

WAGNN: A Weighted Aggregation Graph Neural Network for robot skill learning

Fengyi Zhang^{a,b}, Zhiyong Liu^{a,b,c,*}, Fangzhou Xiong^{a,b}, Jianhua Su^a, Hong Qiao^{a,b,c}

^a State Key Lab of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science, Beijing, 100190, China

^b School of Artificial Intelligence, University of Chinese Academy of Sciences (UCAS), Beijing, 100049, China

^c CAS Centre for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, 200031, China

ARTICLE INFO

Article history:

Available online 5 May 2020

Keywords:

Skill transfer learning
Serial structures
Robot skill learning
Graph Neural Network

ABSTRACT

Robotic skill learning suffers from the diversity and complexity of robotic tasks in continuous domains, making the learning of transferable skills one of the most challenging issues in this area, especially for the case where robots differ in terms of structure. Aiming at making the policy easier to be generalized or transferred, the graph neural networks (GNN) was previously employed to incorporate explicitly the robot structure into the policy network. In this paper, with the help of graph neural networks, we further investigate the problem of efficient learning transferable policies for robots with serial structure, which commonly appears in various robot bodies, such as robotic arms and the leg of centipede. Based on a kinematics analysis on the serial robotic structure, the policy network is improved by proposing a weighted information aggregation strategy. It is experimentally shown on different robotics structures that in a few-shot policy learning setting, the new aggregation strategy significantly improves the performance not only on the learning speed, but also on the control accuracy.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

Owing to the inability to summarize and update skill knowledge, a robot typically needs considerably large number of training samples, which may be impractical in rapidly-changing environments. On the other hand, a key signature of human intelligence is the ability to implement skill transfer learning. People have the ability to make “infinite use of finite means” [1], in which a small set of elements can be adequately composed in boundless ways. This indicates the principle of combinatorial generalization, that is, constructing new skills from known building blocks. Therefore, how to mimic the human learning process to realize transfer learning is crucial for robotic control.

In recent years transfer learning in particular has been considered to be an important direction in robotic skill learning [2]. Some previous works have been proposed to establish the direct mapping between state spaces and transfer skills between robots with different structures [3–5]. The transfer algorithms considered thus far have assumed that a hand-coded mapping between tasks was provided. Nevertheless, these algorithms require specific domain knowledge to form the hand-coded mapping, which makes these algorithms more sophisticated. A paramount

problem is that the policies learned by these methods lack clear structure information, making it difficult to utilize what was learned previously for a new robot with different structures [6].

Wang et al. [7] proposed learning structured policies by incorporating a prior on the structure via graph neural networks (GNN) [8], which can be viewed as a kind of making “infinite use of finite means”. Specifically, as the policy network of the agent, NerveNet [7] first propagates information through the structure of the agent and then outputs actions for different parts of the agent. To verify the transfer or generalization ability of NerveNet from one structure to another, with the goal of running as fast as possible along one direction, Wang et al. directly generalized the policy learned on a bilateral eight-leg centipede to a six-leg one, by just correspondingly dropping two leg modules in the NerveNet. It is easy to understand that when the agent loses its two bilateral legs, it does not affect the direction in which the agent runs. The reason is that the structure of the agent is symmetrical. Although the agent loses two legs, the remaining symmetrical structure still allow it to complete the task. NerveNet can directly achieve transfer learning for the customized task as most of the model weights are shared across the nodes. While, the reason the robots with serial structures are not transferable we consider is because the model weights are not able to be directly reused across the joints. Therefore, the model weights of some robots with serial structures are not transferable, in which case GNN model still needs to prove its potential in transfer learning.

* Corresponding author at: State Key Lab of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Science, Beijing, 100190, China.

E-mail address: zhiyong.liu@ia.ac.cn (Z. Liu).

Serial structures are the most common structures existed in robots [9]. Therefore, there is incentive in establishing techniques to learn transferable policy for robots with serial structures. Robots with serial structures classified into two categories. One type of robots with serial structure have the characteristics of serial in the whole structure (e.g. PR2), while the other type of robots with serial structure have the characteristics of serial in the local structure (e.g. Centipede).

In order to explore the performance of GNN on the tasks whose model weights are not transferable, we explore the problem on both two types of robots with serial structures above. To be specific, we consider the problem of transferring information across robots with different structures, including varying number of joints. Yet withal, the discrepancy in the structure of the robots and the goals of the tasks prevent us from directly reusing policy parameters learned on different robots for a new combination. Instead of throwing away experience learned from past tasks, this work aims at learning structured policies from its past experience to obtain new skills more quickly via GNN model. We further explore the potential of the GNN model in transfer learning tasks. On the other hand, given that the joint information of the serial robotic structure has different importances at different positions, the average aggregation is inapplicable. Hence, a kinematic analysis of robotic arms with serial structures is performed. Based on the physical characteristics of robots with serial structures, we further propose a novel aggregation method of GNN model.

The main contributions of this paper are summarized as follows.

- We investigate the problem of skill transfer learning for the robot with serial structures via graph neural networks.
- For both two types of robots with serial structures, we propose a Weightedly Aggregated Graph Neural Network (WAGNN) based policy network to accelerate the convergence process and improve the learning performance.

The rest of this paper is organized as follows. Preliminaries and a brief review of related work are presented in Section 2. Section 3 shows the weighted aggregation method to learn transferable policies on two simulated tasks. In Section 4, the main experimental result is presented and discussed. Finally, some concluding remarks are given in Section 5.

2. Related works

Recently, researches on transfer learning have been receiving more and more attention [2] because of its potential for reducing the burden of data collection for learning complex policies. Ammar et al. [4] designed a common feature space between the states of two tasks, and learn a mapping between states by using the common feature space. Later research by Ammar et al. [3] applies unsupervised manifold alignment to assign pairings between states for transfer learning. In Gupta et al. [5], the authors tried to improve transfer performance via learning invariant visual features. Efforts have also been made by reusing policy parameters between environments [10–12] to transfer policies. Nevertheless, most of these methods need more domain knowledge to determine how to form the invariant features, making these algorithms more complex. The proposed method is extremely different from these policy transfer methods, since our aim is not to directly transfer a policy, which is typically impossible in the presence of structural differences. This paper adopts GNN model to learn structured policies.

This paper improves on policy network by utilizing graph neural networks [8]. A graph data structure consists of a finite set of vertices (objects) and edges (relationships). It is worth noting that graphs have complex structure with rich potential

information [13]. Researches of graph with machine learning methods have been receiving more and more attention, given that graph structure data is ubiquitous in the real world. GNN was introduced in [8] as a generalization of recursive neural networks that can process graph structure data. Due to its good generalization performance and high interpretability, GNN has become a widely used graph analysis method in recent years. GNN [14–17] has been explored in a diverse range of problem domains, including supervised, semi-supervised, unsupervised, and reinforcement learning settings. GNN has been used to learn the dynamics of physical systems [18–20] and multi-agent systems [21–23]. These GNN models have also been used in both model-free [7] and model-based [24,25] continuous control. GNN models also have potential applications in model-free reinforcement learning [26–29], and for more classical approaches to planning [30].

In this paper, the work is based on the idea of representing a robot as a graph. Here, we define the graph structure of the robot as $G = (u, V, E)$. u is the global attribute of the graph. $V = \{v_i\}_{i=1:N_v}$ is the set of nodes (of cardinality N_v), where each v_i is the attribute of a node. $E = \{e_j, s_j, r_j\}_{j=1:N_e}$ is the set of edges (of cardinality N_e), where each e_j is the attribute of an edge, s_j is the index of the sender node and r_j is the index of the receiver node. In our tasks, the nodes correspond to the joints and the edges correspond to the bodies.

Battaglia et al. [31] presented the Graph Networks (GN) framework that unified and extended various graph neural networks. The GN framework defined a set of functions for relational reasoning on graphs and supported constructing complex structures from simple blocks. The main unit of the GN framework is the GN block which takes a graph as input and returns a graph as output. A GN block contains three “update” functions, ϕ , and three “aggregation” functions, ρ .

$$\begin{aligned} e'_k &= \phi^e(e_k, v_{s_k}, v_{r_k}, u) & \bar{e}'_i &= \rho^{e \rightarrow v}(E'_i) \\ v'_i &= \phi^v(\bar{e}'_i, v_i, u) & \bar{v}' &= \rho^{e \rightarrow u}(E') \\ u' &= \phi^u(\bar{v}', \bar{v}, u) & \bar{v}' &= \rho^{v \rightarrow u}(V') \end{aligned} \quad (1)$$

where $E'_i = \{e'_k, s_k, r_k\}_{r_k=i, k=1:N_e}$, $V' = \{v'_i\}_{i=1:N_v}$, and $E' = \cup_i E'_i = \{e'_k, s_k, r_k\}_{k=1:N_e}$.

As a graph, G , is the input value of a Graph Network, the computations propagate from the edge, to the node and the global level. Algorithm 1 shows the steps of computation for details.

Algorithm 1 Steps of computation in Graph Networks

Input: Graph, $G = (u, V, E)$
for each edge $\{e_j, s_j, r_j\}$ **do**
 Compute updated edge attributes $e'_k \leftarrow \phi^e(e_k, v_{s_k}, v_{r_k}, u)$
end for
for each node $\{n_i\}$ **do**
 Let $E'_i = \{e'_k, s_k, r_k\}_{r_k=i, k=1:N_e}$
 Aggregate edge attributes for each node $\bar{e}'_i \leftarrow \rho^{e \rightarrow v}(E'_i)$
 Compute updated node attributes $v'_i \leftarrow \phi^v(\bar{e}'_i, v_i, u)$
end for
Let $V' = \{v'_i\}_{i=1:N_v}$, $E' = \{e'_k, s_k, r_k\}_{k=1:N_e}$
 Aggregate edge and node attributes globally $\bar{v}' \leftarrow \rho^{e \rightarrow u}(E')$, $\bar{v}' \leftarrow \rho^{v \rightarrow u}(V')$
 Compute updated global attribute $u' \leftarrow \phi^u(\bar{v}', \bar{v}, u)$
Output: Graph, $G' = (u', V', E')$

Recent work by Wang et al. [7] modeled the structure of the reinforcement learning agents using NerveNet model. Relying on the fact that bodies of most robots have discrete graph structures, they aim to transfer policies between robots by learning

Algorithm 2 Steps of computation in NerveNet

Input: Graph, $G = (u, V, E)$
for each node $\{n_i\}$ **do**
 Let $V'_i = \{v'_i\}_{i=1:N_{v_i}}$
 Aggregate attributes from neighbors of node $\bar{m}'_i \leftarrow A(V'_i)$
 Compute updated node attributes $v'_i \leftarrow \phi^v(\bar{m}'_i, v_i)$
end for
Output: Graph, $G' = (u, V', E)$

structured policies. Nodes of the graph represent joints of the robots, and edges represent the physical dependencies between different joints. Particularly, they define the agent's policy using the graph neural networks (GNN) [8], which is a neural network that operates on graph structures. For the customized task in [7], the two legs the agent lost did not affect the direction in which it ran. Although the agent lost two legs, the remaining symmetrical structure still had the ability to complete the task. Therefore, GNN model is not able to show its potential in transfer performance in the customized environment. While for robots with serial structures, whose structures are asymmetric, GNN model cannot achieve satisfactory transfer performance, as witnessed by Section 4.1.1.

Algorithm 1 shows the steps of computation in NerveNet model in [7]. NerveNet receives state vector of joints from the environment and performs a few internal propagation steps in order to output the action to be taken by each joint. In NerveNet, the mean value of state information is used to do the aggregation, which means that function A in the algorithm takes the average function as follows,

$$m'_i = A(V'_i) = \frac{\sum_{j \in N_{v_i}} v'_j}{N_{v_i}} \quad (2)$$

where m'_i is the aggregated state vector which contains the information sent from the neighborhood of the joint, N_{v_i} is the number of joints adjacent to joint i , and v'_j is the state vector of joint which is the neighbor of joint i .

The way of doing the aggregation by the average function is that the information of nodes in the neighborhood is considered without discrimination. However, for robots with serial structures, the information of nodes in the neighborhood may be of quite different importance, as analyzed in the following section. Thus, in this paper we propose the weighted aggregation GNN (WAGNN) to make use of the importance of different nodes.

3. Proposed method

In this section, taking the robotic arm as a typical example, we first give a kinematic analysis on the serial robotic structure. Then based on the results we propose the WAGNN and give specific implementations on two types of robots, i.e., the PR2 arm and Centipede.

3.1. The kinematic analysis on serial structures

The forward kinematics of a robotic arm is to calculate the position and attitude of the end actuator relative to the base coordinate system according to the parameters of each joint. Fig. 1 shows the D-H parameter coordinate system of two adjacent coordinate systems. Where α_{i-1} represents the angle from \hat{Z}_{i-1} to \hat{Z}_i measured about \hat{X}_{i-1} ; a_{i-1} is the distance from \hat{Z}_{i-1} to \hat{Z}_i measured along the \hat{X}_{i-1} direction ($a_i > 0$); θ_i represents the angle from \hat{X}_{i-1} to \hat{X}_i measured about the \hat{Z}_i ; d_i is the distance from \hat{X}_{i-1} to \hat{X}_i measured along the \hat{Z}_i direction.

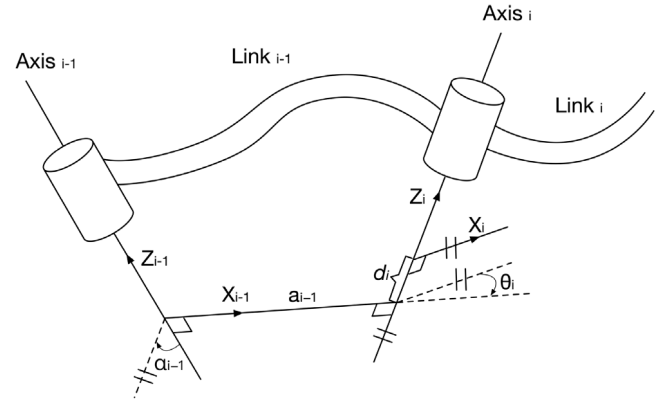


Fig. 1. Coordinate system of D-H parameter.

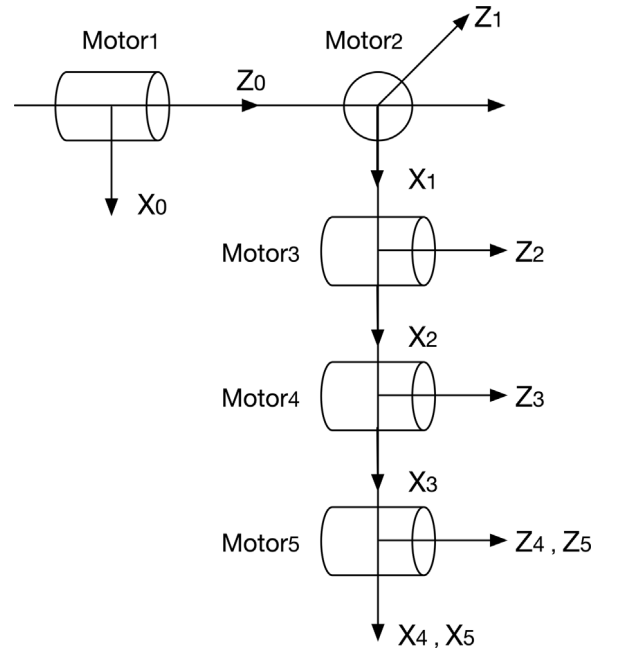


Fig. 2. Coordinate system of D-H parameter of arm.

From Fig. 1, the transformation matrix of the joint coordinate system can be derived, as shown in (2).

$${}^{i-1}_iT = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1})d_i \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1})d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

Eq. (3) is a general matrix representation of the D-H conversion. According to Fig. 2, the D-H parameters determined by the length and position of the connecting joint are shown in Table 1.

From Table 1 and Eq. (2), the forward kinematics formula can be expressed as:

$$T = {}^0_1T {}^1_2T {}^2_3T {}^3_4T {}^4_5T = \begin{bmatrix} R & P_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

In (4), R is a matrix representing the spatial attitude of the endpoint of the robotic arm, and P_x , P_y , and P_z represent the

position coordinates of the endpoint of the robotic arm.

$$P_x = a_3 \cos \theta_1 \cos \theta_2 \cos \theta_3 - a_3 \cos \theta_1 \sin \theta_2 \sin \theta_3 + a_2 \cos \theta_1 \cos \theta_2 + a_1 \cos \theta_1 \quad (5)$$

$$P_y = a_3 \sin \theta_2 \cos \theta_3 + a_3 \cos \theta_2 \sin \theta_3 + a_2 \sin \theta_2 + d_1 \quad (6)$$

$$P_z = -a_3 \sin \theta_1 \cos \theta_2 \cos \theta_3 + a_3 \sin \theta_1 \sin \theta_2 \sin \theta_3 - a_2 \sin \theta_1 \cos \theta_2 - a_1 \sin \theta_1 \quad (7)$$

In (6), the partial derivative of P_y is constructed as follows,

$$\frac{\partial P_y}{\partial \theta_2} = a_3 \cos \theta_2 \cos \theta_3 - a_3 \sin \theta_2 \sin \theta_3 + a_2 \cos \theta_2 \quad (8)$$

$$\frac{\partial P_y}{\partial \theta_3} = a_3 \cos \theta_2 \cos \theta_3 - a_3 \sin \theta_2 \sin \theta_3 \quad (9)$$

$$\frac{\partial P_y}{\partial \theta_2} - \frac{\partial P_y}{\partial \theta_3} = a_2 \cos \theta_2 \quad (10)$$

Given the actual physical structure, we have $\theta_2 \in [0, \frac{\pi}{2}]$. Then, we can obtain

$$\frac{\partial P_y}{\partial \theta_2} > \frac{\partial P_y}{\partial \theta_3} \quad (11)$$

$$P'_y - P_y = \Delta P_y = \frac{\partial P_y}{\partial \theta_2} \Delta \theta \quad (12)$$

$$P''_y - P_y = \Delta P'_y = \frac{\partial P_y}{\partial \theta_3} \Delta \theta \quad (13)$$

With the constraint of (11), we have,

$$\Delta P_y > \Delta P'_y \quad (14)$$

The results, as shown in (14), indicate that θ_3 cause a smaller change in the end position of the robotic arm when joint angle θ_2 and θ_3 change the same angle $\Delta \theta$. When the joint near the pedestal and the joint near the end are identically rotated by $\Delta \theta$, the former one will change the position of the end of the arm more.

3.2. Weighted aggregation graph neural network

Based on the results of kinematic analysis above, we then propose the following weighted aggregation function for graph neural networks,

$$m'_i = A(V'_i) = \sum_{j \in N_{v_i}} \rho_{ji} v'_j \quad (15)$$

where ρ_{ji} is given by

$$\rho_{ji} = \text{softmax}(f_{ji}) = \frac{\exp(f_{ji})}{\sum_{k \in N_{v_i}} \exp(f_{ki})}, \quad (16)$$

where f_{ji} is the serial number of the node in the graph. N_{v_i} is the set of nodes adjacent to node i . ρ_{ji} indicates the importance of node j 's features to node i .

WAGNN can be obtained by substituting ρ_{ji} in Eq. (16) into Eq. (15) yields.

Next, we will give two specific weighted aggregation methods for two tasks of robots with serial structures. For PR2 task, when the joint near the pedestal and the joint near the end are identically rotated by $\Delta \theta$, the latter will change the position of the end of the arm less, as can be seen in Fig. 3. For the task of getting the end of the PR2 arm to the precise point, when the endpoint of the robotic arm is near to the target, we do not need to change θ_2 a lot, just adjust θ_3 . Therefore, we consider the joint information at the end of the robotic arm to be more important. In Eq. (15), the closer the joint is to the end of the robotic arm, the larger the

weight of the node is. For PR2 task, if $u < v$, then the following inequality holds,

$$\rho_{uw} < \rho_{vw} \quad (17)$$

where both joint u and joint v are the neighbors of joint w .

To be specific, for the serial robotic task, the WAGNN with weight1 we consider is to say that in such case, f_{ji} can be obtained by the serial number of the node in the graph as follows,

$$f_{ji} = j \quad (18)$$

where j is the serial number of joint j , which is the neighbor of joint i .

For Centipede task, in the case of the joint near the center axis and the joint near the end being identically rotated by $\Delta \theta$, the former will change the position of the end of the robot more. In order to make Centipede run as fast as possible along a certain direction, we need to make the end of the leg move more. Accordingly, an important point to note here is that the joint information near the center axis of the robot to be more crucial. In Eq. (14), the closer the joint is to the center axis of the robot, the larger the weight of the node is. For Centipede task, if $u < v$, the following inequality holds,

$$\rho_{uw} > \rho_{vw} \quad (19)$$

where both joint u and joint v are the neighbors of joint w .

Concretely, for the robotic task that has the characteristics of serial in the local structure, the WAGNN with weight2 we consider is that f_{ji} has been simply formulated as:

$$f_{ji} = N - j \quad (20)$$

where j is the serial number of joint j , which is the neighbor of joint i . $N + 1$ is the total number of joints in the local serial structure (see Fig. 4).

4. Experimental illustrations

In this section, we first investigate the feasibility of GNN on a transfer learning task. Then, the weighted aggregation method is applied on two simulated robots with serial structures. To fully investigate the transfer performance of the Weightedly Aggregated Graph Neural Network (WAGNN), we build two weighted models for structure transfer learning.

4.1. PR2 task

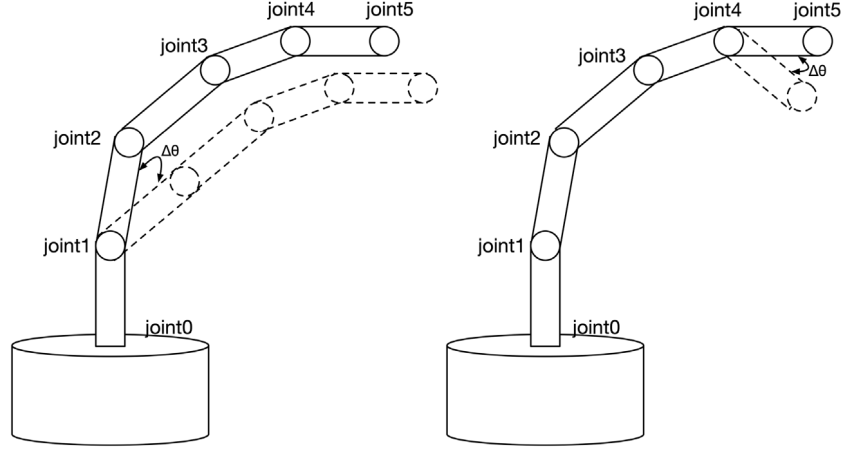
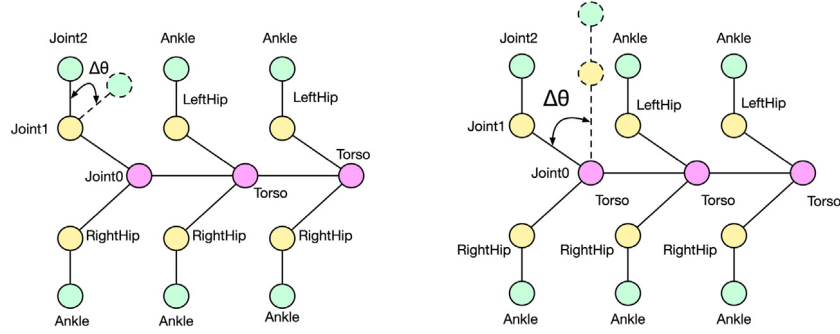
We run experiments on a simulated continuous control task from Gym, Brockman et al. [32], which is based on MuJoCo, Todorov et al. [33]. All of the experiments were implemented on Window 10, NVIDIA GeForce GTX 1060. Particularly, we use a robotic arm task: PR2 arm. The maximum number of training steps is set to be 1 million. In this paper, the proximal policy optimization(PPO) [34] is used to optimize the expected reward.

4.1.1. Transfer in a zero-shot setting

Two types of structural transfer learning tasks are investigated in this section. The first type is to train a model with a robotic arm of small size (small graph) and transfer the learned model to a robotic arm with a larger size. As increasing the size of the robotic arm, state and action space also increase which makes learning more difficult. In the second type of structural transfer task, we first learn a policy for the robotic arm and then transfer it to the robotic arm with a smaller size. Note that for both transfer tasks, none of the environmental factors change except the structure of the robotic arm.

Table 1
D-H parameter table.

Joint	Joint angle θ_i	Offset distance d_i	Twist angle α_{i-1}	Rod length a_{i-1}
1	θ_1	d_1	$-\pi/2$	0
2	θ_2	0	$\pi/2$	a_1
3	θ_3	0	0	a_2
4	θ_4	0	0	a_3
5	θ_5	0	0	0

**Fig. 3.** Different joints with the same angle of rotation for PR2.**Fig. 4.** Different joints with the same angle of rotation for Centipede.

Experimental settings. The environment in which the agent has a similar structure to a PR2 arm is used in this section. The goal of the agent is to get the end of the PR2 arm to the precise point. By linking copies of arms, we create agents with different lengths. Specifically, the shorter arm consists of seven joints and the longer one is made up of eight joints. For each time step, the total reward is the negative value of the distance from the end of the arm to the target point.

Results. In the PR2 environment, we first run experiments of NerveNet models on PR2 with seven joints and PR2 with eight joints to get the learned policies for transfer learning.

This work then explores the transfer performance of NerveNet applied in PR2 environment in a zero-shot setting where zero-shot means directly transferring the policy trained with one structure to the other without any fine-tuning. NerveNet model does not achieve satisfactory transfer performance on PR2 tasks. On one hand, NerveNet has an average reward value of -135.02 when the policy learned from the PR2 arm with 8 joints is transferred to the PR2 arm with 7 joints. On the other hand, NerveNet model has an average reward value of -266.13 when we transfer the policy learned from the PR2 arm with 7 joints to the PR2 arm with 8 joints. Both of them achieve poor transfer performance

since the average reward value of -60 is considered as solved for the PR2 task, which is reported in Table 2. Consequently, NerveNet models can be used to learn structured policies. Furthermore, NerveNet model achieves poor transfer performance for transfer learning tasks in a zero-shot setting.

4.1.2. Transfer in a few-shot setting

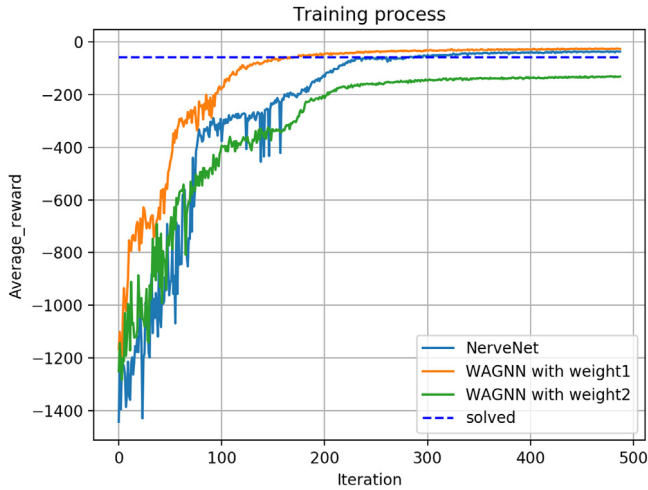
In this section, we show that GNN model has excellent transfer performance for PR2 arms in a few-shot learning setting. In the following experiments, it is experimentally shown that the weighted aggregation method has a better potential of transfer learning by incorporating physical structure prior into the network structure.

Experimental settings. To show the better transfer performance of the weighted aggregation method, we compare the WAGNN with the NerveNet [7] on simulated environment from the Gym. The weighted aggregation method is applied to a robot whose overall structure has a serial feature. More specifically, for the PR2 task whose overall structure has a serial feature, we use an equal number of time steps for each policy's update and calculate the information separately.

Table 2

Performance of zero-shot learning on PR2 task. For each task, we run policy for 10 episodes and record the average reward.

Task	7to8	8to7
Average reward	-135.02	-226.13
Average reward of solving PR2 task	-60	-60

**Fig. 5.** Results of WAGNN for training on the seven-joint PR2 environment.**Fig. 6.** Results of WAGNN for transferring from the seven-joint PR2 to the eight-joint PR2.

Results on training. By including the results of different weighted method, we further evaluate the performance of different weighted aggregation method and show the training curves in Fig. 5.

WAGNN with weight1 reduces about 26.61% time required to reach the level of reward which is considered as solved. At the same time, WAGNN with weight1 increases the control accuracy by about 8.8 percentage points, which are listed in Table 3. Especially for the robotic arm tasks that require precise control, the increase of 8.8 percentage points has important practical application significance. The improvement of WAGNN with weight1 is due to the fact that the joint information at the end of the robotic arm with serial structures to be more important, which not only accelerates the training process, but also improves the control accuracy.

However, as can be seen from Fig. 5, for PR2 task that requires precise control, the WAGNN with weight2 cannot even complete the task in the same time step setting. The results indicate that for the task of getting the end of the PR2 arm to the precise point, the joint information at the end of the robotic arm to be more important than the joint information at the base of the robotic arm. An important point to note here is that the weighted method is so crucial that choosing the appropriate weighted method can get superior learning performance, but instead choosing the unsuitable weighted method will result in worse performance.

Results on transferring. Experiments of all model are run on seven-joint PR2 to get the learned policies for transfer learning. We then fine-tune for eight-joints PR2 to show the transfer performance of WAGNN.

The transfer performance of WAGNN for PR2 is summarized in Fig. 6. From the figure, we can observe that by using the learned policies, WAGNN model has a superior initialization. In addition, when training from scratch, the policy network needs more onerous samples to accomplish the training process compared to transferring. WAGNN model significantly decreases the

number of episodes required to reach the level of reward which is considered as solved. WAGNN model can also achieve satisfactory transfer performance in a few-shot learning setting. From Fig. 6, we notice that WAGNN with weight1 has the ability to achieve comparable performance to NerveNet. Moreover, for WAGNN with weight1, the time required to reach the level of reward which is considered as solved is reduced by 34.98%. Therefore, WAGNN with weight1 can accelerate the training process without affecting the transfer performance.

Similarly, the same experimental phenomena have also been observed in transfer learning. The WAGNN with weight2 achieve poor transfer performance in the same time step setting, as shown in Fig. 7. Results on transfer learning, including an application to PR2 task, demonstrate that sensible weighted method is crucial for WAGNN to achieve satisfactory transfer performance.

4.2. Centipede task

In order to illustrate the transfer ability of the WAGNN model, we use a robotic task: Centipede. The maximum number of training steps is set to be 1 million for the Centipede task.

4.2.1. Transfer in a zero-shot setting

Similar in vein to Section 4.1.1, two types of structural transfer learning tasks are also investigated in this section. The first one is to train a policy for CentipedeSix and transfer the learned policy to CentipedeEight. Another type of structure transfer learning is to learn a policy for CentipedeEight and then transfer the learned policy to CentipedeSix. An important point to note here is that for both two types of transfer tasks, only the structure of the Centipede is changed in the environment.

Experimental settings. We use the second environment in which the agent whose local structure has a serial feature has a similar structure to a centipede, which is common in robotics [7]. The



Results of WAGNN on PR2 environment.

Model	Average reward	Solved time (Number of iteration)
NerveNet	-30.26	248
WAGNN	-27.57	182



Fig. 7. Results of WAGNN for training on the CentipedeSix environment.



Fig. 8. Results of WAGNN for transferring from the CentipedeSix to the CentipedeEight.

goal of the agent is to run as fast as possible along a certain direction in the MuJoCo environment. The agent is composed of three repetitive torso bodies where each one has two legs attached. Each leg of the agent has two joints, accordingly the agent has fifteen joints totally. For Centipede task, the total reward is the speed reward minus the energy cost and force feedback from the ground. CentipedeSix means that the agent has six legs in the structure of the robot. And CentipedeEight refers to the fact that the agent has eight legs.

Results. For the Centipede environment, we first run experiments of NerveNet models on CentipedeSix and CentipedeEight to get the learned policies for transfer learning.

This section then evaluate the transfer performance of NerveNet and WAGNN applied in Centipede environment in a zero-shot setting. The zero-shot transfer performance for Centipede is reported in Table 4. By examining the results videos, we notice that the average reward value of 2500 is considered as solved for Centipede task. Therefore, as can be seen from the table, both NerveNet model and WAGNN model have the ability to achieve satisfactory transfer performance on Centipede tasks in a zero-shot setting.

4.2.2. Transfer in a few-shot setting

In this section, we show that GNN model has excellent transfer performance for Centipede task in a few-shot learning setting. Furthermore, we evaluate the weighted aggregation method on transfer learning by incorporating physical structure prior into the network structure for Centipede task.

Experimental settings. The weighted aggregation method is utilized in a robot whose local structure has a serial feature. For NerveNet, information is aggregated and the mean value of information is applied to update the network. While the WAGNN model adopts the weighted value of information to update the network.

Results on training. The performance of WAGNN for CentipedeSix task is shown in Fig. 7. As can be seen from Table 5, the WAGNN with weight2 reduces about 28.05% time required to reach the level of reward which is considered as solved. In the meantime, it has the ability to improve performance by 6.94%. The improvement of the WAGNN with weight2 is due to the fact that for the Centipede task of running as fast as possible along a certain direction, the joint information at the center axis of the Centipede robot to be more important than the joint information at the end of the Centipede robot, which not only accelerates the training process, but also improves the learning performance.

Nevertheless, from Fig. 7, one can observe that the WAGNN with weight1 does not have the ability to achieve satisfactory performance. One possible reason is that the WAGNN with weight1 learns a policy that does not have the ability to make the Centipede to run as fast as possible. Therefore, it is crucial for specific task to adopt appropriate weighted method to do the aggregation.

Results on transferring. We run experiments of all models on CentipedeSix to get the learned policies for structural transfer learning. Then the fine-tune experiment for CentipedeEight is run and the training process is shown in Fig. 8. From the figure, we can see that by using the learned policy, GNN model has a sensible initialization. The results, as shown in this figure, indicate that the WAGNN with weight2 can achieve comparable performance to the NerveNet. Moreover, the WAGNN with weight2 reduces about 26.11% time required to reach the level of reward which is considered as solved. Accordingly, without reducing the transfer performance, the WAGNN with weight2 have the ability to speed up the convergence process.

Similar in vein to the results on training, note that the WAGNN with weight1 does not even have the ability to complete the transferring task, as shown in Fig. 8. The result indicates that the weight method of the aggregation is related to not only the structure of the robot, but the specific task of the robot. More importantly, the weight method of the aggregation has a

Table 4

Performance of zero-shot learning on Centipede task. For each task, we run policy for 10 episodes and record the average reward.

Task	6to8	8to6
NerveNet	2507.73	2543.87
WAGNN	2567.32	2583.45

Table 5

Results of WAGNN on Centipede environment.

Model	Average reward	Solved time (Number of iteration)
NerveNet	2508.04	303
WAGNN	2682.21	218

significant impact on both learning performance and transferring performance.

5. Conclusions

GNN model is naturally suitable for learning individual feature for each joint by incorporating a prior on the structure of the robot, making it an ideal springboard for tackling state decomposition problems. In this paper the WAGNN is proposed to improve the GNN based robot skill learning on serial structure. The weighted aggregation method in WAGNN learns structured policies by aggregating and propagating information among joints. Aggregation is done by the proposed aggregation method which considers the joints information of the serial structures are of different importance. Extensive experimental comparisons, on both robotic arm and Centipede, witnessed that the WAGNN achieved satisfactory transfer performance on robots with serial structures in a zero-shot setting, and also the policies learned by the WAGNN are significantly better than policies learned by traditional GNN models.

While the WAGNN models were most powerful for robots skill learning on serial structure, they typically needed longer training times than traditional MLP models. A promising direction for future work is to explicitly explore how GNN model performs state decomposition and then use the feature vectors learned by GNN methods to reduce algorithm complexity and training time.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work is supported by National Key Research and Development Plan of China grant 2017YFB1300202, NSFC, China grants U1613213, 61375005, 61503383, 61210009, the Strategic Priority Research Program of Chinese Academy of Science under Grant XDB32050100, and Dongguan core technology research frontier project, China (2019622101001).

References

- [1] N. Chomsky, *Aspects of the Theory of Syntax*, Vol. 11, MIT Press, 2014.
- [2] M.E. Taylor, P. Stone, Transfer learning for reinforcement learning domains: A survey, *J. Mach. Learn. Res.* 10 (Jul) (2009) 1633–1685.
- [3] H.B. Ammar, E. Eaton, P. Ruvolo, M.E. Taylor, Unsupervised cross-domain transfer in policy gradient reinforcement learning via manifold alignment, in: *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.
- [4] H.B. Ammar, M.E. Taylor, Reinforcement learning transfer via common subspaces, in: *International Workshop on Adaptive and Learning Agents*, Springer, 2011, pp. 21–36.
- [5] A. Gupta, C. Devin, Y. Liu, P. Abbeel, S. Levine, Learning invariant feature spaces to transfer skills with reinforcement learning, *arXiv preprint arXiv:1703.02949*.
- [6] C. Devin, A. Gupta, T. Darrell, P. Abbeel, S. Levine, Learning modular neural network policies for multi-task and multi-robot transfer, in: *2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017*, pp. 2169–2176.
- [7] T. Wang, R. Liao, J. Ba, S. Fidler, Nervenet: Learning structured policy with graph neural networks.
- [8] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, *IEEE Trans. Neural Netw.* 20 (1) (2008) 61–80.
- [9] H.D. Taghirad, *Parallel Robots: Mechanics and Control*, CRC Press, 2013.
- [10] S. Daffry, J.A. Bagnell, M. Hebert, Learning transferable policies for monocular reactive mav control, in: *International Symposium on Experimental Robotics*, Springer, 2016, pp. 3–11.
- [11] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, *arXiv preprint arXiv:1606.04671*.
- [12] A. Braylan, M. Hollenbeck, E. Meyerson, R. Miikkulainen, Reuse of neural modules for general video game playing, in: *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [13] A.-L. Barabási, et al., *Network Science*, Cambridge University Press, 2016.
- [14] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, *arXiv preprint arXiv:1511.05493*.
- [15] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, Computational capabilities of graph neural networks, *IEEE Trans. Neural Netw.* 20 (1) (2008) 81–102.
- [16] F. Scarselli, S.L. Yong, M. Gori, M. Hagenbuchner, A.C. Tsoi, M. Maggini, Graph neural networks for ranking web pages, in: *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, 2005, pp. 666–672.
- [17] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, Vol. 2, IEEE, 2005, pp. 729–734.
- [18] M.B. Chang, T. Ullman, A. Torralba, J.B. Tenenbaum, A compositional object-based approach to learning physical dynamics, *arXiv preprint arXiv:1612.00341*.
- [19] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, *arXiv preprint arXiv:1806.01242*.
- [20] P. Battaglia, R. Pascanu, M. Lai, D.J. Rezende, et al., Interaction networks for learning about objects, relations and physics, in: *Advances in Neural Information Processing Systems*, 2016, pp. 4502–4510.
- [21] Y. Hoshen, Vain: Attentional multi-agent predictive modeling, in: *Advances in Neural Information Processing Systems*, 2017, pp. 2701–2711.
- [22] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, R. Zemel, Neural relational inference for interacting systems, *arXiv preprint arXiv:1802.04687*.
- [23] S. Sukhbaatar, R. Fergus, et al., Learning multiagent communication with backpropagation, in: *Advances in Neural Information Processing Systems*, 2016, pp. 2244–2252.
- [24] J.B. Hamrick, A.J. Ballard, R. Pascanu, O. Vinyals, N. Heess, P.W. Battaglia, Metacontrol for adaptive imagination-based optimization, *arXiv preprint arXiv:1705.02670*.
- [25] R. Pascanu, Y. Li, O. Vinyals, N. Heess, L. Buesing, S. Racanière, D. Reichert, T. Weber, D. Wierstra, P. Battaglia, Learning model-based planning from scratch, *arXiv preprint arXiv:1707.06170*.
- [26] J.B. Hamrick, K.R. Allen, V. Bapst, T. Zhu, K.R. McKee, J.B. Tenenbaum, P.W. Battaglia, Relational inductive bias for physical construction in humans and machines, *arXiv preprint arXiv:1806.01203*.
- [27] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, et al., Relational deep reinforcement learning, *arXiv preprint arXiv:1806.01830*.

- [28] Wei Zhang, Bairui Wang, Lin Ma, Wei Liu, Reconstruct and represent video contents for captioning via reinforcement learning, *IEEE Trans. Pattern Anal. Mach. Intell.* (2019).
- [29] Wei Zhang, Ke Song, Xuewen Rong, Yibin Li, Coarse-to-fine uav target tracking with deep reinforcement learning, *IEEE Trans. Autom. Sci. Eng* 16 (4) (2018) 1522–1530.
- [30] S. Toyer, F. Trevizan, S. Thiébaux, L. Xie, Action schema networks: Generalised policies with deep learning, in: *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [31] P.W. Battaglia, J.B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, *arXiv preprint arXiv:1806.01261*.
- [32] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, *arXiv preprint arXiv:1606.01540*.
- [33] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012*, pp. 5026–5033.
- [34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*.



Fengyi Zhang received the bachelor's degree in automation from Harbin Engineering University in 2017. He is now a Ph.D. student in Institute of Automation, Chinese Academy of Sciences, advised by Prof. Zhiyong Liu. His research focuses on machine learning and he is currently studying reinforcement learning and transfer learning.



Zhiyong Liu received the B.E. degree in electronic engineering from Tianjin University, Tianjin, China, in 1997, the M.E. degree in control engineering from the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China, in 2000, and the Ph.D. degree in computer science from the Chinese University of Hong Kong, Hong Kong, in 2003. He is a professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, CASIA, Beijing. His current research interests include machine learning, pattern recognition, computer vision

and bio-informatics.



Fangzhou Xiong received the bachelor's degree in automation from Sun Yat-sen University in 2015. He is now a Ph.D. student in Institute of Automation, Chinese Academy of Sciences, advised by Prof. Zhiyong Liu. His research interest is machine learning and he is currently studying reinforcement learning and multi-task learning.



Jianhua Su received the B.Eng. degree in electronic and information engineering from Beijing Jiaotong University, Beijing, China, in 1999, the M.Eng. degree in Electronic and information engineering from the Beijing Jiaotong University, Beijing, in 2004, and the Ph.D. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2009. He is currently an Assistant Professor with the Institute of Automation, Chinese Academy of Sciences. His background is in the fields of control theory, robotics, automation, and manufacturing. His current research interests include intelligent robot system and train control system.



Hong Qiao received the B.S. and M.S. degrees in Engineering both from Xi'an Jiaotong University, Xian, China, in 1986 and 1989, respectively, the M.Phil. degree from the University of Strathclyde, Strathclyde, U.K. in 1997, and the Ph.D. degree from De Montfort University, Leicester, U.K., in 1995. She held teaching and research positions with universities in the U.K. and Hong Kong, from 1990 to 2004. She is currently a '100-Talents Project' Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China. Her current research interests include robotic manipulation, robotic vision, bio-inspired intelligent robot, brain-like intelligence, etc.