NerveNet: Towards Universal Deep RL Controllers for Robotics

Tingwu Wang

University Of Toronto, Vector Institute

Contents

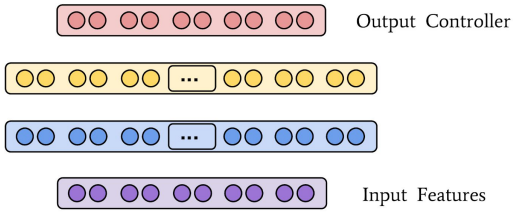
Reinforcement Learning with Neural Network

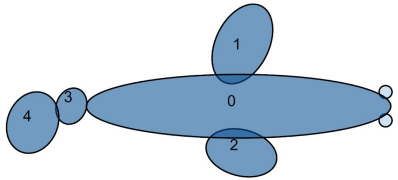
1. One of the most exciting area recently
   a. AlphaGo[1]
   b. Complex humanoid control [2, 3] and industry robot[4, 5]
   c. Video games [6, 7]

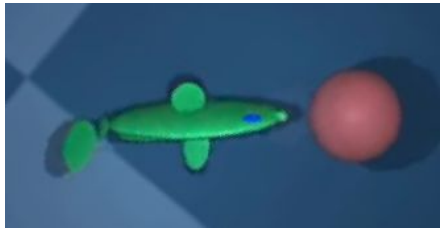
2. Problems do exist [8], for example:
   a. Transferability
      Policy learnt on one agent could not be transferred to agent with different dynamics.
   b. Robustness
      Perturbing the parameters of robot results in catastrophic performance drop.

Reinforcement Learning with Neural Network

3. We constrain the focus to be locomotion control of simulated robot [9]
   a. Input: observations from the sensors of the robot.
      i. Joint angles & velocity, body's Cartesian positions, feedback forces
      ii. ...
   b. Output: the torques applied respectively on each joint.
4. A typical neural network used in these tasks:
   a. Multi-Layer Perceptron Network (MLP)
   b. Ignoring structural information and performing black-box optimization.
      i. PPO [3], TRPO [10], A3C [11], DPG[12], DDPG [13]

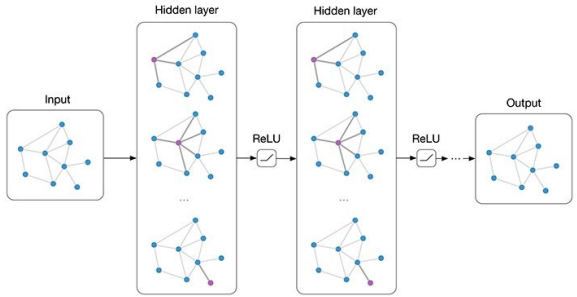


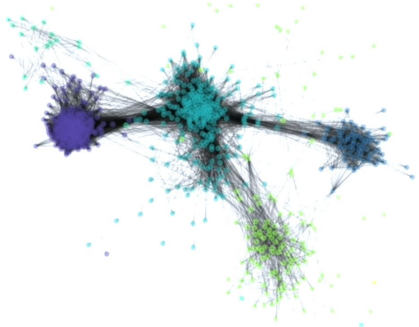




Graph Neural Networks

1. Many types of the input data is a vector or tensor, for example
   a. Observations of the robot (vector, MLP)
   b. Images (tensor, CNN)
2. Many important real-world data could be formulated as graphs:
   a. Social networks, knowledge graphs, the World Wide Web, etc. [14].
   b. A more natural idea is to use graph neural networks to process the data
   c. Example: How likely are you a Warrior supporter or Cavalier supporter?

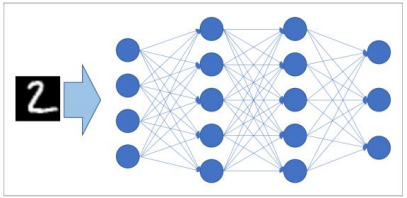


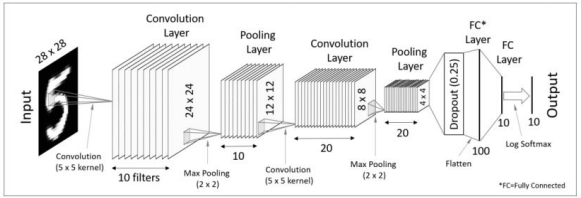


Contents

Basic Idea

1. Use Graph Neural Networks instead of MLPs
   a. MLP Potentially causing the problem on transferability and robustness.
      i. A perhaps similar story: Image classification with MLP or CNN?
      ii. Empirically MLP doesn't scale with data and is easy to overfit
      iii. CNN applies strong structure prior and uses shared weights

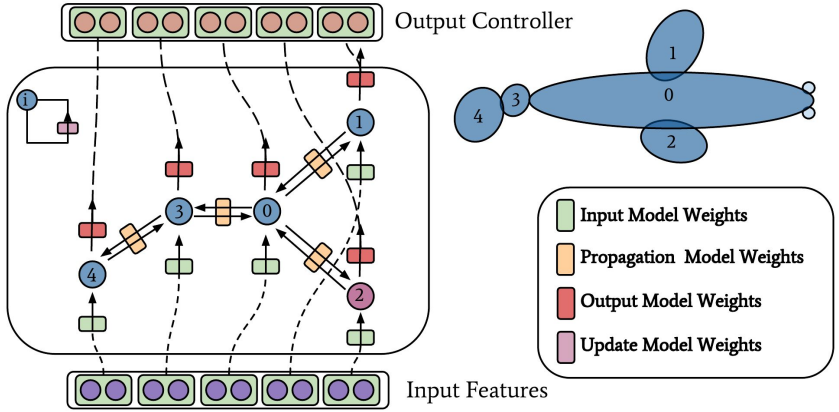




   b. Make use of structural information with GNN
   c. Intuition: to determine the torques applied to one joint
      i. neighbour's absolute position, joint's position & velocity, inertia info, ...

Input Model

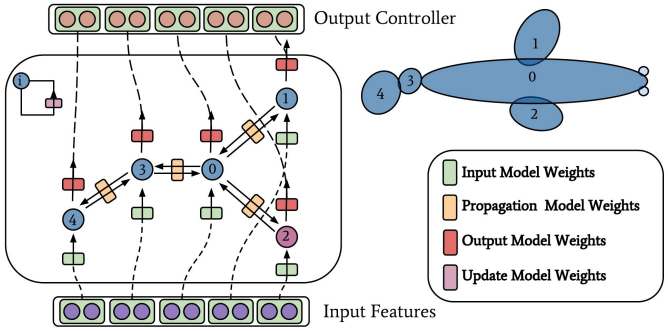
1. For each node (0, 1, 2, 3, 4) of the agent, we fetch the corresponding observation from the input features.
2. Map it into the initial node states using a function F: $h_u^0 = F_{\text{in}}(x_u)$
   a. a matrix, identity function, or MLP.



Propagation Model

1. Let each node communicate with the neighbours by propagating messages.
   a. Message is calculated by passing the current states through a function M.
      $$m^t_{(u,v)} = M_{c_{(u,v)}}(h^t_u)$$
   b. Node aggregate the messages and update its current node states.
      $$\bar{m}^t_u = A(\{h^t_v | v \in \mathcal{N}_{in}(u)\})$$
   c. Ideally we should propagate the messages until the node states converge
      i. In practice we propagate the message for 4 timesteps.

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

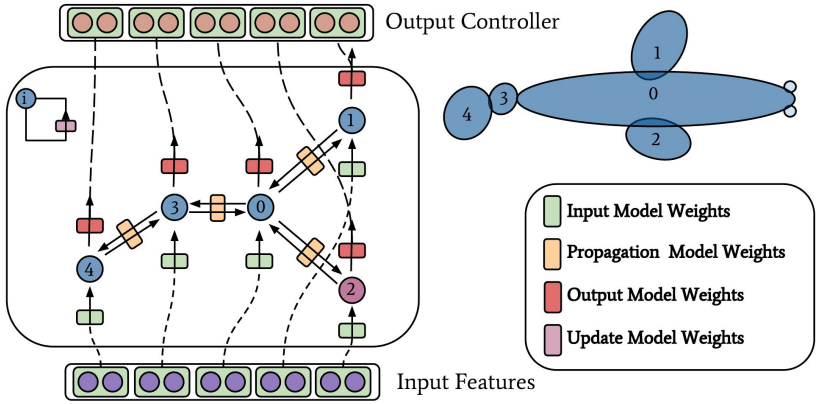


Output Model

1. For each node with a controller, we output the control signal.
   a. The control signal is a function of the node last states.

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

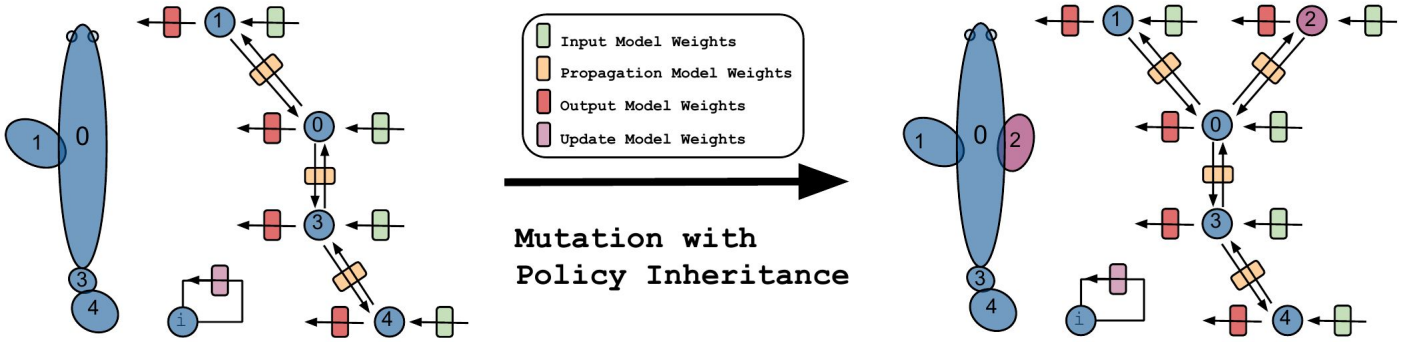
   b. Control signal could be the mean and variance of the gaussian policy.



Contents

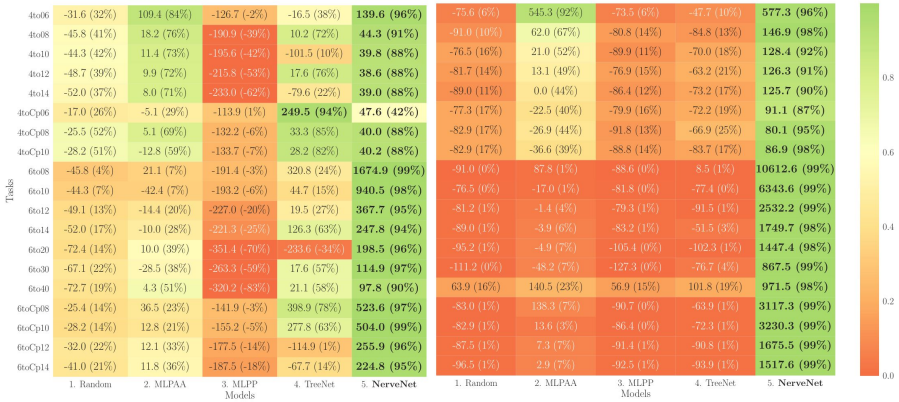
Transfer Learning

1. As we mentioned before, in old methods, policy learnt on one agent could not be transferred to agent with different dynamics. (full result in [15])
   a. Transfer learning benchmarks: snake and centipedes.

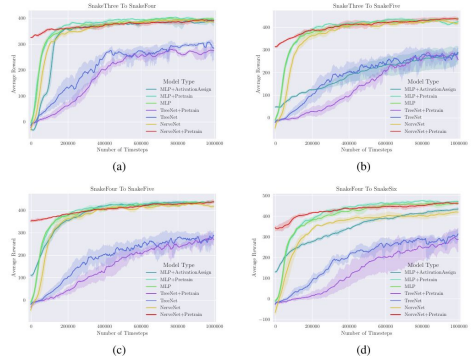


Transfer Learning

1. As we mentioned before, in old methods, policy learnt on one agent could not be transferred to agent with different dynamics. (full result in [15])
   a. Transfer learning benchmarks: snake and centipedes.



(a) Zero-shot average reward. (b) Zero-shot average running-length.



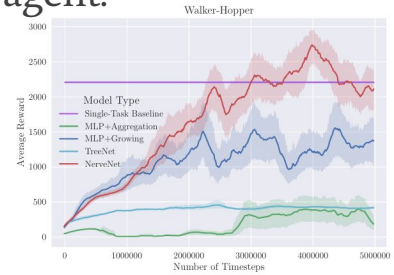
Robustness

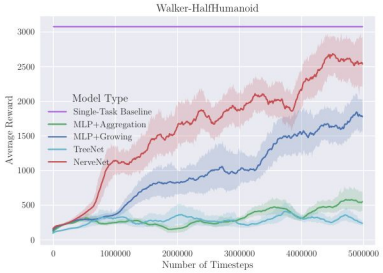
1. We also test the robustness of the policy by perturbing the parameters of the

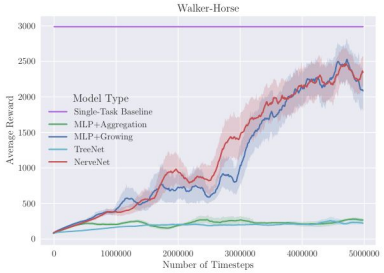
Model		HalfHumanoid	Hopper	Ostrich	Wolf	Horse	Average
MLP	Reward	1775.75	1369.59	1198.88	1249.23	2084.07	/
	Ratio	57.7%	62.0%	48.2%	54.5%	69.7%	58.6%
TreeNet	Reward	237.81	417.27	224.07	247.03	223.34	/
	Ratio	79.3%	98.0%	57.4%	141.2%	99.2%	94.8%
NerveNet	Reward	2536.52	2113.56	1714.63	2054.54	2343.62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

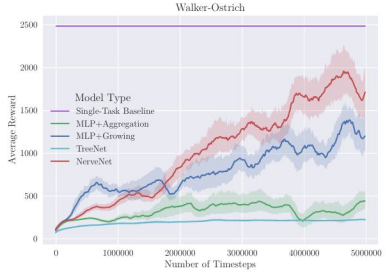
Multi-task Learning

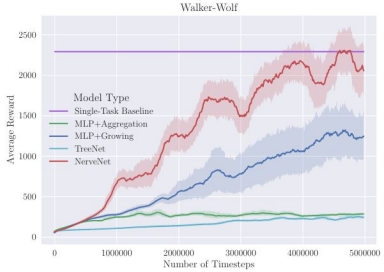
1. We also test the robustness of the policy by perturbing the parameters of the agent.











Contents

Reference

[1] Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." nature 529.7587 (2016): 484-489.
[2] Heess, Nicolas, et al. "Emergence of locomotion behaviours in rich environments." arXiv preprint arXiv:1707.02286 (2017).
[3] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).
[4] Andrychowicz, Marcin, et al. "Hindsight experience replay." Advances in Neural Information Processing Systems. 2017.
[5] Gu, Shixiang, et al. "Q-prop: Sample-efficient policy gradient with an off-policy critic." arXiv preprint arXiv:1611.02247 (2016).
[6] Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602 (2013).
[7] Fernandez, Jose Maria Font, and Tobias Mahlmann. "The Dota 2 Bot Competition." IEEE Transactions on Games (2018).
[8] Rajeswaran, Aravind, et al. "Towards generalization and simplicity in continuous control." Advances in Neural Information Processing Systems. 2017.
[9] Todorov, Emanuel, Tom Erez, and Yuval Tassa. "Mujoco: A physics engine for model-based control." Intelligent Robots and [10] Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012.
[10] Schulman, John, et al. "Trust region policy optimization." International Conference on Machine Learning. 2015.
[11] Mnih, Volodymyr, et al. "Asynchronous methods for deep reinforcement learning." International Conference on Machine Learning. 2016.
[12] Silver, David, et al. "Deterministic policy gradient algorithms." ICML. 2014.
[13] Lillicrap, Timothy P., et al. "Continuous control with deep reinforcement learning." arXiv preprint arXiv:1509.02971 (2015).
[14] Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
[15] Wang, Tingwu, et al. Nervenet: Learning structured policy with graph neural networks. Diss. University of Toronto, 2017.