A Transfer Approach Using Graph Neural Networks in Deep Reinforcement Learning

Tianpei Yang[1,2*], **Heng You**[1], **Jianye Hao**[1*], **Yan Zheng**[1], **Matthew E. Taylor**[2,3]

[1] College of Intelligence and Computing, Tianjin University,
[2]Univeristy of Alberta,
[3]Alberta Machine Intelligence Institute (Amii)
{tpyang,hengyou,yanzheng,jianye.hao}@tju.edu.cn
matthew.e.taylor@ualberta.ca

Abstract

Transfer learning (TL) has shown great potential to improve Reinforcement Learning (RL) efficiency by leveraging prior knowledge in new tasks. However, much of the existing TL research focuses on transferring knowledge between tasks that share the same state-action spaces. Further, transfer from multiple source tasks that have different state-action spaces is more challenging and needs to be solved urgently to improve the generalization and practicality of the method in real-world scenarios. This paper proposes TURRET (Transfer Using gRaph neuRal nETworks), to utilize the generalization capabilities of Graph Neural Networks (GNNs) to facilitate efficient and effective multi-source policy transfer learning in the state-action mismatch setting. TURRET learns a semantic representation by accounting for the intrinsic property of the agent through GNNs, which leads to a unified state embedding space for all tasks. As a result, TURRET achieves more efficient transfer with strong generalization ability between different tasks and can be easily combined with existing Deep RL algorithms. Experimental results show that TURRET significantly outperforms other TL methods on multiple continuous action control tasks, successfully transferring across robots with different state-action spaces.

Introduction

Deep Reinforcement Learning (DRL) has obtained impressive successes in various domains such as video games (Mnih et al. 2015; Silver et al. 2016) and robotics control (Lillicrap et al. 2016). However, DRL still faces the sample inefficiency problem, requiring considerable environmental interactions. Transfer Learning (TL) has emerged as a promising technique to significantly reduce DRL sample complexity by leveraging prior knowledge (Taylor and Stone 2009; Zhu, Lin, and Zhou 2020; Yang et al. 2020a, 2021). Policy transfer is one major class of RL transfer methods, that focuses on leveraging pre-trained policies on source tasks to accelerate learning in a target task (Rusu et al. 2016; Schmitt et al. 2018; Parisotto, Ba, and Salakhutdinov 2016; Yang et al. 2020a,b; Tao et al. 2021). However, these methods assume source tasks share the same state-action space as the target task, which severely limits generalization to more realistic scenarios where state-action space mismatch usually exists, which we call it cross-domain setting.

Recently, several approaches have explored cross-domain TL from the following directions. Some works align states in a common feature space to combat the mismatch using a state encoder. To train the state encoder, Gupta et al. (2017) require to collect paired data of two tasks using pre-trained policies based on time alignment. However, it is expensive and infeasible in real-world problems that the two agents will perform at roughly the same rate. Later, Wan et al. (2020) train the state encoder using mutual information to ensure a high correlation between the state embeddings and current states. However, they ignore capturing the dynamic information of the environment which ultimately leads to insufficient transfer performance. Some other works learn both the state and action mappings for transfer. For example, Chen et al. (2019) capture the semantics of actions using the effects on the environment, which is only applicable in discrete-action scenarios. Later, Zhang et al. (2021) align the environment dynamics using a cycle consistency constraint. However, this method directly reuses the pre-trained source policy on the target task through the mapping in a zero-short transfer manner, which may not achieve optimal performance. Furthermore, all these above methods do not support transferring from multiple source tasks. Recently, CAT (You et al. 2022) has initially done this, but it fails to handle when the number of source policies changes due to its limitation of learning the one-to-one mapping between each source task and the target task. Since previous works didn't solve this problem properly, we argue a suitable method that can handle the multi-source cross-domain TL problem and is suitable for varying numbers of source tasks is urgently needed to improve the generalizability and practicality. Moreover, all the above methods use a multi-layer perception (MLP) based structure, which is not capable of capturing sufficient information for state-action alignment. This may even hinder the transfer performance on target tasks with large state-action spaces. Please refer to the appendix [1] for more information on related work.

To address these challenges, we propose a novel transfer approach called TURRET (Transfer Using gRaph neuRal

*Corresponding author

[1]https://github.com/tianpeiyang/TURRET_code

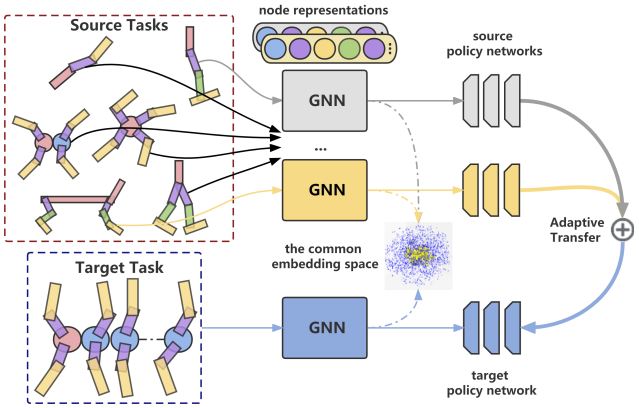


Figure 1: A motivating example of TURRET

nETworks). Figure 1 shows a motivating example of transferring policies across different robots with different sizes and morphologies, where TURRET learns a unified embedding space for all the tasks using Graph Neural Networks (GNNs) and then adaptively accelerates the learning process of the target task with large state-action spaces or completely different morphologies by transferring knowledge from multiple source policies. The key insights of this paper are the two mechanisms proposed: a *Structured Policy Network*, which can improve the training and transfer performance by taking into account the morphological information, and a *adaptive policy transfer*, which can determine when and which source policies should be transferred to the target task. In summary, our contributions are as follows: 1) TURRET adopts an attention mechanism in the structured policy network to capture different relationships of neighboring nodes to learn a more semantic node representation, maintaining sufficient information during the aggregation process and leading to a common state embedding space for all tasks. 2) TURRET measures the distance of states in the unified embedding space to measure the similarity at each state from multiple cross-domain source policies. In this way, TURRET achieves adaptive and delicate transfer. 3) TURRET can be easily combined with existing DRL algorithms. since no additional optimization objectives are required in the training process. Experimental results show that TURRET significantly outperforms the state-of-the-art methods on continuous control tasks.

Preliminaries

Reinforcement Learning RL problems are typically formalized as Markov decision processes (MDPs). An MDP can be described as a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the sets of states and actions, respectively; $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is the transition probability distribution over states; $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}$ is the reward function which gives returns on the agent's performance; and γ is the discount factor for future rewards. A policy $\pi : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ is defined as a state-conditioned probability distribution over actions and the goal of the agent is to find an optimal policy π^* maximizing the expected discounted return $R = \sum_{i=t}^{T} \gamma^{i-t} r_i$.

Policy Gradient Algorithms Policy gradient methods are widely used to directly optimize the policy π parameterized by θ. One of the most effective policy gradient methods is Proximal Policy Optimization (PPO) (Schulman et al. 2017), which can avoid the large deviation of the results caused by the use of importance sampling. PPO attempts to learn a new policy π_θ and make sure that the difference between π_θ and the rollout policy $\pi_{\theta_{\text{old}}}$ is small, which is achieved by introducing a constraint:

$$L^{\theta}_{\text{PPO}} = -\mathbb{E}_\tau \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{A}_t \right) \right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the ratio of the action probabilities under the rollout policy and current policy, and $\hat{A}_t$ is the estimated advantage. The value network V_ψ is updated with temporal difference learning: $L^{\psi}_{\text{PPO}} = -\mathbb{E}_\tau[(V_\psi(s_t) - V_t^{\text{targ}})^2]$. The overall PPO minimization objective is:

$$L_{\text{PPO}}(\theta, \psi) = L^{\theta}_{\text{PPO}} + L^{\psi}_{\text{PPO}}$$

Cross-Domain Transfer In cross-domain transfer, the state-action spaces of different tasks are different, i.e., $\mathcal{M}_{\text{S}} = \langle \mathcal{S}_{\text{S}}, \mathcal{A}_{\text{S}}, \mathcal{R}_{\text{S}}, \mathcal{T}_{\text{S}}, \gamma_{\text{S}} \rangle$ and $\mathcal{M}_{\text{T}} = \langle \mathcal{S}_{\text{T}}, \mathcal{A}_{\text{T}}, \mathcal{R}_{\text{T}}, \mathcal{T}_{\text{T}}, \gamma_{\text{T}} \rangle$. Formally, the problem of multi-source cross-domain TL is defined as follows: it includes a series of source MDPs $\Pi_{\mathcal{M}} = \{\mathcal{M}_1, \mathcal{M}_2, \cdots, \mathcal{M}_n\}$, where $\mathcal{M}_i$ represents the i-th source MDP, and a target MDP $\mathcal{M}_{\text{T}}$. The MDPs share some high-level commonalities (e.g., reptile robots may have qualitatively similar gaits). The goal is to accelerate the learning process on the target task by selectively transferring the most beneficial knowledge from $\Pi_{\mathcal{M}}$.

Transferability Measurement Some works explicitly calculate the similarity between MDPs, which is used to measure the transferability between different tasks (Hu, Gao, and An 2015b,a). However, these methods usually require a known world model and have a high computational complexity, which cannot be applied to more complex continuous control tasks. In contrast, other works implicitly use the average performance on the target task to measure the transferability of each source policy (Fernández and Veloso 2006; Li and Zhang 2018; You et al. 2022). However, using average performance cannot handle the situation where only a part of the information in different source policies is useful. Previous works have supported this point that each source task should be more beneficial in a certain part of the state space (Yang et al. 2020a) or a single state (Rajendran et al. 2017). Nevertheless, all the above works are still constrained by the assumption of the same state-action space.

Graph Neural Networks (GNNs) Traditional methods that use MLP as the policy model cannot handle the cross-domain transfer problem because it can only accept the input and output with the same dimension across tasks. In contrast, under the constraints of different state-action spaces, GNNs have been a natural choice for modeling policies due to their capacity to handle graphs of varying sizes.

A graph is denoted as a tuple $\mathcal{G} = (V, E)$, where V is a set of nodes and edges $E = \{(u, v)|u, v \in V\}$. Each node and edge have the corresponding representation in a

labeled graph, and a GNN is a function that takes a labeled graph as input and outputs a graph with new labels but shares the same topology. A general framework of GNNs is the message-passing neural network (MPNN) (Gilmer et al. 2017), which treats the aggregation process as a K-step message-passing process. An MPNN framework consists of the message functions M^k and update functions U^k, which are used to compute the hidden state h_v^{k+1} and message m_v^{k+1} for each node v, where $k \in \{0, \ldots, K-1\}$. At each step k, the message function aggregates the hidden states and edge features e to compute messages, which are then passed into the update function to compute hidden states for the next step.

NerveNet, Snowflake, and CAT The ability of GNNs to process graphs of arbitrary sizes has been successfully applied to RL in continuous control tasks. One notable work is NerveNet (Wang et al. 2018), which models the morphology of the robot as a graph and uses a GNN following the basic framework of MPNN as the policy network. For example, a Centipede agent has four different node types: *root*, *torso*, *hip*, and *ankle*. The *torso* nodes share the same instance of the input function, and each *torso* sends the same message to the *righthip* and *lefthip*. However, in order to generalize to different agents with varied node and edge types, NerveNet merges all other nodes except *root* as *joint* in the specific implementation. Different nodes are usually correlated with varying degrees, but due to this operation and the characteristics of MPNN, NerveNet loses the ability to capture this correlation, resulting in information loss during the aggregation process. Furthermore, as the number of nodes increases, this phenomenon is exacerbated due to the multi-hop communication in GNNs, thus leading to diminished performance on tasks with large state-action spaces.

Recently, Blake et al. (2021) regard the above phenomenon as caused by the overfitting problem. Therefore, Snowflake freezes network parameters during training, thus facilitating training GNNs on larger graphs for locomotion control in RL. Unfortunately, Snowflake requires extra human effort to determine which parameters to freeze, which would be infeasible in complicated tasks.

More recently, You et al. (2022) firstly propose CAT to solve the multi-source cross-domain transfer problem, which is the same setting as in this paper. CAT learns a one-to-one mapping between each source task and the target task to extract useful knowledge from multiple source policy networks. However, CAT lacks generalization to more realistic scenarios when the number of source tasks changes. Moreover, the transferability measurement used in CAT is not delicate enough to handle the situation where each source policy performs better in only a part of the state space.

Methodology

This section first introduces the whole structure of TURRET and then describes each component of TURRET in detail.

Framework Overview

Figure 2 illustrates the overall framework of TURRET, which contains two main components: (a) ***structured policy network*** and (b) ***adaptive policy transfer***. TURRET facilitates efficient knowledge transfer not only across robots with high similarities in the number of joints and morphological information but also across robots with significantly distinct structures or numbers of joints.

Structured Policy Network As shown in the previous section, previous GNN-based methods are incapable of distinguishing different contributions of neighbor nodes to the central node, resulting in considerable information loss during the aggregation process, and undesired performance in tasks with large state-action spaces. The key insight of the structure policy network is to alleviate the information loss problem during the aggregation process. To combat this, we adopt an *attention mechanism* to learn a more semantic node representation, which is described in the following section. Then, we concatenate node representations and feed them into a readout network F_{read}, thus leading to a common state embedding space (see Figure 2 (a)). The subsequent transfer mechanism is described consequently.

Adaptive Policy Transfer The key insight of this part is that the transferability of each source policy could be reflected by the distance of embeddings in the unified embedding space, which is obtained by our proposed GNN-based structured policy network. Using the obtained distance to calculate the weighting factors of each source policy at each state (Figure 2 (b)), i.e., the *similarity metric*, we can adaptively and delicately extract knowledge from multiple source policies, which is described in detail consequently.

Structured Policy Network

In this section, to handle the mismatch of state spaces of all the tasks, we propose a structured policy network based on GNNs. Further, we assign different weighting factors to neighboring nodes to reduce information loss during the aggregation process. Given a set of N source tasks $\{\chi_1, \chi_2, \cdots, \chi_n\}$ that have arbitrary state-action spaces, the structured policy network is used to learn the policy on each source task, which contains four main models: input model F_{in}, propagation model P, readout model F_{read}, and output model F_{out}. We collectively refer to the first three models as representation model G.

Input Model: s_t is the agent observation vector obtained from the environment, which contains observations x_v of each node v corresponding to each joint (Wang et al. 2018). Then, the initial node representation h_v^0 at propagation step 0 is obtained by placing the node vector into an input network: $h_v^0 = F_{\text{in}}(x_v)$, where F_{in} is an MLP and we keep the fixed-size input for other node observations of different sizes by padding zeros to the vectors.

Propagation Model: As described before, the information loss phenomenon during the aggregation process results in diminished performance on tasks with large state-action spaces, which we'll also show in the empirical section. This phenomenon will be alleviated by considering the different contributions of different neighbor nodes to the central node. To capture different relative weights between nodes, we introduce the attention mechanism (Veličković et al. 2018) (we use multi-head attention in practice):

$$\alpha_{vu}^{k+1} = \text{Softmax}(\eth(\mathbf{a^T}[\mathbf{W^{k+1}}h_v^k || \mathbf{W^{k+1}}h_u^k])),$$

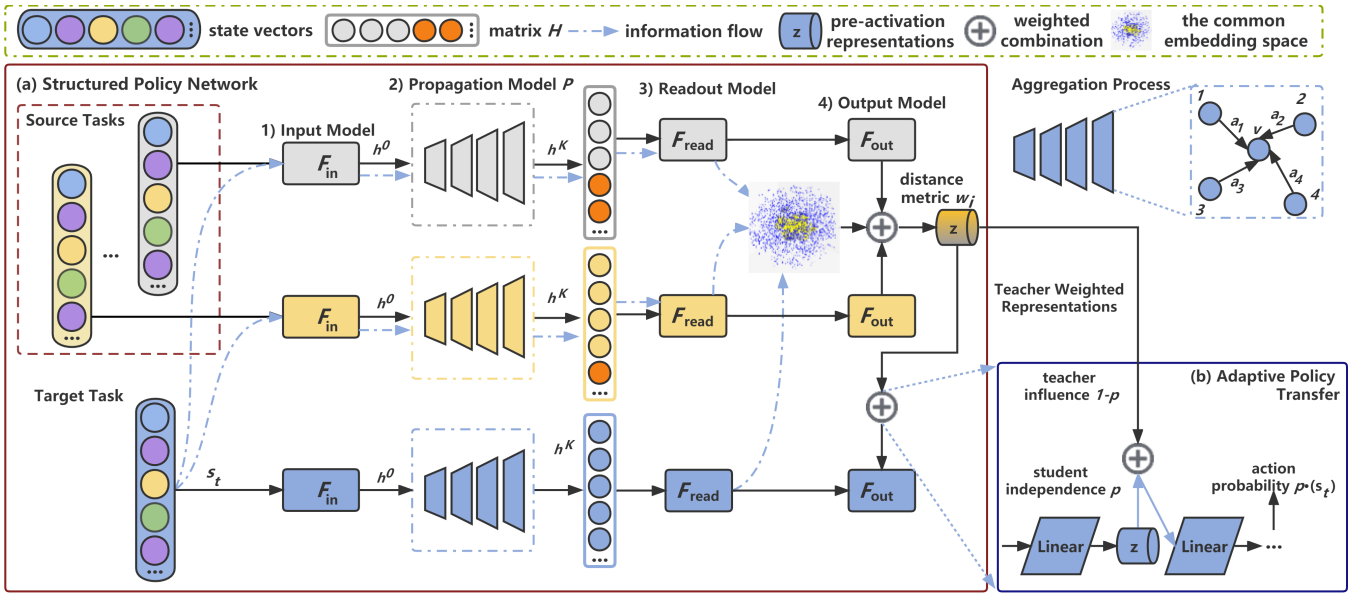


Figure 2: TURRET contains two main components. First, the structured policy network architecture is shown in the brown box, which handles the mismatch of state spaces of all the tasks. Second, the adaptive policy transfer architecture is shown in the blue call-out box, which adaptively and delicately extracts knowledge from multiple source policies. Each task has separate F_{in}, P, F_{read}, F_{out}, and the readout model F_{read} maps the states to a unified embedding space, which is used to measure distance and generate weighting factors.

where $\eth(\cdot)$ is the LeakyReLu activation function, $\mathbf{W^k}$ a weight matrix and $\mathbf{a}$ a vector of learnable parameters. The attention coefficient α_{vu}^k measures the relationship between the node v and its neighbor node u. Then the node update function is as follows:

$$h_v^{k+1} = \sigma\Big(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{k+1} \mathbf{W^{k+1}} h_u^k\Big).$$

The attention mechanism can not only generalize to different tasks but also capture the important relationships between nodes, which reduces information loss during the aggregation process and improves the training and transfer performance relative to other GNN-based methods on large dimensional control tasks.

Readout Model: Inspired by adaptive readout functions proposed by (Buterez et al. 2022), TURRET adopts set transformer readouts to learn an overall state representation. Specifically, once the propagation model obtains the final node representations, the node vectors are first collected into a matrix $H \in \mathbb{R}^{M \times D}$, where M is the maximal number of nodes in source and target tasks and D is the dimension of node representations. For graphs with less than M nodes, we set the padded values to zero to ensure that current states can be fed into the readout models of source tasks. Then the matrix H is fed into the set transformer readout function as input and state embeddings are generated using an attention-based encoder-decoder module (described in appendix):

$$S_{emb} = F_{\text{read}}(H) = \frac{1}{K}\sum_{k=1}^{K}[\text{DECODER}(\text{ENCODER}(H))]_k$$

where $[\cdot]_k$ refers to computation specific to head k. To this point, all the tasks obtain their state representations in a common space, which contains more structural information across tasks and adapts to varying types or numbers of tasks.

Output Model: Previous GNN-based methods predict the actions of each node with the final node representations as input, which only supports zero-shot transfer and is not conducive to the design of the transfer process. Instead, we take state representations into an output network and predict the action distribution for all nodes:

$$\mu_{v \in V} = F_{\text{out}}(S_{emb})$$

where $\mu_{v \in V}$ is the mean of the Gaussian distribution and the standard deviation is a trainable vector. We choose PPO (Schulman et al. 2017) to train all learnable parameters in an end-to-end manner.

Adaptive Policy Transfer

This section describes how to adaptively extract the most relevant knowledge from multiple source structured policies to accelerate the learning process for cross-domain transfer.

Instead of outputting the action of each joint separately, we combine node representations to learn a global state representation through the readout model, which can reflect semantic environmental information. To this end, the distance of states in the embedding space with sufficient semantic information can be seen as a suitable metric to reflect the transferability of each source policy at the current state. More specifically, we first obtain the state representation s_{emb} of the current state s_t through the representation model G in the target task. At the same time, s_t is fed into G of each source

task to obtain $\{s_{emb_1}, s_{emb_2}, \ldots, s_{emb_n}\}$, corresponding to $\{\chi_1, \chi_2, \cdots, \chi_n\}$. The distance between state embeddings in the common space is calculated to generate weights of each source policy as follows:

$$\omega_i = \text{Softmax}((||s_{emb} - s_{emb_i}||_2^2)^{-1}), i \in \{1, 2, \ldots, n\}$$

For example, the current state s_t in an octopod robot is fed into G_1 of a quadruped and G_2 of a hexapod robot, obtaining s_{emb_1}, s_{emb_2} and ω_1, ω_2 (see Figure 2(a)). Through the common embedding space, we can not only handle the mismatch of different state spaces but also achieve adaptive transfer at each state with sufficient interpretability. With the generated weights, TURRET adopts the transfer method of lateral connections between the source and target networks (Liu, Peng, and Schwing 2019; Wan, Gangwani, and Peng 2020; You et al. 2022). Specifically, we denote the pre-activation outputs of the j-th hidden layers of the i-th source policy network as $\{z_i^j, 1 \leq i \leq N, 1 \leq j \leq N_\pi\}$. The policy and value networks have the same number of hidden layers N_π in our setting and we only discuss policy networks here. Then, TURRET combines the representations z^j in the target network with those from source networks at each state following weighting factors ω_i as:

$$z_\pi^j = pz^j + (1-p)\sum_{i=1}^{N} \omega_i z_i^j$$

where p is an increasing factor over time to control the degree of independence of the target network (detailed in appendix). At the beginning of the training process, the agent needs more assistance from source policies. As the agent gains more knowledge, it relies more on itself, which is controlled by a higher value of p. In this way, TURRET adaptively extracts knowledge from multiple source policies and avoids negative transfer, achieving more efficient transfer.

Experiments

In this section, we present four types of transfer learning experiments that cover a wide range of environments to verify the effectiveness of TURRET from various perspectives. The first type is leveraging a set of small source task models to accelerate a larger target task, i.e., *size transfer*. The second type considers source and target tasks where robots have entirely different controls and constructions, i.e., *morphology transfer*. These first two types of experiments evaluate the effectiveness of multi-source cross-domain TL methods. The third type converts trajectories in the source and target tasks into a 3-dimensional space to perform qualitative and quantitative analysis, which shows how TURRET works. Fourth, we present a set of ablation studies to evaluate different components of TURRET. More experiments on the structured policy network training on large agents and the setting of more than two source tasks can be found in appendix. For convenience and computational complexity, we typically set the number of source tasks N to 2. We run 5 random seeds for each algorithm in an experiment, and each seed runs for 10 million environment interactions (i.e., timesteps). More details of network structures and parameter settings can be found in appendix.

Environments: We test our method on a set of continuous control tasks in MuJoCo (Todorov, Erez, and Tassa 2012). In addition to commonly used tasks such as **Walker**, **Ant**, and **Humanoid**, we also use a set of **Centipede-**n tasks, each of which has a robot with $n/2$ torso bodies and n legs (Wang et al. 2018). A more detailed description is in appendix.

Baselines: we consider the following six baselines:

- PPO (Schulman et al. 2017), a mainstream RL method that learns from scratch in the target task;
- CAT (You et al. 2022), which adaptively extracts knowledge from multiple cross-domain source policies;
- NerveNet (Wang et al. 2018), which models the morphology information into GNNs to represent the policy;
- Snowflake (Blake et al. 2021), which extends NerveNet to high-dimensional continuous control environments;
- NerveNet/Snowflake+fine-tune, which directly uses the old weights trained on source models for initialization and then continues to train in the target task.
- SWAT (Hong, Yoon, and Kim 2022), which adopts a transformer structure in multi-task training.

Size Transfer

In *size transfer*, we consider *Centipede-4* and *Centipede-6* as our source tasks. For the target tasks, we choose *Centipede-*$\{$*12, 16, 20*$\}$, which are very difficult to train from scratch.

Figure 3 (a)-(c) depicts the performance of TURRET and other baseline methods across three experimental scenarios. NerveNet exhibits inferior performance due to grappling with the intricacies of dealing with a large number of joints. While CAT demonstrates competitive results in the *Centipede-12* task, it progressively falls behind TURRET as the number of joints increases. This trend highlights the limitations of MLP-based policies in addressing structural disparities in robots with varying joint counts. Furthermore, NerveNet/Snowflake+fine-tune has a slight jumpstart and marginal performance improvement compared to their vanilla forms. Significantly, prior GNN-based approaches falter in transferring knowledge to agents with particularly large differences in the number of joints, which is exactly what our approach seeks to overcome. Similarly, SWAT's inefficient simultaneous aggregation of joint information hampers its performance, compounded by its lack of adaptive transfer capability. Conversely, TURRET consistently outperforms all baselines across all test environments. This can be attributed to TURRET's adeptness at considering diverse neighbor contributions, thus capturing node semantics and being able to measure state distances in a unified embedding space, enabling adaptive knowledge extraction from multiple source policies, which is also validated in the following experiments.

Morphology Transfer

In *morphology transfer*, we consider three combinations of TL experiments: {*Hopper & Centipede-4 → Walker2d*}, {*HalfCheetah & Ant → Centipede-8*} and {*HalfCheetah & Ant → Humanoid*}.

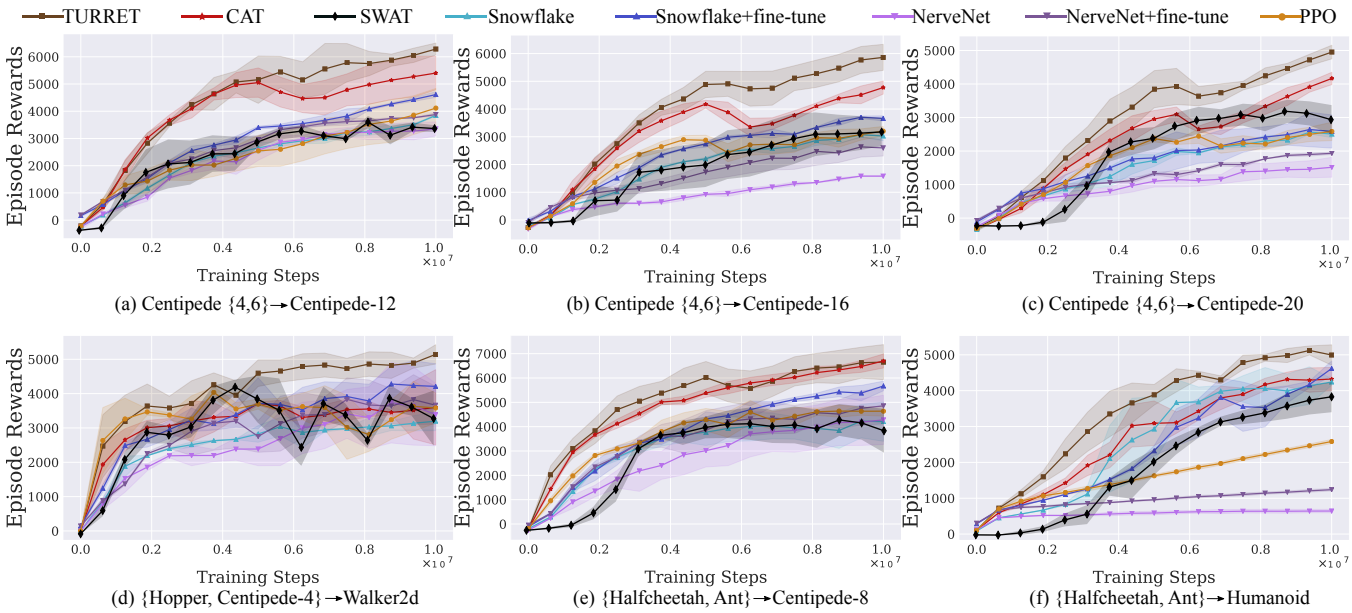


Figure 3: Performance of TURRET and other baselines on different sizes and morphologies of continuous control tasks. We plot the number of timesteps of environment interaction on the x-axis and the average episodic returns on the y-axis (the curves and shadow areas represent the mean and standard deviation over 5 trials, respectively).

We show the training curves of all the methods in Figure 3 (d)-(f). As the results show, CAT achieves comparable performance to TURRET in *Centipede-8*, because both source tasks have relatively similar morphologies to the target task and can provide enough knowledge to accelerate the target task learning. However, CAT performs worse than TURRET in the other two tasks because the two source tasks differ widely from the target task thus difficult to capture commonalities using MLP-based methods. Finally, TURRET significantly improves learning efficiency compared to all baselines in *Humanoid*, which has large state-action spaces and is hard to learn from scratch. All the results indicate that TURRET facilitates effective cross-domain transfer across tasks with completely different morphologies, matching or outperforming all baselines tested.

Visual Analysis

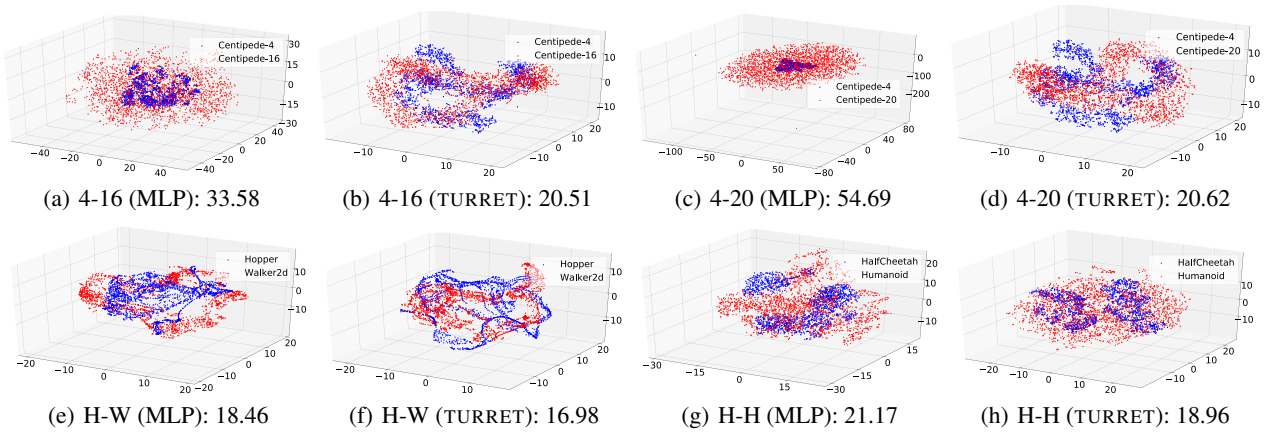
This section considers a post-hoc analysis to better understand why TURRET facilitates effective cross-domain transfer. We first collect trajectories from policies learned with PPO and TURRET in different tasks. Second, we project the source and target trajectories into a 3-dimensional space using *t-SNE*. Third, we calculate the Euclidean distance $\mathcal{D}$ between the corresponding states of two trajectories, which can reflect the similarity between two tasks, i.e., a lower value of $\mathcal{D}$ may mean a more helpful transfer as the projected source and target policies are more similar.

Figure 4 visualizes this analysis of *size transfer* and *morphology transfer*. From Figures 4 (a) and (c), we can see that the source and target policies learned by MLP-based PPO differ significantly when projected into 3 dimensions, which may make transfer difficult. In contrast, Figures 4 (b) and (d) show the difference between source and target policies learned using TURRET, where there is a higher overlap and distance reduction, relative to the PPO trajectories. This phenomenon indicates that TURRET narrows the distance between states with similar semantics in the embedding space, thus facilitating effective transfer. Similarly, the trajectories in the *morphology transfer* setting (i.e., Figures 4 (e) and (g) vs. (f) and (h)) also show that our structured source policy covers more of the target task policy than the MLP-based source policy. We can see that the trajectories of our structured policies are more similar in the project space in all examples, indicating that TURRET produces structured policies that capture semantic commonalities of two tasks and learns representations more useful than PPO.

Ablation Studies

In this section, we analyze the contribution of different parts of TURRET to better verify the effectiveness of our transfer method. We remove the attention mechanism in TURRET to verify it is useful to aggregate information by adopting different weighting factors to neighbor nodes without information loss on tasks with large state-action spaces. We also replace our similarity metric in TURRET with the average performance of each source policy on the target task as a weighting measurement, which is used to verify the effectiveness of our adaptive transfer method. The ablation studies are designed as follows under experiments on *Centipede-*$\{4,6\} \rightarrow 16$:

- TURRET *w/o attention*: Using MPNN without the attention mechanism during the aggregation process.



(a) 4-16 (MLP): 33.58 (b) 4-16 (TURRET): 20.51 (c) 4-20 (MLP): 54.69 (d) 4-20 (TURRET): 20.62

(e) H-W (MLP): 18.46 (f) H-W (TURRET): 16.98 (g) H-H (MLP): 21.17 (h) H-H (TURRET): 18.96

Figure 4: Our visual analysis results on MuJoCo: *Centipede-4* $\rightarrow$ *Centipede-*$\{$*16,20*$\}$, *Hopper* $\rightarrow$ *Walker2d*, and *HalfCheetah* $\rightarrow$ *Humanoid*. The blue and red dots represent the optimal trajectories on the source and target tasks, respectively. For example, "H-H (TURRET) :18.96" represents the distribution of trajectories sampled by our structured policy in the source task *HalfCheetah* and that in the target task *Humanoid*, and the average Euclidean distance $\mathcal{D}$ between two trajectories is 18.96.

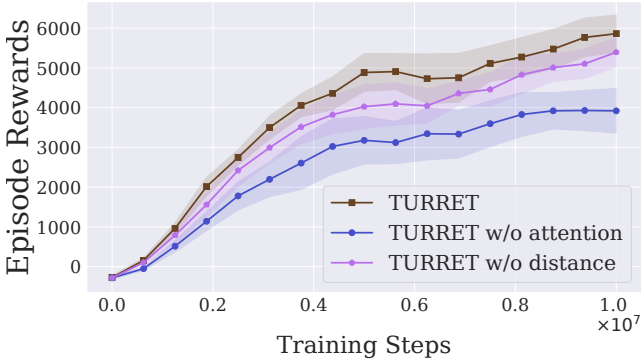


Figure 5: Ablation studies in *Centipede-*$\{$*4,6*$\}$ $\rightarrow$ *16* on the contribution of attention mechanism and similarity metric.

- TURRET *w/o distance*: Using the average performance as weighting factors to extract knowledge.

Figure 5 shows the results of our ablation studies. As we can see, TURRET has a significant performance improvement compared to TURRET *w/o attention*, which confirms the effectiveness of our attention mechanism. Besides, TURRET *w/o distance* performs worse than TURRET. This indicates that using the average performance as a weighting factor cannot handle the situation where only a part of the information in different source policies is useful. Therefore, a delicate transferability metric should be facilitated to improve the transfer performance. All the above results confirm that TURRET achieves more efficient transfer than previous cross-domain transfer methods.

Discussion

This paper focuses on effectively capturing the commonalities in different state-action spaces to discover a feature space that can enhance transferability. This pursuit, however, reveals two intrinsic limitations, which can be alleviated by incorporating additional techniques in TURRET in future work. First, this paper assumes that different tasks have a similar goal, and thus task similarity could be reflected in state similarity. In some cases, different tasks have different goals, e.g., opening the door or pushing the button. In this situation, TURRET could integrate the state similarity and the performance of each source policy to determine which source policy is more suitable. Second, this paper assumes a very clean state composition, where each feature represents the joint of a robot. In practice, states may contain noisy or irrelevant information. Representation learning could be combined with TURRET to discard irrelevant features via uncertainty measurement or adversarial learning to solve this problem.

Conclusion and Future Work

In this work, we propose TURRET, a GNN-based framework to achieve adaptive multi-source cross-domain TL. TURRET contains two main components: a meticulously structured policy network and adaptive knowledge transfer. By adopting the attention mechanism, we capture different relative weights between nodes, enabling the acquisition of a unified state embedding space through set transformer readouts. Based on the similarity metric, TURRET can adaptively and delicately extract knowledge at the state level. We also perform a visual analysis to verify the effectiveness of capturing morphological commonalities across tasks. In this paper, we adopt the attention mechanism in GNNs. Future work could consider designing a specific message-passing mechanism that can fully reflect the robot's morphology information. Another direction is to apply TURRET to more complicated scenarios by employing a more comprehensive similarity measurement to handle the goal mismatch in different tasks. Furthermore, existing methods always require structured information about the robot in advance when dealing with node vectors; thus, how to extract this information automatically is worth further study.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant Nos. 92370132, 62106172) and the National Key R&D Program of China (Grant No. 2022ZD0116402). Part of this work has taken place in the Intelligent Robot Learning (IRL) Lab at the University of Alberta, which is supported in part by research grants from the Alberta Machine Intelligence Institute (Amii); a Canada CIFAR AI Chair, Amii; Compute Canada; Huawei; Mitacs; and NSERC.

References

Blake, C.; Kurin, V.; Igl, M.; and Whiteson, S. 2021. Snowflake: Scaling GNNs to high-dimensional continuous control via parameter freezing. In *Advances in Neural Information Processing Systems 34*, 23983–23992.

Buterez, D.; Janet, J. P.; Kiddle, S. J.; Oglic, D.; and Liò, P. 2022. Graph Neural Networks with Adaptive Readouts. In *Advances in Neural Information Processing Systems*.

Chen, Y.; Chen, Y.; Yang, Y.; Li, Y.; Yin, J.; and Fan, C. 2019. Learning Action-Transferable Policy with Action Embedding. *CoRR*, abs/1909.02291.

Fernández, F.; and Veloso, M. M. 2006. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, 720–727. ACM.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural Message Passing for Quantum Chemistry. In Precup, D.; and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 1263–1272. PMLR.

Gupta, A.; Devin, C.; Liu, Y.; Abbeel, P.; and Levine, S. 2017. Learning Invariant Feature Spaces to Transfer Skills with Reinforcement Learning. In *Proceedings of the 5th International Conference on Learning Representations*.

Hong, S.; Yoon, D.; and Kim, K. 2022. Structure-Aware Transformer Policy for Inhomogeneous Multi-Task Reinforcement Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*.

Hu, Y.; Gao, Y.; and An, B. 2015a. Accelerating Multiagent Reinforcement Learning by Equilibrium Transfer. *IEEE Trans. Cybern.*, 45(7): 1289–1302.

Hu, Y.; Gao, Y.; and An, B. 2015b. Learning in Multi-agent Systems with Sparse Interactions by Knowledge Transfer and Game Abstraction. In Weiss, G.; Yolum, P.; Bordini, R. H.; and Elkind, E., eds., *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 753–761. ACM.

Li, S.; and Zhang, C. 2018. An Optimal Online Method of Selecting Source Policies for Reinforcement Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3562–3570. AAAI Press.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *Proceedings of the 4th International Conference on Learning Representations*.

Liu, I.; Peng, J.; and Schwing, A. G. 2019. Knowledge Flow: Improve Upon Your Teachers. In *Proceedings of the 7th International Conference on Learning Representations*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nat.*, 518(7540): 529–533.

Parisotto, E.; Ba, L. J.; and Salakhutdinov, R. 2016. Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning. In *Proceedings of the 4th International Conference on Learning Representations*.

Rajendran, J.; Lakshminarayanan, A. S.; Khapra, M. M.; Prasanna, P.; and Ravindran, B. 2017. Attend, Adapt and Transfer: Attentive Deep Architecture for Adaptive Transfer from multiple sources in the same domain. In *Proceedings of the 5th International Conference on Learning Representations*.

Rusu, A. A.; Colmenarejo, S. G.; Gülçehre, Ç.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2016. Policy Distillation. In *Proceedings of the 4th International Conference on Learning Representations*.

Schmitt, S.; Hudson, J. J.; Zídek, A.; Osindero, S.; Doersch, C.; Czarnecki, W. M.; Leibo, J. Z.; Küttler, H.; Zisserman, A.; Simonyan, K.; and Eslami, S. M. A. 2018. Kickstarting Deep Reinforcement Learning. *CoRR*, abs/1803.03835.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. *CoRR*, abs/1707.06347.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; Dieleman, S.; Grewe, D.; Nham, J.; Kalchbrenner, N.; Sutskever, I.; Lillicrap, T. P.; Leach, M.; Kavukcuoglu, K.; Graepel, T.; and Hassabis, D. 2016. Mastering the game of Go with deep neural networks and tree search. *Nat.*, 529(7587): 484–489.

Tao, Y.; Genc, S.; Chung, J.; Sun, T.; and Mallya, S. 2021. REPAINT: Knowledge Transfer in Deep Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 10141–10152. PMLR.

Taylor, M. E.; and Stone, P. 2009. Transfer Learning for Reinforcement Learning Domains: A Survey. *J. Mach. Learn. Res.*, 10: 1633–1685.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. IEEE.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wan, M.; Gangwani, T.; and Peng, J. 2020. Mutual Information Based Knowledge Transfer Under State-Action Dimension Mismatch. In *Proceedings of the Thirty-Sixth Conference on Uncertainty in Artificial Intelligence*, volume 124 of *Proceedings of Machine Learning Research*, 1218–1227. AUAI Press.

Wang, T.; Liao, R.; Ba, J.; and Fidler, S. 2018. NerveNet: Learning Structured Policy with Graph Neural Networks. In *Proceedings of the 6th International Conference on Learning Representations*.

Yang, T.; Hao, J.; Meng, Z.; Zhang, Z.; Hu, Y.; Chen, Y.; Fan, C.; Wang, W.; Liu, W.; Wang, Z.; and Peng, J. 2020a. Efficient Deep Reinforcement Learning via Adaptive Policy Transfer. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 3094–3100.

Yang, T.; Hao, J.; Meng, Z.; Zhang, Z.; Hu, Y.; Chen, Y.; Fan, C.; Wang, W.; Wang, Z.; and Peng, J. 2020b. Efficient Deep Reinforcement Learning through Policy Transfer. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, 2053–2055. International Foundation for Autonomous Agents and Multiagent Systems.

Yang, T.; Wang, W.; Tang, H.; Hao, J.; Meng, Z.; Mao, H.; Li, D.; Liu, W.; Chen, Y.; Hu, Y.; et al. 2021. An Efficient Transfer Learning Framework for Multiagent Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34.

You, H.; Yang, T.; Zheng, Y.; Hao, J.; and Taylor, M. E. 2022. Cross-domain Adaptive Transfer Reinforcement Learning Based on State-Action Correspondence. In *Proceedings of the Thirty-eighth Conference on Uncertainty in Artificial Intelligence*.

Zhang, Q.; Xiao, T.; Efros, A. A.; Pinto, L.; and Wang, X. 2021. Learning Cross-Domain Correspondence for Control with Dynamics Cycle-Consistency. In *Proceedings of the 9th International Conference on Learning Representations*.

Zhu, Z.; Lin, K.; and Zhou, J. 2020. Transfer Learning in Deep Reinforcement Learning: A Survey. *CoRR*, abs/2009.07888.