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

- Hip
- Ankle
- Root

- Hip
- Ankle
- Hip
- Ankle
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

\[
\tilde{J}(\theta) = J(\theta) - \beta L_{KL}(\theta) - \alpha L_V(\theta) \\
= \mathbb{E}_{\pi_\theta} \left[ \sum_{\tau=0}^{\infty} \min \left( \hat{A}^\tau r^\tau(\theta), \hat{A}^\tau \text{clip}(r^\tau(\theta), 1 - \epsilon, 1 + \epsilon) \right) \right] \\
- \beta \mathbb{E}_{\pi_\theta} \left[ \sum_{\tau=0}^{\infty} \text{KL} [\pi_\theta(:|s^\tau)|\pi_{\theta_{old}}(:|s^\tau)] \right] - \alpha \mathbb{E}_{\pi_\theta} \left[ \sum_{\tau=0}^{\infty} (V_\theta(s^\tau) - V(s^\tau)_{\text{target}})^2 \right]
\]
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:
  - $\tau$ in RL environment
  - $t$ in NerveNet propagation
Input model

• At each environment time step, for node $u$:

$$h_{u}^{0} = F_{in}(x_{u})$$

• Obtain fixed size vector at propagation step 0
Propagation model (I)

- **Message computation** (for each outgoing edge from $u$):
  \[ m_{(u,v)}^t = M_{c_{(u,v)}}(h_u^t) \]
- Edges of same type *share* instance of message function
Propagation model (II)

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

\[
\bar{m}_u^t = A(\{m_{(u,v)}^t | v \in N_{in}(u)\})
\]
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 $T = 3..6$):
  \[ \mu_{u \in \mathcal{O}} = O_{q_u} (h_u) \]

- Stochastic policy:
  \[
  \pi_\theta(a^T | s^T) = \prod_{u \in \mathcal{O}} \pi_{\theta,u}(a_u^T | s^T) = \prod_{u \in \mathcal{O}} \frac{1}{\sqrt{2\pi\sigma_u^2}} e^{(a_u^T - \mu_u)^2 / (2\sigma_u^2)}
  \]
Putting it all together
Value network

- Obtain state value $V_\theta(s^\tau)$ for observation $s^\tau$
- 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
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
(a) Zero-shot average reward.

(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
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SnakeFour2SnakeFive</td>
<td>20.59 (12%)</td>
<td>89.82 (30%)</td>
<td>-16.54 (3%)</td>
<td>-15.97 (3%)</td>
<td>342.88 (95%)</td>
</tr>
<tr>
<td>SnakeFour2SnakeSeven</td>
<td>-2.28 (7%)</td>
<td>115.12 (40%)</td>
<td>-22.81 (1%)</td>
<td>-22.38 (2%)</td>
<td>313.15 (95%)</td>
</tr>
<tr>
<td>SnakeFour2SnakeSix</td>
<td>7.6 (9%)</td>
<td>105.63 (34%)</td>
<td>-20.45 (2%)</td>
<td>-34.34 (-1%)</td>
<td>351.85 (97%)</td>
</tr>
<tr>
<td>SnakeThree2SnakeFive</td>
<td>20.59 (14%)</td>
<td>51.41 (22%)</td>
<td>-16.48 (3%)</td>
<td>-21.26 (2%)</td>
<td>314.92 (95%)</td>
</tr>
<tr>
<td>SnakeThree2SnakeFour</td>
<td>3.86 (9%)</td>
<td>-30.19 (0%)</td>
<td>-12.36 (4%)</td>
<td>-21.82 (2%)</td>
<td>325.48 (98%)</td>
</tr>
<tr>
<td>SnakeThree2SnakeSeven</td>
<td>-2.28 (9%)</td>
<td>23.53 (17%)</td>
<td>-22.38 (2%)</td>
<td>-42.74 (-4%)</td>
<td>256.29 (95%)</td>
</tr>
<tr>
<td>SnakeThree2SnakeSix</td>
<td>7.6 (11%)</td>
<td>32.2 (18%)</td>
<td>-19.67 (3%)</td>
<td>-42.46 (-3%)</td>
<td>282.43 (94%)</td>
</tr>
</tbody>
</table>
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**
<table>
<thead>
<tr>
<th>Model</th>
<th>Reward</th>
<th>HalfHumanoid</th>
<th>Hopper</th>
<th>Ostrich</th>
<th>Wolf</th>
<th>Horse</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td></td>
<td>1775.75</td>
<td>1369.59</td>
<td>1198.88</td>
<td>1249.23</td>
<td>2084.07</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Ratio</td>
<td>57.7%</td>
<td>62.0%</td>
<td>48.2%</td>
<td>54.5%</td>
<td>69.7%</td>
<td>58.6%</td>
</tr>
<tr>
<td>TreeNet</td>
<td></td>
<td>237.81</td>
<td>417.27</td>
<td>224.07</td>
<td>247.03</td>
<td>223.34</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Ratio</td>
<td>79.3%</td>
<td>98.0%</td>
<td>57.4%</td>
<td>141.2%</td>
<td>99.2%</td>
<td>94.8%</td>
</tr>
<tr>
<td>NerveNet</td>
<td></td>
<td>2536.52</td>
<td>2113.56</td>
<td>1714.63</td>
<td>2054.54</td>
<td>2343.62</td>
<td>/</td>
</tr>
<tr>
<td></td>
<td>Ratio</td>
<td>96.3%</td>
<td>101.8%</td>
<td>98.8%</td>
<td>105.9%</td>
<td>106.4%</td>
<td>101.8%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Halfhumanoid</th>
<th>Hopper</th>
<th>Wolf</th>
<th>Ostrich</th>
<th>Horse</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mass</td>
<td>MLP</td>
<td>33.28%</td>
<td>74.04%</td>
<td>94.68%</td>
<td>59.23%</td>
<td>40.61%</td>
</tr>
<tr>
<td></td>
<td>NerveNet</td>
<td>95.87%</td>
<td>93.24%</td>
<td>90.13%</td>
<td>80.2%</td>
<td>69.23%</td>
</tr>
<tr>
<td>Strength</td>
<td>MLP</td>
<td>25.96%</td>
<td>21.77%</td>
<td>27.32%</td>
<td>30.08%</td>
<td>19.80%</td>
</tr>
<tr>
<td></td>
<td>NerveNet</td>
<td>31.11%</td>
<td>42.20%</td>
<td>42.84%</td>
<td>31.41%</td>
<td>36.54%</td>
</tr>
</tbody>
</table>
Feature distribution

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