Local Learning Rules for Spiking Neural Networks

Jafar Bakhshaliyev

Data Science Group
University of Hildesheim

Supervisor: Prof. Niels Landwehr

July 6, 2026

Outline

Overview

Overview: News

Recent news: Our paper has been accepted at ECML PKDD 2026.

SpikF-GO: Spiking Fourier Graph Operators for Multivariate Time Series Forecasting
Jafar Bakhshaliyev, Niels Landwehr

General research focus:

- Time series forecasting
- Image classification/segmentation
- Positional encodings
- Graph neural networks (with less focus)

I am happy to extend this direction further in the future.

Overview: Why Is Backpropagation a Problem?

Backprop solves credit assignment

$$\Delta W_l = -\eta\, \delta_l h_{l-1}^\top$$

$$\delta_l = \left(W_{l+1}^\top \delta_{l+1}\right) \odot f'(a_l)$$

It computes how each synapse should change to reduce the final error.

But strict backprop needs

$$\text{forward activity} + \text{backward error variables}$$

$$\text{layer } L \to \text{layer } L-1 \to \cdots \to \text{layer } 1$$

So learning depends on a separate backward error pathway.

Biological / implementation issues
1. Weight transport

$$W_{l+1}^\top$$

Feedback weights must match forward weights.
2. Signed error signals

$$\delta_l$$

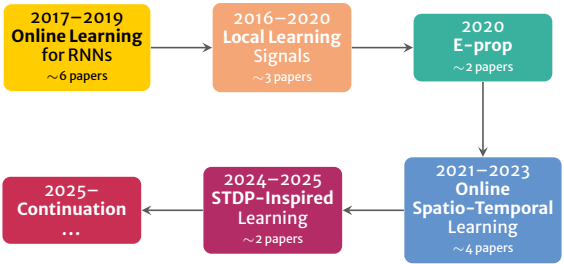
Precise positive/negative error vectors must be sent through layers.
3. Feedback changes activity In the brain, feedback usually modulates neural activity, not only hidden error variables.

Local-learning direction

$$W_{l+1}^\top \delta_{l+1} \quad \longrightarrow \quad \text{local trace} \times \text{approx. learning signal}$$

Key message: We do not abandon credit assignment; we approximate it with biologically and hardware-plausible local signals.

Overview: Literature



Online Learning for RNNs

Recurrent Neural Networks [1]

Recurrent state update

$$\mathbf{a}^{(t)} = F_{\mathbf{w}}\left(\mathbf{a}^{(t-1)}, \mathbf{x}^{(t)}\right)$$

Output/readout

$$\mathbf{y}^{(t)} = F_{\mathbf{w}_o}^{\text{out}}\left(\mathbf{a}^{(t)}\right)$$

Instantaneous loss

$$L^{(t)} = L\left(\mathbf{y}^{(t)}, \mathbf{y}^{*(t)}\right)$$

Total objective

$$\mathcal{L} = \sum_t L^{(t)}$$

Model maps

$$F_{\mathbf{w}} : \mathbb{R}^m \to \mathbb{R}^n$$
$$F_{\mathbf{w}_o}^{\text{out}} : \mathbb{R}^n \to \mathbb{R}^{n_{\text{out}}}$$

$$\mathbf{w} \in \mathbb{R}^P, \qquad \mathbf{w}_o \in \mathbb{R}^{P_o}$$

Easy / natural online:

$$\frac{\partial L^{(t)}}{\partial \mathbf{w}_o}$$

only needs information at time t.

Heart of the problem:

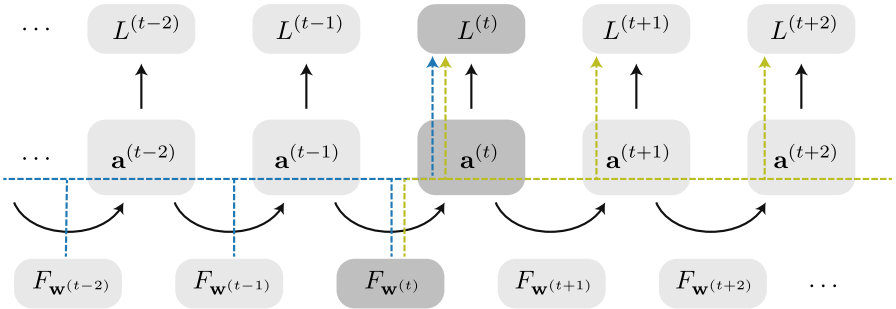
$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$$

because recurrent weights affect future states and losses.

Past-Facing vs. Future-Facing Learning [1]

Past-facing: $\frac{\partial L^{(t)}}{\partial \mathbf{w}} = \sum_{t' \leq t} \frac{\partial L^{(t)}}{\partial \mathbf{w}^{(t')}}$

Future-facing: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(t)}} = \sum_{t' \geq t} \frac{\partial L^{(t')}}{\partial \mathbf{w}^{(t)}}$



Past-facing: current loss receives influence from past weight applications.
Future-facing: current weight application influences future losses.

Real-Time Recurrent Learning (RTRL) [1]

Global recurrent objective

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \sum_t \sum_{s \leq t} \frac{\partial L^{(t)}}{\partial \mathbf{w}^{(s)}} \tag{1}$$

If t is real time: past-facing gradient

$$\nabla_{\mathbf{w}} \mathcal{L}(t) = \sum_{s=0}^{t} \frac{\partial L^{(t)}}{\partial \mathbf{w}^{(s)}} = \frac{\partial L^{(t)}}{\partial \mathbf{w}} \tag{2}$$

Influence matrix

$$M_{kp}^{(t)} = \frac{\partial a_k^{(t)}}{\partial w_p} \tag{3}$$

RTRL influence update

$$\mathbf{M}^{(t)} = \frac{\partial \mathbf{a}^{(t)}}{\partial \mathbf{w}} = \underbrace{\mathbf{J}^{(t)} \mathbf{M}^{(t-1)}}_{\text{past influence}} + \underbrace{\overline{\mathbf{M}}^{(t)}}_{\text{immediate influence}} \tag{4}$$

Dimensions

$$\mathbf{a}^{(t)} \in \mathbb{R}^n$$

$$\mathbf{w} \in \mathbb{R}^P$$

$$\mathbf{M}^{(t)} \in \mathbb{R}^{n \times P}$$

$$\mathbf{J}^{(t)} \in \mathbb{R}^{n \times n}$$

$$\overline{\mathbf{c}}^{(t)} \in \mathbb{R}^n$$

Gradient from current loss

$$\frac{\partial L^{(t)}}{\partial \mathbf{w}} = \underbrace{\frac{\partial L^{(t)}}{\partial \mathbf{a}^{(t)}}}_{\overline{\mathbf{c}}^{(t)}} \underbrace{\frac{\partial \mathbf{a}^{(t)}}{\partial \mathbf{w}}}_{\mathbf{M}^{(t)}} = \overline{\mathbf{c}}^{(t)} \mathbf{M}^{(t)} \tag{5}$$

RTRL: exact online gradient, but expensive.

Memory: $\mathcal{O}(n^3) - \mathbf{M}^{(t)}$ Computation: $\mathcal{O}(n^4) - \mathbf{J}^{(t)} \mathbf{M}^{(t)}$

RTRL Approximations [1]

Continuous-time vanilla RNN

$$\mathbf{a}^{(t)} = (1-\alpha)\mathbf{a}^{(t-1)} + \alpha\phi\Big(\mathbf{W}\hat{\mathbf{a}}^{(t-1)}\Big), \qquad \hat{\mathbf{a}}^{(t-1)} = \operatorname{concat}\Big(\mathbf{a}^{(t-1)}, \mathbf{x}^{(t)}, 1\Big)$$

$$\mathbf{W} \in \mathbb{R}^{n\times m}, \qquad \phi : \mathbb{R}^n \to \mathbb{R}^n, \qquad \alpha \in (0, 1]$$

Influence tensor

$$M^{(t)}_{kij} = \frac{\partial a^{(t)}_k}{\partial W_{ij}}$$

It measures how the k-th hidden unit changes when perturbing the connection from input/source j to unit i.

Immediate influence

$$\overline{M}^{(t)}_{kij} = \frac{\partial a^{(t)}_k}{\partial W^{(t)}_{ij}} = \alpha\delta_{ki}\phi'\left(h^{(t)}_i\right)\hat{a}^{(t-1)}_j$$

Only nonzero when $k = i$: W_{ij} directly affects unit i.

Compressing $M^{(t)}_{kij}$

$$M^{(t)}_{kij} \approx \begin{cases} A^{(t)}_k B^{(t)}_{ij} & \text{UORO} \\ A^{(t)}_j B^{(t)}_{ki} & \text{KF-RTRL} \\ A^{(t)}_i B^{(t)}_{kj} & \text{Reverse KF-RTRL} \\ A^{(t)}_{ki} B^{(t)}_{ij} & \text{KeRNL/RFLO} \\ A^{(t)}_{kj} B^{(t)}_{ij} & \text{Unexplored} \\ A^{(t)}_{ki} B^{(t)}_{kj} & \text{Unexplored} \end{cases}$$

Goal: replace the expensive RTRL tensor with lower-order factors.

UORO: Unbiased Online Recurrent Optimization [1]

Approximation: sensitivity $\times$ efficacy

$$M_{kij}^{(t)} \approx A_k^{(t)} B_{ij}^{(t)}$$

$$A_k^{(t)} B_{ij}^{(t)} = \left(\rho_0 \sum_{k'} J_{kk'}^{(t)} A_{k'}^{(t-1)} + \rho_1 \nu_k \right) \left(\rho_0^{-1} B_{ij}^{(t-1)} + \rho_1^{-1} \sum_{k'} \nu_{k'} \overline{M}_{k'ij}^{(t)} \right) \tag{6}$$

$$\Rightarrow \mathbb{E}\left[A_k^{(t)} B_{ij}^{(t)} \right] = \sum_{k'} J_{kk'}^{(t)} \mathbb{E}\left[A_{k'}^{(t-1)} B_{ij}^{(t-1)} \right] + \overline{M}_{kij}^{(t)} = M_{kij}^{(t)} \tag{7}$$

Properties:

$$A^{(t)} \in \mathbb{R}^n, \qquad B^{(t)} \in \mathbb{R}^{n \times m} \qquad \Rightarrow \qquad \text{Memory } \mathcal{O}(n^2)$$

$$\sum_{k'} J_{kk'}^{(t)} A_{k'}^{(t-1)} \qquad \Rightarrow \qquad \text{Time } \mathcal{O}(n^2)$$

Stochastic due to random vector ν and closed-form update.

KF-RTRL: Kronecker-Factored Approximation [1]

More natural factorization

$$M_{kij}^{(t)} \approx A_j^{(t)} B_{ki}^{(t)}$$

$$\overline{M}_{kij}^{(t)} = \alpha \delta_{ki} \phi'(h_i^{(t)}) \hat{a}_j^{(t-1)} = D_{ki}^{(t)} \hat{a}_j^{(t-1)}$$

Update equations

$$A_j^{(t)} = \nu_0 \rho_0 A_j^{(t-1)} + \nu_1 \rho_1 \hat{a}_j^{(t-1)} \tag{8}$$

$$B_{ki}^{(t)} = \nu_0 \rho_0^{-1} \sum_{k'} J_{kk'}^{(t)} B_{k'i}^{(t-1)} + \nu_1 \rho_1^{-1} \alpha \delta_{ki} \phi'(h_i^{(t)}) \tag{9}$$

Properties:

$$A^{(t)} \in \mathbb{R}^m, \qquad B^{(t)} \in \mathbb{R}^{n \times n} \qquad \Rightarrow \qquad \text{Memory } \mathcal{O}(n^2)$$

$$\sum_{k'} J_{kk'}^{(t)} B_{k'i}^{(t-1)} \qquad \Rightarrow \qquad \text{Time } \mathcal{O}(n^3)$$

Stochastic due to random variables ν_0, ν_1, but closed-form update.

KeRNL: Eligibility Traces and Sensitivity [1]

Approximation

$$M_{kij}^{(t)} \approx A_{ki} B_{ij}^{(t)}$$

Eligibility trace

$$B_{ij}^{(t)} = (1-\alpha_i) B_{ij}^{(t-1)} + \alpha \phi'(h_i^{(t)}) \hat{a}_j^{(t-1)} \tag{10}$$

Sensitivity matrix

$$\frac{\partial a_k^{(t)}}{\partial a_i^{(t')}} \approx A_{ki} (1-\alpha_i)^{(t-t')} \tag{11}$$

$$A_{ki}(1-\alpha_i) \approx \sum_{k'} J_{kk'}^{(t)} A_{k'i} \tag{12}$$

For $t = t'$:

$$A_{ki} \approx \begin{cases} \delta_{ki} \\ (1-\alpha_i)^{-1} \sum_{k'} J_{kk'}^{(t)} A_{k'i} \end{cases} \tag{13}$$

$$A_{ki} B_{ij}^{(t)} \approx \sum_{k'} J_{kk'}^{(t)} A_{k'i} B_{ij}^{(t-1)} + \alpha \delta_{ki} \phi'(h_i^{(t)}) \hat{a}_j^{(t-1)} = M_{kij}^{(t)} \tag{14}$$

A_{ki} and α_i are learned by gradient descent. The $B^{(t)}$ update is closed-form, while KeRNL is deterministic and numerical due to SGD.

Summary of Past-Facing Algorithms [1]

Algorithm	Facing	Tensor	Update	Memory	Time
RTRL	Past	M_{kij}	Deterministic, closed-form	$\mathcal{O}(n^3)$	$\mathcal{O}(n^4)$
UORO	Past	$A_k B_{ij}$	Stochastic, closed-form	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
KF-RTRL	Past	$A_j B_{ki}$	Stochastic, closed-form	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
R-KF-RTRL	Past	$A_i B_{kj}$	Stochastic, closed-form	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
r-OK	Past	$\sum_{l=1}^{r} A_{lj} B_{lki}$	Stochastic, numerical	$\mathcal{O}(rn^2)$	$\mathcal{O}(rn^3)$
KeRNL	Past	$A_{ki} B_{ij}$	Deterministic, numerical	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
RFLO	Past	$\delta_{ki} B_{ij}$	Deterministic, closed-form	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$

Future-Facing View [1]

If s is real time: future-facing gradient

$$\nabla_{\mathbf{w}} \mathcal{L}(s) = \sum_{t=s}^{\infty} \frac{\partial L^{(t)}}{\partial \mathbf{w}^{(s)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(s)}} \tag{15}$$

Credit assignment vector

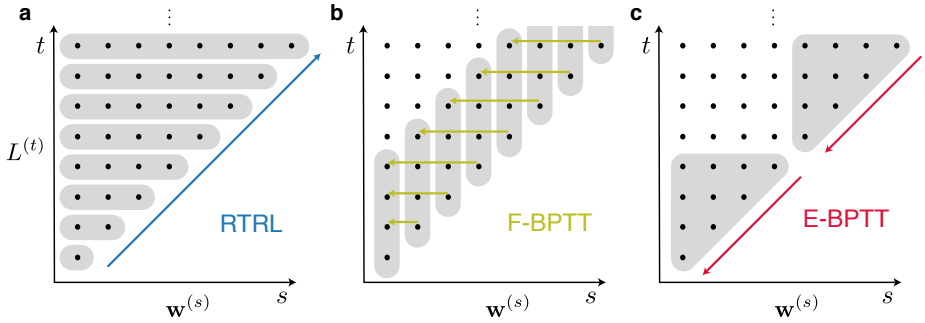
$$\mathbf{c}^{(t)} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(t)}} = \underbrace{\frac{\partial L^{(t)}}{\partial \mathbf{a}^{(t)}}}_{\bar{\mathbf{c}}^{(t)}} + \mathbf{c}^{(t+1)} \mathbf{J}^{(t+1)} \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{(t)}} \frac{\partial \mathbf{a}^{(t)}}{\partial \mathbf{w}^{(t)}} = \mathbf{c}^{(t)} \overline{\mathbf{M}}^{(t)} \tag{17}$$

Trade-off: Past-facing can truly run online because it follows the forward-time direction. Memory requirements favor future-facing, because $\mathcal{L}$ is scalar while $\mathbf{w}$ is high-dimensional.

Efficient PF: compress $\mathbf{M}^{(t)}$ Efficient FF: predict $\mathbf{c}^{(t+1)}$

Future Facing Algorithms [1]



DNI: Decoupled Neural Interfaces [1]

Linear prediction of credit assignment

$$c_i^{(t)} \approx \sum_l \tilde{a}_l^{(t)} A_{li}, \qquad \tilde{\mathbf{a}}^{(t)} = \operatorname{concat}\left(\mathbf{a}^{(t)}, \mathbf{y}^{*(t)}, 1\right)$$

$$L_{\mathrm{SG}}^{(t)} = \frac{1}{2} \left\| \sum_l \tilde{a}_l^{(t)} A_{li} - c_i^{(t)} \right\|^2 \tag{18}$$

But this begs the question: how do we get $c^{(t)}$ without future information?

$$\mathbf{c}^{(t)} = \overline{\mathbf{c}}^{(t)} + \mathbf{c}^{(t+1)} \mathbf{J}^{(t+1)} \tag{19}$$

$$\Delta A_{li} \propto -\tilde{a}_l^{(t)} \left[\sum_{l'} \tilde{a}_{l'}^{(t)} A_{l'i} - \left(\overline{c}_i^{(t)} + \sum_m \sum_{l'} \tilde{a}_{l'}^{(t+1)} A_{l'm} J_{mi}^{(t+1)} \right) \right] \tag{20}$$

DNI: future-facing, deterministic, numerical update.

Summary of Future-Facing Algorithms [1]

Algorithm	Facing	Tensor	Update	Memory	Time
E-BPTT	–	–	Deterministic, closed-form	$\mathcal{O}(nT)$	$\mathcal{O}(n^2)$
F-BPTT	Future	–	Deterministic, closed-form	$\mathcal{O}(nT)$	$\mathcal{O}(n^2T)$
DNI	Future	A_{li}	Deterministic, numerical	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$

Local Learning Signals

Local Learning Signals [2]

Goal in this part: We do not focus on fully training SNNs or RNNs here. Instead, we study local learning signals that can be used when a prediction module needs credit assignment.

Backpropagation gives strong gradients, but introduces two non-local constraints:

$$\delta \mathbf{y}_k = \mathbf{W}_{k+1}^T \delta \mathbf{z}_{k+1}$$

- **Weight symmetry / transport:** the backward pathway needs access to forward weights.
- **Update locking:** layers must wait until the forward and backward passes are completed.
- This requires storing activations and moving gradient information backward.

Local-learning direction: replace exact backward transport with simpler learning signals:

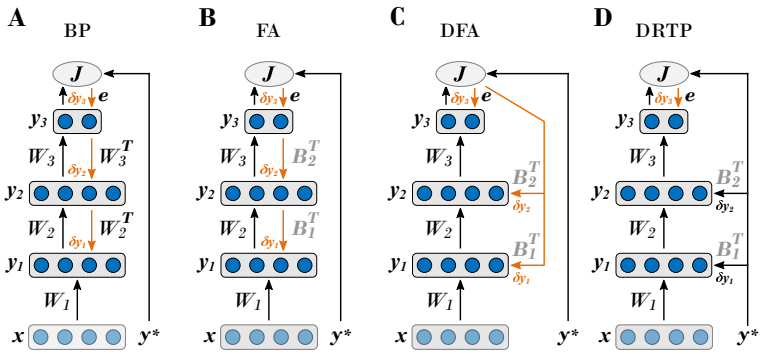
$$\mathbf{W}_{k+1}^T \delta \mathbf{z}_{k+1} \quad \longrightarrow \quad \mathbf{B}_k^T \mathbf{e} \quad \text{or} \quad \mathbf{B}_k^T \mathbf{y}^*$$

Key idea: fixed random feedback or target projections can provide useful layerwise teaching signals, without exact weight symmetry.

Local Learning Signals [2]

Layer equations

$$z_k = W_k y_{k-1} + b_k$$
$$y_k = f_k(z_k)$$
$$\delta z_k = \delta y_k \circ f_k'(z_k)$$



	A BP	B FA	C DFA	D DRTP
δy_k	$\frac{\partial J}{\partial y_k} = W_{k+1}^T \delta z_{k+1}$	$B_k^T \delta z_{k+1}$	$B_k^T e$	$B_k^T y^*$
Weight-transport-free	×	✓	✓	✓
Update-unlocked	×	×	×	✓

Local Learning Signals [2]

Backpropagation is powerful, but not local:

- Requires symmetric forward/backward weights
- Requires backward error propagation
- Layers are update-locked until the backward pass

Feedback alignment relaxes weight symmetry:

$$\mathbf{W}_{k+1}^{T}\delta\mathbf{z}_{k+1} \quad \longrightarrow \quad \mathbf{B}_{k}^{T}\mathbf{e}$$

DRTP goes further:

$$\mathbf{B}_{k}^{T}\mathbf{e} \quad \longrightarrow \quad \mathbf{B}_{k}^{T}\mathbf{y}^{*}$$

Key message: Useful local teaching signals can be generated without exact feedback weights and without waiting for full backpropagation.

E-prop

E-prop: The Learning Dilemma [3]

Target setting: recurrent spiking neural networks (RSNNs)

A synapse W_{ji} at time t can influence:

future spikes, future hidden states, and future losses.

Standard solution: BPTT

$$t = 1, 2, \ldots, T$$

unroll the RSNN through time, then propagate gradients backward through the unrolled graph.

Why this is problematic

store all intermediate states

backward-in-time gradient propagation

So the paper formulates the **learning dilemma**:

BPTT works well, but it is not brain-like or hardware-friendly.

E-prop idea: instead of sending gradients backward through time, each synapse keeps a forward-in-time **eligibility trace**, and a top-down **learning signal** gates the update.

E-prop: Eligibility Trace + Learning Signal [3]

Central decomposition

$$\frac{dE}{dW_{ji}} = \sum_t L_j^t \, e_{ji}^t$$

$$L_j^t = \frac{dE}{dz_j^t}, \qquad e_{ji}^t = \left[\frac{dz_j^t}{dW_{ji}}\right]_{\text{local}}$$

Interpretation

$$\boxed{\text{weight update} = \text{learning signal} \times \text{eligibility trace}}$$

$$\Delta W_{ji}^t \propto -\, L_j^t e_{ji}^t$$

Eligibility trace e_{ji}^t

- local memory at the synapse
- depends on presynaptic activity, postsynaptic state, membrane/adaptation variables, spike history

Learning signal L_j^t

- but exact future credit is not available online

Supervised approximation

$$\tilde{L}_j^t = \sum_k B_{jk}(y_k^t - y_k^{*,t})$$

Thus, e-prop uses **local synaptic memory** + **approximate top-down feedback**.

E-prop: Online Algorithm and Complexity [3]

Forward dynamics

$$h_j^t = M(h_j^{t-1}, z^{t-1}, x^t, W_j), \qquad z_j^t = f(h_j^t)$$
$$y_k^t = \sum_j W_{kj}^{\text{out}} z_j^t, \qquad E^t = \ell(y^t, y^{*,t})$$

Eligibility vector recursion

$$\epsilon_{ji}^t = \frac{\partial h_j^t}{\partial h_j^{t-1}} \epsilon_{ji}^{t-1} + \frac{\partial h_j^t}{\partial W_{ji}}$$

$$e_{ji}^t = \frac{\partial z_j^t}{\partial h_j^t} \epsilon_{ji}^t \approx \psi_j^t \, \epsilon_{ji}^t$$

Online update

$$\Delta W_{ji}^t = -\eta \, \tilde{L}_j^t \, e_{ji}^t \qquad \Longrightarrow \qquad \Delta W_{ji} = -\eta \sum_t \tilde{L}_j^t e_{ji}^t$$

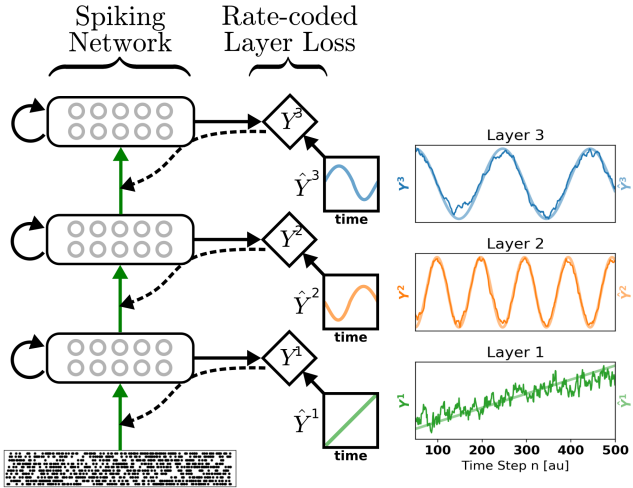
For a simple LIF neuron, this becomes the **three-factor rule**

$$\Delta W_{ji}^t = -\eta \underbrace{\tilde{L}_j^t}_{\text{top-down learning signal}} \underbrace{\psi_j^t}_{\text{postsynaptic sensitivity}} \underbrace{\bar{z}_i^t}_{\text{presynaptic trace}}$$

E-prop memory: $\mathcal{O}(S) \approx \mathcal{O}(n^2)$, E-prop per-step time: $\mathcal{O}(S) \approx \mathcal{O}(n^2)$

Key advantage: e-prop is online: memory does not grow with sequence length T.

Deep Continuous Local Learning (DECOLLE) [4]



Online Spatio-Temporal Learning

OSTL: Online Spatio-Temporal Learning [5]

Forward dynamics

$$s_l^t = W_l y_{l-1}^t + H_l y_l^{t-1} + d\, s_l^{t-1} \odot (1 - y_l^{t-1})$$

$$y_l^t = \Theta(s_l^t + b_l), \qquad \Psi_l^t = \frac{\partial y_l^t}{\partial s_l^t} \approx \mathrm{diag}(\psi_l^t)$$

Full BPTT gradient

$$E = \sum_t E^t(y_k^t, \hat{y}^t), \qquad \theta_l \in \{W_l, H_l, b_l\}$$

$$\frac{dE}{d\theta_l} = \sum_t \frac{\partial E^t}{\partial y_k^t} \left[\Psi_k^t \frac{ds_k^t}{d\theta_l} + \frac{\partial y_k^t}{\partial \theta_l} \right]$$

The difficult object is $\frac{ds_k^t}{d\theta_l}$: it contains all temporal and spatial paths from θ_l to the final-layer state.

OSTL: Online Spatio-Temporal Learning [5]

Same-layer temporal sensitivity

$$\epsilon_l^{t,\theta_l} = \frac{ds_l^t}{d\theta_l}$$

$$\frac{ds_l^t}{d\theta_l} = \sum_{\tau \leq t} \left(\prod_{r=\tau+1}^{t} \frac{ds_l^r}{ds_l^{r-1}} \right) \left[\frac{\partial s_l^\tau}{\partial \theta_l} + \frac{\partial s_l^\tau}{\partial y_l^{\tau-1}} \frac{\partial y_l^{\tau-1}}{\partial \theta_l} \right]$$

Forward recursion

$$\epsilon_l^{t,\theta_l} = \frac{ds_l^t}{ds_l^{t-1}} \epsilon_l^{t-1,\theta_l} + \left[\frac{\partial s_l^t}{\partial \theta_l} + \frac{\partial s_l^t}{\partial y_l^{t-1}} \frac{\partial y_l^{t-1}}{\partial \theta_l} \right]$$

Output eligibility

$$e_l^{t,\theta_l} = \Psi_l^t \epsilon_l^{t,\theta_l} + \frac{\partial y_l^t}{\partial \theta_l}$$

e_l^{t,θ_l} tells how θ_l affected the current output y_l^t through the past dynamics of layer l.

OSTL: Online Spatio-Temporal Learning [5]

Spatial learning signal

$$L_k^t = \frac{\partial E^t}{\partial y_k^t}$$

$$L_l^t = L_{l+1}^t \Psi_{l+1}^t W_{l+1}$$

$$\text{temporal credit} \rightarrow e_l^{t,\theta_l}, \qquad \text{spatial credit} \rightarrow L_l^t$$

OSTL gradient

$$\frac{dE}{d\theta_l} = \sum_t L_l^t e_l^{t,\theta_l} + R \qquad \Rightarrow \qquad \frac{dE}{d\theta_l} \approx \sum_t L_l^t e_l^{t,\theta_l}$$

Residual R

$$D_{m,l}^t = \frac{ds_m^t}{d\theta_l}, \qquad m > l$$

$$\theta_l \rightarrow y_l^\tau \rightarrow s_{l+1}^\tau \rightarrow s_{l+1}^{\tau+1} \rightarrow \cdots \rightarrow E^t$$

$R =$ mixed spatial-temporal credit across layers

For a single recurrent layer, there are no upper-layer recurrent states, so $R = 0$.

ETLP: Event-Based Three-Factor Local Plasticity [6]

Main idea: ETLP combines an e-prop-like temporal trace with a DRTP-like target projection, but adapts both to spiking, event-driven learning.

Temporal side

- Uses local spike traces instead of backpropagation through time.
- For ALIF neurons, the trace also includes threshold adaptation.
- The local synapse keeps: presynaptic activity, postsynaptic voltage sensitivity, and adaptation state.

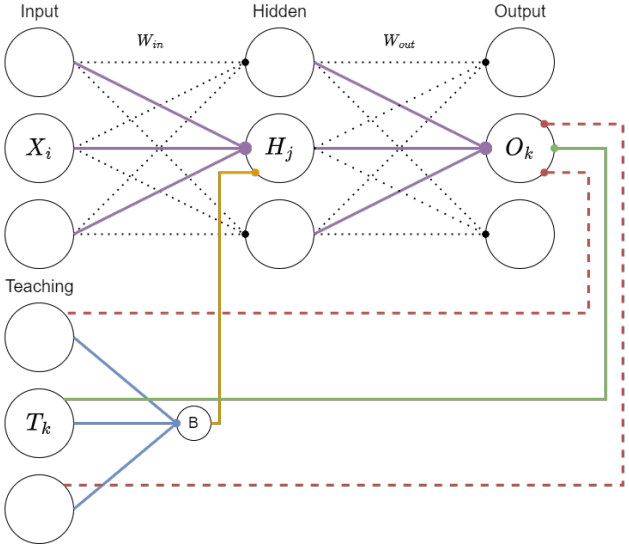
Spatial side

- No hidden-layer error backpropagation.
- Labels are represented as teaching neurons.
- Fixed random projections send teaching spikes directly to hidden neurons.

$$\text{ETLP} = \text{spike trace} + \text{postsynaptic surrogate factor} + \text{projected-label teaching signal}$$

Difference from DRTP: DRTP projects targets in non-spiking ANNs; ETLP turns this idea into event-triggered plasticity for SNNs, with ALIF traces and teaching spikes.

ETLP: Event-Based Three-Factor Local Plasticity [6]



OSTTP: Online Spatio-Temporal Learning with Target Projection [7]

Main idea

$$\text{OSTTP} = \text{OSTL eligibility traces} + \text{DRTP target projection}$$

OSTTP keeps the temporal eligibility trace from OSTL:

$$e_l^{t,\theta_l}$$

but replaces the spatial backpropagation signal with a projected target:

$$L_l^t = B_l y^{t,*}$$

So the update becomes:

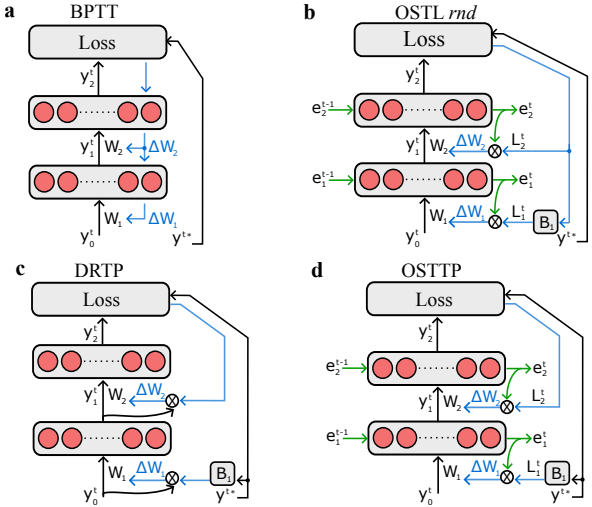
$$\Delta\theta_l \approx \sum_t (B_l y^{t,*})\, e_l^{t,\theta_l}$$

What changes compared to OSTL?

$$\text{spatial backprop signal} \quad \longrightarrow \quad \text{fixed random target projection}$$

Key message: OSTTP removes backward-through-time and backward-through-layer learning signals, while preserving temporal credit through OSTL-style eligibility traces.

OSTTP: Online Spatio-Temporal Learning with Target Projection [7]



OSTTP: Algorithm Comparison [7]

Algorithm	No symmetric weights	No update-locking in time	No update-locking in space
BPTT	×	×	×
FA	√	–	×
DFA	√	–	×
DRTP	√	–	√
e-prop	√	√	×
OSTL	×	√	×
OSTL *rnd*	√	√	×
DECOLLE	√	√	√
ETLP	√	√	√
OSTTP	√	√	√

OSTTP combines temporal online learning with target-projection-based spatial locality.

OTTT: Online Training Through Time

OTTT: From BPTT with SG to Online Temporal Learning [8]

LIF neuron

$$u_i[t+1] = \lambda\left(u_i[t] - V_{\text{th}} s_i[t]\right) + \sum_j w_{ij} s_j[t] + b_i$$

$$s_i[t+1] = H\left(u_i[t+1] - V_{\text{th}}\right)$$

BPTT with surrogate gradients

$$\frac{\partial L}{\partial W_l} = \sum_{t=1}^{T} \frac{\partial L}{\partial s_{l+1}[t]} \frac{\partial s_{l+1}[t]}{\partial u_{l+1}[t]} \left(\frac{\partial u_{l+1}[t]}{\partial W_l} + \sum_{\tau<t} \cdots \right)$$

The non-differentiable spike function is handled by a surrogate derivative:

$$\frac{\partial s}{\partial u} \approx \text{rectangular or sigmoid-like surrogate}$$

Why BPTT is expensive

store all time-step states $\implies$ backward through time

$$\text{Memory}_{\text{BPTT}} = \mathcal{O}(TLn)$$

Goal of OTTT: keep strong supervised training, but remove backpropagation through time.

OTTT: Decoupling Temporal Dependency [8]

Key OTTT idea
In the temporal dependency path, do *not* use surrogate derivatives:

$$\frac{\partial u_{l+1}[i+1]}{\partial s_{l+1}[i]} \frac{\partial s_{l+1}[i]}{\partial u_{l+1}[i]} \approx 0$$

because the derivative of the Heaviside spike function is zero almost everywhere.

Then the temporal dependency reduces to only the leak path:

$$\frac{\partial u_{l+1}[i+1]}{\partial u_{l+1}[i]} = \lambda I$$

So the past contribution can be summarized by a tracked presynaptic activity:

$$\hat{a}_l[t] = \sum_{\tau \leq t} \lambda^{t-\tau} s_l[\tau]$$

with online update:

$$\hat{a}_l[t+1] = \lambda \hat{a}_l[t] + s_l[t+1]$$

Instantaneous loss

$$L[t] = \frac{1}{T} L(s_N[t], y)\,, \qquad L = \sum_{t=1}^{T} L[t]$$

This makes gradients computable independently at each time step, without backward-through-time.

OTTT: Online Gradient and Three-Factor Form [8]

Instantaneous OTTT gradient

$$\nabla_{W_l} L[t] = g_{u_{l+1}}[t] \hat{a}_l[t]^T$$

where

$$g_{u_{l+1}}[t] = \left(\frac{\partial L[t]}{\partial s_{l+1}[t]} \frac{\partial s_{l+1}[t]}{\partial u_{l+1}[t]} \right)^T$$

Interpretation

postsynaptic error/sensitivity $\times$ presynaptic trace

At neuron/synapse level:

$$\nabla_{W_{ij}} L[t] = \hat{a}_i[t] \, f(u_j[t]) \, \delta_j[t] \tag{7}$$

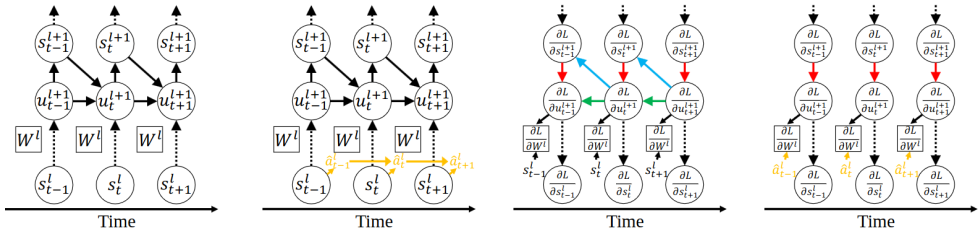
So OTTT has a three-factor Hebbian form:

presynaptic trace $\times$ postsynaptic factor $\times$ learning signal

Important

OTTT is online in time, but it still uses instantaneous spatial backpropagation.

BPTT vs. OTTT [8]



(a) BPTT Forward (b) OTTT Forward (c) BPTT Backward (d) OTTT Backward

OTTT: Algorithm [8]

Algorithm 1 One iteration of OTTT training for a feedforward network

Require: Network parameters $\{W_l\}$, $\{b_l\}$; input $\mathbf{x}$; label y; time steps T

for $t = 1, 2, \ldots, T$ **do**
  for $l = 1, 2, \ldots, N$ **do**
    Update membrane potentials $u^l[t]$ and spikes $s^l[t]$
    Update tracked presynaptic activity

$$\hat{a}^l[t] = \lambda \hat{a}^l[t-1] + \hat{s}^l[t]$$

  end for
  for $l = N, N-1, \ldots, 1$ **do**
    Compute instantaneous backpropagated error $g_u^l[t]$
    Compute instantaneous gradient

$$\nabla_{W_{l-1}} L[t] = g_u^l[t](\hat{a}^{l-1}[t])^T$$

    if online update **then**
      Update weights immediately (OTTT$_O$)
    else
      Accumulate gradients (OTTT$_A$)
    end if
  end for
end for
if not online update **then**
  Update parameters with accumulated gradients
end if

OTTT_A vs. OTTT_O and Why No BatchNorm [8]

Two variants

$$\text{OTTT}_A: \quad \nabla_W L = \sum_{t=1}^{T} \nabla_W L[t]$$

$$\text{OTTT}_O: \quad W \leftarrow W - \eta \nabla_W L[t] \quad \text{at each time step}$$

Variant	Update behavior
OTTT_A	accumulate gradients, update once after sequence
OTTT_O	update immediately at every time step

Why no temporal BatchNorm?
Temporal BatchNorm needs statistics across all time steps, so it breaks the online property.

Instead, the paper uses:

scaled Weight Standardization (sWS)

$$\hat{W}_{i,j} = \gamma \frac{W_{i,j} - \mu_{W_{i,\cdot}}}{\sigma_{W_{i,\cdot}} \sqrt{N}}$$

sWS stabilizes training without needing future time-step statistics.

STDP-Inspired Local Learning

Spike-Timing Dependent Plasticity (STDP) [9]

SNN forward model

$$u_i[t] = \gamma\left(u_i[t-1] - v_{\text{th}} y_i[t-1]\right) + w_{ij} x_j[t]$$

$$y_i[t] = \Theta\left(u_i[t] - v_{\text{th}}\right)$$

Interpretation

$$x_j[t] \rightarrow u_i[t] \rightarrow y_i[t]$$

The membrane integrates current input and leaky past state; if threshold is crossed, the neuron spikes and resets.

STDP timing function

$$\Phi(t_i, t_j) = \begin{cases} \alpha_{\text{pre}} \lambda_{\text{pre}}^{t_i - t_j}, & t_i \geq t_j \\ \alpha_{\text{post}} \lambda_{\text{post}}^{t_j - t_i}, & t_i < t_j \end{cases}$$

Causal term

$t_j \rightarrow t_i$ pre before post

Non-causal term

$t_i \rightarrow t_j$ post before pre

$\alpha_{\text{post}} < 0 \Rightarrow$ favor causality $\alpha_{\text{post}} > 0 \Rightarrow$ favor synchrony

S-TLLR: From STDP to a Local Eligibility Trace [9]

STDP written online

$$\Delta w_{ij}[t] = \alpha_{\text{pre}} y_i[t] \sum_{t'=0}^{t} \lambda_{\text{pre}}^{t-t'} x_j[t'] + \alpha_{\text{post}} x_j[t] \sum_{t'=0}^{t-1} \lambda_{\text{post}}^{t-t'} y_i[t']$$

$$w_{ij}[t+1] = w_{ij}[t] + \Delta w_{ij}[t]$$

$$\Delta w_{ij}[t] = \alpha_{\text{pre}} y_i[t]\, \text{tr}(x_j)[t] + \alpha_{\text{post}} x_j[t] \big(\text{tr}(y_i)[t] - y_i[t]\big)$$

STDP is temporal-local and uses both causal and non-causal timing.

BPTT recap

$$\frac{d\mathcal{L}}{dw} = \sum_t \frac{\partial \mathcal{L}}{\partial y[t]} \frac{\partial y[t]}{\partial u[t]} \frac{\partial u[t]}{\partial w}$$

Generic three-factor rule

$$e_{ij}[t] = \beta e_{ij}[t-1] + f(y_i[t]) g(x_j[t])$$

$$\Delta w_{ij} = \sum_t \delta_i[t]\, e_{ij}[t]$$

S-TLLR keeps the three-factor form, but replaces the eligibility trace.

S-TLLR technical eligibility trace

$$e_{ij}[t] = \alpha_{\text{pre}} \Psi(u_i[t]) \sum_{t'=0}^{t} \lambda_{\text{pre}}^{t-t'} x_j[t'] + \alpha_{\text{post}} x_j[t] \sum_{t'=0}^{t-1} \lambda_{\text{post}}^{t-t'} \Psi(u_i[t'])$$

S-TLLR: Learning Signal and Update [9]

Learning signal

$$\delta_i^{(l)}[t] = \begin{cases} \dfrac{\partial \mathcal{L}(\mathbf{y}^L[t], \mathbf{y}^*)}{\partial y_i^{(l)}[t]}, & t \geq T_l \\ 0, & \text{otherwise} \end{cases}$$

Synaptic update

$$w_{ij} := w_{ij} + \rho \sum_{t=T_l}^{T} \delta_i[t]\, e_{ij}[t] \qquad (13)$$

Interpretation

$$\Delta w_{ij}[t] = \underbrace{\delta_i[t]}_{\text{top-down learning signal}} \quad \underbrace{e_{ij}[t]}_{\text{STDP-inspired local eligibility}}$$

Backpropagation occurs through layers, not through time.

Learning-signal choices

- BP through layers
- or random feedback / DFA

Complexity

Memory: $\mathcal{O}(n)$

Time: $\mathcal{O}(n^2)$

Why low memory? S-TLLR drops the recurrent synapse-level eligibility state and keeps only neuron-level trace terms.

Recurrent extension Also derived for recurrent synaptic connections:

$$w_{ik}^{\text{rec}}$$

with the same STDP-inspired principle.

Key message: S-TLLR is supervised, temporal-local, low-memory, and explicitly uses both causal and non-causal spike timing.

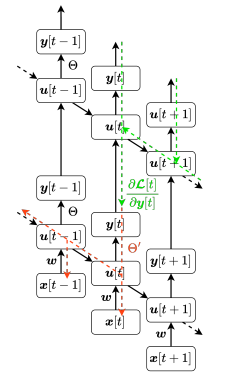
S-TLLR Algorithm [9]

Algorithm 2 S-TLLR

Require: $\mathbf{x}, \mathbf{y}^*, \mathbf{w}, T, T_l, \rho$

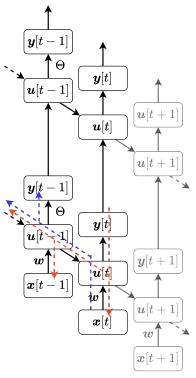
for $t = 1, 2, \ldots, T$ **do**
  for $l = 1, 2, \ldots, L$ **do**
    Update $\mathbf{u}^{(l)}[t]$, $\mathbf{y}^{(l)}[t]$
    Update eligibility trace $\mathbf{e}^{(l)}[t]$
  end for
  if $t \geq T_l$ **then**
    Compute output learning signal $\boldsymbol{\delta}^{(L)}$
    for $l = L-1, L-2, \ldots, 2$ **do**
      Compute $\boldsymbol{\delta}^{(l)}$ by BP or random feedback
    end for
    $\Delta w_{ij}^{(L)}[t] = \delta_i^{(L)}\, y_j^{(L-1)}[t]$
    $\Delta w_{ij}^{(l)}[t] = \delta_i^{(l)}\, e_{ij}^{(l-1)}[t]$
  end if
end for
$\mathbf{w}^{(l)} := \mathbf{w}^{(l)} + \rho \sum_{t=1}^{T} \Delta \mathbf{w}^{(l)}[t]$
return $\mathbf{w}$

BPTT vs. STDP vs. S-TLLR [9]



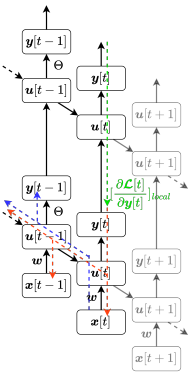
$$\Delta \boldsymbol{w}[t] = \frac{\partial \mathcal{L}}{\partial \boldsymbol{y}[t]} \; \Theta'(\boldsymbol{u}[t]) \sum_{t'=0}^{t} \gamma^{t-t'} \boldsymbol{x}[t']$$

BPTT



$$\Delta \boldsymbol{w}[t] = \alpha_{pre} \boldsymbol{y}[t] \sum_{t'=0}^{t} \lambda_{pre}^{t-t'} \boldsymbol{x}[t'] + \alpha_{post} \boldsymbol{x}[t] \sum_{t'=0}^{t-1} \lambda_{post}^{t-t'} \boldsymbol{y}[t']$$

STDP



$$\Delta \boldsymbol{w}[t] = \left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{y}[t]}\right] \alpha_{pre} \boldsymbol{\Psi}(\boldsymbol{u}[t]) \sum_{t'=0}^{t} \lambda_{pre}^{t-t'} \boldsymbol{x}[t'] + \alpha_{post} \boldsymbol{x}[t] \sum_{t'=0}^{t-1} \lambda_{post}^{t-t'} \boldsymbol{\Psi}(\boldsymbol{u}[t'])$$

S-TLLR

S-TLLR: Complexity Comparison [9]

Method	Memory Complexity	Time Complexity	Temporal Local	Leverage Non-Causality
BPTT	Tn	Tn^2	No	No
RTRL	n^3	n^4	Yes	No
e-prop	n^2	n^2	Yes	No
OSTL	n^2	n^2	Yes	No
ETLP	n^2	n^2	Yes	No
OSTTP	n^2	n^2	Yes	No
OTTT	n	n^2	Yes	No
S-TLLR	$\mathbf{n}$	$\mathbf{n^2}$	**Yes**	**Yes**

Main point: S-TLLR keeps temporal locality with linear memory in the number of neurons.

S-TLLR: Effect of Non-Causal Terms [9]

Dataset	Model	$\alpha_{\text{post}} = 0$	$\alpha_{\text{post}} = +1$	$\alpha_{\text{post}} = -1$
DVS Gesture	VGG9	$94.61 \pm 0.73\%$	$94.01 \pm 1.10\%$	$\mathbf{95.07 \pm 0.48}\%$
DVS CIFAR10	VGG9	$72.93 \pm 0.94\%$	$73.42 \pm 0.50\%$	$\mathbf{73.93 \pm 0.62}\%$
N-CALTECH101	VGG9	$62.24 \pm 1.22\%$	$53.42 \pm 1.50\%$	$\mathbf{66.33 \pm 0.86}\%$
SHD	RSNN	$77.09 \pm 0.33\%$	$\mathbf{78.23 \pm 1.84}\%$	$74.69 \pm 0.47\%$

Vision tasks benefit from $\alpha_{\text{post}} < 0$, while SHD benefits from $\alpha_{\text{post}} > 0$.

S-TLLR: Performance Comparison I [9]

Dataset	Method	Model	Accuracy
DVS CIFAR10	BPTT baseline	VGG-9	$75.44 \pm 0.76\%$
DVS CIFAR10	S-TLLR ($T_l = 5$, $\alpha_{\text{post}} = -1$)	VGG-9	$73.93 \pm 0.62\%$
DVS CIFAR10	S-TLLR ($T_l = 0$, $\alpha_{\text{post}} = -1$)	VGG-9	$\mathbf{75.6 \pm 0.10}\%$
DVS CIFAR10	S-TLLR ($T_l = 0$, $\alpha_{\text{post}} = 0$)	VGG-9	$74.8 \pm 0.15\%$
DVS CIFAR10	BPTT baseline	ResNet18	$72.68 \pm 0.87\%$
DVS CIFAR10	S-TLLR ($T_l = 5$)	ResNet18	$71.94 \pm 0.75\%$
DVS CIFAR10	S-TLLR ($T_l = 0$)	ResNet18	$\mathbf{74.5 \pm 0.64}\%$
DVS Gesture	BPTT baseline	VGG-9	$95.58 \pm 1.08\%$
DVS Gesture	S-TLLR	VGG-9	$\mathbf{97.72 \pm 0.38}\%$
DVS Gesture	BPTT baseline	ResNet18	$94.92 \pm 0.38\%$
DVS Gesture	S-TLLR	ResNet18	$94.92 \pm 0.61\%$
N-CALTECH101	BPTT baseline	VGG-9	$65.92 \pm 0.82\%$
N-CALTECH101	S-TLLR	VGG-9	$\mathbf{66.058 \pm 0.92}\%$
N-CALTECH101	BPTT baseline	ResNet18	$60.89 \pm 0.89\%$
N-CALTECH101	S-TLLR	ResNet18	$\mathbf{61.65 \pm 0.99}\%$

S-TLLR: Performance Comparison II [9]

Dataset	Method	Model	Accuracy / Metric
SHD	ETLP	ALIF-RSNN	$74.59 \pm 0.44\%$
SHD	OSTTP	LIF-RSNN	$77.33 \pm 0.8\%$
SHD	BPTT baseline	LIF-RSNN	$70.57 \pm 0.96\%$
SHD	S-TLLR$_{\mathrm{BP}}$	LIF-RSNN	$\mathbf{78.24 \pm 1.84}\%$
SHD	S-TLLR$_{\mathrm{DFA}}$	LIF-RSNN	$74.60 \pm 0.52\%$

S-TLLR stays competitive with BPTT while using much less memory.

TESS: Temporally and Spatially Local Learning [10]

Previous temporal-local rules

e-prop, OSTL, S-TLLR

avoid backpropagation through time by using eligibility traces.

But many of them still need a spatial learning signal from upper layers:

$$\left[\frac{\partial \mathcal{L}}{\partial s^{(l)}[t]}\right]_{\text{local in time}}$$

TESS goal

$$\boxed{\text{local temporal credit} + \text{local spatial credit}}$$

Main idea

$$\boxed{\text{TESS} = \text{S-TLLR-style low-memory traces} + \text{local learning signal generation}}$$

Key difference from S-TLLR: S-TLLR is temporal-local but still uses BP/DFA for the learning signal; TESS generates the learning signal inside each layer.

TESS: Forward Model and Three-Factor Form [10]

LIF forward dynamics

$$u_i^{(l)}[t] = \gamma \left(u_i^{(l)}[t-1] - v_{\text{th}} o_i^{(l)}[t-1] \right) + \sum_j W_{ij}^{(l)} o_j^{(l-1)}[t]$$

$$o_i^{(l)}[t] = \Theta \left(u_i^{(l)}[t] - v_{\text{th}} \right)$$

Three-factor update

$$\Delta W_{ij}^{(l)} = \sum_t m_i^{(l)}[t] \, e_{ij}^{(l)}[t]$$

learning signal $\times$ eligibility trace

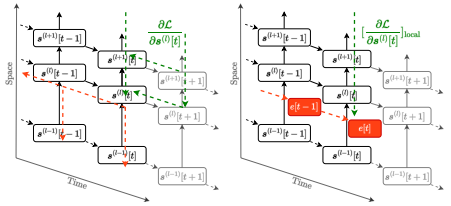
TESS changes both factors

$e_{ij}^{(l)}[t] \Rightarrow$ low-memory causal/non-causal traces

$m_i^{(l)}[t] \Rightarrow$ locally generated layer-wise learning signal

TESS is fully local: no BPTT through time and no BP through layers.

TESS: Learning Rule Strategies [10]



(a) Non-local learning rule

(b) Temporally local learning rule

(c) Fully local learning rule

TESS: Temporal Traces and Local Learning Signal [10]

Causal presynaptic trace

$$q^{(l)}[t] = \lambda_{\text{pre}} q^{(l)}[t-1] + o^{(l-1)}[t]$$

$$e^{(l)}_{\text{pre}}[t] = \alpha_{\text{pre}} \Psi(u^{(l)}[t]) \otimes q^{(l)}[t]$$

Non-causal postsynaptic trace

$$h^{(l)}[t] = \lambda_{\text{post}} h^{(l)}[t-1] + \Psi(u^{(l)}[t-1])$$

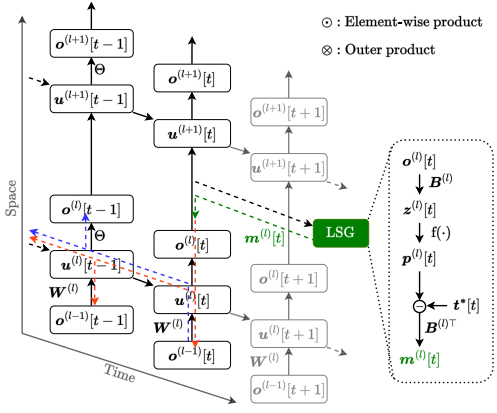
$$e^{(l)}_{\text{post}}[t] = \alpha_{\text{post}} h^{(l)}[t] \otimes o^{(l-1)}[t]$$

Local learning signal generation

$$m^{(l)}[t] = B^{(l)T} \left(f(B^{(l)} o^{(l)}[t]) - y^* \right)$$

Each layer maps its own spikes to task space, compares with the target, and projects the error back locally.

TESS: Method Overview [10]



$$\Delta \boldsymbol{W}^{(l)}[t] = (\boldsymbol{m}^{(l)}[t] \odot \alpha_{\text{pre}} \Psi(\boldsymbol{u}^{(l)}[t])) \otimes \sum_{t'=0}^{t} \lambda_{\text{pre}}^{t-t'} \boldsymbol{o}^{(l-1)}[t'] + (\boldsymbol{m}^{(l)}[t] \odot \sum_{t'=0}^{t-1} \lambda_{\text{post}}^{t-t'} \Psi(\boldsymbol{u}^{(l)}[t'])) \otimes \alpha_{\text{post}} \boldsymbol{o}^{(l-1)}[t]$$

TESS: Memory Cost [10]

BPTT stores states for all time steps

$$\text{Mem}_{\text{BPTT}} = T \sum_{l=0}^{L} n^{(l)} \approx \mathcal{O}(TLn)$$

S-TLLR stores two neuron-level traces

$$\text{Mem}_{\text{S-TLLR}} = 2 \sum_{l=0}^{L} n^{(l)} \approx \mathcal{O}(Ln)$$

TESS stores the same type of vector traces

$$q^{(l)}[t], \qquad h^{(l)}[t]$$

$$\text{Mem}_{\text{TESS}} = 2 \sum_{l=0}^{L} n^{(l)} \approx \mathcal{O}(Ln)$$

TESS stores neuron-level traces, not synapse-level traces.

TESS: Computational Cost [10]

BPTT learning-signal cost

$$\text{MAC}_{\text{BPTT}} = T \sum_{l=1}^{L} n^{(l)} n^{(l-1)} \approx \mathcal{O}(TLn^2)$$

S-TLLR still propagates errors through layers

$$\text{MAC}_{\text{S-TLLR}} = (T - t_l) \sum_{l=1}^{L} n^{(l)} n^{(l-1)} \approx \mathcal{O}(Ln^2)$$

TESS generates the learning signal locally

$$m^{(l)}[t] = B^{(l)T} \left(f(B^{(l)} o^{(l)}[t]) - y^* \right)$$

$$\text{MAC}_{\text{TESS}} = (T - t_l) \sum_{l=1}^{L} 2Cn^{(l)} \approx \mathcal{O}(LCn)$$

Since usually $C \ll n$, TESS is much cheaper than layer-wise backpropagation.

TESS: Complexity Comparison [10]

Method	Memory Complexity	Time Complexity	Temporal Locality	Spatial Locality
BPTT	TLn	TLn^2	No	No
e-prop	Ln^2	Ln^2	Yes	No
OSTL	Ln^2	Ln^2	Yes	No
ETLP	Ln^2	LCn	Yes	Yes
OSTTP	Ln^2	LCn	Yes	Yes
OTTT	Ln	Ln^2	Yes	No
S-TLLR	Ln	Ln^2	Yes	No
TESS	$\mathbf{Ln}$	$\mathbf{LCn}$	**Yes**	**Yes**

TESS combines the memory advantage of S-TLLR with the spatial locality of local signal generation.

TESS: Non-Causal Term Ablation [10]

Dataset	T	$\alpha_{\text{post}} = -1$	$\alpha_{\text{post}} = 0$	$\alpha_{\text{post}} = +1$
CIFAR10-DVS	10	$\mathbf{75.00 \pm 0.69}\%$	$75.00 \pm 0.65\%$	$74.36 \pm 0.87\%$
DVS Gesture	20	$98.56 \pm 0.41\%$	$98.33 \pm 0.57\%$	$\mathbf{98.56 \pm 0.31}\%$
CIFAR10	6	$89.93 \pm 0.31\%$	$91.99 \pm 0.19\%$	$\mathbf{92.55 \pm 0.16}\%$
CIFAR100	6	$62.49 \pm 1.05\%$	$68.19 \pm 0.55\%$	$\mathbf{70.00 \pm 0.34}\%$

The best non-causal setting is dataset-dependent; positive inclusion helps CIFAR10 and CIFAR100.

TESS: Performance and Cost I [10]

Dataset	Method	Local Learning	Accuracy	MACs ($\times \mathbf{10^6}$)
CIFAR10-DVS	BPTT baseline	No	$76.40 \pm 0.66\%$	13589.59
CIFAR10-DVS	S-TLLR baseline	Partial	$75.14 \pm 1.37\%$	13589.59
CIFAR10-DVS	TESS	Yes	$75.00 \pm 0.65\%$	$\mathbf{22.15}$
DVS Gesture	BPTT baseline	No	$97.95 \pm 0.68\%$	12079.69
DVS Gesture	S-TLLR baseline	Partial	$98.48 \pm 0.37\%$	12079.69
DVS Gesture	TESS	Yes	$\mathbf{98.56 \pm 0.31\%}$	$\mathbf{22.65}$

On DVS Gesture, TESS is fully local and slightly outperforms both baselines.

TESS: Performance and Cost II [10]

Dataset	Method	Local Learning	Accuracy	MACs ($\times 10^6$)
CIFAR10	BPTT baseline	No	$92.55 \pm 0.06\%$	3623.90
CIFAR10	S-TLLR baseline	Partial	$91.88 \pm 0.28\%$	3623.90
CIFAR10	TESS	Yes	$\mathbf{92.55 \pm 0.16\%}$	$\mathbf{5.48}$
CIFAR100	BPTT baseline	No	$69.28 \pm 0.37\%$	3624.18
CIFAR100	S-TLLR baseline	Partial	$68.00 \pm 0.71\%$	3624.18
CIFAR100	TESS	Yes	$\mathbf{70.00 \pm 0.34\%}$	$\mathbf{17.64}$

TESS matches BPTT on CIFAR10 and exceeds the BPTT baseline on CIFAR100.

Future Directions

Future Directions

Removed due to internal discussion

This part was intentionally omitted from the shared version of the slides.

Closing

Conclusion

- I summarized key papers from 2016–2025 and identified three main research directions for my next work.
- I plan to build a codebase that includes existing local learning methods and carefully study their implementations.
- Understanding local learning mechanisms in SNNs can support cross-domain knowledge transfer and inspire new architectures.
- Developing a new learning rule with strong performance is challenging, but I will continue exploring it alongside my other research directions.

Thanks for listening!

Any Questions?

References I

[1] Owen Marschall, Kyunghyun Cho, and Cristina Savin. *A Unified Framework of Online Learning Algorithms for Training Recurrent Neural Networks*. 2019. arXiv: 1907.02649 [cs.LG]. URL: https://arxiv.org/abs/1907.02649.

[2] Charlotte Frenkel, Martin Lefebvre, and David Bol. "Learning Without Feedback: Fixed Random Learning Signals Allow for Feedforward Training of Deep Neural Networks". In: *Frontiers in Neuroscience* 15 (Feb. 2021). ISSN: 1662-453X. DOI: 10.3389/fnins.2021.629892. URL: http://dx.doi.org/10.3389/fnins.2021.629892.

[3] Guillaume Bellec et al. "A solution to the learning dilemma for recurrent networks of spiking neurons". In: *Nature Communications* 11.1 (2020), p. 3625. DOI: 10.1038/s41467-020-17236-y. URL: https://doi.org/10.1038/s41467-020-17236-y.

[4] Jacques Kaiser, Hesham Mostafa, and Emre Neftci. "Synaptic Plasticity Dynamics for Deep Continuous Local Learning (DECOLLE)". In: *Frontiers in Neuroscience* 14 (May 2020). ISSN: 1662-453X. DOI: 10.3389/fnins.2020.00424. URL: http://dx.doi.org/10.3389/fnins.2020.00424.

References II

[5] Thomas Bohnstingl et al. *Online Spatio-Temporal Learning in Deep Neural Networks*. 2020. arXiv: 2007.12723 [cs.LG]. URL: https://arxiv.org/abs/2007.12723.

[6] Fernando M Quintana et al. "ETLP: event-based three-factor local plasticity for online learning with neuromorphic hardware". In: *Neuromorphic Computing and Engineering* 4.3 (Aug. 2024), p. 034006. ISSN: 2634-4386. DOI: 10.1088/2634-4386/ad6733. URL: http://dx.doi.org/10.1088/2634-4386/ad6733.

[7] Thomas Ortner et al. *Online Spatio-Temporal Learning with Target Projection*. 2023. arXiv: 2304.05124 [cs.NE]. URL: https://arxiv.org/abs/2304.05124.

[8] Mingqing Xiao et al. *Online Training Through Time for Spiking Neural Networks*. 2022. arXiv: 2210.04195 [cs.NE]. URL: https://arxiv.org/abs/2210.04195.

[9] Marco Paul E. Apolinario and Kaushik Roy. *S-TLLR: STDP-inspired Temporal Local Learning Rule for Spiking Neural Networks*. 2024. arXiv: 2306.15220 [cs.NE]. URL: https://arxiv.org/abs/2306.15220.

References III

[10] Marco P. E. Apolinario, Kaushik Roy, and Charlotte Frenkel. “TESS: A Scalable Temporally and Spatially Local Learning Rule for Spiking Neural Networks”. In: *2025 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2025, pp. 1–9. DOI: 10.1109/ijcnn64981.2025.11227652. URL: http://dx.doi.org/10.1109/IJCNN64981.2025.11227652.