NerveNet: Learning Structured Policy with Graph Neural Networks

Tingwu Wang, Renjie Liao, Jimmy Ba & Sanja Fidler
ICLR 2018 [openreview.net/forum?id=S1sqHMZCb]

Presented by Cătălina Cangea

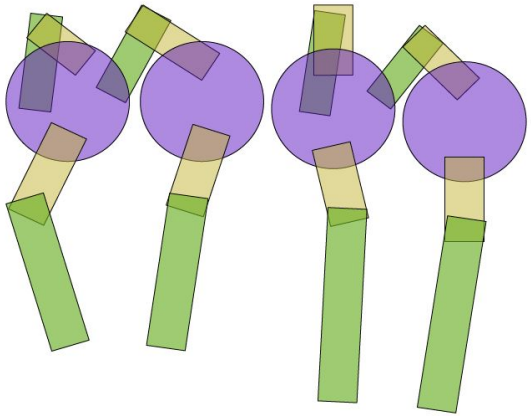
RL agents with dependent controllers

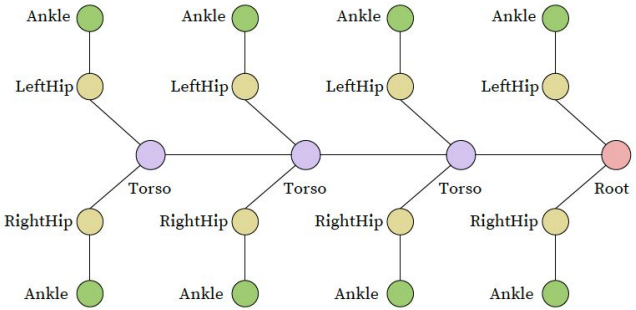
- Common RL: concatenation of observations passed through an MLP to learn agent policy
- Longer training times and exposure to environment needed to discover latent relationships between observations, such as…

→ **agent structure**! (i.e. joints with links/dependencies)

RL agents with dependent controllers

- Actions taken by a joint/body should depend on:
  - Its own observations (previous work)
  - **Actions of other joints (this paper)**





NerveNet

- Exploit the body structure of an agent and physical dependencies
- Agent policy → Graph Neural Network model
- Main idea
  - Propagate information between different parts of the body *along the graph structure*
  - Output action probabilities for each part

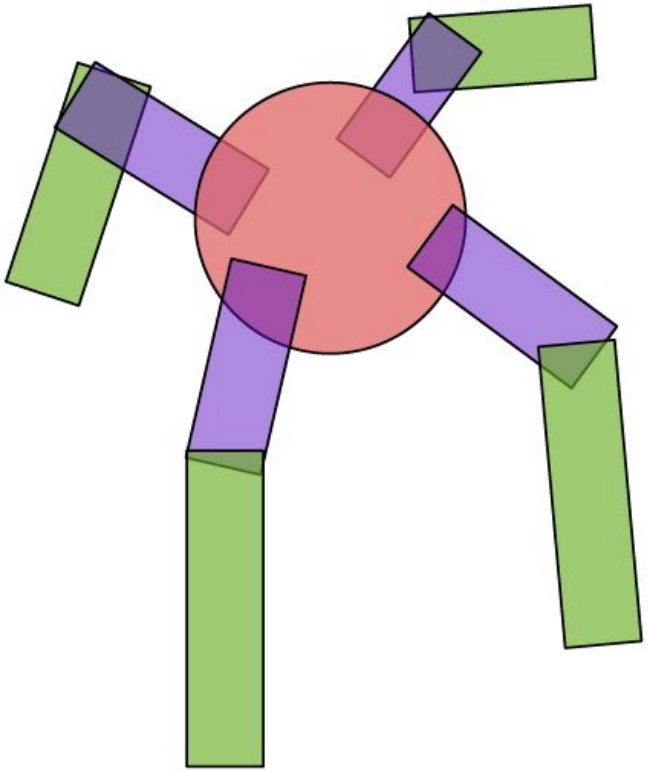
Contributions

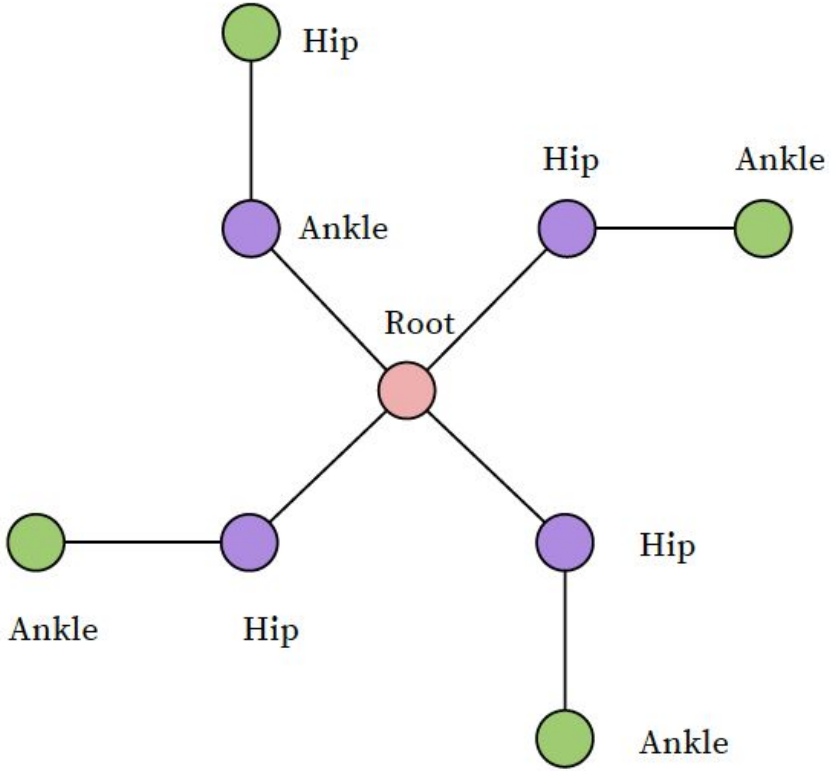
- *Transferable and generalized* features obtained by placing prior on agent structure using GNN model
- *RL benchmarks*: results comparable to SOTA MLP-based models
- *Structure transfer* and *multi-task learning*: better results

Graph construction

- Use tree graphs from MuJoCo
- Assume two types of nodes:
  - `body` (e.g. `Thigh`, `Shin` in humanoid)
  - `joint` (e.g. `Knee`)
- Single node of type `root` - observes additional information

Ant





Overview of RL setting

- States: graph structure with incoming node features
- Actions: move `joint` nodes
- Observations:
  - for `joint` nodes: angular velocity, twist angle, torque, position
  - for `body` nodes: velocity, inertia, force
- Rewards: task-defined (e.g. +speed, -energy cost)

Overview of RL setting

- PPO (proximal policy optimization) (Schulman et al. (2017))
- Maximize cumulative reward: $J(\theta) = \mathbb{E}_{\pi}\left[\sum_{\tau=0}^{\infty} \gamma^{\tau} r(s^{\tau}, a^{\tau})\right]$
- Minimize KL-divergence of new/old policy penalty term and value function loss

$$
\begin{aligned}
\tilde{J}(\theta) =& J(\theta) - \beta L_{KL}(\theta) - \alpha L_{V}(\theta) \\
=& \mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{\infty} \min\left(\hat{A}^{\tau} r^{\tau}(\theta), \hat{A}^{\tau}\ \text{clip}\left(r^{\tau}(\theta), 1-\epsilon, 1+\epsilon\right)\right)\right] \\
& - \beta \mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{\infty} \text{KL}\left[\pi_\theta(:|s^{\tau}) | \pi_{\theta_{old}}(:|s^{\tau})\right]\right] - \alpha \mathbb{E}_{\pi_\theta}\left[\sum_{\tau=0}^{\infty} \left(V_\theta(s^{\tau}) - V(s^{\tau})^{\text{target}}\right)^2\right]
\end{aligned}
$$

NerveNet as policy network

- 3 component models
  - Input
  - **Propagation** (GNN with synchronous message-passing)
  - Output
- At each (RL environment) step: receive observation, perform T propagation steps, decide action for each node

Notation

- $G = (V, E)$, outgoing edges $\mathcal{N}_{out}(u)$, incoming $\mathcal{N}_{in}(u)$
- Node types $p_u \in \{1, 2, \ldots, P\}$
- Edge types $c_{(u,v)} \in \{1, 2, \ldots, C\}$
- Time:
  - τ in RL environment
  - t in NerveNet propagation

Input model

- At each environment time step, for node *u*:

$$h_u^0 = F_{\text{in}}(x_u)$$

- Obtain fixed size vector at propagation step 0

Propagation model (I)

- **Message computation** (for each outgoing edge from *u*):

$$m^t_{(u,v)} = M_{c_{(u,v)}}(h^t_u)$$

- Edges of same type *share* instance of message function

Propagation model (II)

- **Message aggregation** (from all incoming edges of *u*):

$$\bar{m}_u^t = A(\{m_{(u,v)}^t | v \in \mathcal{N}_{in}(u)\})$$

Propagation model (III)

- **States update** on every node's state vector:

$$h_u^{t+1} = U_{p_u}(h_u^t, \bar{m}_u^t)$$

- U can be GRU, LSTM, MLP
- Nodes of same type *share* instance of update function

Output model

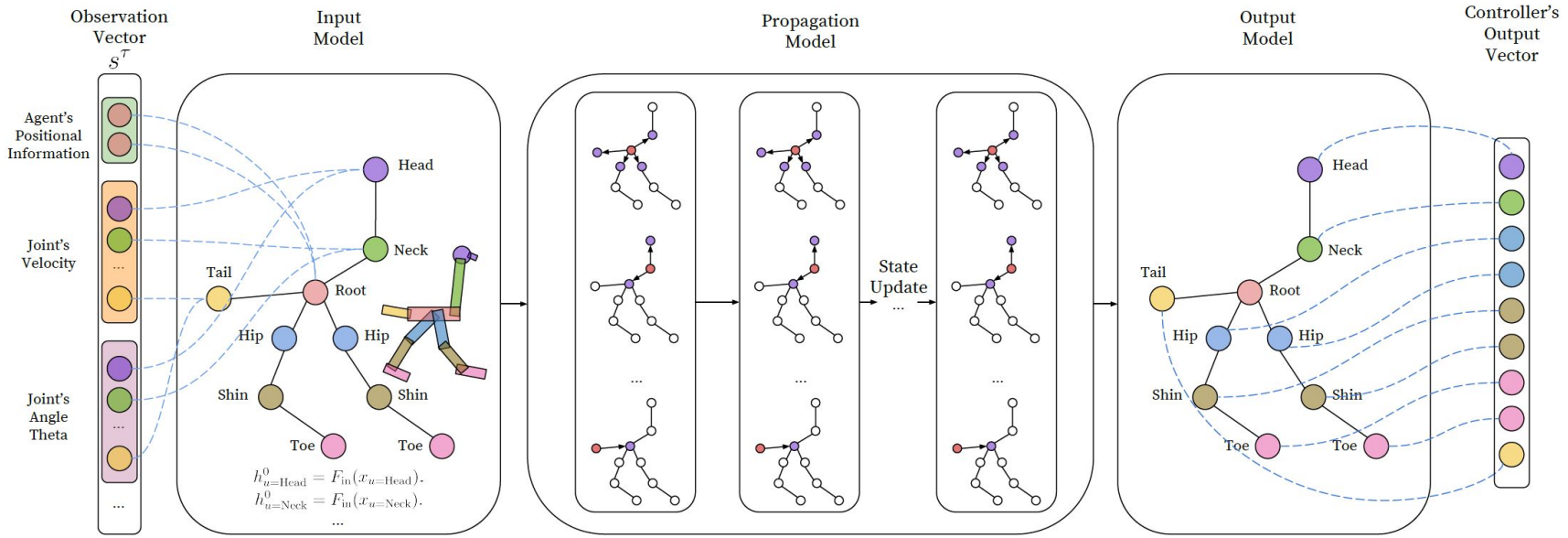
- For *each* node *u* with an associated controller, predict *mean value* of action applied to actuator (with $T = 3..6$):

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

- Stochastic policy:

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

Putting it all together



Value network

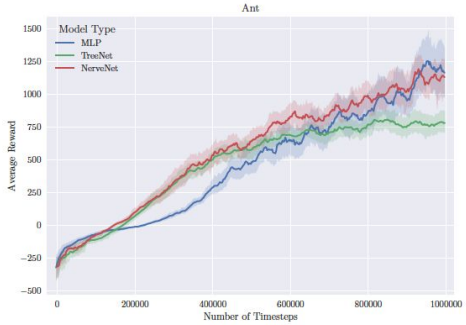
- Obtain state value $V_\theta(s^\tau)$ for observation s^τ
- Three alternatives
  - NerveNet-MLP (GNN for policy, MLP for value)
  - NerveNet-2 (*separate* GNNs for policy and value)
  - NerveNet-1 (*same* GNN for policy and value)

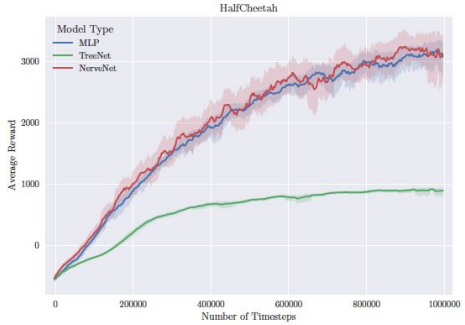
Evaluation

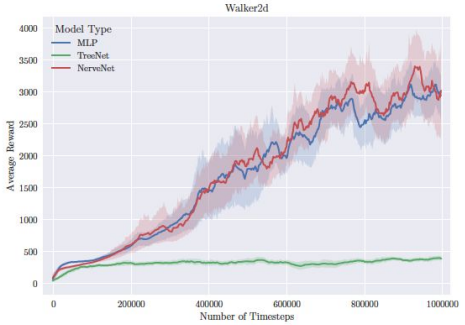
1. Comparison on standard benchmarks of MuJoCo in Gym
2. Structure transfer learning
3. Multi-task learning
4. Robustness
5. Interpretation
6. Comparison of model variants

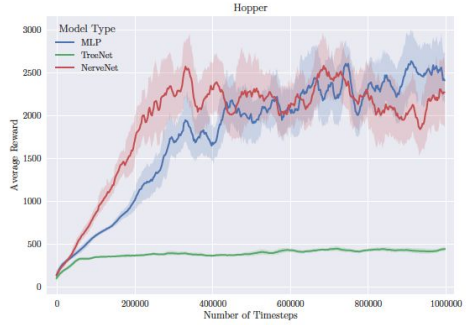
Comparison on continuous control MuJoCo

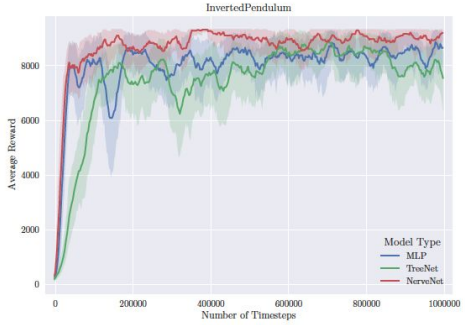
- Baselines
  - **Standard MLP** models (Schulman et al. (2017))
  - **TreeNet**:
    - remove physical structure, introduce *super-node* as `root`, connect it to all nodes in the graph
    - similar propagation model: (1) aggregate messages from all children; (2) give state vector of root to output model

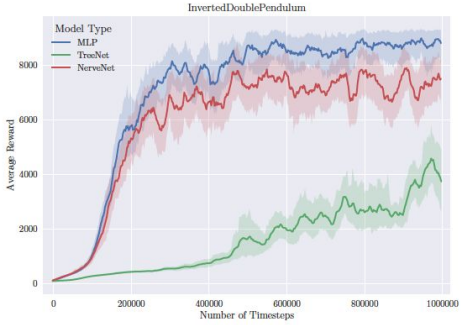

Ant
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps

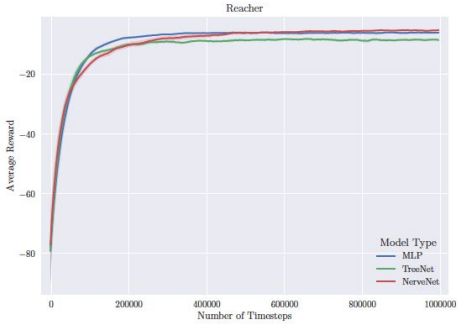

HalfCheetah
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps

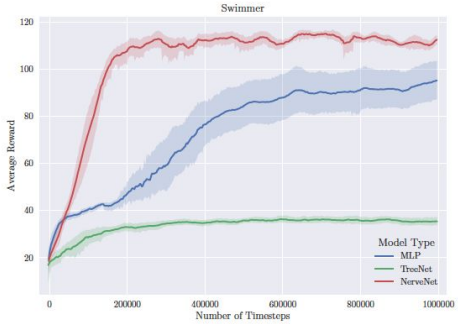

Walker2d
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps


Hopper
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps


InvertedPendulum
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps


InvertedDoublePendulum
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps


Reacher
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps


Swimmer
Model Type
MLP
TreeNet
NerveNet
Average Reward
Number of Timesteps

Structure transfer learning tasks

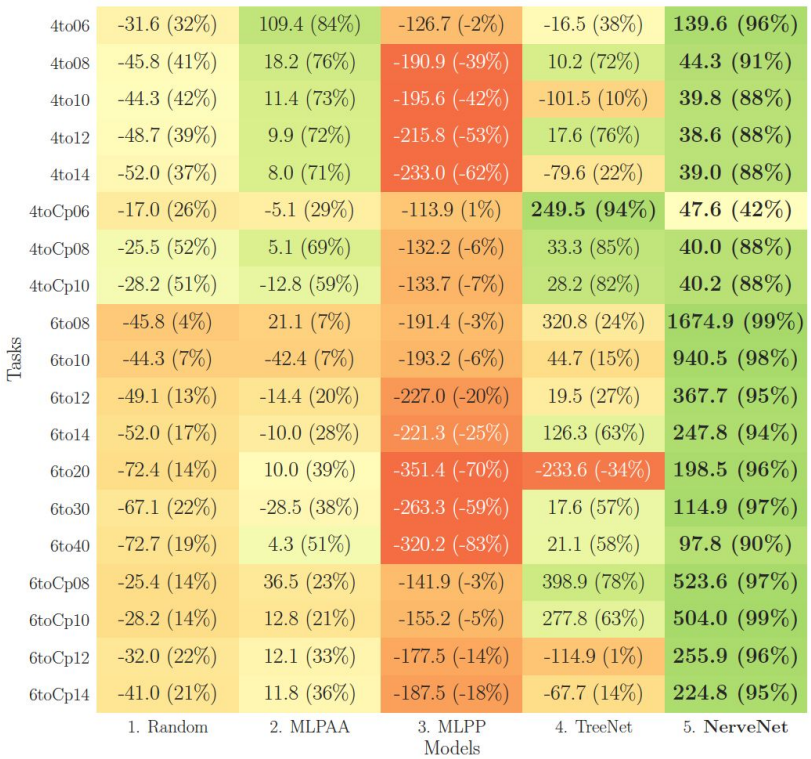
- *Size* transfer:
  - train on smaller agent
  - apply on larger agent
- *Disability* transfer:
  - train on original agent
  - apply on “crippled” agent (some components disabled)

Models

- **NerveNet:**
  - reuse all weights of small-agent model (corresponding joints)
- **MLP pre-trained (MLPP):**
  - reuse weights from hidden to output layer
- **MLP activation assigning (MLPAA):**
  - reuse weights of small-agent model as partial weights of the large-agent model, set the remaining to 0
- **TreeNet:**
  - same reassignment as for MLPAA

Environments (I)

- `Centipede`
  - Repetitive body structure (torso with 2 legs attached)
  - Leg = `thigh` + `shin`
  - Goal: run as fast as possible in the y-direction
  - Agents of different lengths (4..40) for *size transfer*
  - `CrippleCentipede` (2 back legs off) for *disability transfer*



Tasks	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. **NerveNet**
4to06	-31.6 (32%)	109.4 (84%)	-126.7 (-2%)	-16.5 (38%)	**139.6 (96%)**
4to08	-45.8 (41%)	18.2 (76%)	-190.9 (-39%)	10.2 (72%)	**44.3 (91%)**
4to10	-44.3 (42%)	11.4 (73%)	-195.6 (-42%)	-101.5 (10%)	**39.8 (88%)**
4to12	-48.7 (39%)	9.9 (72%)	-215.8 (-53%)	17.6 (76%)	**38.6 (88%)**
4to14	-52.0 (37%)	8.0 (71%)	-233.0 (-62%)	-79.6 (22%)	**39.0 (88%)**
4toCp06	-17.0 (26%)	-5.1 (29%)	-113.9 (1%)	**249.5 (94%)**	**47.6 (42%)**
4toCp08	-25.5 (52%)	5.1 (69%)	-132.2 (-6%)	33.3 (85%)	**40.0 (88%)**
4toCp10	-28.2 (51%)	-12.8 (59%)	-133.7 (-7%)	28.2 (82%)	**40.2 (88%)**
6to08	-45.8 (4%)	21.1 (7%)	-191.4 (-3%)	320.8 (24%)	**1674.9 (99%)**
6to10	-44.3 (7%)	-42.4 (7%)	-193.2 (-6%)	44.7 (15%)	**940.5 (98%)**
6to12	-49.1 (13%)	-14.4 (20%)	-227.0 (-20%)	19.5 (27%)	**367.7 (95%)**
6to14	-52.0 (17%)	-10.0 (28%)	-221.3 (-25%)	126.3 (63%)	**247.8 (94%)**
6to20	-72.4 (14%)	10.0 (39%)	-351.4 (-70%)	-233.6 (-34%)	**198.5 (96%)**
6to30	-67.1 (22%)	-28.5 (38%)	-263.3 (-59%)	17.6 (57%)	**114.9 (97%)**
6to40	-72.7 (19%)	4.3 (51%)	-320.2 (-83%)	21.1 (58%)	**97.8 (90%)**
6toCp08	-25.4 (14%)	36.5 (23%)	-141.9 (-3%)	398.9 (78%)	**523.6 (97%)**
6toCp10	-28.2 (14%)	12.8 (21%)	-155.2 (-5%)	277.8 (63%)	**504.0 (99%)**
6toCp12	-32.0 (22%)	12.1 (33%)	-177.5 (-14%)	-114.9 (1%)	**255.9 (96%)**
6toCp14	-41.0 (21%)	11.8 (36%)	-187.5 (-18%)	-67.7 (14%)	**224.8 (95%)**

(a) Zero-shot average reward.

1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. **NerveNet**
-75.6 (6%)	545.3 (92%)	-73.5 (6%)	-47.7 (10%)	**577.3 (96%)**
-91.0 (10%)	62.0 (67%)	-80.8 (14%)	-84.8 (13%)	**146.9 (98%)**
-76.5 (16%)	21.0 (52%)	-89.9 (11%)	-70.0 (18%)	**128.4 (92%)**
-81.7 (14%)	13.1 (49%)	-76.9 (15%)	-63.2 (21%)	**126.3 (91%)**
-89.0 (11%)	0.0 (44%)	-86.4 (12%)	-73.2 (17%)	**125.7 (90%)**
-77.3 (17%)	-22.5 (40%)	-79.9 (16%)	-72.2 (19%)	**91.1 (87%)**
-82.9 (17%)	-26.9 (44%)	-91.8 (13%)	-66.9 (25%)	**80.1 (95%)**
-82.9 (17%)	-36.6 (39%)	-88.8 (14%)	-83.7 (17%)	**86.9 (98%)**
-91.0 (0%)	87.8 (1%)	-88.6 (0%)	8.5 (1%)	**10612.6 (99%)**
-76.5 (0%)	-17.0 (1%)	-81.8 (0%)	-77.4 (0%)	**6343.6 (99%)**
-81.2 (1%)	-1.4 (4%)	-79.3 (1%)	-91.5 (1%)	**2532.2 (99%)**
-89.0 (1%)	-3.9 (6%)	-83.2 (1%)	-51.5 (3%)	**1749.7 (98%)**
-95.2 (1%)	-4.9 (7%)	-105.4 (0%)	-102.3 (1%)	**1447.4 (98%)**
-111.2 (0%)	-48.2 (7%)	-127.3 (0%)	-76.7 (4%)	**867.5 (99%)**
63.9 (16%)	140.5 (23%)	56.9 (15%)	101.8 (19%)	**971.5 (98%)**
-83.0 (1%)	138.3 (7%)	-90.7 (0%)	-63.9 (1%)	**3117.3 (99%)**
-82.9 (1%)	13.6 (3%)	-86.4 (0%)	-72.3 (1%)	**3230.3 (99%)**
-87.5 (1%)	7.3 (7%)	-91.4 (1%)	-90.8 (1%)	**1675.5 (99%)**
-96.5 (1%)	2.9 (7%)	-92.5 (1%)	-93.9 (1%)	**1517.6 (99%)**

(b) Zero-shot average running-length.

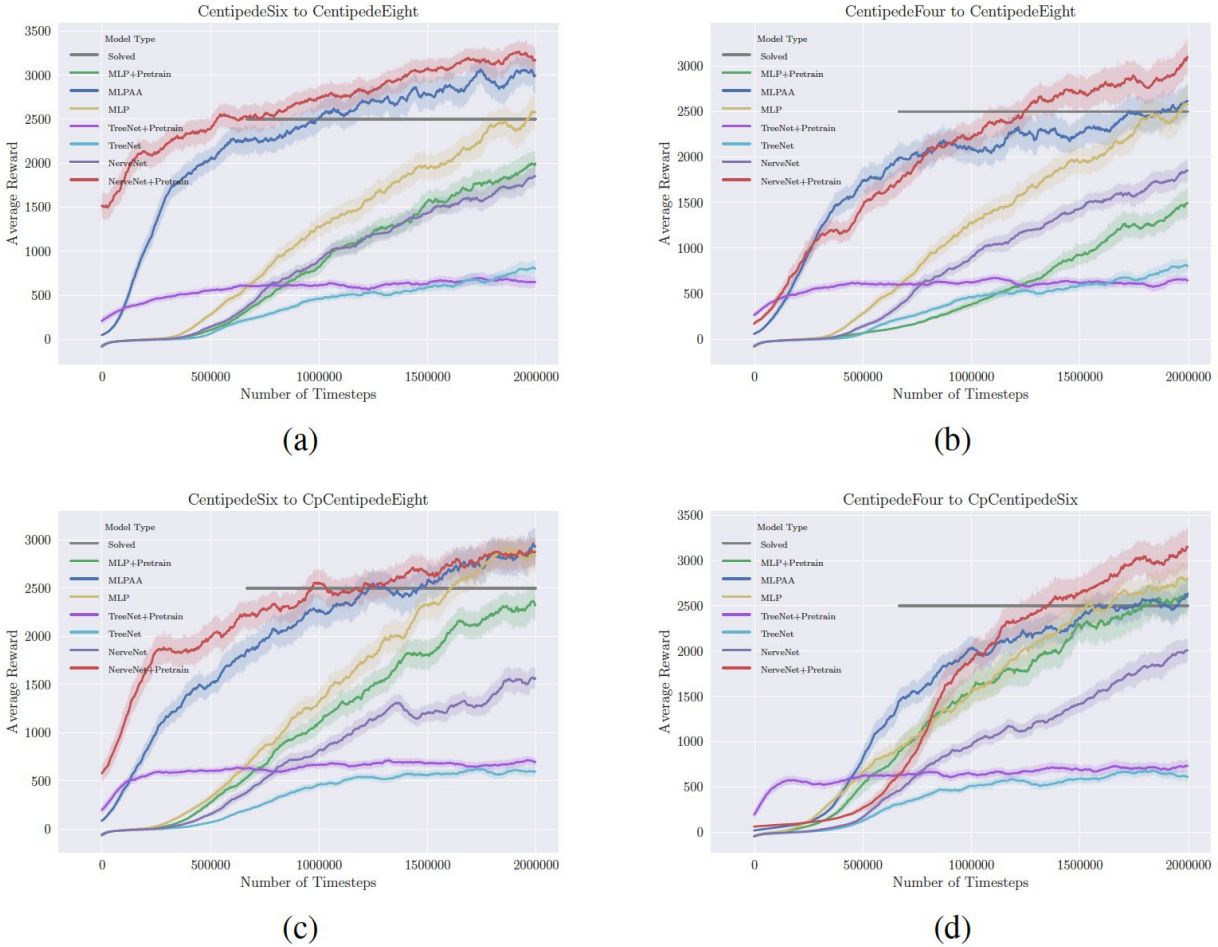
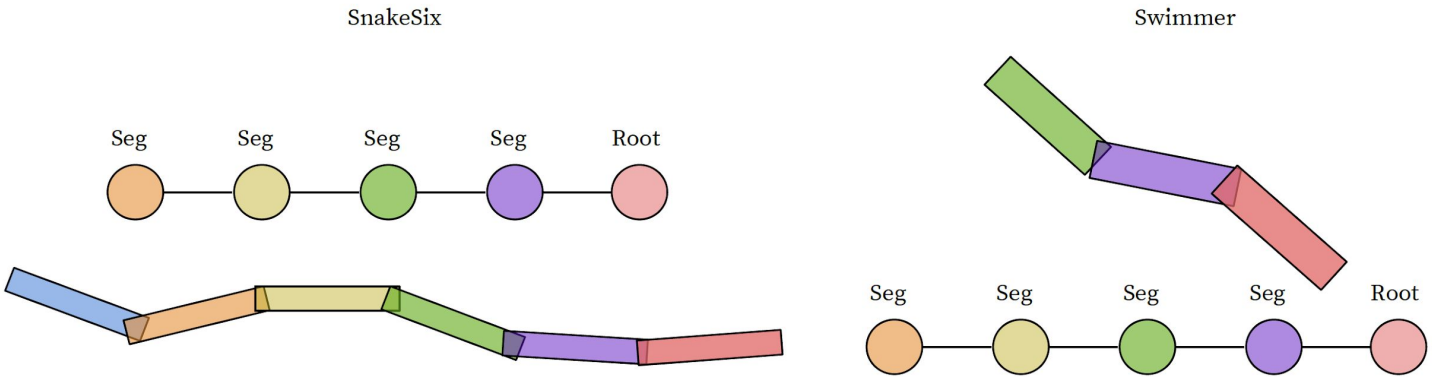


Figure 5: (a), (b): Results of fine-tuning for *size transfer* experiments. (c), (d) Results of fine-tuning for *disability transfer* experiments.

Environments (II)

- `Snake`
  - Based on `Swimmer` model in Gym
  - Goal: move as fast as possible



	1. Random	2. MLPAA	3. MLPP	4. TreeNet	5. **NerveNet**
SnakeFour2SnakeFive	20.59 (12%)	89.82 (30%)	-16.54 (3%)	-15.97 (3%)	**342.88 (95%)**
SnakeFour2SnakeSeven	-2.28 (7%)	115.12 (40%)	-22.81 (1%)	-22.38 (2%)	**313.15 (95%)**
SnakeFour2SnakeSix	7.6 (9%)	105.63 (34%)	-20.45 (2%)	-34.34 (-1%)	**351.85 (97%)**
SnakeThree2SnakeFive	20.59 (14%)	51.41 (22%)	-16.48 (3%)	-21.26 (2%)	**314.92 (95%)**
SnakeThree2SnakeFour	3.86 (9%)	-30.19 (0%)	-12.36 (4%)	-21.82 (2%)	**325.48 (98%)**
SnakeThree2SnakeSeven	-2.28 (9%)	23.53 (17%)	-22.38 (2%)	-42.74 (-4%)	**256.29 (95%)**
SnakeThree2SnakeSix	7.6 (11%)	32.2 (18%)	-19.67 (3%)	-42.46 (-3%)	**282.43 (94%)**

Models

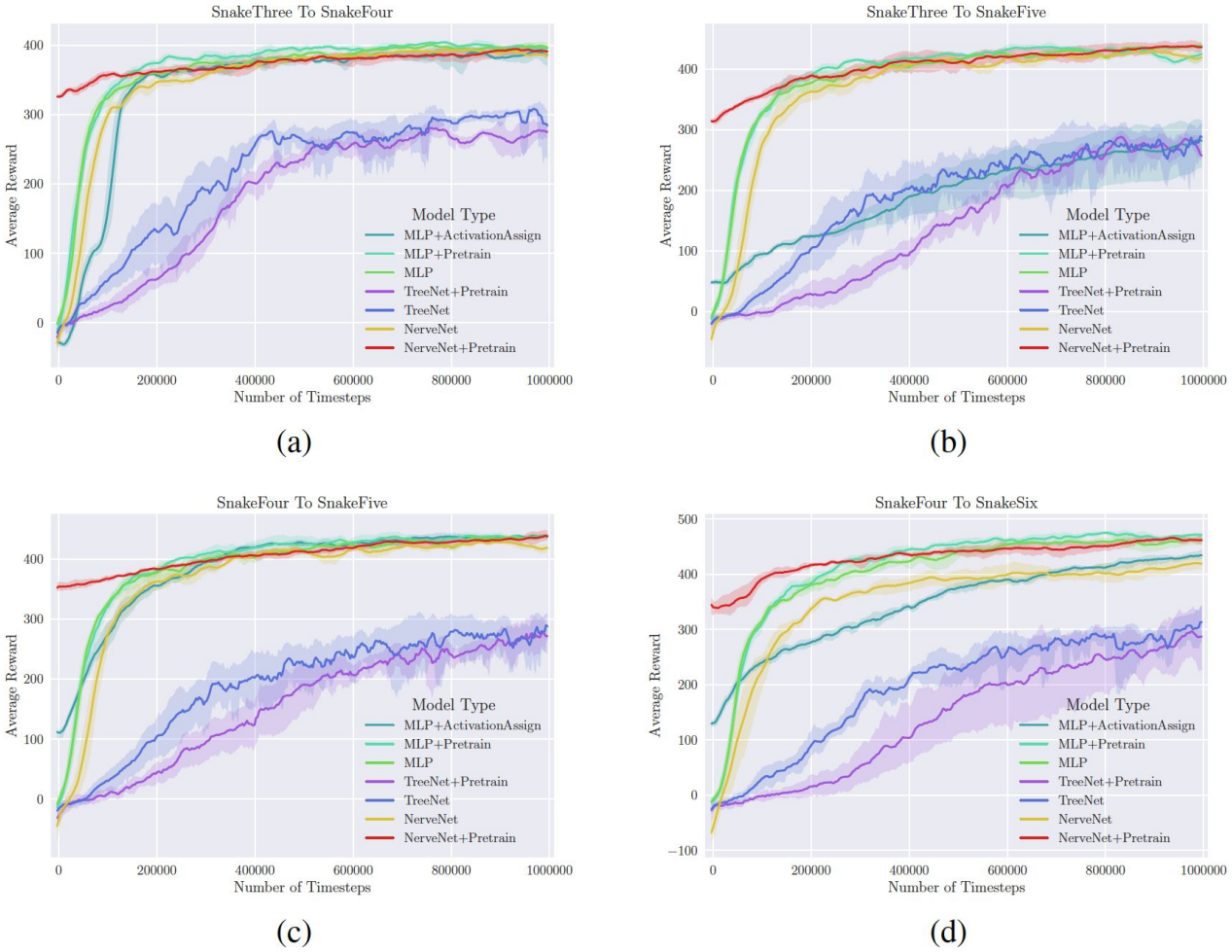
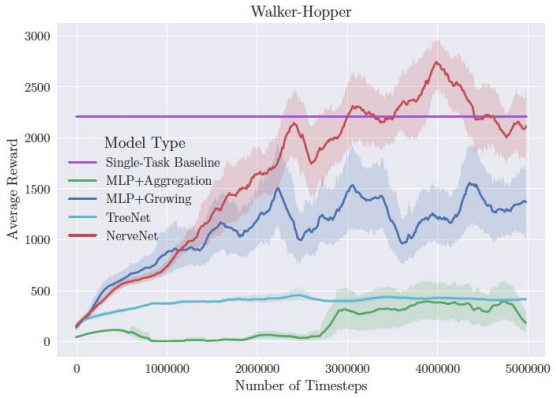
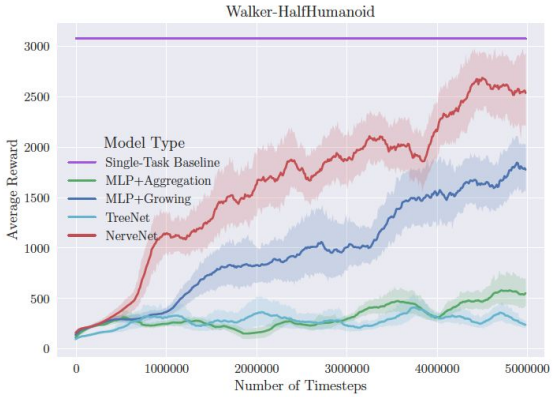


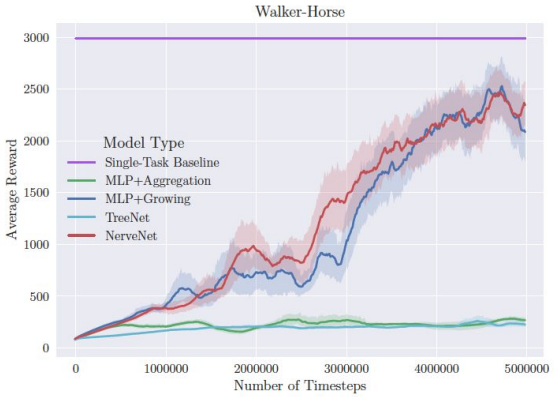
Figure 7: Results of finetuning on snake environments.

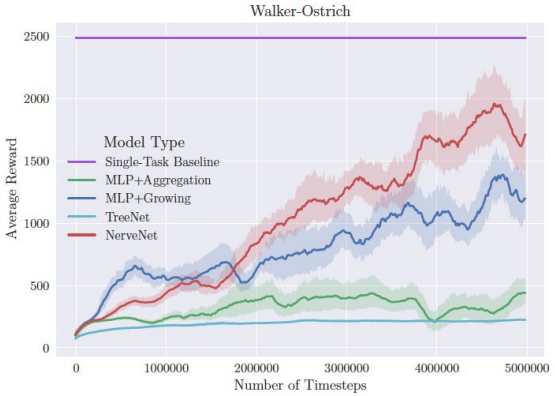
Multi-task learning

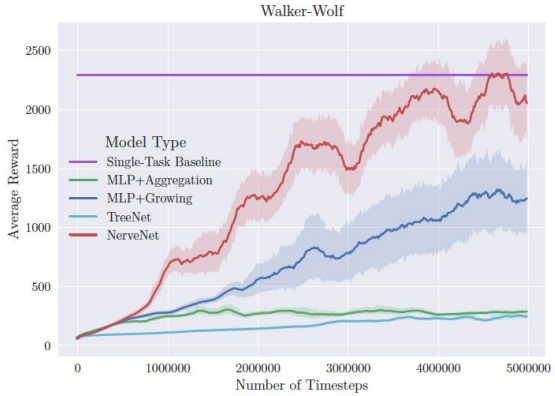
- `Walker` task-set
  - five 2D walkers: `{HalfHumanoid, Hopper, Horse, Ostrich, Wolf}`
- Very different dynamics across agents
  - First two: MuJoCo
  - Last three: natural animals
- **Test ability to control multiple agents with one network**


Walker-Hopper
Average Reward
Number of Timesteps
Model Type
Single-Task Baseline
MLP+Aggregation
MLP+Growing
TreeNet
NerveNet


Walker-HalfHumanoid
Average Reward
Number of Timesteps
Model Type
Single-Task Baseline
MLP+Aggregation
MLP+Growing
TreeNet
NerveNet


Walker-Horse
Average Reward
Number of Timesteps
Model Type
Single-Task Baseline
MLP+Aggregation
MLP+Growing
TreeNet
NerveNet


Walker-Ostrich
Average Reward
Number of Timesteps
Model Type
Single-Task Baseline
MLP+Aggregation
MLP+Growing
TreeNet
NerveNet


Walker-Wolf
Average Reward
Number of Timesteps
Model Type
Single-Task Baseline
MLP+Aggregation
MLP+Growing
TreeNet
NerveNet

Model		HalfHumanoid	Hopper	Ostrich	Wolf	Horse	Average
MLP	Reward	1775.75	1369.59	1198.88	1249.23	2084.07	/
	Ratio	57.7%	62.0%	48.2%	54.5%	69.7%	58.6%
TreeNet	Reward	237.81	417.27	224.07	247.03	223.34	/
	Ratio	79.3%	98.0%	57.4%	141.2%	99.2%	94.8%
NerveNet	Reward	2536.52	2113.56	1714.63	2054.54	2343.62	/
	Ratio	96.3%	101.8%	98.8%	105.9%	106.4%	101.8%

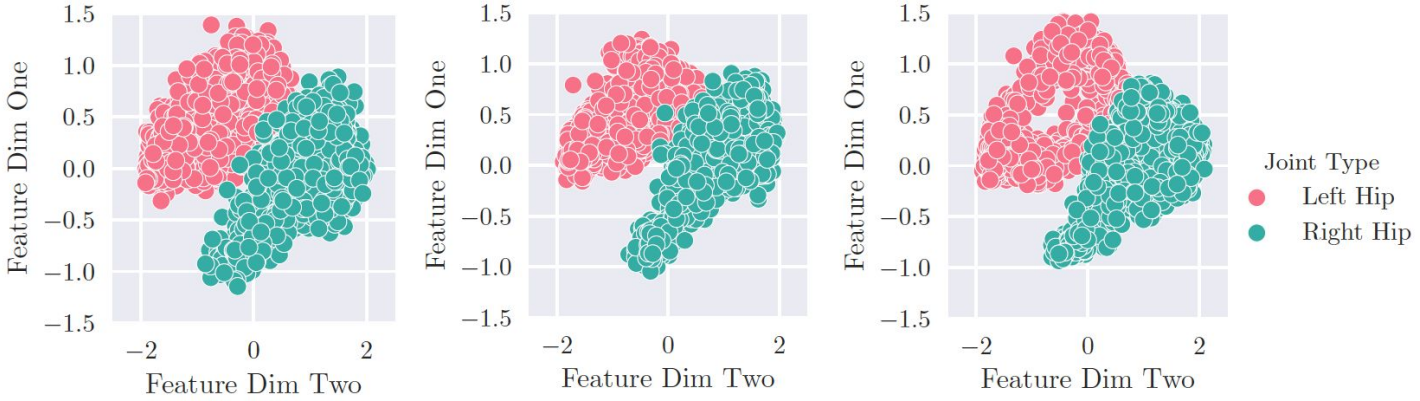
Table 2: Results of Multi-task learning, with comparison to the single-task baselines. For three models, the first row is the mean reward of each model of the last 40 iterations. The second row indicates the percentage of the performance of the multi-task model compared with the single-task baseline of each model.

Robustness of learned policies

- Perturb agent parameters to simulate realistic setting
  - Mass of rigid bodies in MuJoCo
  - Scale of forces on the joints

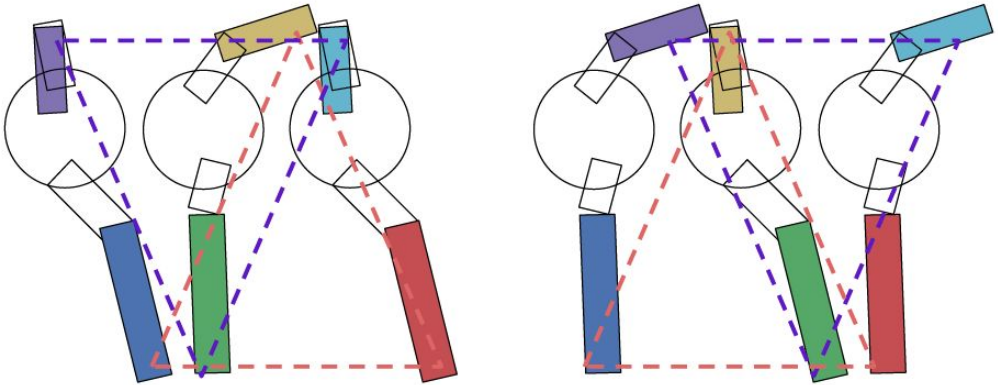
Model		Halfhumanoid	Hopper	Wolf	Ostrich	Horse	Average
Mass	MLP	33.28%	74.04%	94.68%	59.23%	40.61%	60.37%
	NerveNet	95.87%	93.24%	90.13%	80.2%	69.23%	85.73%
Strength	MLP	25.96%	21.77%	27.32%	30.08%	19.80%	24.99%
	NerveNet	31.11%	42.20%	42.84%	31.41%	36.54%	36.82%

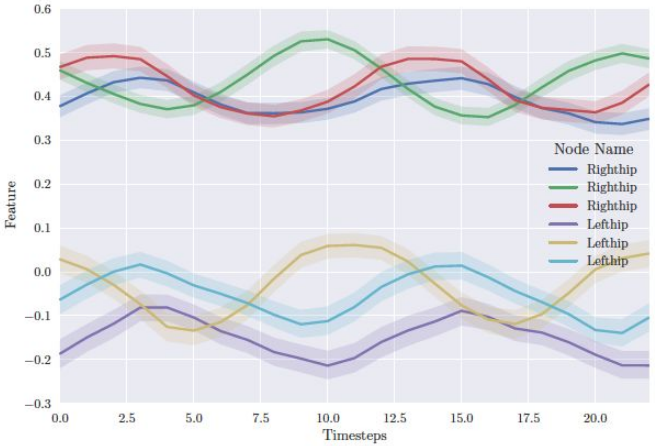
Feature distribution



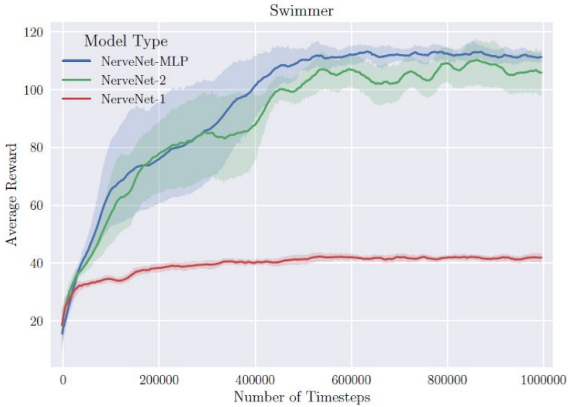
- PCA → represent state vectors of output layer
- Invariant representation across pairs of legs

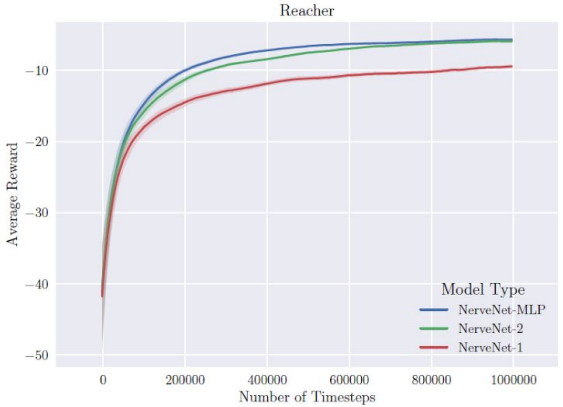
Walk cycle

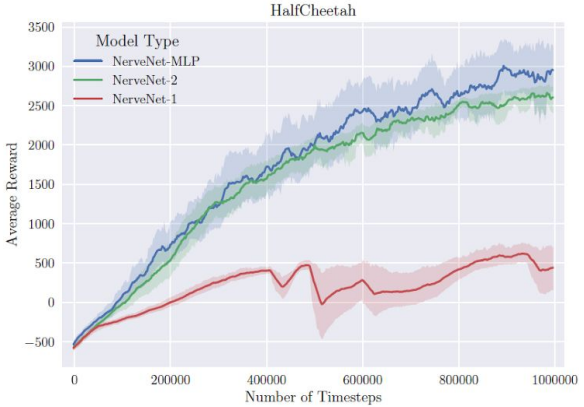



0.6
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
Feature
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Timesteps
Node Name
Righthip
Righthip
Righthip
Lefthip
Lefthip
Lefthip

Comparison of model variants







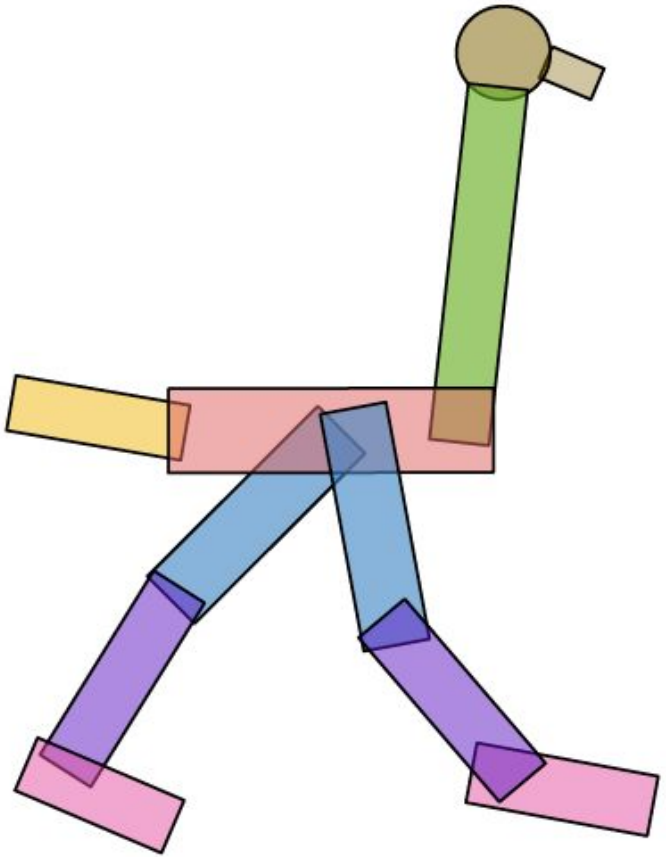
- NerveNet-1 policy/value network sharing affects PPO

Main takeaways

- Exploit physical structure of RL agents
- GNN for agent policy
- Message passing via edges corresponding to physical links
- SOTA on MuJoCo benchmarks
- Strong transfer capabilities (including zero-shot)

Thank you!

Questions?



Catalina.Cangea@cst.cam.ac.uk

www.cl.cam.ac.uk/~ccc53/