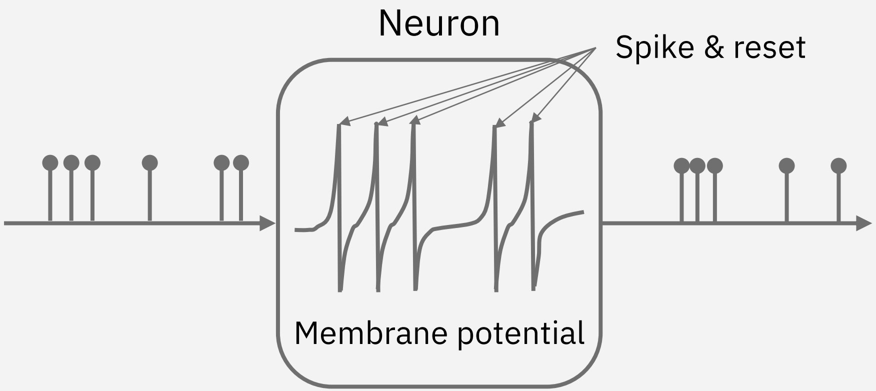
A Differentiable Point Process with its Application to Spiking Neural Networks

Hiroshi Kajino
IBM Research - Tokyo

Opportunities and challenges in spiking neural networks (SNNs)

Spiking neural networks (SNNs)

- Neurons communicate with spikes
- Neuron has a dynamical state (= membrane potential) evolving in continuous time



Opportunities

SNNs are often reported to be more energy-efficient

- Spike-based communication
- Efficient information encoding into spikes

Challenges

- Difficult to train due to discrete nature of spikes
- Time-consuming to simulate SNNs
  - Computation time
  - Step-size parameter to discretize time axis

Our approach: Probabilistic model of SNNs

Point process (PP) = a probalistic model of event seq.

$$\mathcal{T}^{\leq t_N} = \{t_1, \dots, t_N\} \subset [0, T]$$

PP is specified by a **conditional intensity function**:

$$\lambda(t \mid \mathcal{T}^{\leq t_n})\mathrm{d}t$$
$$= \Pr[t_{n+1} \in [t, t + \mathrm{d}t] \mid \mathcal{T}^{\leq t_n} \text{ and } t_{n+1} \notin (t_n, t)]$$

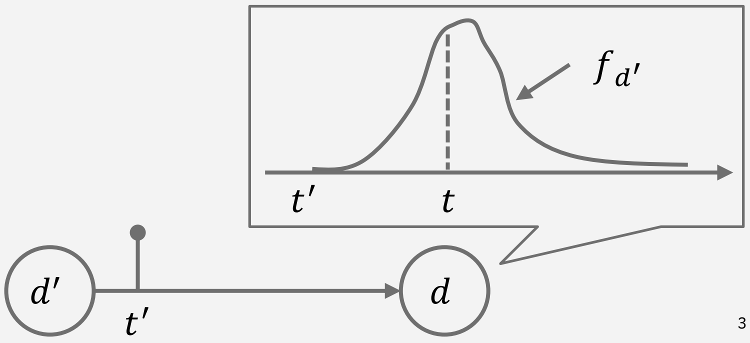
Event in $[t, t + \mathrm{d}t]$

$$= \frac{f(t \mid \mathcal{T}^{\leq t_n})\,\mathrm{d}t}{1 - F(t \mid \mathcal{T}^{\leq t_n})}$$

No event in (t_n, t)

A **probabilistic spiking neuron** is defined as a PP:

- Conditional intensity function ∝ membrane potential
$$\lambda_d(t \mid \mathcal{T}^{\leq t_n}) = \sigma\big(u_d(t \mid \mathcal{T}^{\leq t_n})\big)$$
- Membrane potential = spike response model
$$u_d(t \mid \mathcal{T}^{\leq t_n}) = \overline{u_d} + \sum_{(t',d') \in \mathcal{T}^{\leq t_n}} f_{d'}(t - t')$$



😄 Probabilistic SNNs can be simulated exactly

Thinning algorithm [Lewis+,79] [Ogata,81]

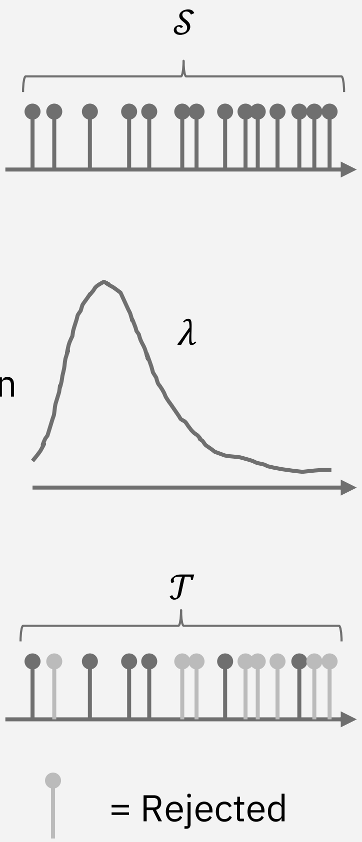
Input: $\lambda(t \mid \mathcal{T}^{\leq t_n})$, upperbound $\bar{\lambda} \geq \lambda(t \mid \mathcal{T}^{\leq t_n})$
Output: $\mathcal{T}$ Realization of $\mathcal{PP}(\lambda)$

$k \leftarrow 1, \mathcal{S} \leftarrow \emptyset, \mathcal{T} \leftarrow \emptyset$
Repeat:
  Sample $s_k \in \mathbb{R} \sim \mathcal{PP}(\bar{\lambda} \mid \mathcal{S})$
  Sample $d_k \in \{0,1\} \sim \mathrm{Bernoulli}\left(\frac{\lambda(s_k \mid \mathcal{T})}{\bar{\lambda}}\right)$
  If $d_k = 1$ then $\mathcal{T} \leftarrow \mathcal{T} \cup \{s_k\}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_k\}$
Return $\mathcal{T}$

Base Poisson point process

Conditional intensity function

Output of the thinning algo.



😢 SNNs w/ hidden neurons are difficult to train

Optimize -ELBO by SGD [Rezende+,11] [Rezende+, 14]

$$\ell(\theta, \phi) = \mathbb{E}_{q(\mathcal{T}_H;\phi)}\underbrace{[-\log p(\mathcal{T}_O, \mathcal{T}_H; \theta) + \log q(\mathcal{T}_H; \phi)]}_{\equiv \hat{\ell}(\theta, \phi; \mathcal{T}_O, \mathcal{T}_H)}$$

Observable Hidden

- $p(\mathcal{T}_O, \mathcal{T}_H; \theta)$: SNN to be trained
- $q(\mathcal{T}_H; \phi)$: Variational distribution (point process)

Existing approach by Rezende+ (2014)

- Gradient w.r.t. θ is easy to estimate
- Gradient w.r.t. ϕ is estimated by

$$\frac{\partial \ell}{\partial \phi} = \frac{\partial}{\partial \phi} \mathbb{E}_{q(\mathcal{T}_H;\phi)}[\hat{\ell}(\theta, \phi; \mathcal{T}_O, \mathcal{T}_H)]$$

$$= \mathbb{E}_{q(\mathcal{T}_H;\phi)}[\hat{\ell} \cdot \nabla_\phi \log q(\mathcal{T}_H; \phi) + \nabla_\phi \hat{\ell}]$$

whose variance is known to be high ☹

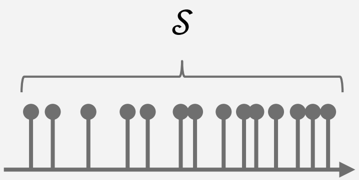
A remedy: Reparameterization trick
To make a realization of $q(\mathcal{T}_H; \phi)$ differentiable

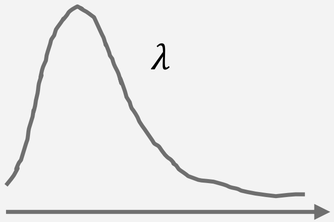
😄 Our differentiable point process enables the reparameterization trick

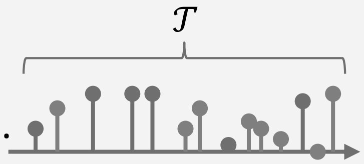
Sampling algorithm of a differentiable PP

Input: $\lambda(\, t \mid \mathcal{T}^{\le t_n}\,)$, upperbound $\bar{\lambda} \ge \lambda(\, t \mid \mathcal{T}^{\le t_n}\,)$
Output: $\mathcal{T}$ Realization of $\partial\mathcal{PP}(\lambda)$

$k \leftarrow 1, \mathcal{S} \leftarrow \emptyset, \mathcal{T} \leftarrow \emptyset$
Repeat:
  Sample $s_k \in \mathbb{R} \sim \mathcal{PP}(\bar{\lambda} \mid \mathcal{S})$
  Sample $d_k \in [0,1] \sim \text{Gumbel\_softmax}_\tau \left(\frac{\lambda(\, s_k | \mathcal{T}\,)}{\bar{\lambda}}\right)$
  $\mathcal{T} \leftarrow \mathcal{T} \cup \{(s_k, d_k)\}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s_k\}$
Return $\mathcal{T}$

Base Poisson point process


Conditional intensity function


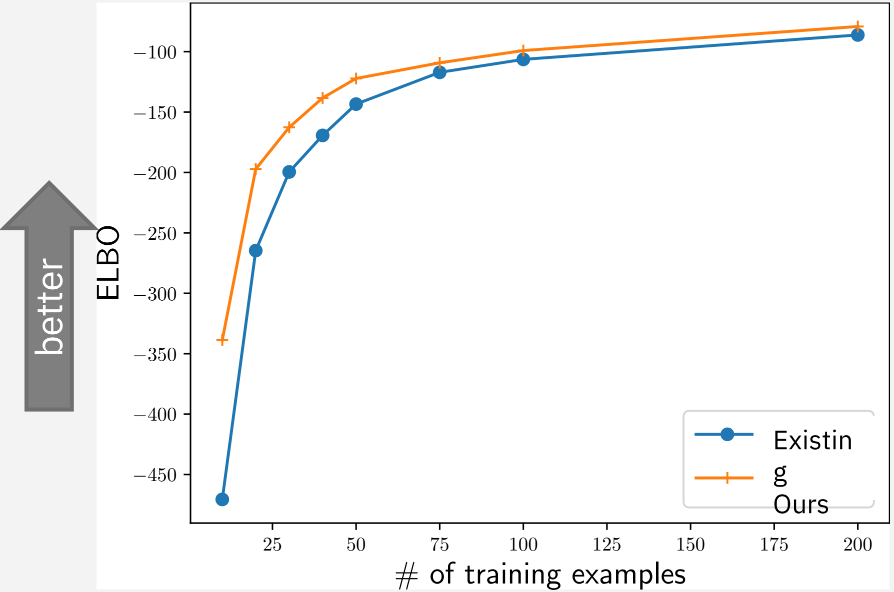
Output of our sampling algo.


Empirical studies

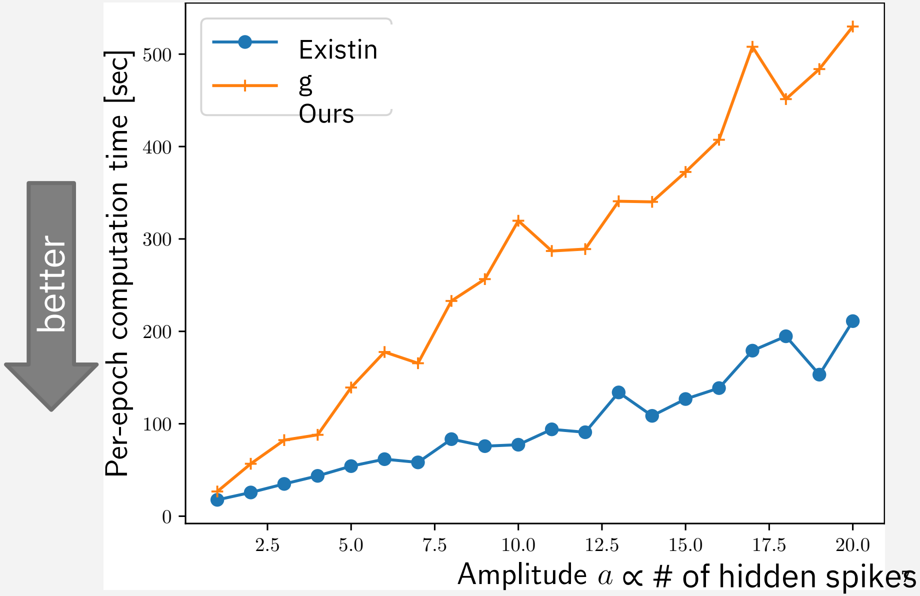
1. Standard deviations of gradient estimators

66.3 (Ours) vs. 2,490 (Existing)

2. Predictive performance



3. Computation time



Summary

Our contributions

- A differentiable point process
- A better learning algorithm of probabilistic SNNs

Findings (vs. the existing work)

- Our gradient estimator has smaller variance
- Smaller var. leads to the better predictive performance
- 2.8x computational overhead

Code is available under MIT license!

https://github.com/ibm-research-tokyo/diffsnn