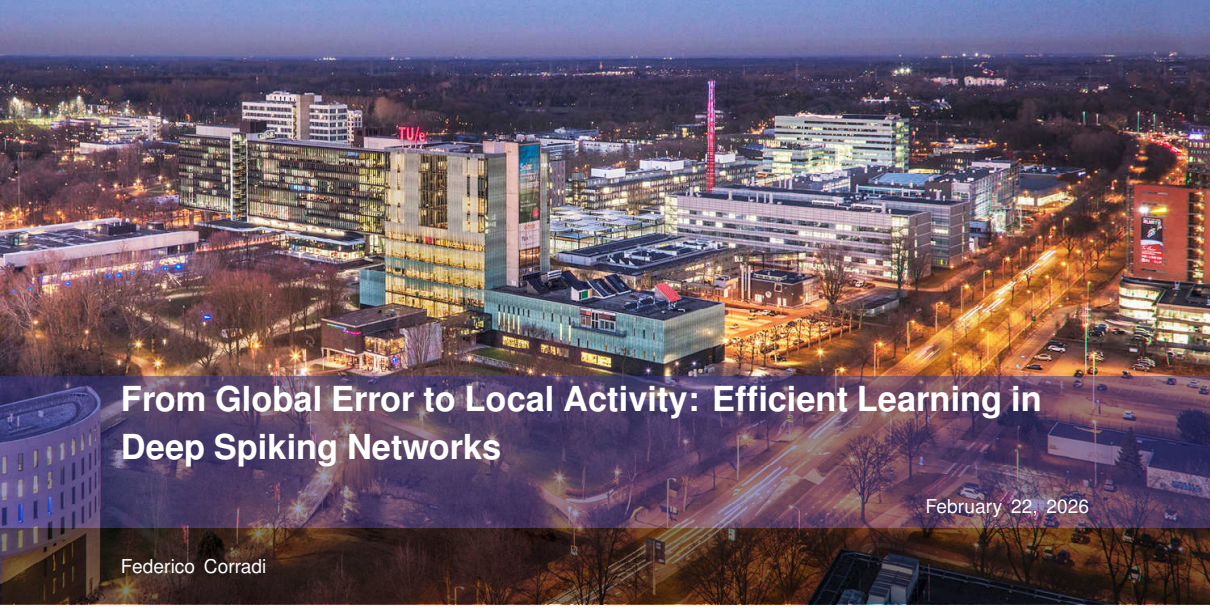


From Global Error to Local Activity: Efficient Learning in Deep Spiking Networks

February 22, 2026

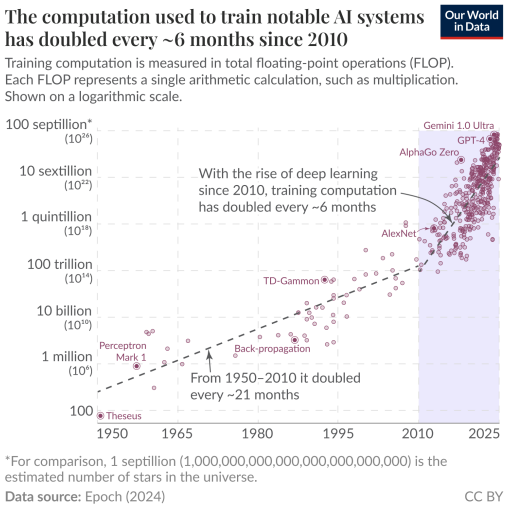
Federico Corradi

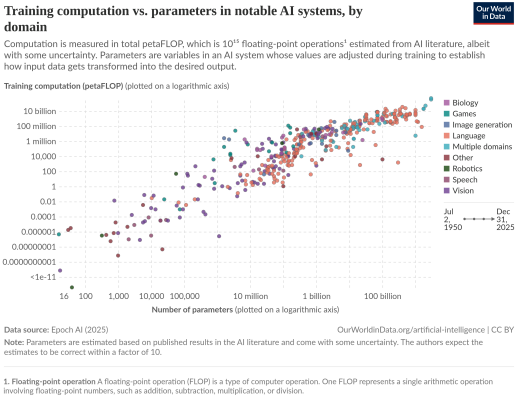
Eindhoven University of Technology, Neuromorphic Edge Computing Systems Lab

Institute of Neuroinformatics, Zurich, 2026

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

Deep Learning Seems to Have No Limits: Compute & Data





How much Training Data? [Epoch AI]

$C \sim 6ND$ (C compute, N param., D tokens)

The Efficiency Gap: LLMs vs Brains

State-of-the-Art LLM (Llama 3 405B)

1. Data Throughput

15T Tokens (Text-only)

$\approx$ 50 TB

2. Energy to Train

Cluster (54 days)

21,600 MWh *(Meta, "The Llama 3 Herd of Models", 2024)*

3. Result

Static Statistical Predictor

The 4-Year-Old Child (Vision)

1. Data Throughput (Vision only)

LeCun's: ~~1 PB~~ Optic nerve: $\sim$1 MB/s $\rightarrow$ $\sim$**60 TB**

(Koch et al., 2006; $\sim$12 h/day awake)

2. Energy (entire brain)

20 W $\times$ 4 Years

0.7 MWh *(incl. all functions)*

3. Result

Adaptive Multi-modal Learner

The Verdict: Orders-of-Magnitude Gap

Comparable data throughput (same order of magnitude), yet:

Biology uses $\sim$**30,000**$\times$ **less energy** (0.7 vs 21,600 MWh) even **counting all brain functions**, not just learning.

We need architectures and algorithms that extract meaning from continuous event streams.

The Opportunity: Neuromorphic Gatekeepers

The Problem: Blind Processing

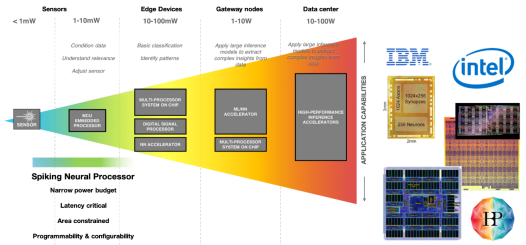
Today's Cloud AI is a "dumb pipe":

- **Redundant** raw data streams.
- **Kilowatts** on static information.
- Relies on **offline**, frozen models.

The Alternative: Intelligent Gatekeepers

The same principles that make biological learning efficient **event-driven, local, online** also enable efficient inference.

- **Reject Redundancy**.
- **Learn at the Edge**.
- **Continual adaptation**.



Innatera Nanosystems B.V., [Corradi et al, ISLPED, 2026]

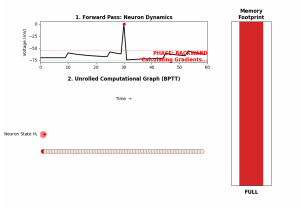
"The key is learning at the edge on continuous stream of compressed data."

Spiking or Event-Driven NN are the natural substrate — but how do we train them with local, online rules?

The Problem: Current Training for SNN is Inefficient

Standard: Backpropagation Through Time (BPTT)

- Global Error Propagation: Needs entire network state.
- Time Unrolling: Must store history of all activations.
- The Result: Massive memory footprint $\mathcal{O}(T \cdot H \cdot L)$.



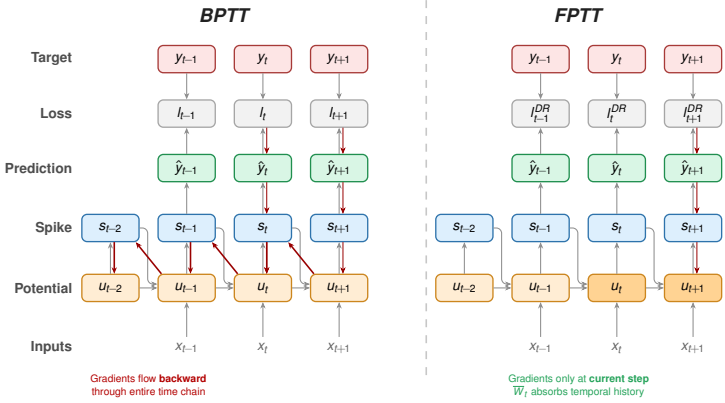
[Animation Link]

The Memory Bottleneck

- Memory is the primary constraint.
- Latency cannot tolerate backward locking.

TU/e

Forward Propagation Through Time (FPTT) vs BPTT



Variables: u_t = membrane potential, s_t = spike, $\hat{y}_t$ = prediction

Key: FPTT replaces temporal backprop chain with dynamic regularization $l_t^{DR} = \mathscr{L}_t + \mathscr{R}(\overline{W}_t)$

[B. Yin, F. Corradi, S. Bohte, Nature Machine Intelligence, 2023]

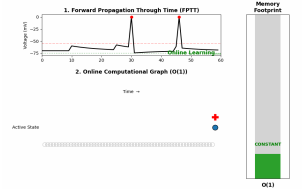
TU/e

Forward Propagation Through Time (FPTT)

$$l_t^{dyn} = \underbrace{\mathscr{L}_t}_{\text{Instantaneous Loss}} + \underbrace{\mathscr{R}(\overline{W}_t)}_{\text{Dynamic Regularization}} \tag{1}$$

- **Instantaneous Loss ($\mathscr{L}_t$):** The standard objective (e.g., Cross-Entropy) measuring the gap between prediction $\hat{y}_t$ and target y_t.
- **Dynamic Regularization ($\mathscr{R}_t$):** A penalty term based on the running average of past weight updates ($\overline{W}_t$). It forces the network to reach a consensus on optimal weights.

Result: Gradients depend only on current state $\rightarrow \mathscr{O}(H \cdot L)$ memory.



[Animation Link]

FPTT: Dynamic Loss & Update Rule

2. The Gradient Update

Unlike BPTT, the gradient for FPTT does not depend on a chain of history. The update is computed online as:

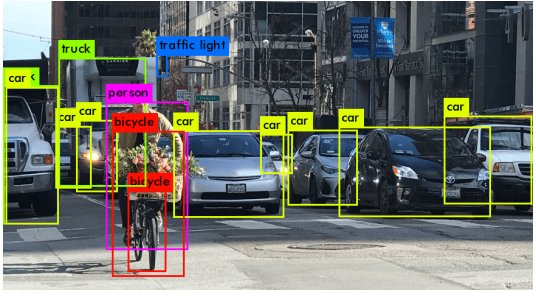
$$\frac{\partial l_t^{dyn}}{\partial W} = \frac{\partial l_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial u_t} \cdot \frac{\partial u_t}{\partial W} \tag{2}$$

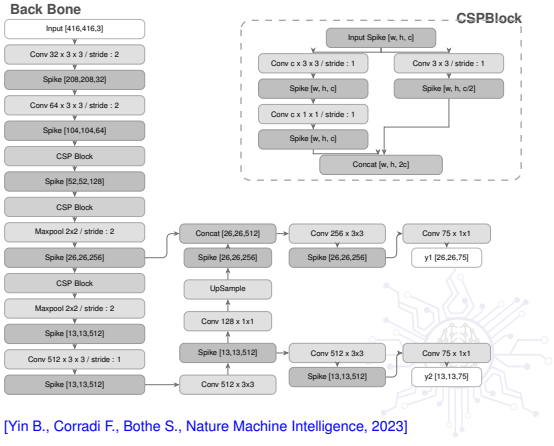
- **No Time Dependency:** The dynamic regularization absorbs temporal history into $\overline{W}_t$, so the gradient at time t depends only on current states (u_t, s_t), no chain through $u_{t-1}, u_{t-2}, \ldots$
- Eliminates vanishing/exploding gradients over time.

Deep Spiking Neural Networks Models (SpyYoloV4)

SpyYoloV4

- 6.2 M spiking neurons
- Conv, fully connected, cross-stage partial subnets
- 14 M parameters
- Approaching DNNs

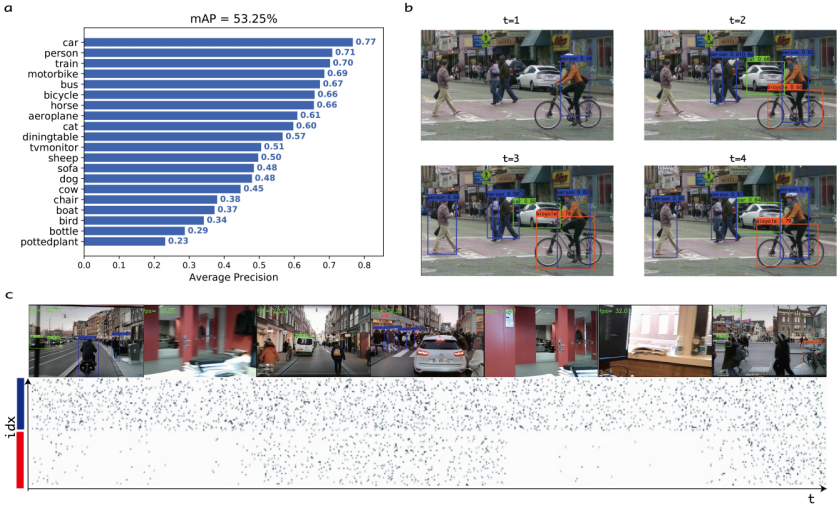




[Yin B., Corradi F., Bothe S., Nature Machine Intelligence, 2023]

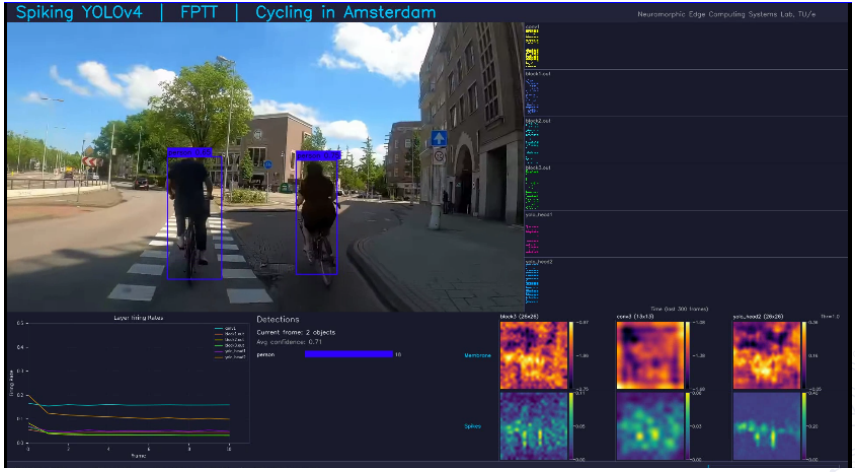
TU/e

FPTT Performance: Scaling to Deep Networks



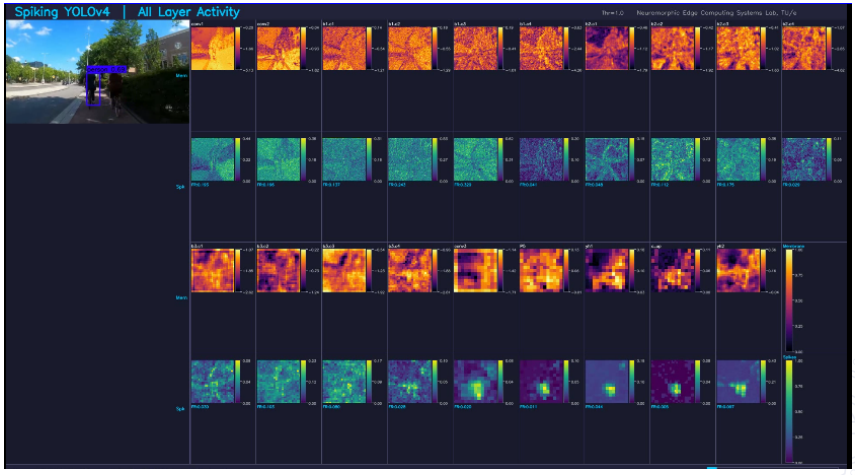
[B. Yin, F. Corradi, S. Bohte, 2023 NMI]

FPTT Performance: Scaling to Deep Networks



[B. Yin, F. Corradi, S. Bohte, 2023 NMI]
[Animation Link]

FPTT Performance: Scaling to Deep Networks



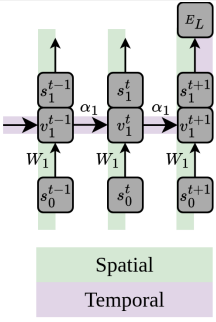
[B. Yin, F. Corradi, S. Bohte, 2023 NMI]

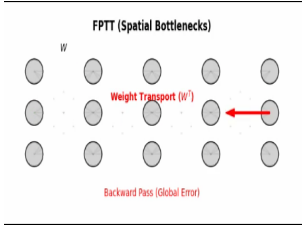
[Animation Link]

The Spatial Bottleneck

FPTT Solved Time, but...

- We still have **Spatial Credit Assignment** issues.
- Update Locking: Layers must wait for the forward pass to finish.
- Weight Transport: Backward weights must match forward weights (biologically implausible).

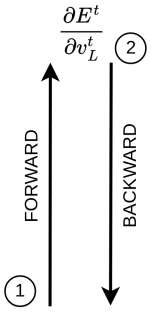




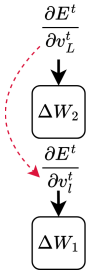
[Animation Link]

We need a solution that is Local in Time AND Space.

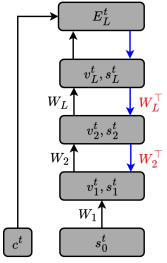
Spatial Credit Assignment Bottlenecks and Mitigations



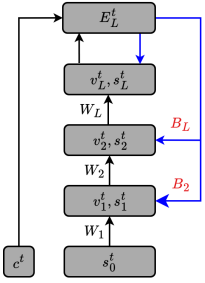
Update locking



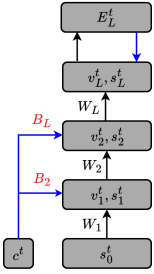
Sequential updates



Weight transport



Direct feedback alignment

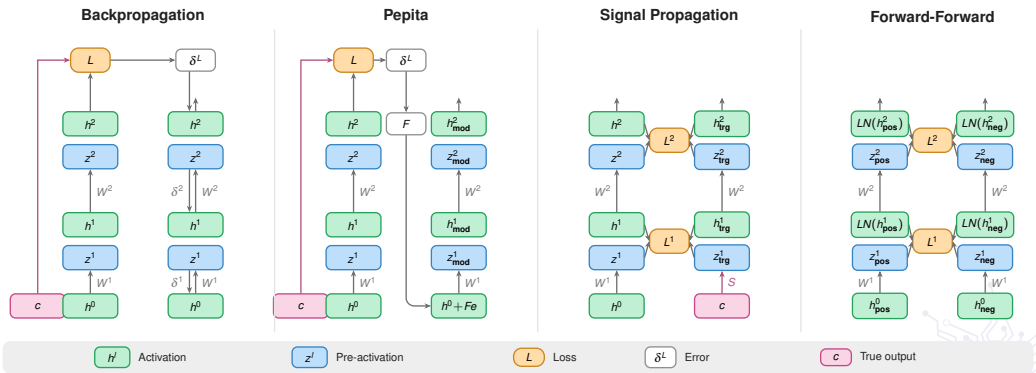


Direct-random-target projection

DFA [Nøkland A, 2016,] [Lee J. et al., 2020] , DRTP [Frenkel C. et al, 2021], showing potential but..

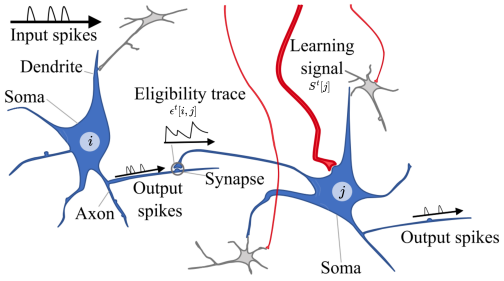
- Accuracy gap vs. BP $\rightarrow$ limited to small datasets
- Both require extra projection matrices (B_l)

Spatial Credit Assignment: Forward-Only Algorithms (ANN)



Dellaferrera & Kreiman, ICML 2022 | Kohan et al., TNNLS 2023 | Hinton, arXiv 2022

The Paradigm Shift: From Backpropagation to Local Learning



Adapted from OSTL, Bohnstingl et al., 2022

Lillicrap et al., Nature Reviews Neuroscience (2020)

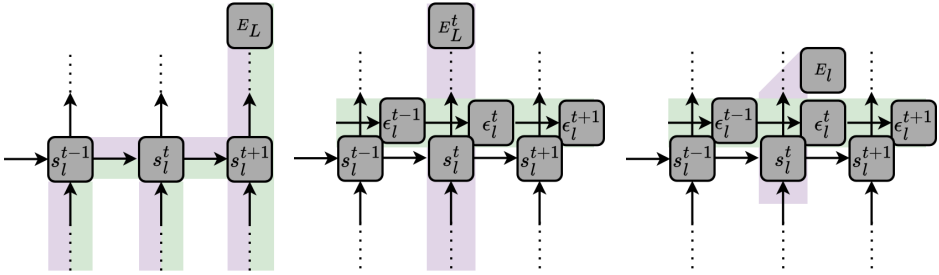
The Vision: Activity-Based Learning

- **Activity, Not Error:** the brain uses feedback to induce *activity changes*, not explicit error derivatives.
- **True Locality:**
  - **Space:** No Weight Transport.
  - **Time:** Continuous updates (No Locking).

  Efficiency: Reduces memory complexity to linear $O(H \cdot L)$ via eligibility traces.

Strict backpropagation is biologically problematic, the brain may implement its core principles by using feedback connections to induce neural activity differences that locally approximate error signals.

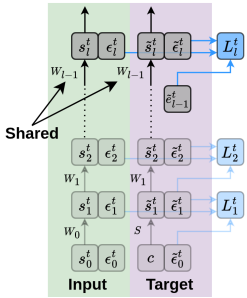
Traces Propagation (TP)



Fully Local Learning

- **No Backpropagation**: Forward-only passes.
- **Local Loss**: Layer-wise contrastive objectives.
- **Eligibility Traces**: Capture temporal dynamics at the synapse.

Traces Propagation: How it Works



spike: $s_l^t = \Theta(v_l^t - V_{th})$

trace: $e_l^t = \beta e_l^{t-1} + s_l^t$

spike: $\tilde{s}_l^t = \Theta(\tilde{v}_l^t - V_{th})$

trace: $\tilde{e}_l^t = \beta \tilde{e}_l^{t-1} + \tilde{s}_l^t$

$$L_l^t = CE(\underbrace{e_l^t(\tilde{e}_l^t)^T}_{y_t^l}, \underbrace{Softmax(\tilde{e}_{l-1}^t(\tilde{e}_{l-1}^t)^T)}_{z_t^l})$$

$$\frac{dL_l^t}{dW_{l-1}} = \underbrace{(y_l^t - z_l^t)}_{\text{Modulation signal}} \left[\left(e_l^t \Theta'(v_l^t - V_{th}) s_{l-1}^t\right) + \left(\tilde{e}_l^t \Theta'(\tilde{v}_l^t - V_{th}) \tilde{s}_{l-1}^t\right) \right]$$

Mechanism

- **Modulation Signal**: Drives input traces to align with target traces for same-class examples.
- **Divergence**: Pushes traces apart for different classes.
- **Result:** Linear separability emerges locally at each layer.

Traces Propagation: Layer-wise Contrastive Loss

1. The Local Loss Function (E_l^t)

Trace Propagation (TP) defines a local loss at each layer l and time t, aiming to align input traces with target traces:

$$E_l^t = -\sum_{b,b'} y_l^t[b,b'] \log(\text{Softmax}(z_l^t[b,b'])) \tag{3}$$

- **Similarity Score (z_l^t):** Measures alignment between the input trace of example b and the target trace of example b' via dot product: $z_l^t = \varepsilon_l^t(\tilde{\varepsilon}_l^t)^T$.
- **Target Distribution (y_l^t):** A probability distribution derived from the previous layer's target traces, ensuring examples from the same class remain close.

Traces Propagation: Local Update Rule

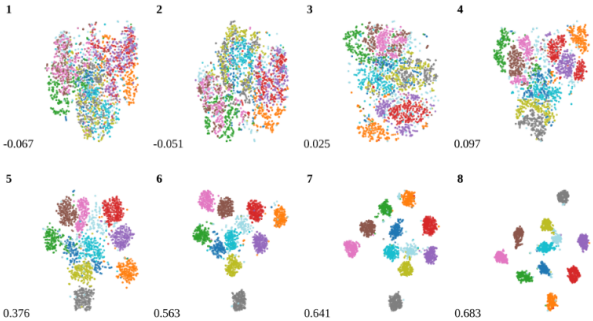
2. The Gradient Update

The weight update is computed using only locally available information (traces and spikes), eliminating the backward pass:

$$\Delta W_l \propto \underbrace{\left(\text{Softmax}(z_l^t) - y_l^t\right)}_{\text{Modulation Signal}} \cdot \underbrace{\tilde{\varepsilon}_l^t}_{\text{Post-syn. Trace}} \cdot \underbrace{\Theta'(v_l^t)}_{\text{Surrogate}} \cdot \underbrace{s_{l-1}^t}_{\text{Pre-syn. Spike}} \tag{4}$$

- **Modulation Signal:** Pushes input traces of the same class together and different classes apart (Green/Red arrows).
- **Three-Factor Rule:** The update combines a global (layer-wise) error signal with local pre- and post-synaptic activity.

Intuition: Separation in Feature Space



t-SNE Visualization

- Visualizing input traces at different layers.
- **Layer 1:** Mixed, hard to separate.
- **Layer 8:** Clear clusters for 11 classes.
- Experimental results: local rules can learn global representations.

Traces propagation Complexity and Memory

Table: $L \rightarrow$ layers, $H \rightarrow$ neu, $T \rightarrow$ seq. length, and $O \rightarrow$ out classes.

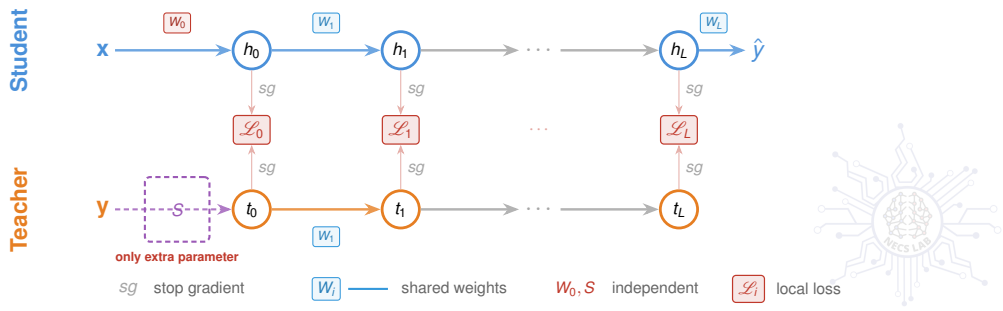
Model	Update Locking	Weight Transport	Time Local	Space Local	Space Complexity	Time Complexity	Auxiliary Matrices
BPTT	✗	✗	✗	✗	TLH	TLH^2	–
E-prop [Bellec 2020]	✗	✗	✓	✗	LH^2	LH^2	–
E-prop(*rnd*) [Bellec 2020]	✗	✓	✓	✗	LH^2	LOH	LOH
OSTL [Ortner 2022]	✗	✗	✓	✗	LH^2	LH^2	–
OSTL(*rnd*) [Ortner 2022]	✗	✓	✓	✗	LH^2	LOH	LOH
S-TLLR [Apolinario 2023]	✗	✗	✓	✗	LH	LH^2	–
DECOLLE [Kaiser 2020]	✓	✓	✓	✓	LH	LOH	LOH
ETLP [Quintana 2024]	✓	✓	✓	✓	LH	LOH	LOH
OSTTP [Ortner 2023	✓	✓	✓	✓	LH^2	LOH	LOH
TESS [Apolinario 2023]	✓	✓	✓	✓	LH	LOH	LOH
TP (ours)	✓	✓	✓	✓	LH	LH	OH

TU/e

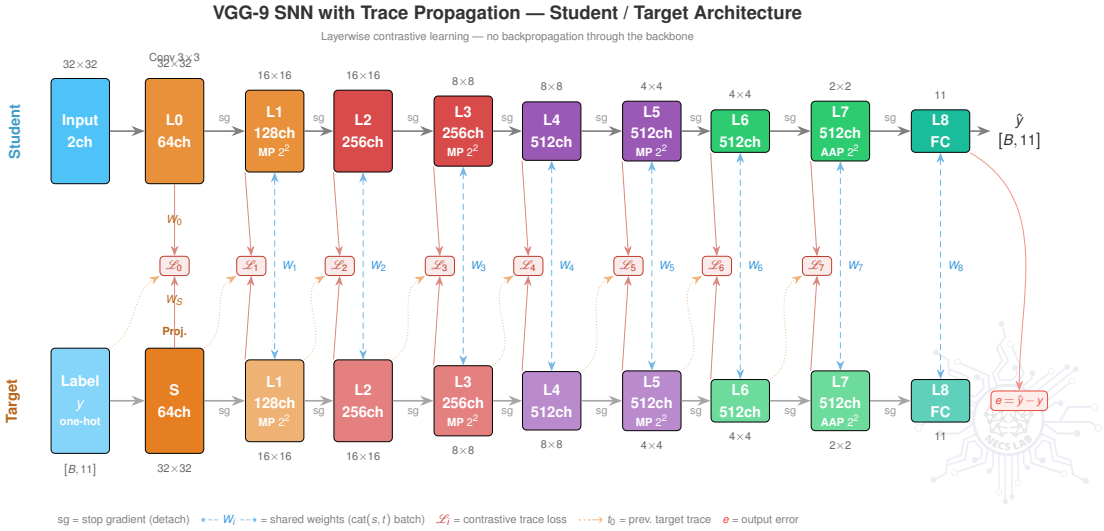
Completing the Picture: The Complexity of Learning

Addressing the Learning Gap

TP only requires one extra matrix S during training. The teacher network shares the same exact weight as the student and it exists only during training.

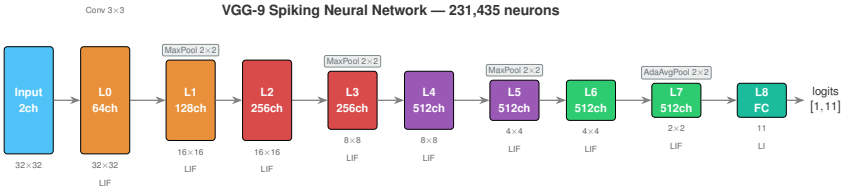


Quantitative Results: TP on Dynamic Data and Deep SNNs



TU/e

Quantitative Results: TP on Dynamic Data and Deep SNNs



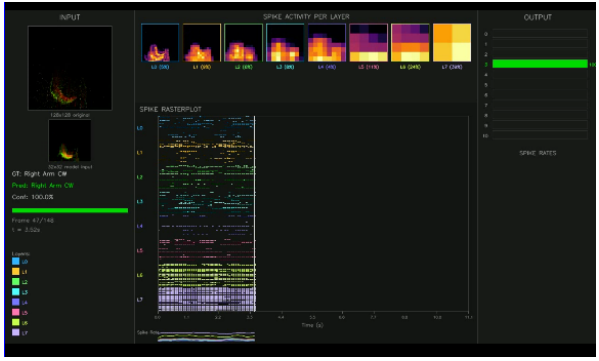
Model	Architecture Type	Neuron Type	Local Learning	Time Steps	Number of Epochs	Test Accuracy
			DVS-GESTURE			
BPTT (ours)	VGG-9	LIF	✘	20	200	98.56 ± 0.15
OTTT [7]	VGG-9	LIF	Partial (time)	20	200	96.88
S-TLLR [9]	VGG-9	LIF	Partial (time)	20	200	98.48 ± 0.37
DECOLLE [12]	3-CONV	LIF	✔	1800	200	95.54 ± 0.16
TESS [13]	**VGG-9**	**LIF**	✔	**20**	**200**	$\mathbf{98.56 \pm 0.31}$
TP(**ours**)	VGG-9	LIF	✔	20	200	98.18 ± 0.15
			DVS-CIFAR10			
BPTT [13]	VGG-9	LIF	✘	10	200	76.40 ± 0.66
OTTT [7]	VGG-9	LIF	Partial (time)	10	200	76.27 ± 0.05
S-TLLR [9]	VGG-9	LIF	Partial (time)	10	200	75.14 ± 1.37
TESS [13]	**VGG-9**	**LIF**	✔	**10**	**200**	$\mathbf{75.00 \pm 0.65}$
TP(**ours**)	VGG-9	LIF	✔	10	200	71.39 ± 0.19

[L. Pes, B. Yin, S. Stuijk, F. Corradi, IOP NCE, 2026]

Performance

- **DVS Gestures**: Competitive with BPTT.
- **DVS CIFAR-10**: Scalable to complex vision tasks.
- **Complexity**: $\mathcal{O}(H \cdot L)$ memory (vs. $\mathcal{O}(T \cdot H \cdot L)$ for BPTT).
- **No update Locking** - unlike FPTT.
- **No weight transport** - unlike FPTT.

Traces Propagation: Spiking VGG-9 for DVS Gesture Example



[Animation Link]

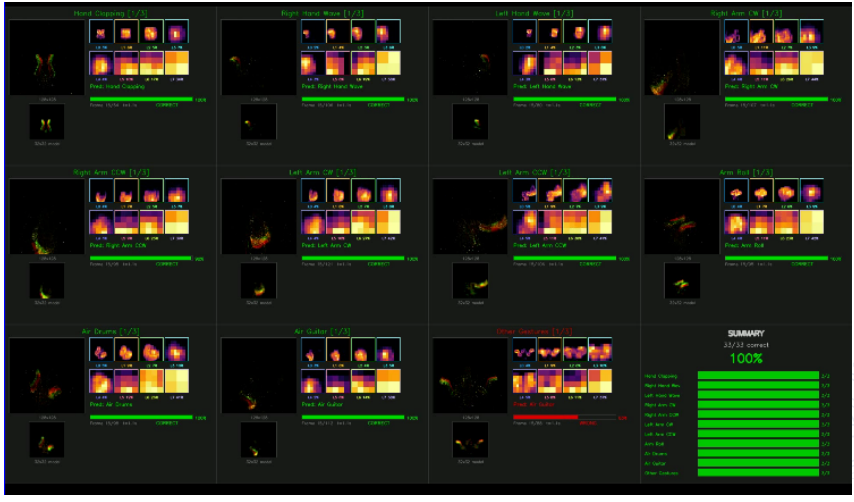
[L. Pes, B. Yin, S. Stuijk, F. Corradi, IOP NCE, 2026]

Layer, Type	Ch. × Spatial	# Neu.
L0, Conv+Pool	$64 \times 32 \times 32$	65,536
L1, Conv+Pool	$128 \times 16 \times 16$	32,768
L2, Conv	$256 \times 16 \times 16$	65,536
L3, Conv+Pool	$256 \times 8 \times 8$	16,384
L4, Conv	$512 \times 8 \times 8$	32,768
L5, Conv+Pool	$512 \times 4 \times 4$	8,192
L6, Conv	$512 \times 4 \times 4$	8,192
L7, Conv+Pool	$512 \times 2 \times 2$	2,048
L8, FC	11	11
Total		231,435

- heatmaps: average across all channels
- raster: 50 randomly sampled neurons per layer

TU/e

Traces Propagation: Spiking VGG-9 for DVS Gesture



[Animation Link]

TU/e

Real-World Adaptation: Fine-Tuning and One-Shot Learning

[Animation Link]

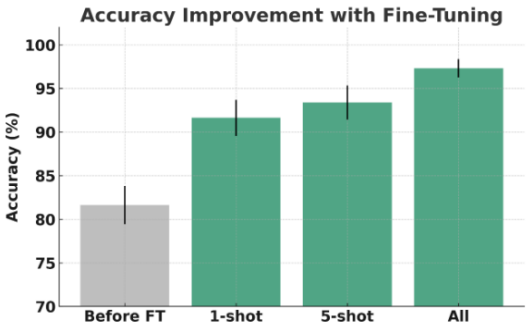
[L. Pes, B. Yin, S. Stuijk, F. Corradi, IOP NCE, 2026]

- TP-MLP: 1 hidden layer, 450 recurrent LIF neurons, 36 output classes
- Input: 40-bin mel-spectrogram + $1^{st}/2^{nd}$ order deltas = 120-dim, 101 time steps (1s audio)
- Pre-trained on 84,527 samples from 2,112 speakers → 81% on new speaker
- 1-shot adaptation: 26 samples (1 per class), 3 epochs with TP → **96%**
- Test set: same speaker, different recordings (80/20 split)
- GSC v0.02 dataset (35 words + unknown)

Real-World Adaptation: On-Device Fine-Tuning

The Use Case

- Pre-train on general data (Cloud).
- Deploy to Edge.
- **Adapt Locally**: Fine-tune for the specific user using Trace Propagation.



Example: Google Speech Commands adaptation recovers accuracy for new users with just a few shots.

Conclusions Part I

From Global Error to Local Activity and Learning

- **FPTT** eliminates temporal unrolling: $\mathcal{O}(T \cdot H \cdot L) \rightarrow \mathcal{O}(H \cdot L)$ memory, enabling deep online SNN training.
- **Trace Propagation** removes spatial backprop: forward-only, no update locking, no weight transport — fully local in space & time.
- **On-device fine-tuning** with TP enables few-shot adaptation at the edge (speech, gestures, bio-signals).

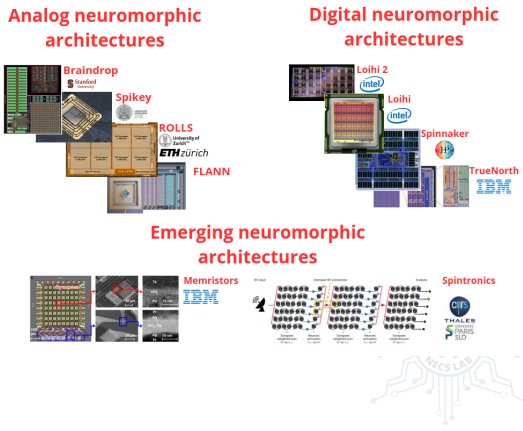
Next Step: Silicon

- **Algorithm-Hardware Co-Design**
- **Exploiting Extreme Sparsity**
- **Native On-Chip Learning**

NM Hardware Spectrum & System Efficiency & Development Cycles

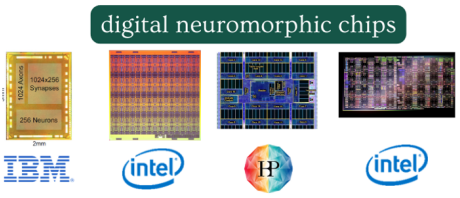
Neuromorphic Computing Systems

- Digital Neuromorphic Hardware
  - Software models.
  - Easy to design, scalable.
- Mixed-Signal Neuromorphic Hardware
  - Bio-realistic models.
  - Hard to design, noise, mismatch.
- Emerging Technologies
  - Non-volatile memories.
  - Spintronics.
  - New design styles (e.g., in-memory computing).



TU/e

Neuromorphic Computing Hardware, many approaches!

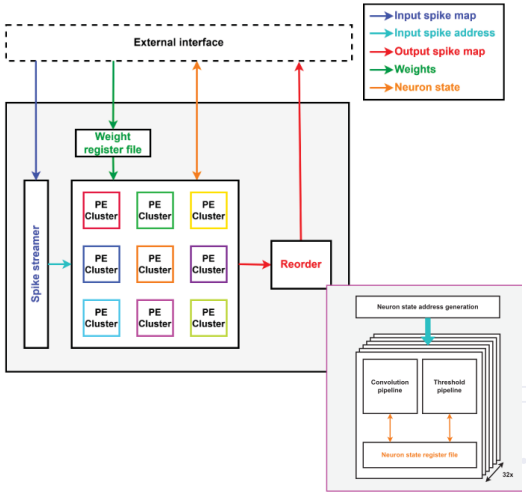


Fully digital:

- Benefits:
  - High-fidelity (SW vs. HW)
  - Denser than analog
  - Fast design
- Downsides:
  - Many little von Neumann cores
  - Large on-chip mem

Mega: Digital SNN Accelerator for SpyYoloV4 (TU/e)

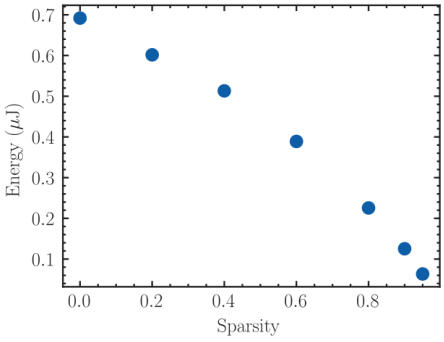
- 3x3 spiking convolution
  - Event-based: operations happen when there is a spike
  - Each PE cluster updates 32 neurons in parallel
- Memory hierarchy
  - Small, fast memories near the PEs
  - Larger memory for flexibility
- 0.501 pJ/SOP
  - Post-synthesis in GF 22 nm
  - 625 MHz @ 0.8V

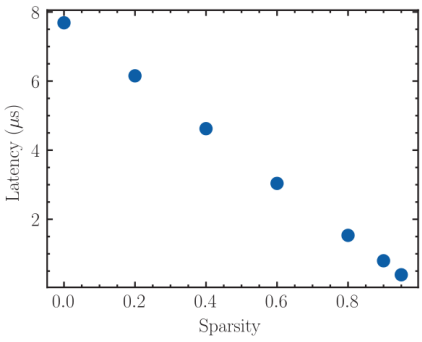


CONVOLVE

[Luiken R., unpublished]

Mega: What Happens if Spiking is Sparse?





- Energy and latency for a 96x48 spike map.
- SpyYoloV4 application.

[Luiken R., unpublished]

Digital Neuromorphic Chips: Strengths & Bottlenecks

Key Advantages

- **1:1 Equivalence:** Exact software-to-hardware mapping.
- **Standard Scaling:** Fast development via standard EDA & CMOS nodes.
- **Flexibility:** Supports deep, complex networks (e.g., Spiking YOLO).

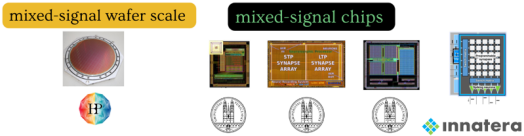
Design Bottlenecks

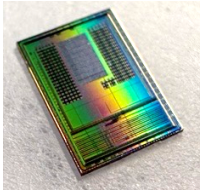
- **Memory Wall:** Massive SRAM needed for states and weights.
- **Data Movement:** Memory fetches dominate energy over compute.
- **Clock Overhead:** Power-hungry clock trees (motivates async).

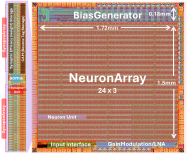
The Takeaway

*Digital provides the precision for deep SNNs, but efficiency requires **memory-minimal algorithms** (e.g., FPTT) and **sparsity-aware architectures**.*

Neuromorphic Computing Hardware, many approaches!







TU/e TU/e

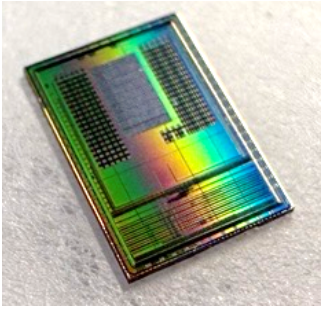
[Max K. et al, IOP NCE, 2025]

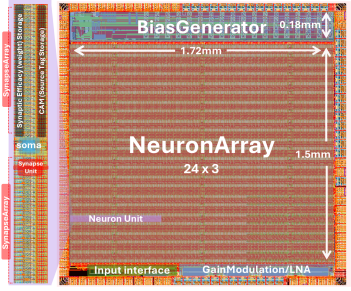
[Ye S. et al, ISCAS, 2026]

Analog mixed-signal:

- Benefits:
  - High level of biorealism
  - Long time scales
  - Ultra-low-power
- Downsides:
  - Sensitivity to noise (PVT)
  - Expensive (area & design time)

Bio-sensing (olfaction) with a Tiny Spiking Neural Network





BioPad: SkyWater 130 nm

- Low-Noise Amplifier (Olfactory Sensing)
- Tiny SNN (16 neu, 512 syn)
- RISC-V (mcu)

[Max K. et al, IOP NCE, 2025]

AURA: IHP 130nm

- 8 analog inputs (LNA+filter+Event-Coding)
- 72 neurons + 64K syn (4-bit)
- Time continuous & barrier synch.

[Ye S. et al, ISCAS (in-press), 2026]

Mixed-Signal Analog Neuromorphic Chips: Strengths & Bottlenecks

Key Advantages

- **Ultra-Low Power:** Continuous inference at μW or nW energy budgets.
- **Native Time Constants:** RC circuits natively match slow bio-signals (e.g., ECG, olfaction) without clocks.

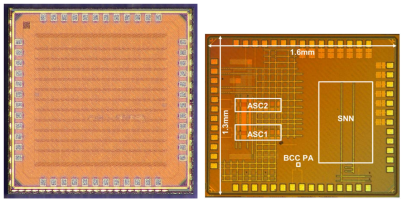
Design Bottlenecks

- **Rigid Design:** Highly application-specific; lacks general flexibility.
- **Calibration Overhead:** Mismatch requires extensive post-fab tuning.
- **PVT Sensitivity:** Algorithms must tolerate physical noise.

The Takeaway

*Analog excels at the sensor interface, but its noise sensitivity demands **robust algorithms, on-chip learning, and self-calibration**.*

Synthesizable mixed-signal

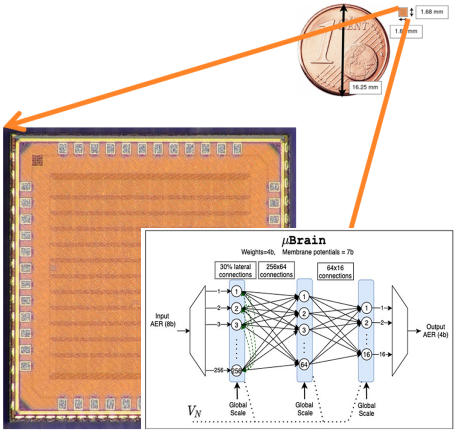


[Stuijt J. et al, Front. Neurosc., 2021]
[He Y. et al, JSSC, 2022]

Synthesizable mixed-signal:

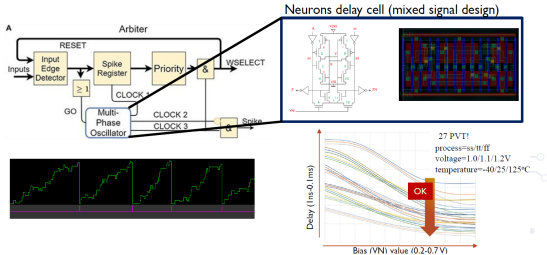
- Benefits:
  - High-fidelity (SW vs. HW)
  - Fully asynchronous (sensor driven)
  - Fast development-to-deployment cycle
  - Ultra-low-power
- Downsides:
  - Larger area (than MCUs)
  - Difficult to train (using BPTT!)

μBrain: Asynchronous Ultra-Low-Power Processor



μBrain

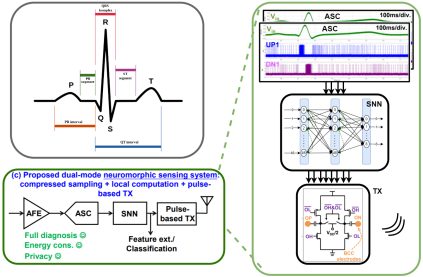
- 1.68 mm^2 (60% syn, 4% neu, 36% arb)
- 300 neurons, 200,000 synapses
- $\leq$ 100 μW

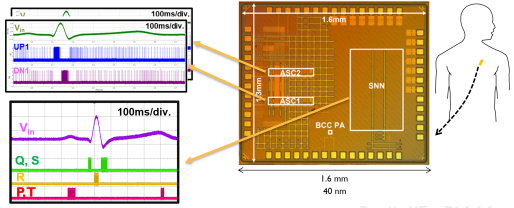


[Stuijt et al., Front. Neurosc., 2021]

TU/e

μBrain Application: Bio-signal Processing





	This work		[2] Luo, TBIOCAS'19	[3] Kim, TBIOCAS'14		Reveal LINQ [10]
Tech. (nm)	40		130	180		N.A.
Supply (V)	0.9/1/1.1		0.9	1.2		N.A.
System	ADC+SNN+TX		AFE+ADC+ DSP+TX	AFE+ADC+DSP+TX		N.A.
Applications	Insertable		Wearable	Wearable		Insertable
	Daily monit.	Diagnostic		Daily monit.	Diagnostic	
ADC architecture	LC		SAR	SAR		N.A.
ADC sample rate × nr. of bits (bps)	~ 140 (event-driven) × 2b =280		256 × 12b =3072	512/64 (Adaptive) × 12b =877 (7× compression)		256× 16b =4096
ADC ENOB (bits)	5	9.6	10.32	N.A.		N.A.
TX technology	Pulse based BCC		QPSK/BFSK	BLE		Bluetooth
Power con. (μW)						
TX	**1.2**	**1.5**	69.4~317.4[B]	1000	13302	-
DSP/NN/Memory	13	15	-	42	0	-
ADC	14	34	4.6	18	38	-
AFE	-		4.6	21	21	-
Total (excl. AFE)	**28.2**	**50.5**	74~322	1060	13340	N.A.[C]
Nr. of ECG chan.	2		2	3		N.A.
Core area (mm^2)	**0.32[A]**		6.25	23.5		N.A.
System area (mm^2)	**72**		N.A.	~500		322[D]

TU/e

The Design Bottleneck: Automating Neuromorphic Synthesis

Challenges in Custom Silicon

- **High NRE Costs:** Complex SoC integration for data routing and control.
- **Manual Layout:** Asynchronous (QDI) and mixed-signal logic demand bespoke, by-hand design.
- **Toolchain Friction:** Standard synchronous EDA tools struggle with unclocked, event-driven logic.

Emerging Enablers

- **Open-Source EDA:** Maturation of highly customizable, scriptable toolchains (Yosys, OpenROAD).
- **Domain-Specific Synthesis:** The success of High-Level Synthesis (HLS) mapping Python directly to hardware.

The Objective

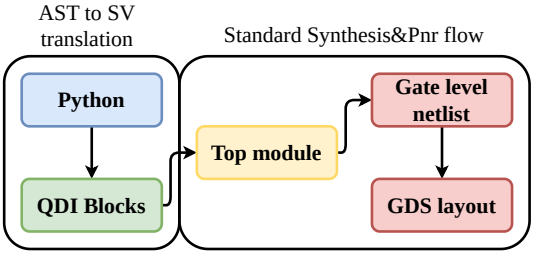
How can we systematically bridge the semantic gap between event-driven algorithms and asynchronous silicon?

A Framework for Synthesizing Neuromorphic Asynch. Hardware

The Apocalypse Framework

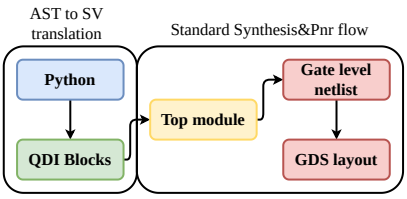
Goal: Develop a domain-specific compilation framework to synthesize neuromorphic hardware directly from high-level behavioral models (e.g., Python) down to physical layout.

Apocalypse: An end-to-end compilation toolchain that translates high-level Python behavioral models into synthesizable SystemVerilog.



- Syntax-directed mapping strategy
- Event-driven neuromorphic constructs
- Quasi-Delay-Insensitive (QDI) standard cells
- And mixed-signal macros

Apocalypse: From Python to Asynchronous Silicon



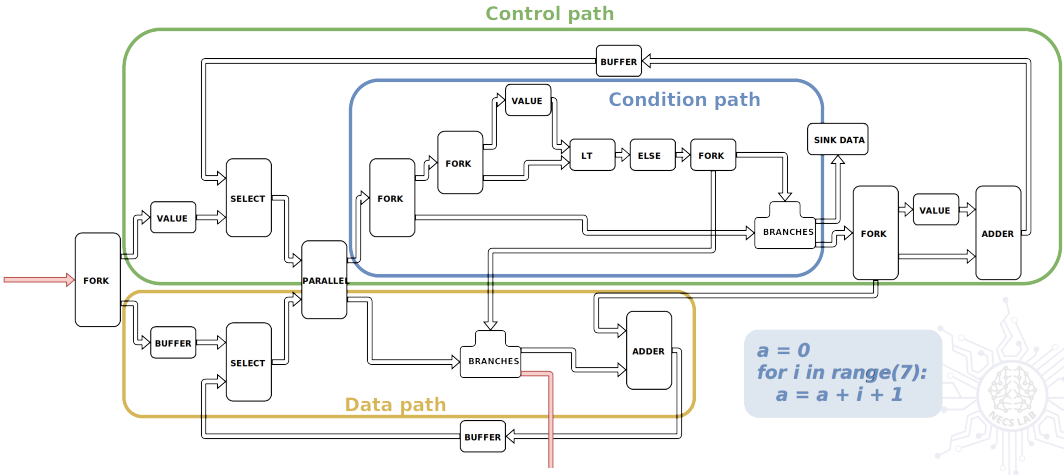
Automated Translation (3 Phases)

1. **AST Analysis**: Parse Python, extract variable lifetimes and read/write dependencies
2. **Syntax-Directed Mapping**: Map operations to spatial QDI dataflow graph using pre-verified blocks
3. **Netlist Emission**: Flat SystemVerilog → standard synthesis & P&R

QDI Standard Cell Library (27 blocks)

- **Data handling**: QDI_buffer, QDI_sink_data, QDI_fork
- **Logic ops**: QDI_adder, QDI_lt, QDI_le, QDI_equal
- **Flow control**: QDI_select (merge), QDI_branch, QDI_parallel
- **Branching**: QDI_else_condition (token steering for if/else)

QDI Example: For-Add-Loop Structure



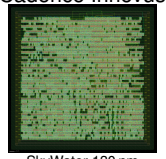
QDI Example: For-Add-Loop Structure

Python → 22 QDI Blocks

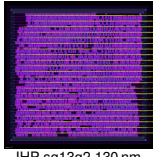
```
for i in range(7):
    a = a + i + 1
```

- **QDI_select**: loop entry (merges init + feedback)
- **QDI_adder**: computes `a + i + 1`
- **QDI_parallel**: synchronizes loop variables
- **QDI_branch**: loop termination steering

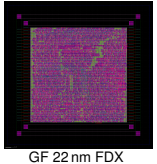
Cadence Innovus



SkyWater 130 nm

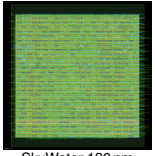


IHP sg13g2 130 nm

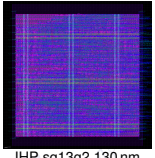


GF 22 nm FDX

Librelane Open-Source



SkyWater 130 nm



IHP sg13g2 130 nm

N/A

GF 22 nm FDX

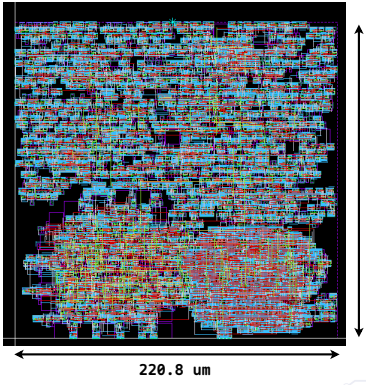
QDI Example: Integrate-and-Fire Neuron (16 Synapses)

IF Neuron Model

```
while potential < threshold:
  if input_spike[i]:
    potential += w[i]
  if potential >= threshold:
    fire = 1; potential = 0
```

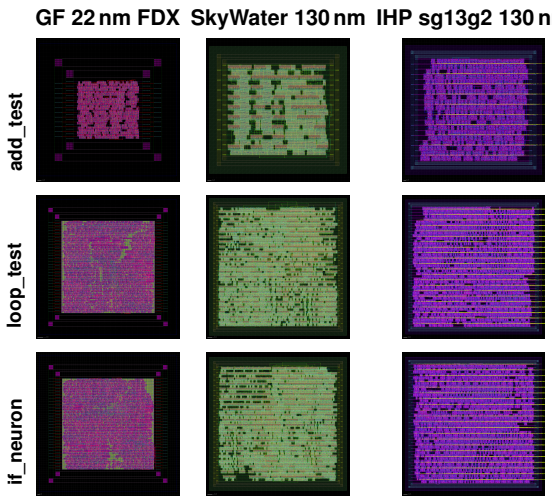
QDI Circuit (20 blocks)

- Fully asynchronous: spikes when $V_{\text{mem}} \geq \theta$
- 16 synaptic inputs with accumulation
- Layout: 230.8×64 μm (IHP 130 nm)
- $\sim$48 ns per iteration (post-synth SDF)

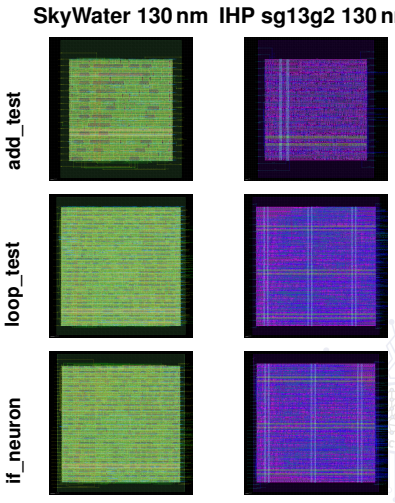


IHP sg13g2 130 nm (Librelane Open-Source)

Layout Gallery: All Technologies Two Flows



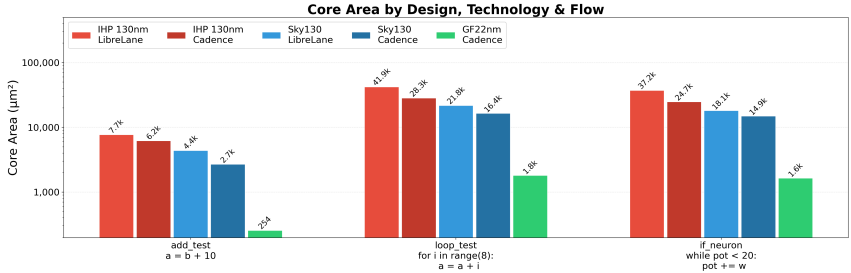
All layouts: Cadence Genus + Innovus | 0 DRC violations | PostLayout Sim OK

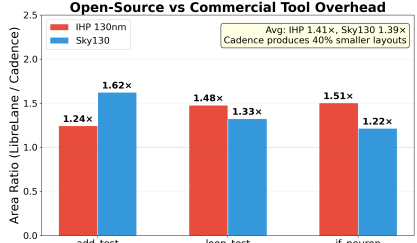


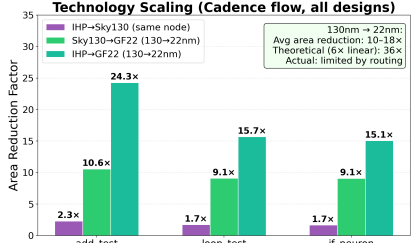
All layouts: Librelane + OpenROAD | 0 DRC violations | PostLayout Sim OK

TU/e

Apocalypse: QDI Async. Circuit Scaling Technologies & Toolflows







Apocalypse: QDI Async. Circuit Scaling Technologies & Toolflows

Analysis Results

- Cadence vs open-source: Cadence (Genus+Innovus) on average 40% smaller than the open-source LibreLane flow (Yosys+OpenROAD) on the same PDK.
- Technology scaling (130nm $\rightarrow$ 22nm): From Sky130 130nm to GF22 22nm: 9-16x area reduction (Cadence flow). Theoretical scaling from $(130/22)^2 \sim 36$x is not fully realized (routing and I/O overhead).
- IHP130 vs Sky130 (same node): Sky130 achieves 1.7x smaller area than Cadence on IHP, custom c_element and MUTEX macros that reduce cell count significantly (e.g. 176 vs 322 stdcells for add_test).

Recap

Algorithms: From Global Error to Local Activity

- FPTT: Online training, $\mathcal{O}(T \cdot H \cdot L) \rightarrow \mathcal{O}(H \cdot L)$ memory.
- Trace Propagation: Fully local in space & time — no backprop, no weight transport.
- On-device fine-tuning: Few-shot adaptation at the edge.

Hardware Implementations Across the Stack

- Digital event-based (Mega): sparsity exploitation, 0.501 pJ/SOP (GF 22 nm).
- Mixed-signal (μBrain, AURA): ultra-low-power always-on sensing (≤ 100 μW).

Design Automation: Apocalypse Framework

- Python $\rightarrow$ asynchronous silicon: automated QDI synthesis for neuromorphic hardware.
- Multi-PDK (IHP 130 nm, SkyWater 130 nm, GF 22 nm) — open-source & commercial flows.
- Control blocks (Branch, Merge) 3–33$\times$ more compact than existing frameworks.

Future Directions & Open Challenges

1. What is Missing in the Field?

- **True Co-Design:** Algorithms and silicon are still largely developed in silos. We lack standardized frameworks that seamlessly link continuous-time models to hardware.
- **Dynamic Benchmarks:** Over-reliance on converted static datasets (e.g., nm CIFAR-100) rather than continuous, highly temporal real-world tasks.

2. The Major Hurdles

- **The Programming Paradigm Gap:** Mapping event-driven, unclocked algorithms onto existing synchronous EDA tools is highly inefficient.
- **On-Chip Learning at Scale:** Tolerating PVT (Process, Voltage, Temperature) variations and quantization constraints while maintaining convergence stability.

Future Directions & Open Challenges

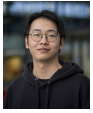
3. How Our Research Contributes

- **Algorithmic Locality:** FPTT and Trace Propagation completely eliminate spatial weight transport and temporal unrolling, making on-device learning viable.
- **Bridging the Gap:** The Apocalypse framework democratizes asynchronous hardware synthesis, drastically reducing NRE costs and design time.

4. A Call to the Community

- Embrace **Open-Source EDA & Software** (Yosys, OpenROAD, SnnTorch) to build a collaborative, scalable infrastructure for mixed-signal neuromorphic design.
- Shift focus toward **embodied AI** at the (extreme) edge (e.g., bio-signal monitoring, autonomous robotics) where SNNs could proof undeniable advantage over standard Deep Learning.

Neuromorphic Edge Computing Systems Lab



Zhanbo Shen

Lorenzo Pes

Dr. Bojian Yin

Shimeng Ye

Yvonne Vullers

Stefano Chiavazza

Noah Marti

Dr. Roel Jordans

Dr. Federico Corradi

Collaborators

- **Prof. Sander Bohte** (CWI)
- **Dr. Yao-Hong Liu** (IMEC)
- **Dr. Bojian Yin** (Chen Institute)

Interested? Reach out!

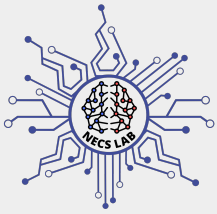
Neuromorphic Edge Computing Systems Lab

Federico Corradi

Eindhoven University of Technology

f.corradi@tue.nl, +31(0)402 472 556

Neuromorphic Edge Computing Systems Lab



Join the Team!

We have two vacancies

- **PhD in Event-Based Sensor Fusion Algorithms for Real-Time Perception and Control**
- **PhD in Event-Based Neuromorphic Hardware for Real-Time Control (FPGA)**