A Survey of Robotics Control Based on Learning-Inspired Spiking Neural Networks

Overview:

- How does a neuron work? Neurotransmitters, synapses, inactive period, biological background.
- What are spiking neural networks? Event-driven learning - more efficient instead of using ALL information. Data is pulse coded, spikes control the system.
- Why use SNNs?
    - closer to biology
    - speed and energy efficient
    - computationally efficient
    - **ability to encode time**
- How easy is SNN creation? - Bahar
    - Depends on the problem, convert problem to pulses.
    - Certain problems easier to model this way.
    - Two ways to go around this: convert to pulses then use SNNs vs use richer data and use ANNs (Bahar).

- Modeling SNNs. Four steps: neuron modeling, info encoding, synaptic plasticity, network models.
- Neuron modeling:
    - When you have a spike, we care less about amplitude but more about frequency. (Bahar)
    - When you go over threshold, reset.
- Info coding:
    - Several schemes: binary coding, population, temporal, rate.
    - Eg. convert real world data into neuron activity (like firing rate).
- Network models:
    - Feed forward - information travels from input nodes to output nodes.
    - Recurrent - recursive networks in a directed graph.

- Learning: Hebbian, RL, etc.
- Hebbian based learning:
    - “cells that fire together, wire together”
    - supervised vs unsupervised
    - how to decide threshold values? (Bahar)
    - trained using trial error, doesn’t scale well (Semanti)
- Classical conditioning, operant conditioning, reward modulated.

- RL and other algorithms:
    - Temporal diff in RL, lets us look ahead in time
    - Evolutionary, self-organization algorithms, liquid state machines.

- Simulators and Platforms:
    - Large scale neuromorphic HW - Spinnaker, TrueNorth
    - other platforms - Neural Grid, NeuroFlow

- Open research areas:
    - Biological mechanisms - how does brain process info?
    - How do we generalize the training process? ANNs have back-propagation, but what about SNNs?
    - How can we use neuromorphic devices?

- Conclusion:
    - SNNs have a lot of potential if we can figure out how to train SNNs generally and can use them for robotic algorithms.

Interesting points raised:

- **Need for a uniform training method**. Several people raised concerns about the need for a uniform training method and how the lack of one could lead to problems in practical uses of SNNs. Sam also raised a point that one needs to maybe have a single generalized piece of hardware on board the robot for computation. Prof. Bahar agreed and suggested that one could use reconfigurable hardware. Karpur highlighted the lack of detail regarding energy efficiency in generic vs specific training models.

- **Offloading computation to cloud.** Qishen mentioned robots could offload computation to cloud. Prof. Bahar countered that given the large amount of information one needs to transmit, latency requirements, and energy efficiency, this wouldn't be possible. Semanti argued that lack of an internet connection also makes this unviable.

- **SNNs vs ANNs.** Several people wanted to see concrete results for improvement in accuracy/efficiency in SNNs. The paper doesn't compare SNNs to ANNs at all.

- **Training inefficiency.** Both Semanti and Jiwon mentioned that training by trial and error would be inefficient and therefore this cannot scale to more than single/two layer networks. They also argued that encoding/decoding information to pulses is a difficult task and still needs to be figured out before we can use ANNs. Mark also raised a point about training.

- **ANNs for brain simulations.** Jiwon and Karpur argued that one could use ANNs for brain simulations similar to how quantum computers can be used for quantum

simulations.

- **Infeasibility of SNNs.** Yash argued that there is need for neuromorphic computing HW to be able to use SNNs in a real world scenario. Given the large amounts information needed to be processed, one cannot offload computation to cloud.