

# Training Event-Based Neural Networks with Exact Gradients via Differentiable ODE Solving in JAX

Lukas König<sup>1</sup>, Manuel Kuhn<sup>1</sup>,  
David Kappel<sup>1</sup> and Anand Subramoney<sup>2</sup>

<sup>1</sup>Faculty of Computer Science  
University of Bielefeld

<sup>2</sup>Faculty of Computer Science  
Royal Holloway University of London

NICE, March 2026

Event-based neural networks are formulated in **continuous time** and compute via **discrete spike events**.

Training them with gradient-based methods is challenging:

- Spikes are **discontinuous**: gradients must pass through both continuous dynamics and discrete resets
- Spike **timing precision** matters for temporal coding
- Most frameworks don't handle both **flexibility** in neuron model choice while working in **continuous time**

**Discrete Time**  
(Surrogate Gradients)

Flexible models

Biased gradients

Time binned

vs.

**Continuous Time**  
(Closed Form)

Continuous time

Exact gradients\*

Less flexible

Can we use the continuous time formulation for **training arbitrary neuron models** with backpropagation?

Simulate on a **fixed time grid**. Use **surrogate gradients** to handle discontinuities.

- ✓ Fast on GPU
- ✓ Flexible model choice
- ✓ Simple to implement

- ≈ Surrogate gradients (bias)
- ✗ Fixed time grid
- ✗ Memory/resolution trade-off

# Continuous Time (Closed Form)

Use **closed-form solutions** for spike times and state evolution. Derive gradients via closed form solutions e.g. from using the adjoint method.

- ✓ Scales with number of events
- ✓ Continuous time
- ≈ Exact gradients<sup>\*</sup>

- ✗ Only specific models (i.e. LIF, QIF ...)
- ✗ Requires closed form solution

# Continuous Time (ODE Solver). What our work (Eventax) uses.

**ODE solver** to simulate state between events. Use **root finder** to locate exact spike times. (Diffraction<sup>1</sup> handles both)

Gradients through jumps via **implicit function theorem**. Two options for backprop through the solver:

- ① Through solver steps (BPTT-like)
- ② Via backward ODE

---

<sup>1</sup>Kidger, *On Neural Differential Equations*, PhD thesis, University of Oxford, 2021

# Approach 1: Backprop Through Solver Steps

## Differentiate through the ODE solver steps

- ✓ Continuous time
- ✓ Exact w.r.t. forward simulation<sup>\*</sup>
- ✓ Many ODE-defined neuron models
- ✗ Memory scales with solver steps
- ✗ Slower than closed form

→ **This is what Eventax does using Diffrax.**

We extend part of the approach of Holberg & Salvi<sup>1</sup> to a general framework.

---

<sup>1</sup>Holberg & Salvi, *Exact Gradients for Stochastic Spiking Neural Networks Driven by Rough Signals*, NeurIPS 2024

# Approach 1: Backprop Through Solver Steps

## Differentiate through the ODE solver steps

- ✓ Continuous time
  - ✓ Many ODE-defined neuron models
  - ✗ Memory scales with solver steps
  - ✗ Slower than closed form
- ≈ Exact w.r.t. forward simulation\*

→ **This is what Eventax does using Diffrax.**

We extend part of the approach of **Holberg & Salvi<sup>1</sup>** to a general framework.

**check this out: Implementation + Theory  
of a Stochastic LIF using SDEs**

---

<sup>1</sup>Holberg & Salvi, *Exact Gradients for Stochastic Spiking Neural Networks Driven by Rough Signals*, NeurIPS 2024

## Approach 2: Backprop via Backward ODE

**Solve an adjoint ODE backward** in time instead of storing forward states.

✓ Continuous time

✓ Memory scales with events

✗ Inexact gradients

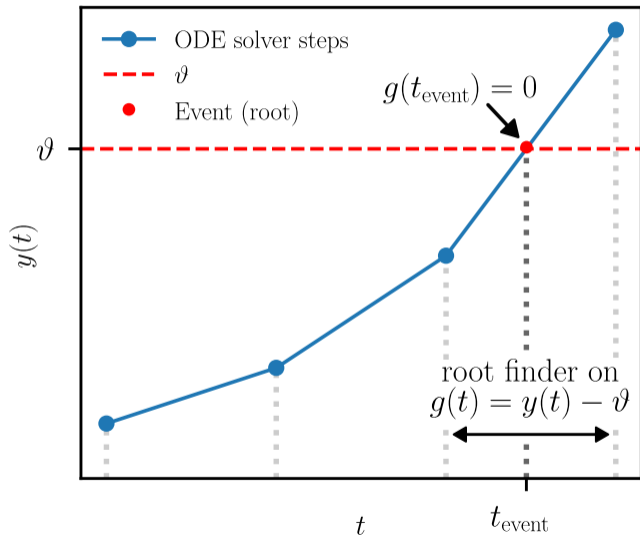
✗ Not yet supported with events in Diffrax

Future work.

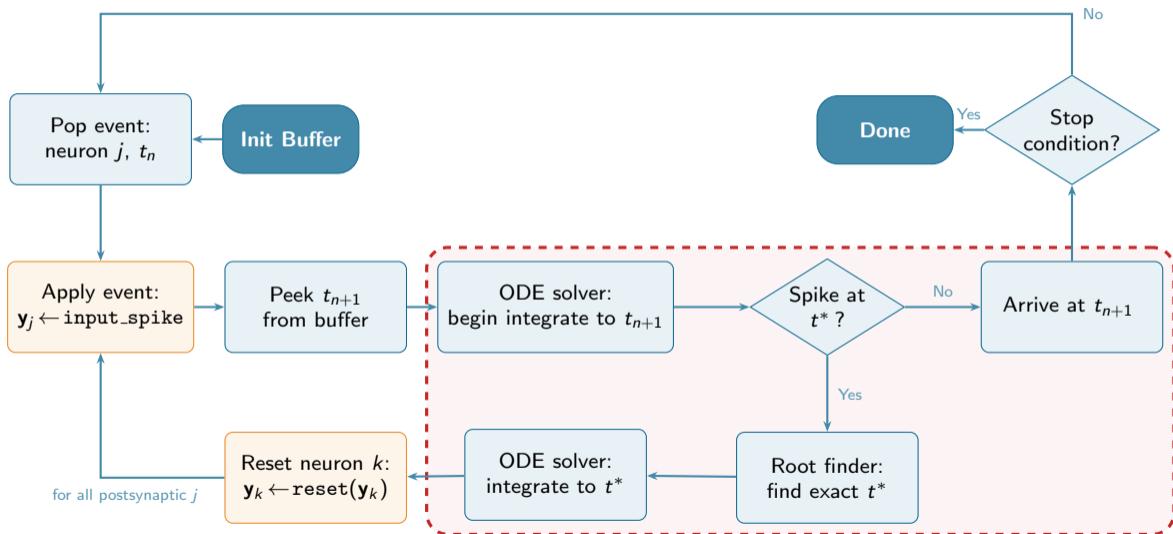
# What Does “Exact” Gradient Mean?

<b>Method</b>	<b>Exact w.r.t.</b>	<b>possible Model</b>
Discrete BPTT	neither (surrogate bias)	many
Closed form	continuous-time model	specific
<b>Eventax (ours)</b>	forward simulation	<b>many ODE</b>

# DiffraX Event Handling



# Eventax Main Loop

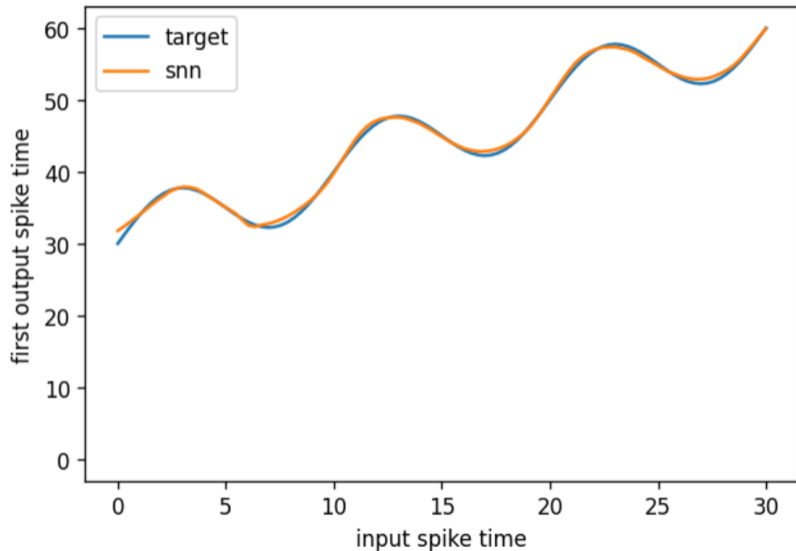


Diffraction

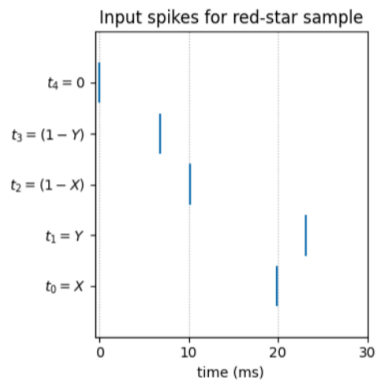
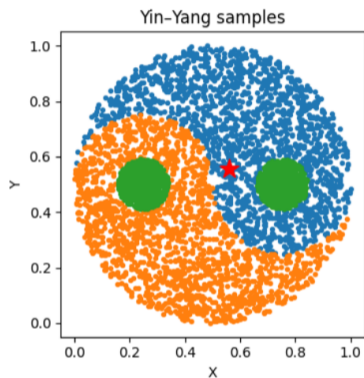
```
1 class NeuronModel(eqx.Module):
2     def init_state(self, n_neurons: int) -> Array: ...
3     def dynamics(self, t: float, y: Array, args: Dict) -> Array: ...
4     def spike_condition(self, t: float, y: Array, **kwargs) -> Array: ...
5     def input_spike(self, y: Array, from_idx: Int,
6                     to_idx: Int, valid_mask: Bool) -> Array: ...
7     def reset_spiked(self, y: Array, spike_mask: Bool) -> Array: ...
```

The `NeuronModel` interface. Users define custom neuron models by implementing: initial state, dynamics, spike condition, input spike handling, and post-spike reset.

# Example 1: Temporal Regression



## Example 2: Yin-Yang Dataset



# Temporal and State-based Classification

All losses use softmax classification.

$$\mathcal{L} = -\log \frac{e^{z_y}}{\sum_{c=1}^C e^{z_c}}$$

Temporal:[1]

$$z_c^{\text{temp}} = -t_c/t_0 \quad (+ \text{late spike penalty, not shown})$$

State based:[2]

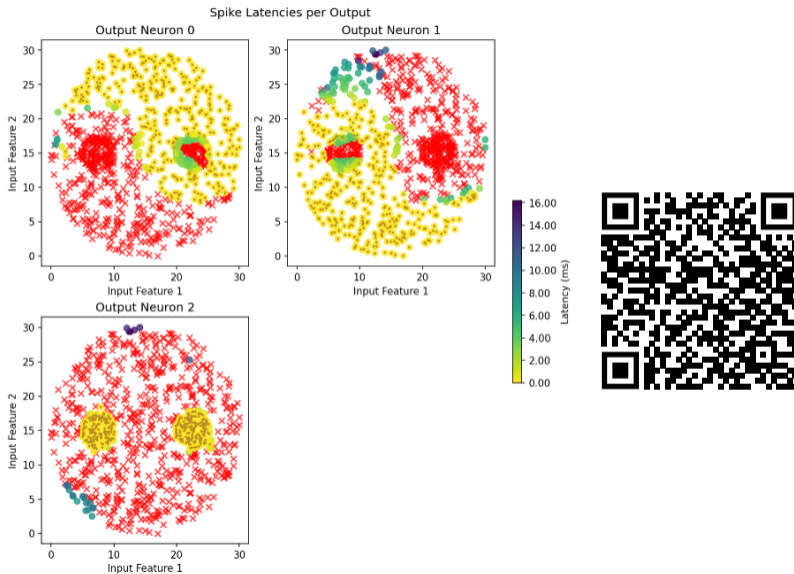
$$\begin{aligned} z_c^{\text{max}} &= \max_{t \in [0, T]} V_c(t), \\ z_c^{\text{int}} &= \int_0^T V_c(t) dt, \\ z_c^{\text{exp}} &= \int_0^T e^{-\lambda t} V_c(t) dt. \end{aligned}$$

**Eventax** handles both temporal and state-based losses out of the box thanks to **continuous-time formulation** and use of **DiffraX**.

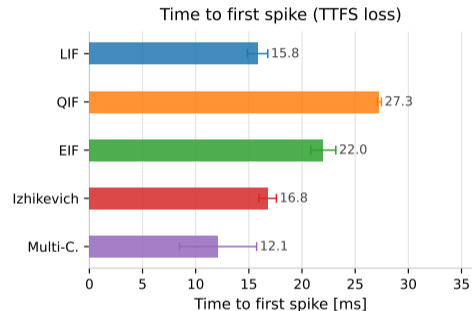
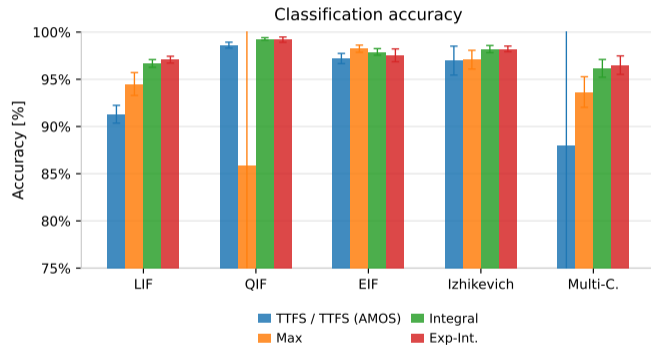
[1] Wunderlich & Pehle, *Event-based backpropagation can compute exact gradients*, 2021

[2] Nowotny, Turner & Knight, *Loss shaping enhances exact gradient learning with EventProp*, 2022

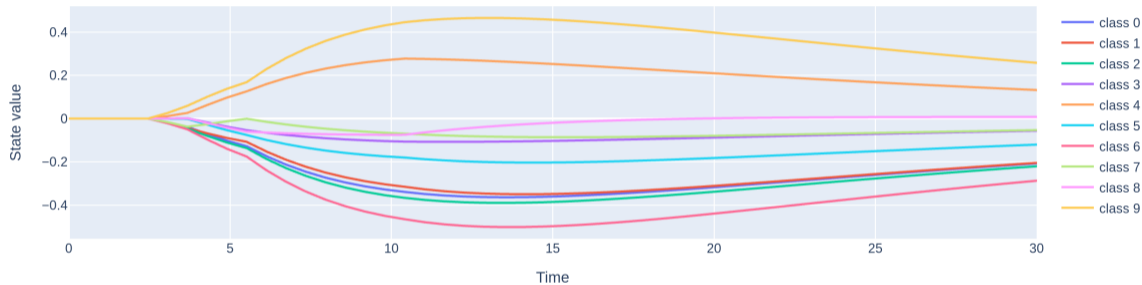
# Results (QIF)



# Results (Other Models)

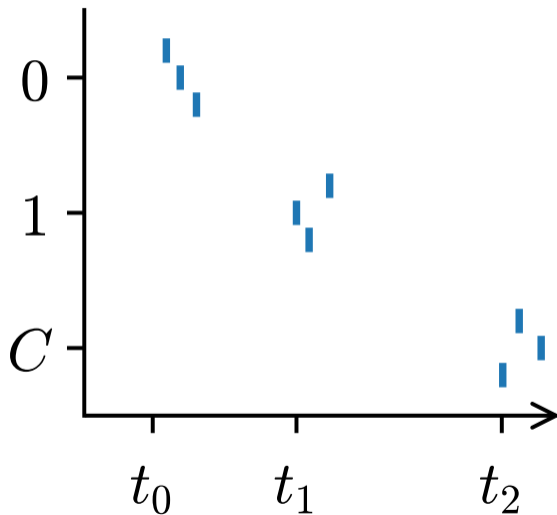


## Example 3: MNIST (LIF)



Eventax achieves  $97.50 \pm 0.07\%$  Accuracy using LIF with exp-integral loss

## Example 4: XOR (EGRU)



- Implement synaptic delays: Continuous time formulation is a natural fit for this.
- Implement Klos and Memmesheimer's method to tackle dead neuron problem for certain models.
- Group neurons into populations:
  - Support more complex architectures like CNNs and mixtures thereof
  - Benefit from sequential structures (i.e. feed-forward)
- Look into good initialization schemes for synaptic parameters (i.e. weights) to speed up training and mitigate dead neurons.

Thank you for your attention. Any questions?