)}\}$  and  $\mathcal{T}^{(j)} = \{\mathcal{S}^{(j)}, \mathcal{A}^{(j)}, p^{(j)}, r^{(j)}, \gamma, \rho_0^{(j)}\}$   
857 are two MDPs sampled from the distribution of tasks  $p(\mathcal{T})$ .  $\pi^{(i)}$  is a deterministic policy  
858 on  $\mathcal{T}^{(i)}$  and  $\pi^{(j)}$  is a deterministic policy on  $\mathcal{T}^{(j)}$ . We assume there exist state equivalence  
859 relation  $B \in \mathcal{S}^{(i)} \times \mathcal{S}^{(j)}$  and a state translator function  $G$  defining a one-to-one mapping  
860 from  $\mathcal{S}_B^{(i)}$  to  $\mathcal{S}_B^{(j)}$ . Suppose that the reward function  $r^{(i)}(s^{(i)}, a^{(i)}, s'^{(i)}) = r^{(i)}(s^{(i)}, s'^{(i)})$   
861 and  $r^{(j)}(s^{(j)}, a^{(j)}, s'^{(j)}) = r^{(j)}(s^{(j)}, s'^{(j)})$ . If  $s^{(j)} = G(s^{(i)})$  and  $s'^{(j)} = G(s'^{(i)})$ ,  
862  $r^{(i)}(s^{(i)}, s'^{(i)}) = r^{(j)}(s^{(j)}, s'^{(j)})$ . Let  $M = \sup_{s^{(i)} \in \mathcal{S}^{(i)}} |r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)})|$  and  
863  $d = \sup_{s^{(i)} \in \mathcal{S}_B^{(i)}} 2MD_{TV}(p^{(i)}(\cdot | s^{(i)}, \pi^{(i)}(s^{(i)})), p^{(j)}(G(\cdot) | s^{(j)}, \pi^{(j)}(s^{(j)})))$ .

864 Then  $\forall s^{(i)} \in \mathcal{S}_B^{(i)}, s^{(j)} = G(s^{(i)})$ , we have

$$\left| V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)}) \right| \leq \frac{d}{1-\gamma}$$

865 *Proof.* Let  $a^{(i)} = \pi^{(i)}(s^{(i)})$  and  $a^{(j)} = \pi^{(j)}(s^{(j)})$ .  $s'^{(i)}$  denotes the next state following state  $s^{(i)}$ .

866 Because the reward solely depends on the current and next state, we rewrite the value function:

$$\begin{aligned}
V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) &= r^{(i)}(s^{(i)}, a^{(i)}) + \gamma \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)} | s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\
&= \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)} | s^{(i)}, a^{(i)}) r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)} | s^{(i)}, a^{(i)}) V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \\
&= \sum_{s'^{(i)} \in \mathcal{S}^{(i)}} p^{(i)}(s'^{(i)} | s^{(i)}, a^{(i)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right]
\end{aligned}$$

867 Then we derive the value difference:

$$\begin{aligned}
& V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)}) \\
&= \sum_{s'^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&- \sum_{s'^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ r^{(j)}(s^{(j)}, s'^{(j)}) + \gamma V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right] \\
&\quad \text{*minus and plus } \sum_{s'^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&= \sum_{s'^{(i)}} p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&- \sum_{s'^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&+ \sum_{s'^{(i)}} p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&- \sum_{s'^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ r^{(j)}(s^{(j)}, s'^{(j)}) + \gamma V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right] \\
&\quad \text{*combine first two terms, rewrite the third term given invertible function } G \\
&= \sum_{s'^{(i)}} \left[ p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) - p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \right] \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&+ \sum_{s'^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ r^{(i)}(G^{-1}(s^{(j)}), G^{-1}(s'^{(j)})) + \gamma V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) \right] \\
&- \sum_{s'^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ r^{(j)}(s^{(j)}, s'^{(j)}) + \gamma V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right] \\
&\quad \text{*combine last two terms, note the assumption of reward function} \\
&= \sum_{s'^{(i)}} \left[ p^{(i)}(s'^{(i)}|s^{(i)}, a^{(i)}) - p^{(j)}(G(s'^{(i)})|s^{(j)}, a^{(j)}) \right] \left[ r^{(i)}(s^{(i)}, s'^{(i)}) + \gamma V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) \right] \\
&+ \gamma \sum_{s'^{(j)}} p^{(j)}(s'^{(j)}|s^{(j)}, a^{(j)}) \left[ V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right]
\end{aligned}$$

868 Therefore, the absolute value of value difference can be upper bounded. The proof is similar to the  
869 proof of Theorem 2.

$$\begin{aligned}
\left| V^{\pi^{(i)}}(s^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s^{(j)}, \mathcal{T}^{(j)}) \right| &\leq 2MD_{TV}(p^{(i)}(\cdot|s^{(i)}, a^{(i)}), p^{(j)}(G(\cdot)|s^{(j)}, a^{(j)})) \\
&+ \gamma \sup_{s'^{(j)} \in \mathcal{S}_B^{(j)}} \left| V^{\pi^{(i)}}(G^{-1}(s'^{(j)}), \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(s'^{(j)}, \mathcal{T}^{(j)}) \right| \\
&\leq d + \gamma \sup_{s'^{(i)} \in \mathcal{S}_B^{(i)}} \left| V^{\pi^{(i)}}(s'^{(i)}, \mathcal{T}^{(i)}) - V^{\pi^{(j)}}(G(s'^{(i)}), \mathcal{T}^{(j)}) \right| \\
&\leq \frac{d}{1-\gamma}
\end{aligned}$$

870

□

871 Obviously, if the state equivalence relation is only true for identical states (i.e.  $G$  is an identity  
872 mapping,  $s^{(i)}Bs^{(j)}$  if and only if  $s^{(i)} = s^{(j)}$ ), then Proposition 2 degenerates into Proposition 1. If  
873 we optimize the action translator  $H$  to minimize  $d$  for policy  $\pi^{(j)}$  and  $\pi^{(i)}(s^{(i)}) = H(s^{(j)}, \pi^{(j)}(s^{(j)}))$ ,

the policy value for correspondent states  $s^{(i)}$  and  $s^{(j)}$  can be close. Minimizing  $d$  means finding actions leading to next states remaining correspondent.

## E.2 Method

According to Proposition 2, we not only learn an action translator  $H$ , but also state translators  $G$  mapping target states  $s^{(i)}$  to the equivalent states on source task  $\mathcal{T}^{(j)}$  and  $G^{-1}$  identifying correspondent state on target task  $\mathcal{T}^{(i)}$ . We additionally learn a discriminator network  $D$  to assist learning of state translator.

Given transition data  $s^{(j)}$  on source task and  $s^{(i)}$  on target task, the adversarial objective is:

$$\min_G \max_D \mathcal{L}_{adv}(G, D) = \log D(s^{(j)}) + \log(1 - D(G(s^{(i)})))$$

$G$  aims to map target state  $s^{(i)}$  to the distribution of states on source task, while  $D$  tries to distinguish translated state  $G(s^{(i)})$  and real states in the source task. To build state equivalence, the translated state should be translated back to the source state. We further leverage cycle consistency loss to learn the one-to-one mapping on states across tasks:

$$\mathcal{L}_{back} = |G^{-1}(G(s^{(i)})) - s^{(i)}| + |G(G^{-1}(s^{(j)})) - s^{(j)}|$$

Drawn upon Proposition 2, we extend our transfer loss  $\mathcal{L}_{trans}$  to  $\mathcal{L}_{trans,s,a}$ . Formally,

$$\mathcal{L}_{trans,s,a} = -\log F(\tilde{s}_{t+1}^{(i)} | \tilde{s}_t^{(i)}, \tilde{a}_t^{(i)})$$

where  $\tilde{s}_{t+1}^{(i)} = G^{-1}(s_{t+1}^{(j)})$ ,  $\tilde{s}_t^{(i)} = G^{-1}(s_t^{(j)})$ , and  $\tilde{a}_t^{(i)} = H(s_t^{(j)}, a_t^{(j)})$ .  $\mathcal{L}_{trans,s,a}$  is applied to optimize the state translator  $G^{-1}$  and action translator  $H$ .

In this way, given the state  $s_t^{(j)}$  on source task, we first get the correspondent state  $\tilde{s}_t^{(i)}$  on target task. Then the translated action  $\tilde{a}_t^{(i)}$  make transition to next state  $\tilde{s}_{t+1}^{(i)}$  on target task still correspondent to next state  $s_{t+1}^{(j)}$  on source task. The objective function  $\mathcal{L}_{trans,s,a}$  drives the next state distribution on the target task  $p^{(i)}(\cdot | \tilde{s}_t^{(i)}, \tilde{a}_t^{(i)})$  to be close to the distribution of correspondent next state on the source task  $p^{(j)}(G(\cdot) | s_t^{(j)}, a_t^{(j)})$ . This is implicitly minimizing  $d$  in Proposition 2.

In practice, we may need the action translator network  $H$  or the state translator network  $G$  and  $G^{-1}$  reasonably initialized, in order to prevent the joint training collapsing to a trivial solution. The implementation details of learning the context model, forward dynamics model and action translator are the same as we explained in Appendix C.2. During training of the state translator, the weight of  $\mathcal{L}_{adv}$ ,  $\mathcal{L}_{back}$ ,  $\mathcal{L}_{trans,s,a}$  is 10, 30, 100 respectively, the same as default hyper-parameters in [41]. The similar technique of learning state translator and action translator has been mentioned in [41]. Yet, our theorems shed light on its underlying mechanism and our objective function for learning the action translator is simpler.

## E.3 Experiments on Tasks Differing in Reward Function

When the tasks share **the same state space and action space but the reward function varies**, we combine our action translator with a state translator for policy transfer.

To investigate this scheme for policy transfer, we conduct experiments on MetaWorld task moving the robot arm to a goal location. We set the source and target task with different goal locations and hence with different reward functions. Tab. 15 lists the goal locations on the source and target tasks. Specifically, on the same state  $s$  of the agent’s current location  $(x, y, z)$ , the reward varies across tasks, because it is inversely proportional to the distance from the current location to goal. The initial location of robot arm is randomly sampled between  $[-0.1, 0.6, 0.02]$  and  $[0.1, 0.7, 0.02]$ . The state is current location of the robot arm. The action is the moving vector of the robot arm.

We compare our method and [41] learning both state translator and action translator. We initialize the state translator networks by assigning  $G(s = (x, y, z)) = G^{-1}(s = (x, y, z)) = (-x, y, z)$ . As observed in Tab. 15, ours compares favorably with [41] and achieves satisfactory cumulative

episode reward on the target task. We conclude that, for source and target tasks with different reward functions depending on the state and next state, learning state translator and action translator jointly is promising for policy transfer.

Source Task	Target Task	Source policy on source task	Source policy on target task	Transferred policy [41] on target task	Transferred policy (Ours) on target task
$[-0.1, 0.8, 0.2]$	$[0.1, 0.8, 0.2]$	4855.7	947.5	1798.2( $\pm 592.4$ )	<b>3124.3</b> ( $\pm 1042.0$ )
$[-0.1, 0.8, 0.2]$	$[0.05, 0.8, 0.2]$	4855.7	1470.2	1764.0( $\pm 316.3$ )	<b>1937.1</b> ( $\pm 424.5$ )
$[-0.1, 0.8, 0.2]$	$[0.1, 0.8, 0.05]$	4855.7	1040.8	<b>2393.7</b> ( $\pm 869.8$ )	2315.7( $\pm 1061.5$ )
2-leg	3-leg	5121.4	NA	1957.8( $\pm 298.4$ )	<b>2018.2</b> ( $\pm 50.8$ )

Table 15: Mean ( $\pm$  standard error) of episode rewards over 3 runs, comparing source and transferred policy on target task. This is expanding Tab. 5 in the main text.

#### E.4 Experiments on Tasks Differing in State and Action Space

For tasks with **different state space and action space**, we investigate the proposed idea on MuJoCo environment HalfCheetah. The HalfCheetah agent by default has 2 legs in the source task and we modify the agent to have 3 legs in the target task. Because the agents have different numbers of joints in the source and target task, the dimensions of state space and action space also differ, as explained in [41]. Again, we compare our method and [41] learning both state translator and action translator. We assign a good initialization for the action translator in both methods as [41] introduced. We remark that ours with a simpler objective function and fewer components than the baseline method can transfer the source policy to perform well on the target task.