SOLSA: Neuromorphic Spatiotemporal Online Learning for Synaptic Adaptation

Zhenhang Zhang, Jingang Jin,
Haowen Fang, Qinru Qiu

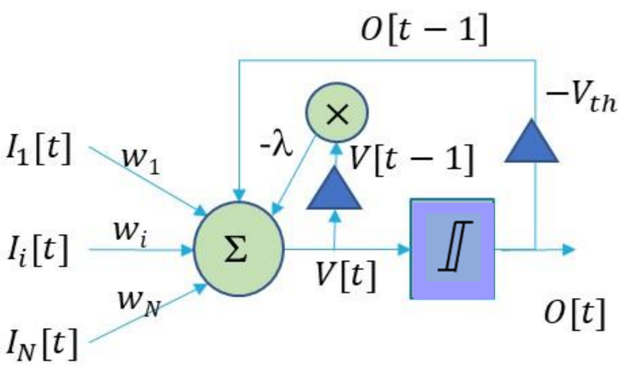
Syracuse University, USA

Outline

- Introduction
- Proposed Method
- Experiments
- Conclusion

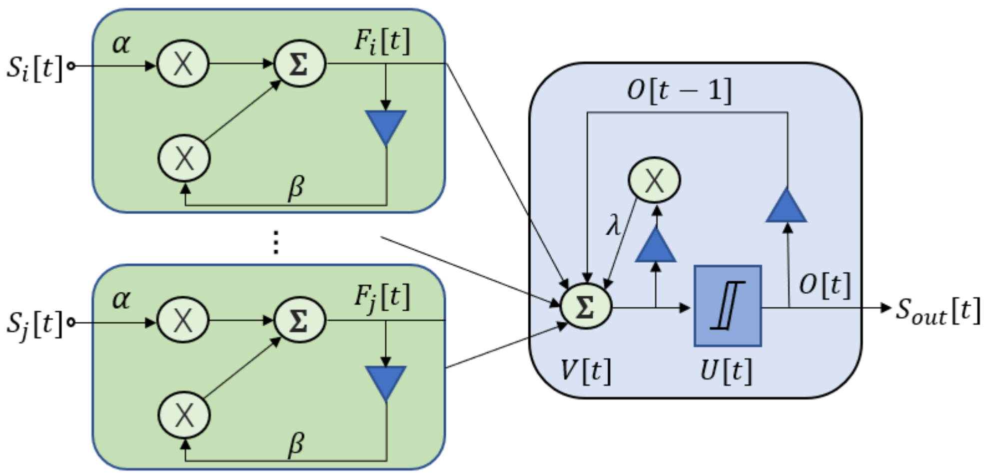
Basics of Spiking Neural Networks

- Spiking Neural Network (SNN) is inspired by biological system.
  - Unlike ANN, which is a highly abstracted model, SNN incorporates more biological aspects.
- Neurons communicate through event-triggered and asynchronous spikes
  - Input spikes causes membrane potential to charge
  - Output spikes are generated if membrane potential exceeds a threshold
  - Event driven operation brings energy efficiency
- Leaky Integrate and Fire (LIF) neuron has exponential decay
  - Membrane potential:
  - $V_i^l[t] = \lambda V_i^l[t-1] + \sum_j^{N_{l-1}} w_{i,j}^l O_j^{l-1}[t] - V_{th} O_i^l[t-1]$
  - Output: $O_i^l[t] = U\left(V_i^l[t] - V_{th}\right)$
  - Heaviside activation: $U(x) = 0, x < 0 \text{ otherwise } 1$



A More Realistic Neuron Model

- The computation power of biological neuron is much richer than IF or LIF model
- Dendritic trees in individual neuron have the capacity to perform computations. A more realistic model of a synapse is a low-pass filter
- LIF neuron with post-synaptic potential (PSP)
  - Synapse filter: $F_{ij}^{l}[t] = \beta F_{ij}^{l}[t-1] + \alpha O_{ij}^{l-1}[t-1]$
  - Membrane potential: $V_i^l[t] = \lambda V_i^l[t-1] + \sum_j^{N_{l-1}} w_{i,j}^l F_j^l[t] - V_{th} O_i^l[t-1]$

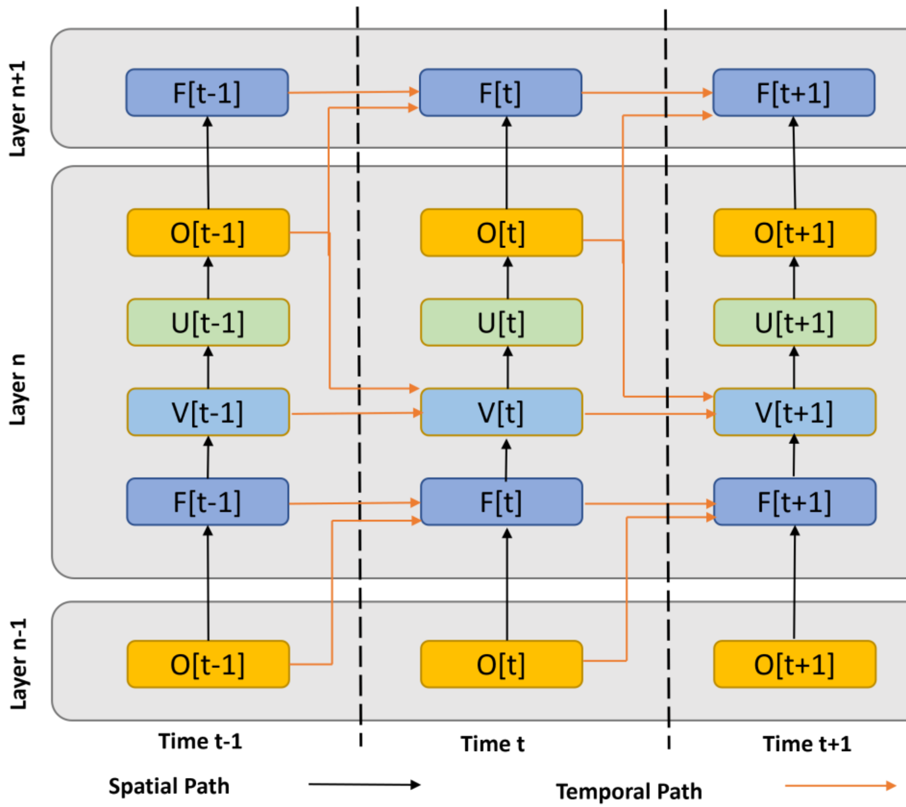


SNN Training Algorithms

- Hebbian Learning
  - Increase the synaptic efficacy when a presynaptic neuron repeatedly and persistently stimulates a postsynaptic neuron.
  - Vanilla Hebbian Learning is slow and inefficient
- Backpropagation Through Time(BPTT)
  - Train SNN as a Recurrent Neural Network(RNN)
  - Needs to unroll the network along the temporal axis
    - High memory requirement, high latency and computation complexity
    - Not suitable for edge devices and online learning
  - Variant: Truncated BPTT
    - Ignores historical temporal information causes performance degradation
- Three-Factor Hebbian Learning
  - E-prop(Bellec et. al. 2020), OTTT (Xiao et. al. 2022)
  - Relies on presynaptic and postsynaptic activities, and neuromodulated global error signal
  - Does not require network unrolling
  - Considers only simplified neuron model

Temporal and Spatial Path in the SNN

- In forward pass, data propagate from lower left corner to upper right corner in the data graph
  - Spatial propagation (bottom to top): From input layer to output layer
  - Temporal propagation (left to right): From the past to the future time



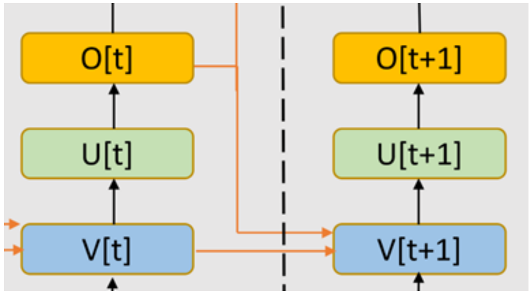
SOLSA Learning Rule

- E as the error(the difference between the actual output and the expected output)
- The gradient with the respect to the weight

$$\frac{dE}{dw_{ij}^l} = \sum_t \frac{dE}{dV_i^l[t]} \frac{\partial V_i^l[t]}{\partial w_{ij}^l} \longrightarrow F_{ij}^l[t]$$

- First term: impact of the membrane potential on the error

$$\frac{dE}{dV_i^l[t]} = \underbrace{\frac{dE}{dO_i^l[t]} \frac{\partial O_i^l[t]}{\partial V_i^l[t]}}_{\text{Spatial Path}} + \underbrace{\frac{dE}{dV_i^l[t+1]} \frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}}_{\text{Temporal Path}}$$



Spatial Gradient

- Spatial path gradient is approximated by vertical back propagation

$$\frac{dE}{dO_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]} \approx \sum_k \frac{\partial E[t]}{\partial O_k^{l+1}[t]}\frac{\partial O_k^{l+1}[t]}{\partial V_k^{l+1}[t]}\frac{\partial V_k^{l+1}[t]}{\partial F_{ik}^{l+1}[t]}\frac{\partial F_{ik}^{l+1}[t]}{\partial O_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]} = \mu_i^l[t]$$

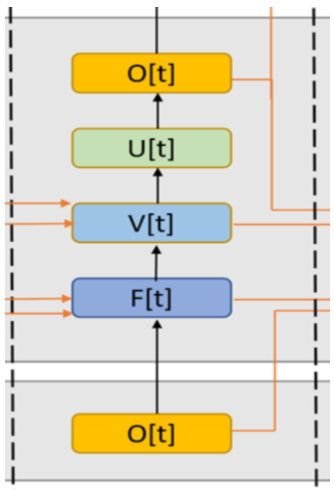
w_{ki}^{l+1} β_{ki}^{l+1} $\epsilon_i^{l(+1)}[t]$: Surrogate gradient for Heaviside activation

- Surrogate gradient function for Heaviside activation

$$P(V + z > V_{th}) = \frac{1}{2}\,\mathrm{erfc}\left(\frac{V_{th} - V}{\sqrt{2}\sigma}\right)$$

  - Surrogate gradient

$$\epsilon_i^l[t] \approx \frac{dP\left(V_i^l[t] + z > V_{th}\right)}{dV_i^l[t]}$$



Spatial and Temporal Gradient

- Combining the spatial and temporal gradient

Spatial gradient Temporal gradient

$$\frac{dE}{dV_i^l[t]} = \underbrace{\frac{dE}{dO_i^l[t]}\frac{\partial O_i^l[t]}{\partial V_i^l[t]}}_{\mu_i^l[t]} + \frac{dE}{dV_i^l[t+1]}\underbrace{\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}}_{\text{Membrane potential change rate}}$$

Recursive decomposition

$$= \mu_i^l[t] + \left(\mu_i^l[t+1] + \frac{dE}{dV_i^l[t+2]}\frac{\partial V_i^l[t+2]}{\partial V_i^l[t+1]}\right)\frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}$$

Rearrange

$$= \sum_{t \le t' \le T} \mu_i^l[t'] \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]} \frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]} \cdots \frac{\partial V_i^l[t+1]}{\partial V_i^l[t]}$$

The gradient needs information on time step *t* and all the future time steps

Calculate Gradient in Forward Pass

- Revisit the gradient with the respect to the weight

$$\frac{dE}{dw_{ij}^l} = \sum_t \frac{dE}{dV_i^l[t]} \frac{\partial V_i^l[t]}{\partial w_{ij}^l}$$

$$= \sum_t \sum_{t \le t' \le T} \mu_i^l[t'] \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]} \frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]} \cdots \frac{\partial V_i^l[t+1]}{\partial V_i^l[t]} F_{ij}^l[t]$$

Switch the order of summation

$$= \sum_{t' < T} \mu_i^l[t'] \underbrace{\sum_{t \le t'} \frac{\partial V_i^l[t']}{\partial V_i^l[t'-1]} \frac{\partial V_i^l[t'-1]}{\partial V_i^l[t'-2]} \cdots \frac{\partial V_i^l[t+1]}{\partial V_i^l[t]} F_{ij}^l[t]}_{\varepsilon_{i,j}^l[t'] \text{ (eligibility trace)}}$$

For every time step t', the gradient calculation only needs historical information in $t \le t'$

Proposed Method: Gradient

- $\varepsilon_{i,j}^{l}$ can be updated incrementally

$$\varepsilon_{i,j}^{l}[t'] = \frac{\partial V_j^l[t']}{\partial V_j^l[t'-1]} \varepsilon_{i,j}^{l}[t'-1] + F_{i,j}^{l}[t']$$

$$= \left(\lambda - v_{th}\epsilon_i^l[t']\right)\varepsilon_{i,j}^{l}[t'-1] + F_{i,j}^{l}[t']$$

- SOLSA update rule

$$\frac{dE}{dw_{ij}^{l}} = \sum_{t'} \mu_i^l[t'] \cdot \varepsilon_{i,j}^{l}[t']$$

  - The weight change relies only on information from current and last time step
  - No need to record historical neuron activities

Adaptive Synapse Filter Kernel

- Parameters α and β of synapse filter has significant impact on the learning performance
- Adapt the value of α and β by accumulating gradients over time
  - Calculate instantaneous gradient in each time step t

$$\nabla_{\alpha_{ij}^l}[t] = \frac{\partial E[t]}{\partial \alpha_{ij}^l} = \mu_i^l[t] \frac{\partial V_i^l[t]}{\partial F_{ij}^l[t]} \frac{\partial F_i^l[t]}{\partial \alpha_{ij}^l} = \mu_i^l[t] \cdot w_{ij}^l \cdot F_i^l[t-1],$$

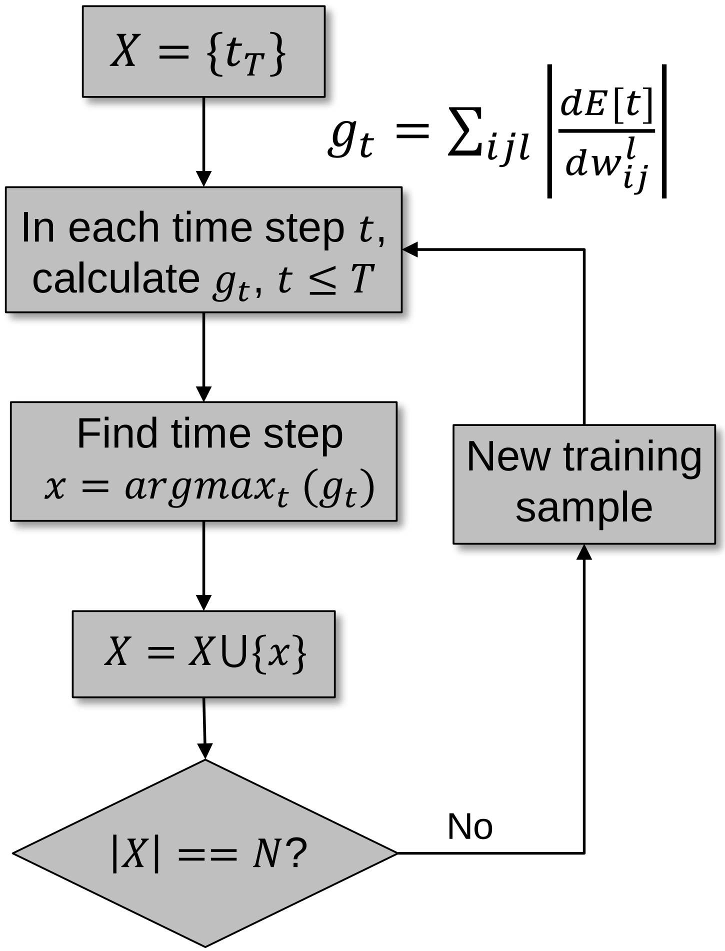
$$\nabla_{\beta_{ij}^l}[t] = \frac{\partial E[t]}{\partial \beta_{ij}^l} = \mu_i^l[t] \frac{\partial V_i^l[t]}{\partial F_{ij}^l[t]} \frac{\partial F_i^l[t]}{\partial \beta_{ij}^l} = \mu_i^l[t] \cdot w_{ij}^l \cdot O_j^{l-1}[t-1].$$

  - Accumulate the impact of the gradient over time with an assumption that later gradients are more important

$$\frac{dE}{d\alpha_{ij}^l} = \sum\nolimits_t \nabla_{\alpha_{ij}^l}[t] \cdot \frac{1-\gamma^{t+1}}{1-\gamma} \qquad \frac{dE}{d\beta_{ij}^l} = \sum\nolimits_t \nabla_{\beta_{ij}^l}[t] \cdot \frac{1-\gamma^{t+1}}{1-\gamma}$$

Scheduled Weight Update

- Every time step t, SOLSA calculate the partial gradient
  - $\frac{dE}{dw_{ij}^l}[t] = \sum_{t' \le t} \mu_i^l[t'] \cdot \varepsilon_{i,j}^l[t']$
- Rationale: adjusts network parameters multiple times using the partial gradient can accelerate the training speed
  - Frequent adjustment at each time step introduces noise due to local variance
  - Selecting the right time to update is crucial
- Scheduled weight update selects the right time to update the network
  - The total number of update points is a hyperparameter



Early-Stop Training

- Rationale: data outside the signature pattern are noise to the training process
  - Stop the training process on the given input sequence if the model can already make correct prediction consistently and robustly
- Monitor the accuracy at each update point t:
  - $A(t) = (\sum_{i=0}^{t} O_y[i]) / (\sum_k \sum_{i=0}^{t} O_k[i])$
- Increase counter if $A(t)$ surpasses a predefined threshold
- Stop training a sample when counter exceeds (or equals) 50% of update points

Experiment Setup: Datasets

- All datasets are processed by fully connected network

Dataset Name		Input size	Sequence length	Input format	SNN architecture
Regular	EMG gesture	8	100	Spikes	8-150-150-7
	Finger mov.	28	50	Current	28-100-100-2
	Basic motion	6	100	Current	6-100-100-4
	Epilepsy	3	207	Current	3-100-100-4
	Jap. Vowel	12	29	Current	12-100-100-9
	RacketSports	6	30	Current	6-100-100-4
	DVS128	4096	100	Spikes	4096-100-100-11
Long	Self reg. scp	6	896	Current	6-100-100-2
	EMG action	8	1000	Current	8-200-200-10

Experiment Setup: Baselines

- BPTT and Truncated-BPTT(20 steps)
  - BPTT needs huge memory. T-BPTT performs unstably.
- E-prop(Bellec et. al. 2020)
  - E-prop ignores synapse filter and update at the end.
- Online Training Through Time(Xiao et. Al. 2022)
  - No reset and synapse filter and update at every time step.
- Ablation Study variants

Feature	SOLSA	E-prop	OTTT	SOLSA variant 1	SOLSA variant 2	SOLSA variant 3
Synapse filter	✓			✓	✓	✓
Adaptive weight	✓	✓	✓	✓	✓	✓
Adaptive kernel	✓				✓	
Impact of reset	✓	✓		✓	✓	✓
Scheduled updates	✓			✓	✓	✓
Early stop	✓					✓

Experiment Results

- Comparison with Baselines
  - SOLSA achieves a 5% higher average accuracy with a 72% reduction in memory cost, compared with BPTT
  - T-BPTT(20 steps) performs much worse
  - On average SOLSA outperform E-prop by 30% and OTTT by 66.5%

Dataset	BPTT based		Three factor Hebbian			Memory usage (MB)	
	BPTT	TBPTT	SOLSA	E-prop	OTTT	BPTT	SOLSA
EMG gesture	0.956	0.664	**0.985**	0.675	0.672	13.4	6.6
Finger mov.	0.58	0.56	**0.64**	0.58	0.59	9.0	7.1
Basic motion	1	0.25	**1**	0.925	1	13.4	7.5
Epilepsy	0.941	0.676	**0.971**	0.816	0.904	22.7	9.9
Jap. Vowel	0.926	0.951	**0.981**	0.944	0.969	7.1	6.6
RacketSports	0.809	0.769	**0.907**	0.388	0.796	7.4	6.2
DVS128	0.959	0.6	**0.979**	0.819	0.875[a]	105.8	67.2
Self reg. scp	0.836	0.866	**0.897**	0.876	0.89	79.6	22.2
EMG action	0.973	0.696	**0.979**	0.77	0.161	158.4	44.1

Experiment Results: Ablation Study

- Compare variant 1 with its unscheduled version (only update at the end of the sequence), scheduled updated gives ~20% improvement
- Comparing SOLSA with variant 3, adaptive filter gives ~5% improvement
- Comparing SOLSA with variant 2, early-stop training gives ~17% improvement

Dataset	Accuracy				
	Unscheduled	variant 1	variant 2	variant 3	SOLSA
EMG gesture	0.912	0.942	0.671	0.957	**0.985**
Finger mov.	0.56	**0.65**	0.59	0.58	0.64
Basic motion	0.95	1	1	1	**1**
Epilepsy	0.794	0.934	0.713	0.958	**0.971**
Jap. Vowel	0.869	0.975	0.619	0.96	**0.981**
RacketSports	0.598	0.855	0.901	0.835	**0.907**
DVS128	0.895	0.93	0.959	0.93	**0.979**
Self reg. scp	0.88	0.894	0.897	0.893	**0.897**
EMG action	0.134	0.93	0.946	0.848	**0.979**

Conclusion

- We designed an SNN online learning algorithm called SOLSA (Spatiotemporal Online Learning for Synaptic Adaptation)
- SOLSA can train SNNs with temporal filters on both synapses and neuron membrane potentials
- At each time step, SOLSA only relies on information on current time step and previous time step
- We further proposed unique training techniques such as scheduled weight update and early stop
- Experiment results showed that SOLSA requires less memory, and has better & more robust performance