

NerveNet: Learning Structured Policy with Graph Neural Networks

Wang, Tingwu, et al. ICLR2018

Present by Kibeom Kim

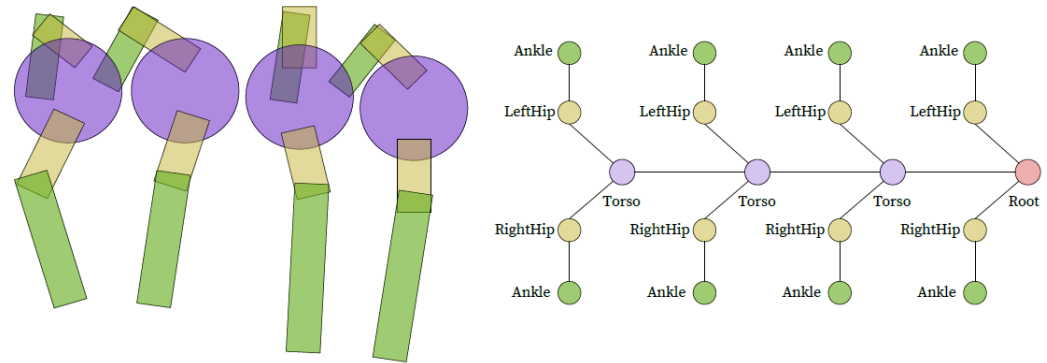
Interdisciplinary Program in Neuroscience
Seoul National University

NerveNet: Learning Structured Policy with GNN

- Typically use MLP to learn the agent's policy in RL
 - Concatenation of observations from the environment
 - This leads to longer training times, requiring more data
- To exploit the body structure of an agent, and physical dependencies that naturally exist in such agents.

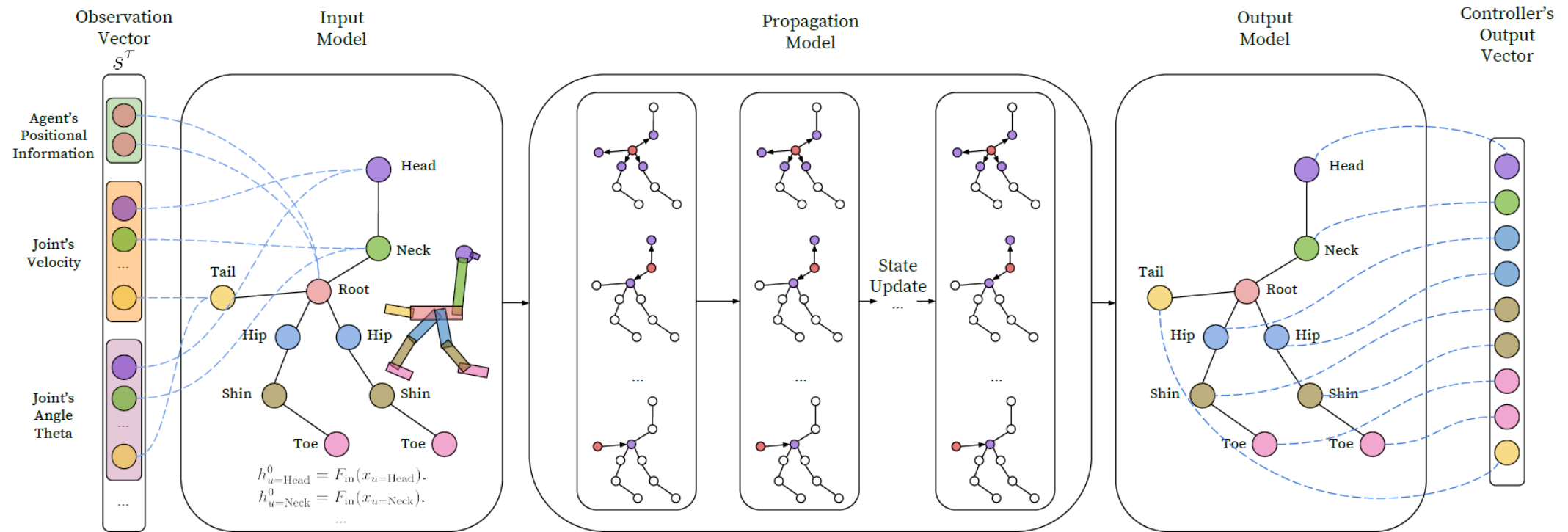
NerveNet: Learning Structured Policy with GNN

- Bodies of most robots and animals have a discrete **graph structure**.
- NerveNet first **propagates information over the structure** of the agent and **then predict actions for different parts** of the agent.
- Contribution:
 - Policy Network as GNN
 - Transfer learning ability



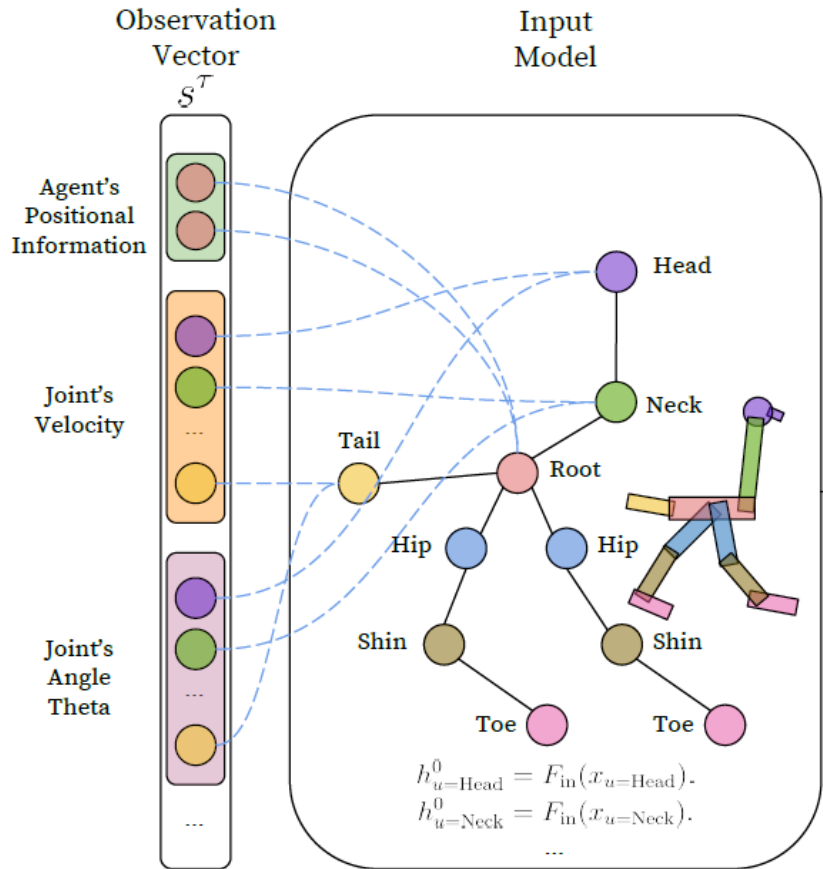
Visualization of the graph structure of *CentipedeEight*

NerveNet: Learning Structured Policy with GNN



Example of NerveNet

NerveNet: Learning Structured Policy with GNN



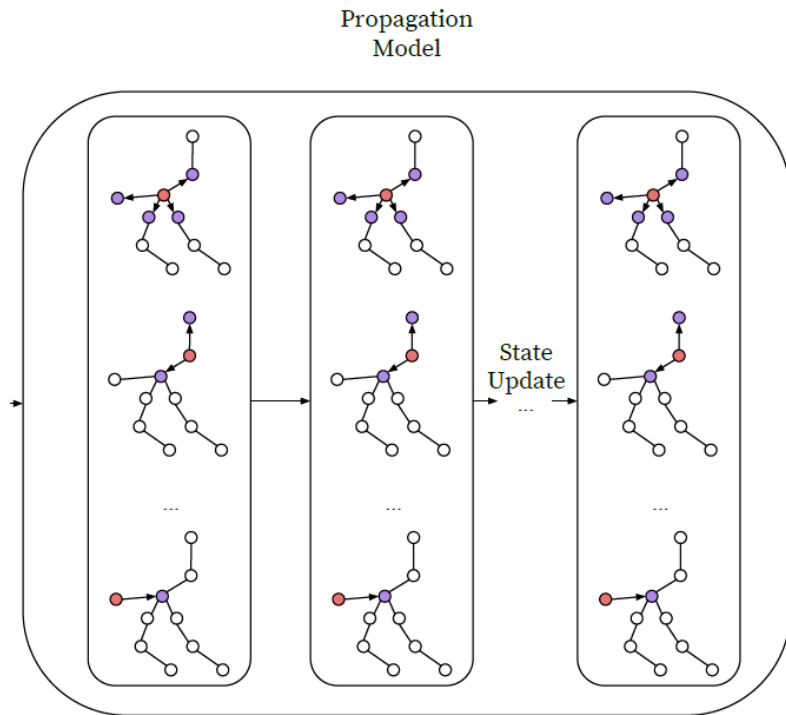
Input model of NerveNet

■ Input Model

- State vector $h_u^0 = F_{in}(x_u)$
 - x_u is the elements of observation vector corresponding to node u .
 - F_{in} is MLP.
 - Zero padding if observation size is different

NerveNet: Learning Structured Policy with GNN

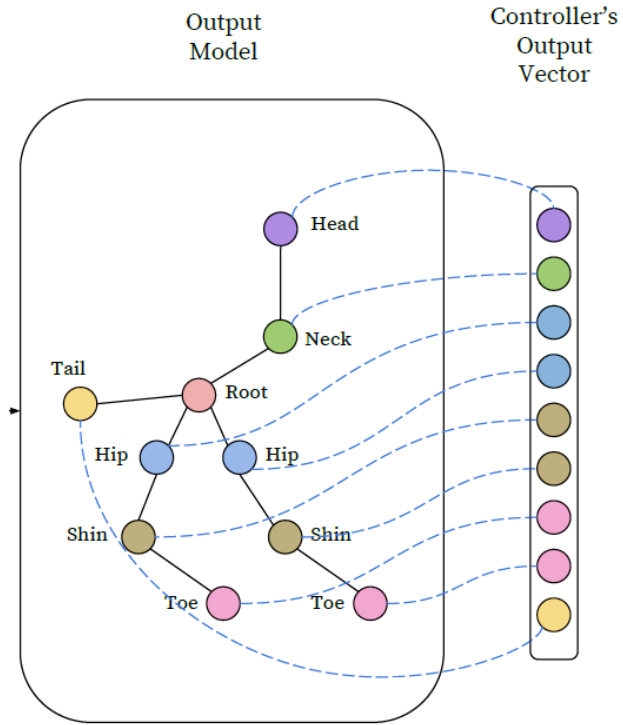
■ Propagation Model



Propagation model of NerveNet

- Message Computation $m_{(u,v)}^t = m_{c(u,v)}(h_u^t)$
 - $c(u,v)$ indicates that edges of the same edge type share the same instance of the message function. $m_{c(u,v)}$ is MLP.
- Message Aggregation $\bar{m}_u^t = A(\{h_v^t | v \in \mathcal{N}_{in}(u)\})$
 - $A()$ is the aggregation function which may be a summation, average or max-pooling function.
- States Update $h_u^{t+1} = U_{p_u}(h_u^t, \bar{m}_u^t)$
 - p_u is nodes of the same node type share the same instance of the update function. U_{p_u} is LSTM or GRU.

NerveNet: Learning Structured Policy with GNN



Output model of NerveNet

■ Output Model

- $\mu_{u \in \mathcal{O}} = O_{q_u}(h_u^T)$

- $\mu_{u \in \mathcal{O}}$ is the mean value for action applied on each actuator.

- O_{q_u} is MLP.

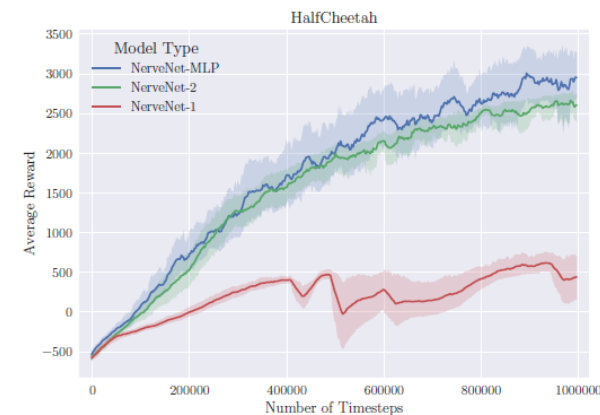
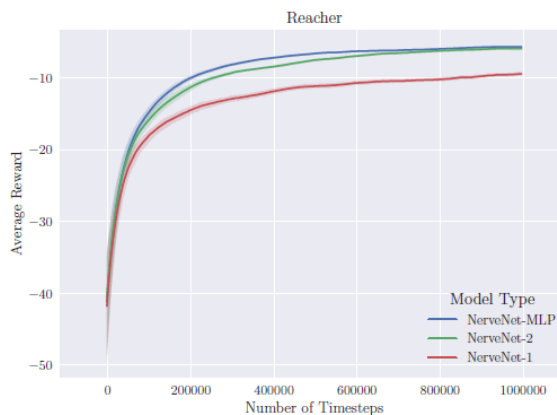
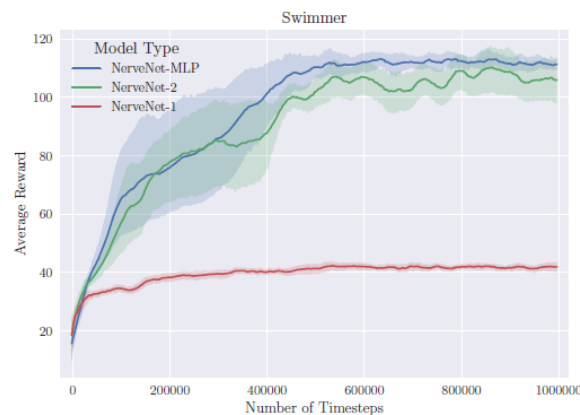
■ Policy as Gaussian distribution

- $\pi_{\theta}(a^T | s^T) = \prod_{u \in \mathcal{O}} \pi_{\theta,u}(a_u^T | s^T) = \prod_{u \in \mathcal{O}} \frac{1}{\sqrt{2\pi\sigma_u^2}} e^{(a_u^T - \mu_u)^2 / (2\sigma_u^2)}$

- $a^T \in \mathcal{A}$ is the output action, and σ_u is the variable standard deviation for each action

NerveNet: Learning Structured Policy with GNN

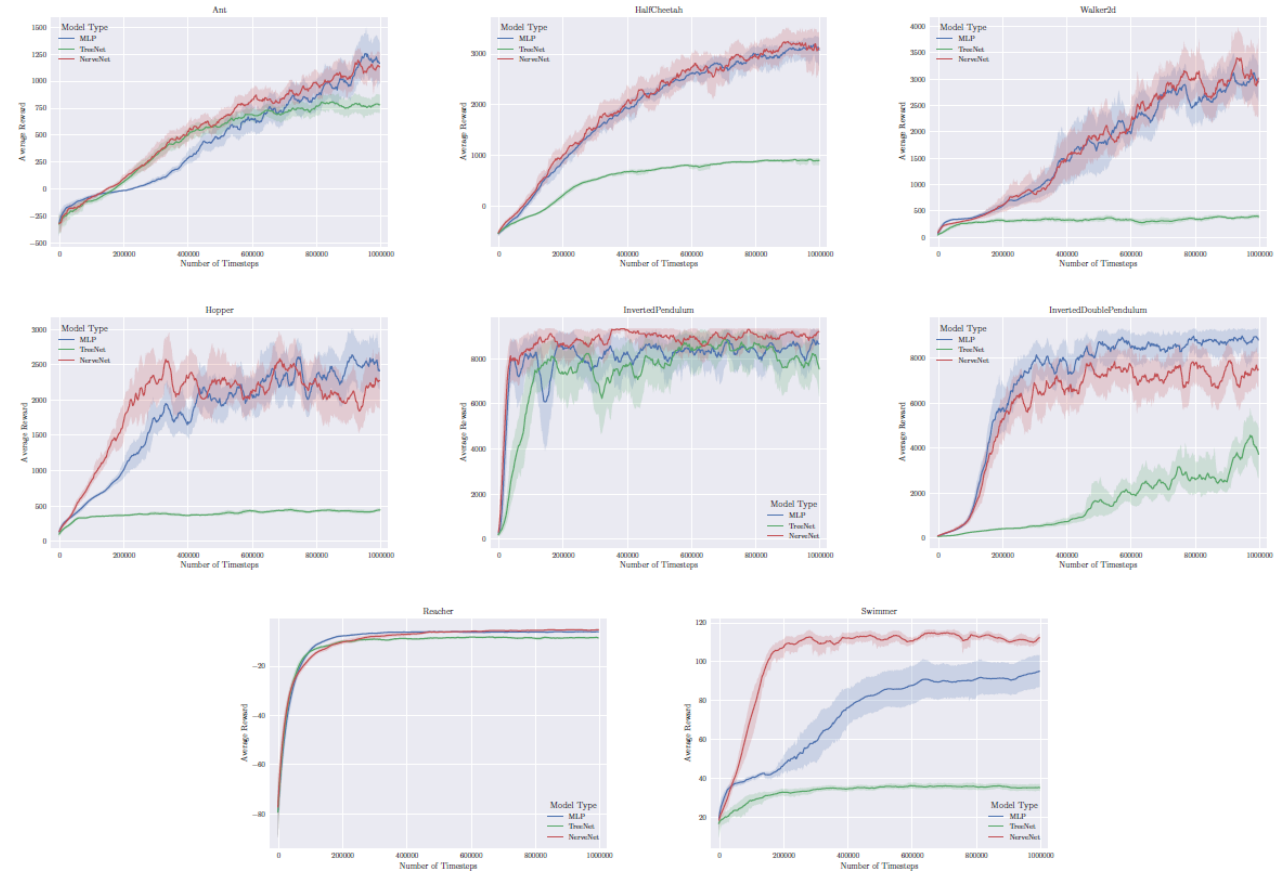
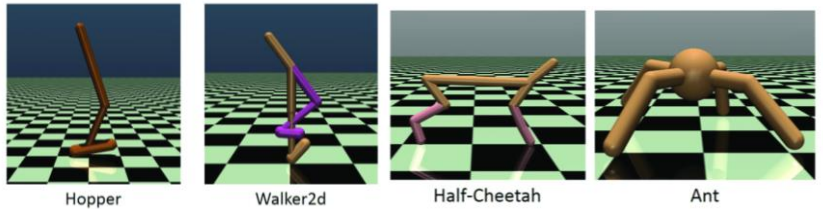
- Learning Algorithm
 - PPO (Proximal Policy Optimization)
- Value Network
 - NerveNet-MLP - GNN as the policy network, MLP as the value network
 - NerveNet-2 – GNN as the policy network, another GNN as the value network
 - NerveNet-1 – GNN as both policy and value network



Results of several variants of NerveNet

Experiments

- Comparison on standard benchmark
 - 8 continuous control benchmarks
 - MuJoCo simulation
- TreeNet
 - singly rooted depth-1 tree
 - Aggregates the information and then feeds the state vector to the output model



Results of **MLP**, **TreeNet** and **NerveNet** on 8 continuous control benchmarks from the gym

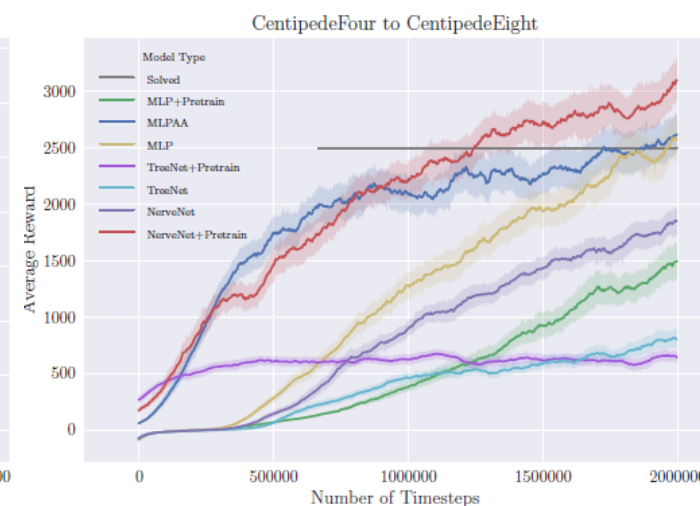
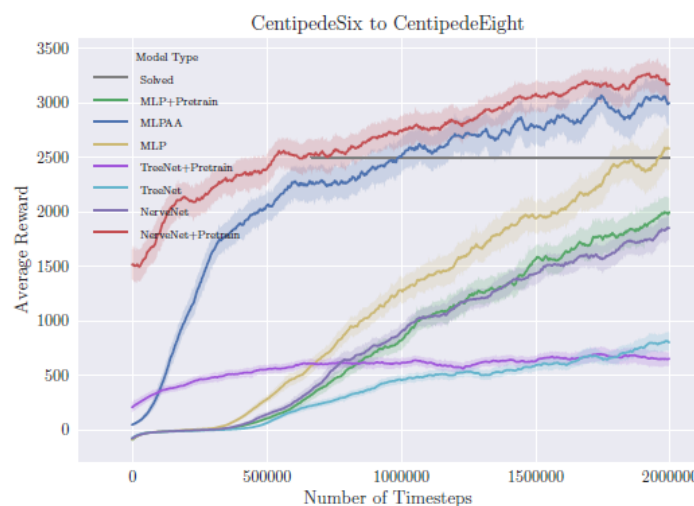
Experiments

- Structure Transfer Learning
 - Size transfer
 - To train a model with a **small size** agent and apply learned model to **larger size**
 - NerveNet+Pretrain
 - Large agent has repetitive structure
 - MLPAA
 - Learning large-agent model and setting the remaining weight to be zero
 - MLP, Solved
 - MLP+Pretrain, NerveNet

Tasks	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. NerveNet	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. NerveNet
4to6	-31.6 (32%)	109.4 (84%)	-126.7 (-2%)	-16.5 (38%)	139.6 (96%)	-75.6 (6%)	545.3 (92%)	-73.5 (6%)	-47.7 (10%)	577.3 (96%)
4to8	-45.8 (41%)	18.2 (76%)	-190.9 (-39%)	10.2 (72%)	44.3 (91%)	-91.0 (10%)	62.0 (67%)	-80.8 (14%)	-84.8 (13%)	146.9 (98%)
4to10	-44.3 (42%)	11.4 (73%)	-195.6 (-42%)	-101.5 (10%)	39.8 (88%)	-76.5 (16%)	21.0 (52%)	-89.9 (11%)	-70.0 (18%)	128.4 (92%)
4to12	-48.7 (39%)	9.9 (72%)	-215.8 (-53%)	17.6 (76%)	38.6 (88%)	-81.7 (14%)	13.1 (49%)	-76.9 (15%)	-63.2 (21%)	126.3 (91%)
4to14	-52.0 (37%)	8.0 (71%)	-233.0 (-62%)	-79.6 (22%)	39.0 (88%)	-89.0 (11%)	0.0 (44%)	-86.4 (12%)	-73.2 (17%)	125.7 (90%)
4toCp06	-17.0 (26%)	-5.1 (29%)	-113.9 (1%)	249.5 (94%)	47.6 (42%)	-77.3 (17%)	-22.5 (40%)	-79.9 (16%)	-72.2 (19%)	91.1 (87%)
4toCp08	-25.5 (52%)	5.1 (69%)	-132.2 (-6%)	33.3 (85%)	40.0 (88%)	-82.9 (17%)	-26.9 (44%)	-91.8 (13%)	-66.9 (25%)	80.1 (95%)
4toCp10	-28.2 (51%)	-12.8 (59%)	-133.7 (-7%)	28.2 (82%)	40.2 (88%)	-82.9 (17%)	-36.6 (39%)	-88.8 (14%)	-83.7 (17%)	86.9 (98%)
6to8	-45.8 (4%)	21.1 (7%)	-191.4 (-3%)	320.8 (24%)	1674.9 (99%)	-91.0 (0%)	87.8 (1%)	-88.6 (0%)	8.5 (1%)	10612.6 (99%)
6to10	-44.3 (7%)	-42.4 (7%)	-193.2 (-6%)	44.7 (15%)	940.5 (98%)	-76.5 (0%)	-17.0 (1%)	-81.8 (0%)	-77.4 (0%)	6343.6 (99%)
6to12	-49.1 (13%)	-14.4 (20%)	-227.0 (-20%)	19.5 (27%)	367.7 (95%)	-81.2 (1%)	-1.4 (4%)	-79.3 (1%)	-91.5 (1%)	2532.2 (99%)
6to14	-52.0 (17%)	-10.0 (28%)	-221.3 (-25%)	126.3 (63%)	247.8 (94%)	-89.0 (1%)	-3.9 (6%)	-83.2 (1%)	-51.5 (3%)	1749.7 (98%)
6to20	-72.4 (14%)	10.0 (39%)	-351.4 (-70%)	-233.6 (-34%)	198.5 (96%)	-95.2 (1%)	-4.9 (7%)	-105.4 (0%)	-102.3 (1%)	1447.4 (98%)
6to30	-67.1 (22%)	-28.5 (38%)	-263.3 (-59%)	17.6 (57%)	114.9 (97%)	-111.2 (0%)	-48.2 (7%)	-127.3 (0%)	-76.7 (4%)	867.5 (99%)
6to40	-72.7 (19%)	4.3 (51%)	-320.2 (-83%)	21.1 (58%)	97.8 (90%)	63.9 (16%)	140.5 (23%)	56.9 (15%)	101.8 (19%)	971.5 (98%)
6toCp08	-25.4 (14%)	36.5 (23%)	-141.9 (-3%)	398.9 (78%)	523.6 (97%)	-83.0 (1%)	138.3 (7%)	-90.7 (0%)	-63.9 (1%)	3117.3 (99%)
6toCp10	-28.2 (14%)	12.8 (21%)	-155.2 (-5%)	277.8 (63%)	504.0 (99%)	-82.9 (1%)	13.6 (3%)	-86.4 (0%)	-72.3 (1%)	3230.3 (99%)
6toCp12	-32.0 (22%)	12.1 (33%)	-177.5 (-14%)	-114.9 (1%)	255.9 (96%)	-87.5 (1%)	7.3 (7%)	-91.4 (1%)	-90.8 (1%)	1675.5 (99%)
6toCp14	-41.0 (21%)	11.8 (36%)	-187.5 (-18%)	-67.7 (14%)	224.8 (95%)	-96.5 (1%)	2.9 (7%)	-92.5 (1%)	-93.9 (1%)	1517.6 (99%)
	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. NerveNet	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. NerveNet

(a) Zero-shot average reward.

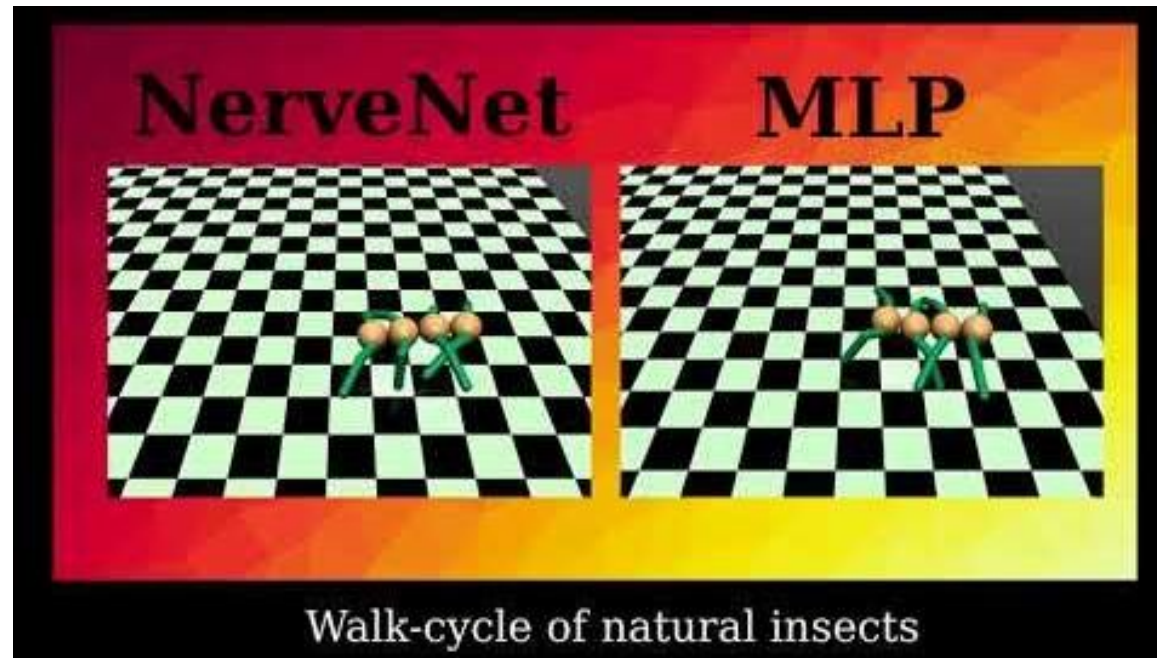
(b) Zero-shot average running-length.



Finetuning for Size transfer

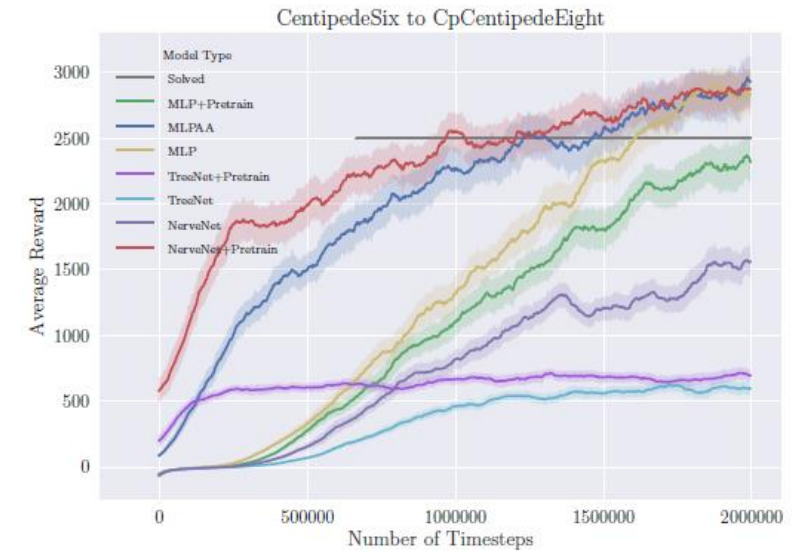
Experiments

- Size transfer
 - 6 to 8 legs
 - MLP+Pretrain
 - TreeNet
 - NerveNet

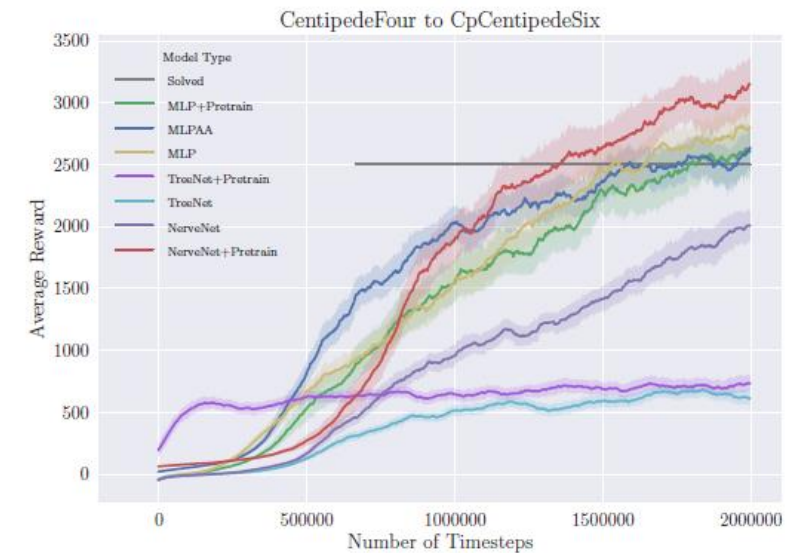


Experiments

- Structure Transfer Learning
 - Disability transfer
 - Learn a model for the original agent and then apply it some components disabled
 - two back legs are disabled
 - If overfits, disabling some components of the agent might bring catastrophic performance degradation.
 - NerveNet+Pretrain, MLPAA, MLP, Solved



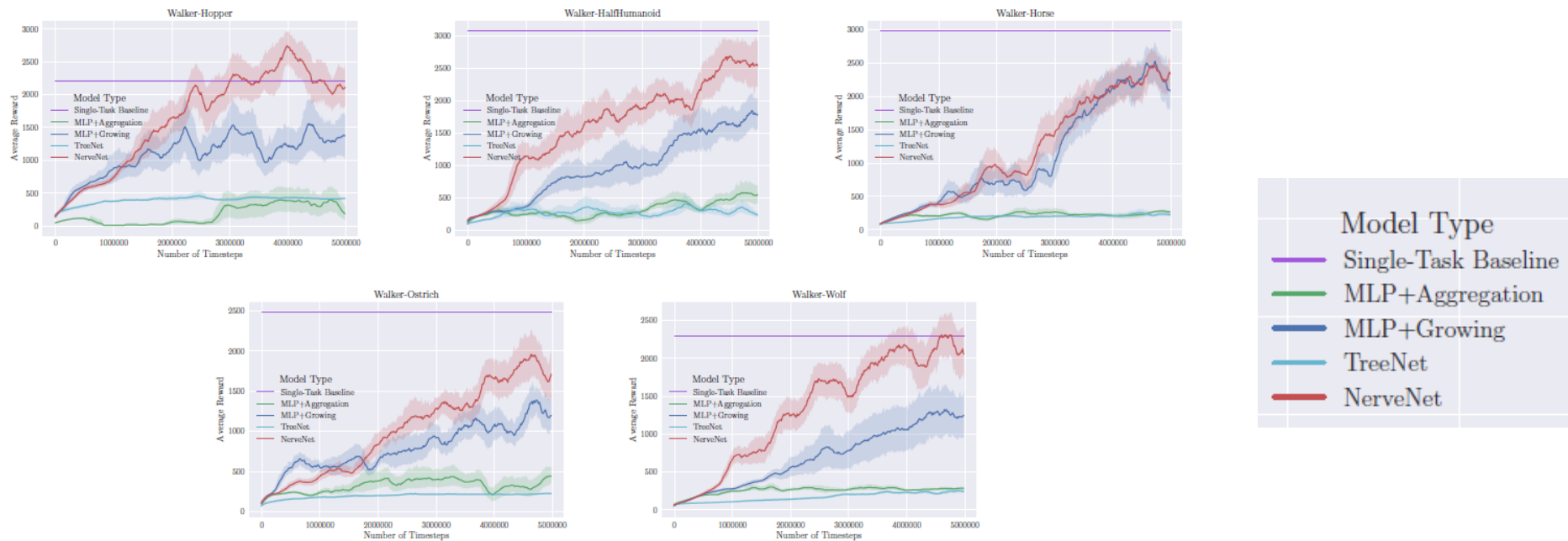
(c)



(d)

Experiments

- Multi-task experiments
 - Constrain the Walker multi-task learning
 - Aim to test the ability to control multiple agents using one unified network.



Results of Multi-task learning. We train the networks simultaneously on five different tasks from Walker.

My Opinion on this paper

- Pros
 - Structured Policy
 - Transfer learning experiments
- Cons
 - Only for graph structured agents
 - No significant difference compared to MLP

Thanks!

