tinyML® EMEA

Enabling Ultra-low Power Machine Learning at the Edge

June 26 - 28, 2023

TINY ML

www.tinyML.org

Monostable Multivibrator Networks: extremely low power inference at the edge with timer neurons

L. Keuninckx[1], M. Hartmann[1], P. Detterer[2], A. Safa[1,3], I. Ocket[1]

1 imec Leuven, Belgium; 2 imec Holst Centre, Eindhoven, The Netherlands;
3 Department of Electrical Engineering, Katholieke Universiteit Leuven (KUL), Belgium.

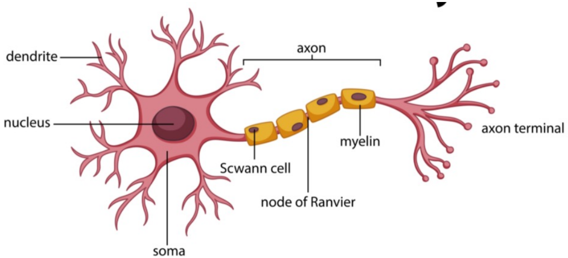
tinyML EMEA Innovation Forum - June 26-28, 2023.

What's all this neuromorphic stuff, anyway?

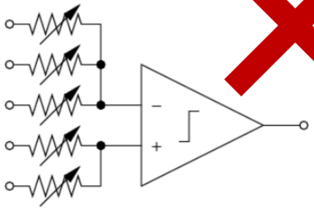
Energy/inference ↓ ⇒ applications, market opportunities ↑

- Neurons are living cells first and only then computational units
- In the end, the only thing that counts are the computational properties of the units we employ and how easy we can build & organize them in networks
- What if… **we reverse the question**:

From: how to implement neurons (+ synapses) in electronics?



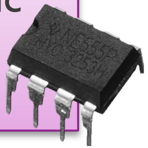
Model:
Hodgkin-Huxley,
Fitzhugh-Nagumo,
Leaky I&F,…



To: which known electronic building blocks can be useful as artificial "neuron"?

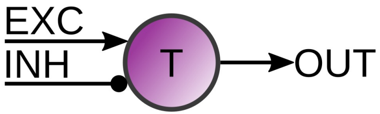
Neuron-like computational properties?

Known electronic building block

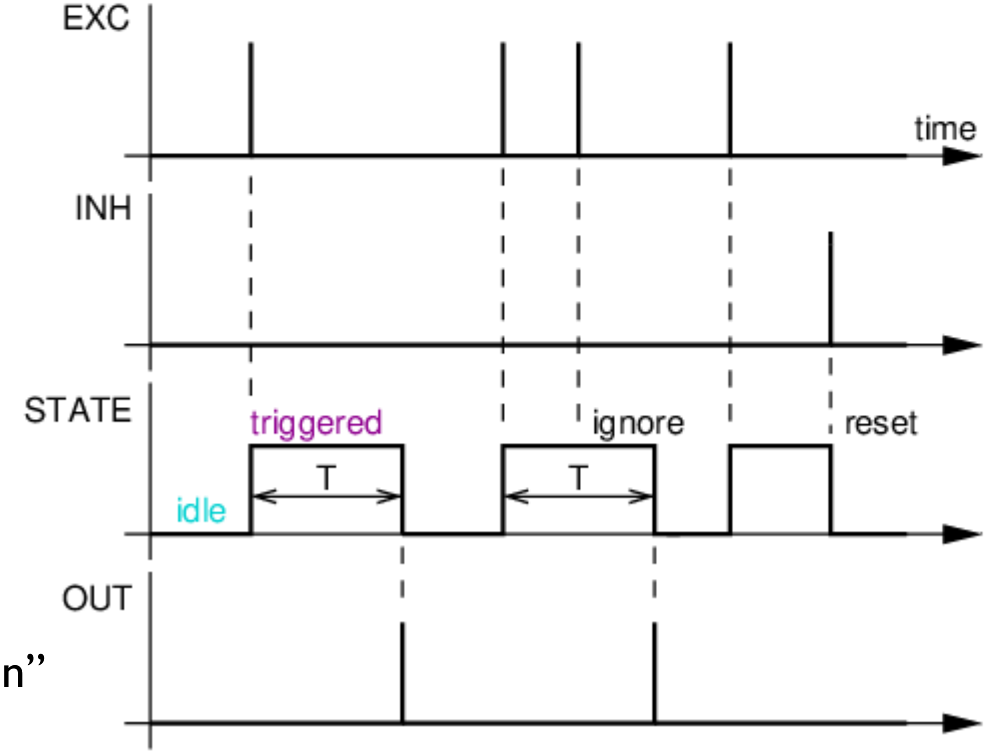


Non-retriggerable* monostable multivibrator

It's just a timer!



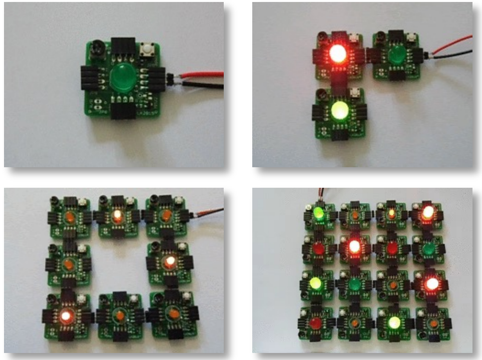
- Two possible states: idle, triggered
- Two inputs: EXCitatory and INHibitory
- If idle and EXC spike in → triggered
- *If triggered and EXC in → ignore
- After **T**: Spike and return to idle
- If triggered and INH spike in
  → return to idle, don't spike
- **OR** signals together
  → overlapping input spikes? Don't care!
- "Non-biologically inspired spiking neuron"



Motivation: MMV = just a counter in digital hardware

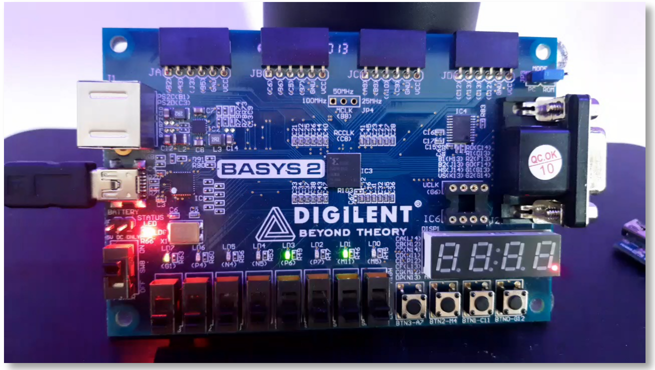
Earlier work:

- Dynamical behavior of MMV networks:
  rate equations, equilibria of recurrent networks...
- **Rate-based** backpropagation
- Many fun problems for mathematicians & physicists ☺



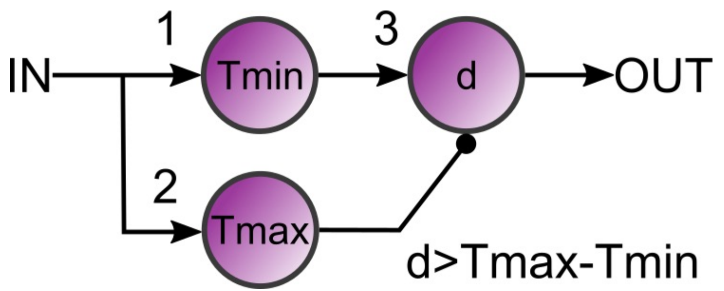
"Monostable multivibrators as novel artificial neurons", Neural Networks, 2018.

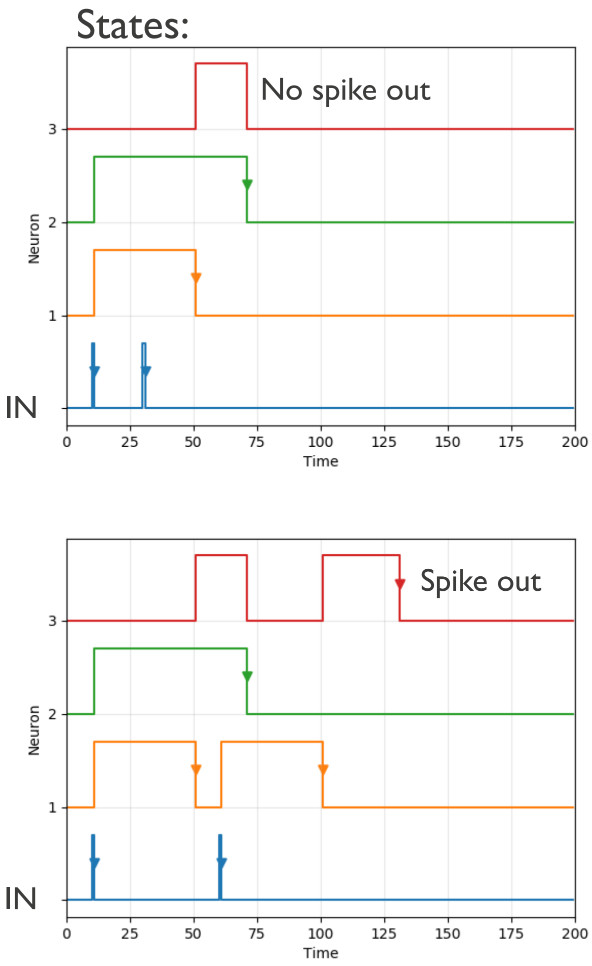
- #MMVs ≈ #Flipflops/$\log_2(T_{max})$
- Artix XC7A100T FPGA (ARTY-board) → ~10k
- ~100s of MMVs in a recurrent network:



How to put MMVs to good use in **event-driven** networks to actually solve problems?

Example: a spike-interval discriminator



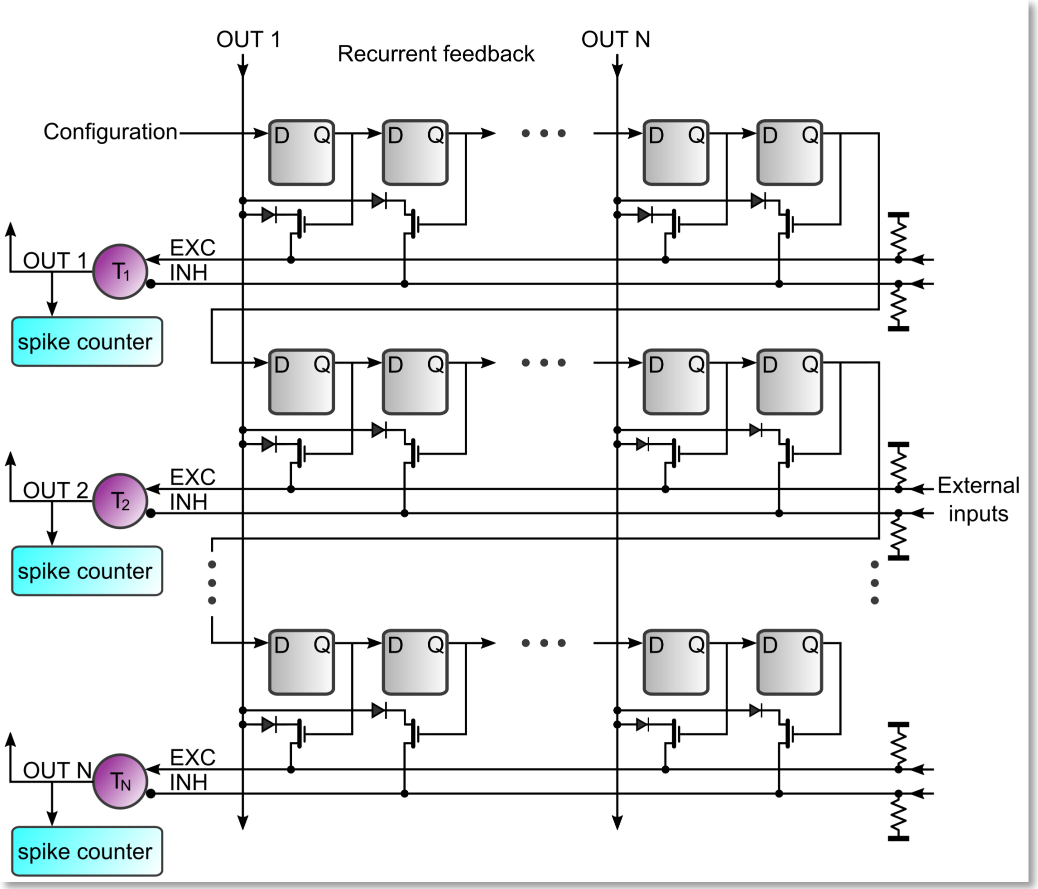


- Spike out if input spike interval between Tmin, Tmax
- Conditions: d>Tmax – Tmin, Tmin>Tmax/2
- But...how to train large event-driven networks?

Goal: Optimized Recurrent MMV Networks

- OR-ing type crossbar
- No synaptic addition in the network
- Overlapping spikes? We don't care!
- How to find connections and T's?

Look Mom,
No multipliers!
No adders!



Training method overview

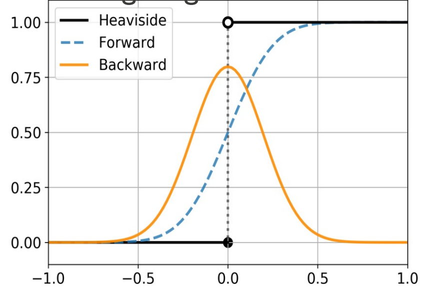
MMV model:

- Backpropagation with surrogate gradients
  → "get over" step-function discontinuity
- Spike, EXC, INH = 3 separate step-functions needed
- Use multiplicative integration rather than additive timing process
  → retain insensitivity to long T

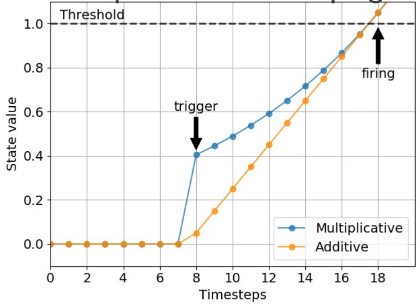
Network:

- Separate EXC/INH inputs and handle as events via surrogate gradients
  → handle overlaps/OR-ed signals
- Slow binarization of connections & rounding of periods over many epochs
  → force connections {0,1} vs. real-valued weights

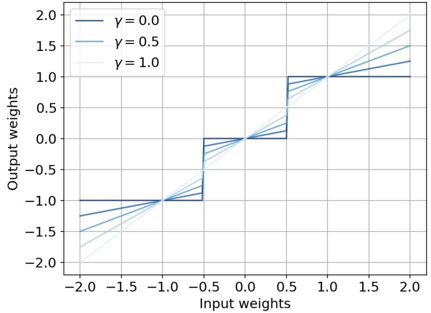
Surrogate gradient



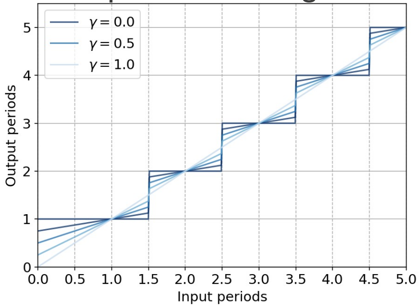
Multiplicative timekeeping



Slow binarization

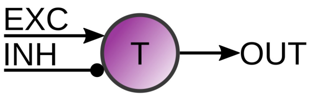


Slow period rounding



ALL elements are needed for the training to work!

TRICK #1: the NRMMV model



Trigger, Reset, Spike = 3 discontinuous events

Activation function:

Algorithm 1 Activation Function

FORWARD_ACT(x)
  save x **as context**
  return 1 **if** $x > 0$, **else** 0

BACKWARD_ACT(g)
hyperparameters: amplitude:A, spread: σ
  $x \leftarrow$ **retrieve context**
  return $g \cdot A \exp(-x^2/\sigma^2)$

Surrogate gradient

Leaky Integrate&Fire:

Algorithm 2 Leaky Integrate & Fire Neuron

UPDATE_LIF_STATE(a)
hyperparameters: leakiness: $\alpha \leq 1$, threshold: L
  state $\leftarrow \alpha \cdot$ state $+ a$
  spike $=$ FORWARD_ACT(state $- L$)
  state $\leftarrow 0$ if spike $= 1$, else state
  return spike

True adder needed!

NRMMV:

Algorithm 3 Monostable Multivibrator Neuron

UPDATE_MMV_STATE(in_exc, in_inh)
parameter: period: T
hyperparameters: growth factor: $\alpha = 1.1$, input threshold: $\beta = 0.99$
  $\delta = \alpha^{-T+1/2}$
  state $\leftarrow \alpha \cdot$ state
  spike $=$ FORWARD_ACT(state $- 1$)
  state $\leftarrow 0$ if spike $= 1$, else state
  exc $=$ FORWARD_ACT(in_exc $- \beta$)
  inh $=$ FORWARD_ACT(in_inh $- \beta$)
  state $\leftarrow$ exc $\cdot \delta$ if state $= 0$, else state
  state $\leftarrow 0$ if inh $= 1$, else state
  return spike

Separate EXC/INH inputs

Multiplying accumulator

EXC and INH are separate EVENTS

Processing order determines “corner cases”

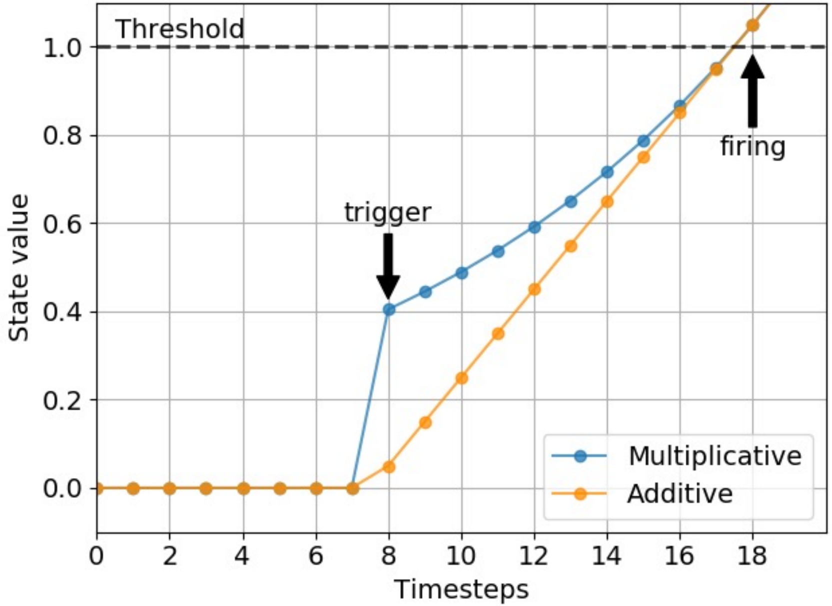
TRICK #1½ : multiplicative vs additive accumulation*

Preserve gradient sensitivity to input spikes in the past

From PyTorch Autogradient

```
#Multiplicative:
#m =  1.0
#dm/dx =  0.9999998807907104   --> keeps sensitivity!
#dm/dT =  -0.09531016647815704 --> -ln(ALPHA)

#Additive:
#m =  1.0000001192092896
#dm/dx =  0.05000000074505806 --> 1/T, less sensitive!
#dm/dT =  -0.05000000074505806  --> 1/T
```



*This is only for the model

Unrelated to the hardware implementation!

TRICK #2: network model

Split synaptic activation in EXC, INH and process separately

start with random real-valued weight matrices

Split EXC (>0) and INH (<0) weights

Process separately, inputs are positive, Equivalent to #activated lines

Algorithm 4 MMV network propagation

NET_FORWARD(input_spikes, internal_spikes)
parameter: connections C_{in}, C_{net}
split:
$C_{\text{in}}^{\text{EXC}}, C_{\text{in}}^{\text{INH}} \xleftarrow{\text{Eqs. (9,10)}} C_{\text{in}}$
$C_{\text{net}}^{\text{EXC}}, C_{\text{net}}^{\text{INH}} \xleftarrow{\text{Eqs. (9,10)}} C_{\text{net}}$
calculate activations:
$\text{act\_exc} = C_{\text{in}}^{\text{EXC}} \cdot \text{input\_spikes} + C_{\text{net}}^{\text{EXC}} \cdot \text{internal\_spikes}$
$\text{act\_inh} = C_{\text{in}}^{\text{INH}} \cdot \text{input\_spikes} + C_{\text{net}}^{\text{INH}} \cdot \text{internal\_spikes}$
calculate new internal spikes:
internal_spikes =
UPDATE_MMV_STATE(act_exc, act_inh)
return internal_spikes

$$C_{\text{net},i,j}^{\text{EXC}} = \begin{cases} C_{\text{net},i,j} & \text{if } C_{\text{net},i,j} \geq 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$C_{\text{net},i,j}^{\text{INH}} = \begin{cases} -C_{\text{net},i,j} & \text{if } C_{\text{net},i,j} < 0, \\ 0 & \text{otherwise,} \end{cases}$$

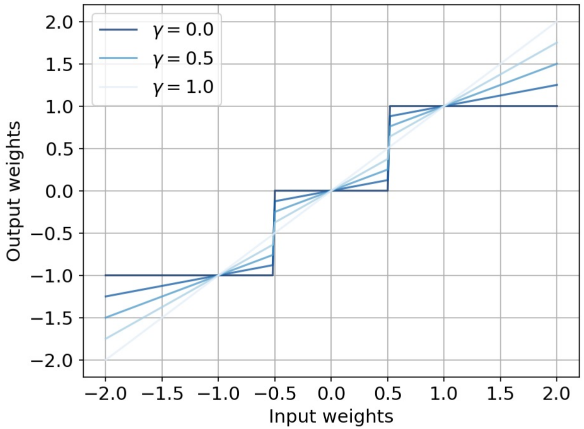
An MMV cannot distinguish between 1,2 or more simultaneous EXC/INH inputs!
There is NO addition of synaptic inputs
Hence NO adders are needed in the network!
Synapses are simple OR-operations

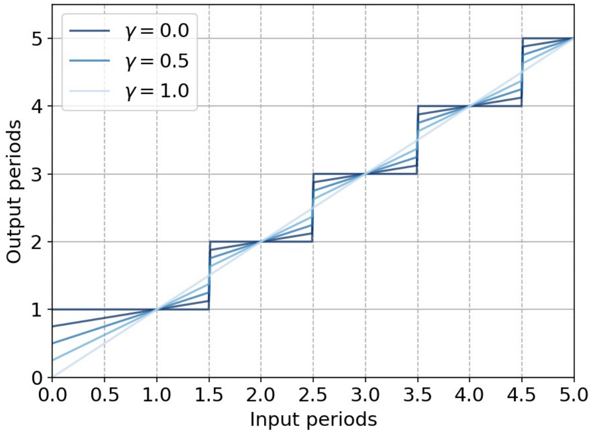
TRICK #3: slow weight binarization & period rounding

- Slowly "crystalize" the weights over many epochs, γ: 1→0, to connections
- Forward step: use partially quantized weights and rounded periods
- Backward and update step: operate on full precision C

$$C_{i,j} \leftarrow \begin{cases} \gamma C_{i,j} & \text{if } -\frac{1}{2} \leq C_{i,j} \leq \frac{1}{2}, \\ \gamma C_{i,j} + 1 - \gamma & \text{if } C_{i,j} > \frac{1}{2}, \\ \gamma C_{i,j} - 1 + \gamma & \text{if } C_{i,j} < -\frac{1}{2}, \end{cases}$$

$$T_i \leftarrow \gamma T_i + (1-\gamma) \max\{1, \lfloor T_i \rceil\}$$

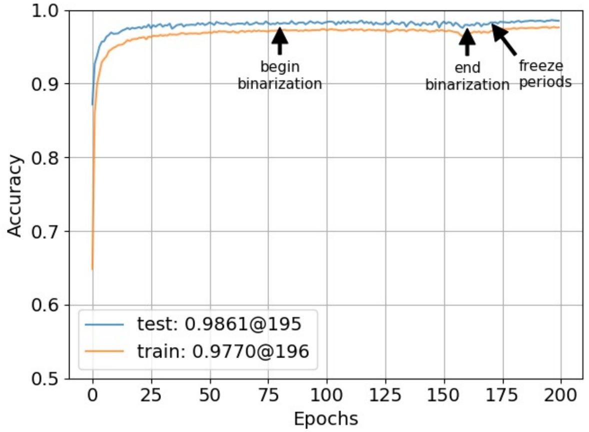


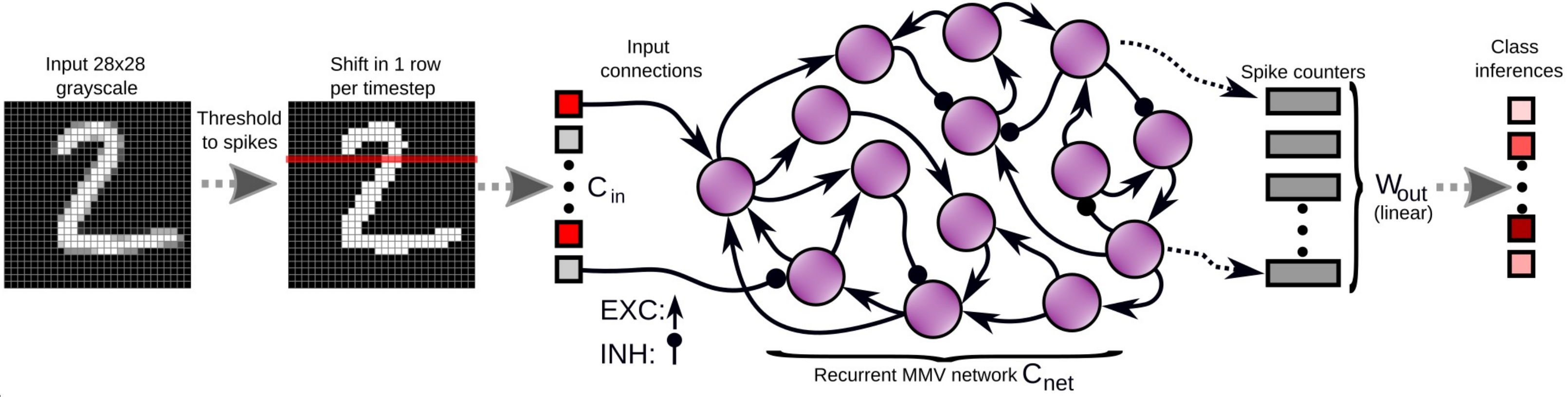


MNIST handwritten digits

- Threshold pixels at 0.5
- Stream in row or column-wise
- 500 NRMMVs
- Count spikes
- Linear readout: ~98.6%
- Population count readout: ~97.3%

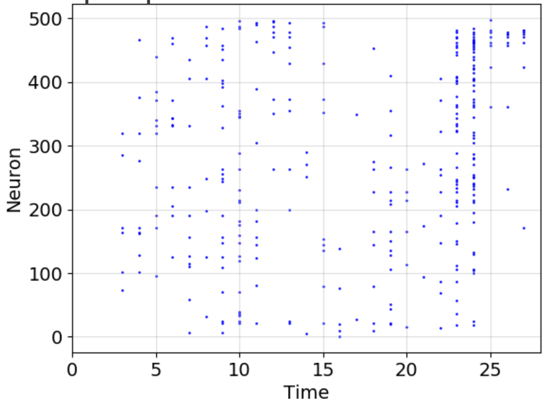
Label \ Inference	0	1	2	3	4	5	6	7	8	9
0	99.5					0.1	0.1	0.1	0.2	
1		99.5	0.2				0.2			
2	0.2	0.4	98.2	0.4				0.4	0.3	
3			0.2	99.0		0.2		0.4		
4			0.1		98.4		0.4	0.1	0.1	0.9
5			0.1	0.6		98.2	0.2	0.1	0.4	0.3
6	0.4	0.3	0.1	0.1	0.3	0.1	98.4		0.2	
7		0.5	0.9					98.4		
8	0.2		0.2	0.2	0.3	0.1	0.1	0.2	98.2	0.4
9		0.2		0.2	0.7			0.4	0.4	98.0



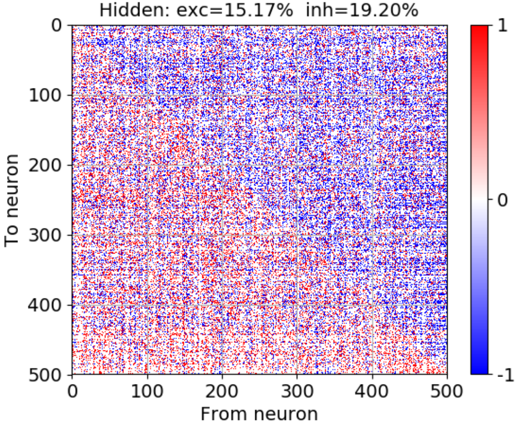


MNIST digits - some network metrics

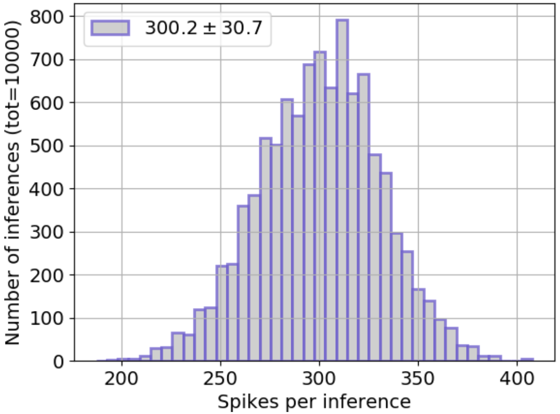
Spike pattern



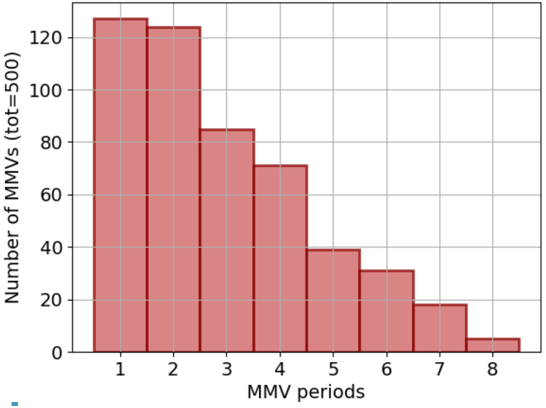
Network connections



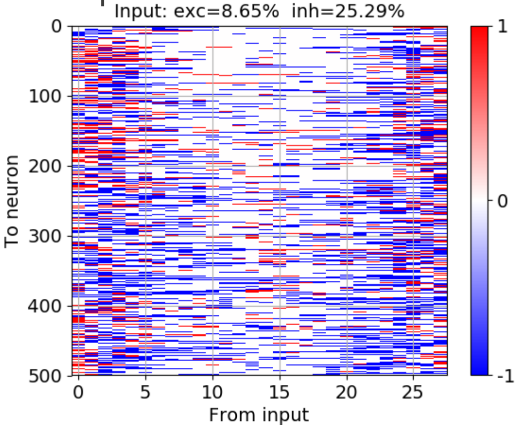
Spikes/inference distribution



Period distribution

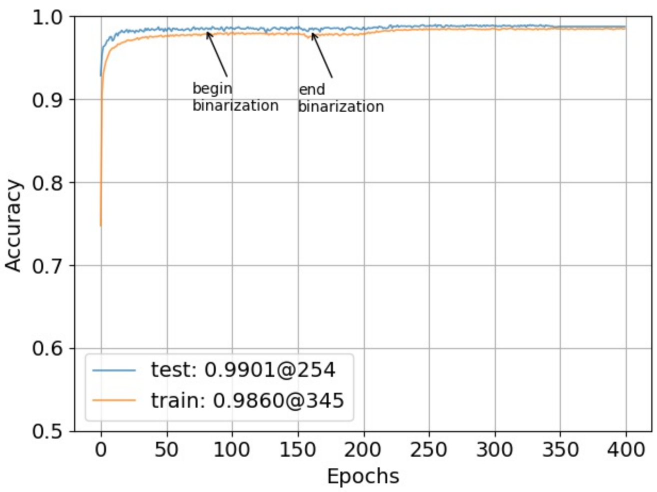


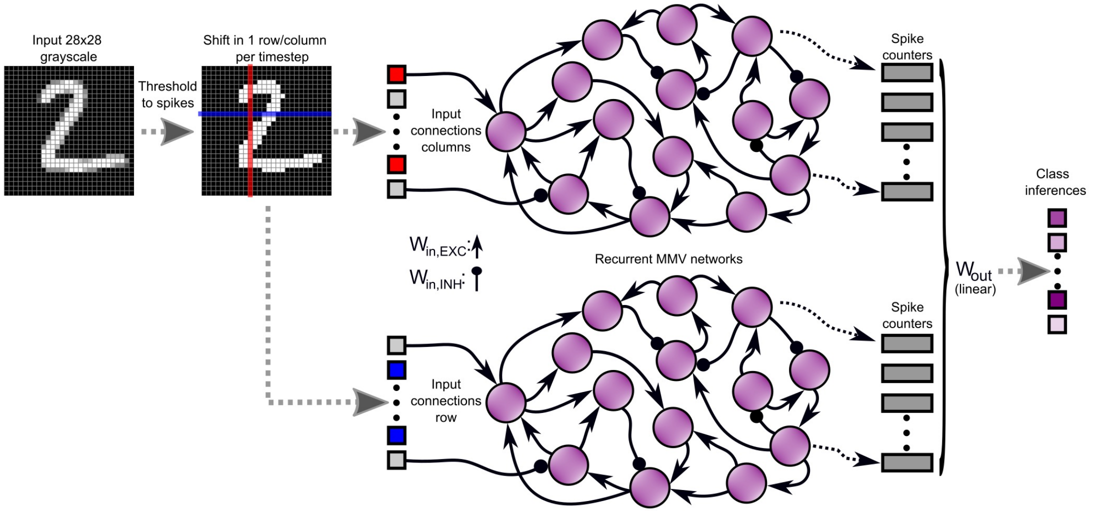
Input connections



MNIST handwritten digits

- Streaming in rows & columns
- 2 x 500 NRMMVs + linear readout 1000 →10
- Data augment train set every epoch: shift ±2, rotate ±15° → ~**99%**



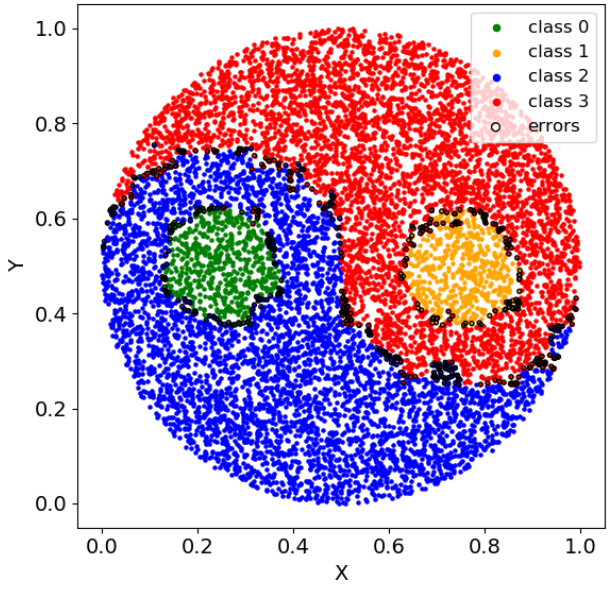


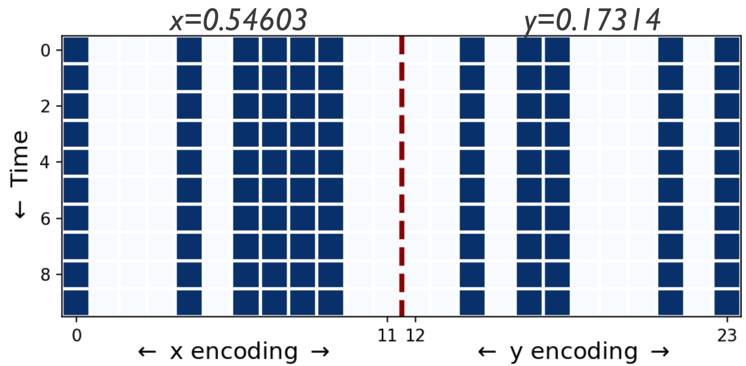
Yin-Yang segmentation task

A nicer "XOR"

- Recurrent NRMMV net + linear readout layer
- Direct binary encoding [0,1] → $0..2^{12}$ -1 (12 bits)
- 82 ± 24 spikes, active connections: ~26%

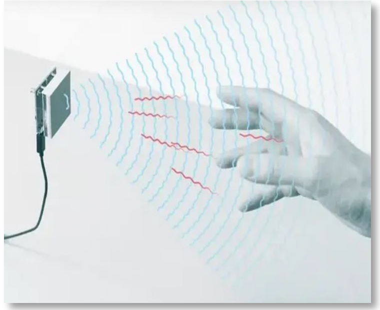
Network size	Accuracy [%]
100	96.26
200	98.43
500	99.16



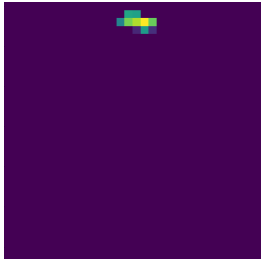


Google Soli radar gestures

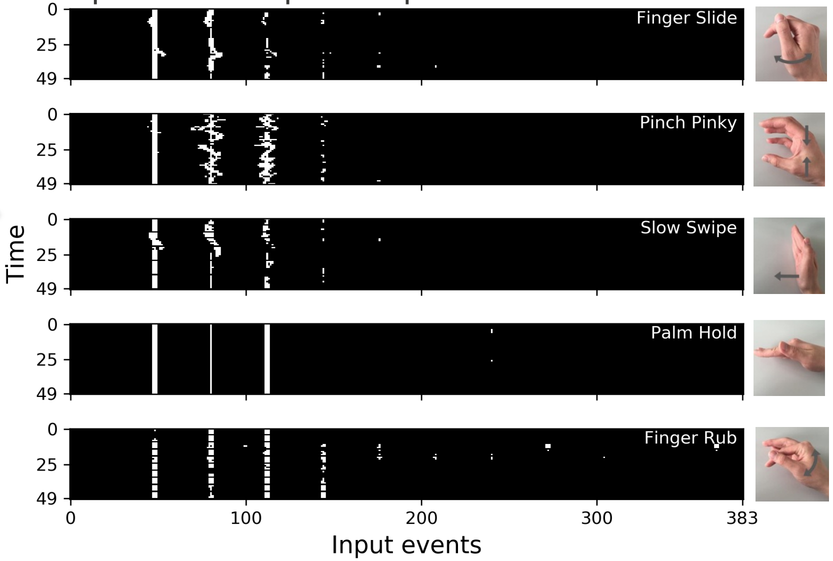
- Input: 32×32 Range-Doppler maps [0..1], 25ms/map, 11 gestures
- Preprocessing:
  - limit input image to first 384 pixels
  - #timesteps (frames) rescaled from 28...145 → 50
  - Any value>0 → spike in



Pinch Pinky



Input: 50 timesteps x 384 pixels

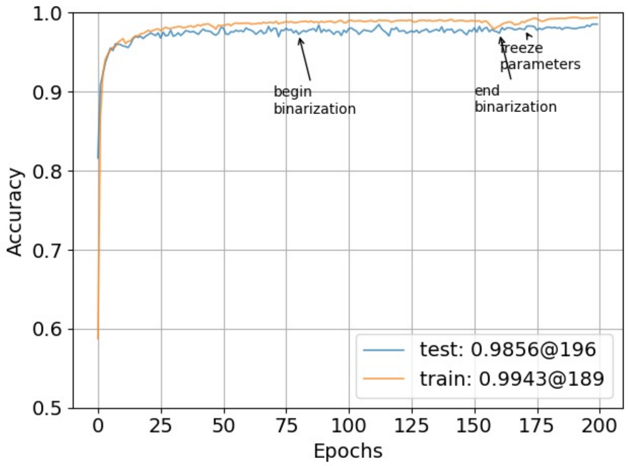


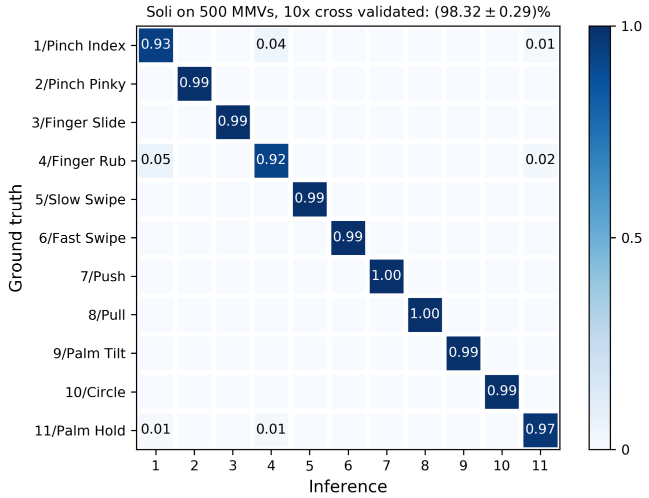
[Wang et al., Proc. UIST 2016.]

Google Soli radar gestures

- Stream in recurrent NRMMV net + linear readout
- 10× cross validated

Reference	Network/Algorithm	Accuracy [%]
This work	**100 MMVs + linear readout**	**97.62 ± 0.33**
This work	**250 MMVs + linear readout**	**98.01 ± 0.26**
This work	**500 MMVs + linear readout**	**98.32 ± 0.29**
Yin *et al.*, Nat. Mach. Intell 2021	Spiking RNN	91.9
Tsang *et al.*, MDPI Electronics 2021	LSM with 460 neurons	98.6 ± 0.7
Wang *et al.*, UIST 2016	CNN+RNN	87.17





IBM DVS128 gestures

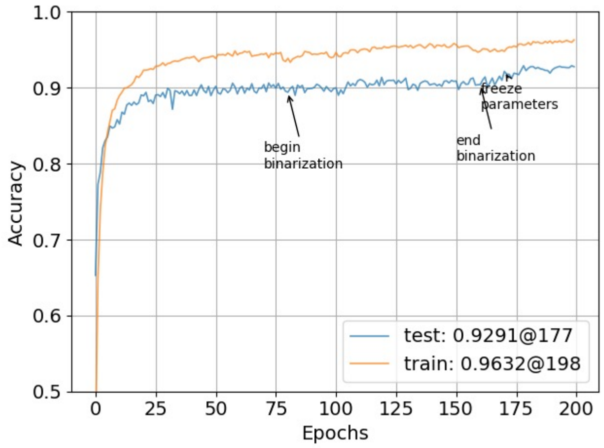
- Group events in frames of 50ms
- pol=1 → pixel++, pol=0 → pixel--, clamp at 0, threshold to spikes
- 2.0s/gesture, 40 frames, 0.25s hop, 32x32 pixels → flatten to 1024 wide x 40 timesteps
- 22347 train, 5870 test gestures

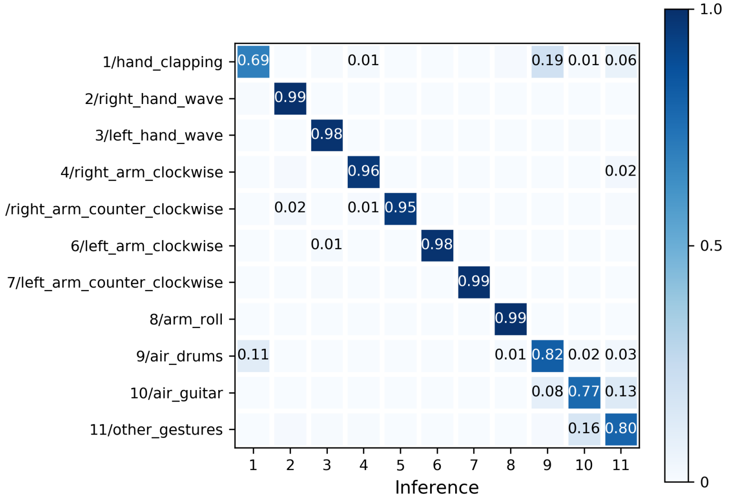
https://research.ibm.com/interactive/dvsgesture/

IBM DVS128 gestures

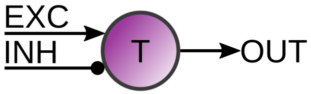
- 500 NRMMVs + linear readout
- Data augmentation: shift ±3 pixels, rotate ±15°
- 22347 train, 5870 test gestures

Reference	Network/Algorithm	Accuracy [%]
This work	**500 MMVs + linear readout**	**92.91**
Safa et al., 2022 [arXiv.2111.00791]()	LIF+STDP	92.5
Amir et al., IEEE CVPR 2017	CNN	91.77
Amir et al., IEEE CVPR 2017	CNN+postprocessing	94.59
Samadzadeh et al., 2021 [arXiv.2003.12346]()	Spiking ResNet	96.7





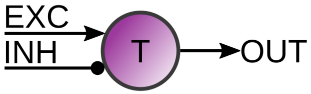
MNIST digits - energy/inference



- PyTorch → Verilog
- Mapping to TSMC 28nm HPC+, 500 MMV recurrent reconfigurable network with D-FF & OR-ing trees: **855pJ/inf.**
- Linear/population count readout **guesstimate** using figures from another project: 2.8pJ/MAC, 1.4pJ/ADD, BF16 format, 100MHz

Reference	Method/Conditions	Accuracy [%]	Time per inference	Energy per inference
This work	500 MMVs → linear readout, 28nm TSMC HPC+	98.61	330ns+$t_{readout}$=330ns+(500x10+10)x10ns=50.43µs	855pJ+$E_{readout}$= 855pJ+10x(500x2.8pJ+1.4pJ)= **14.9nJ**
This work	500 MMVs → population coding readout, 28nm TSMC HPC+	97.32	330ns+$t_{readout}$= 330ns+500x10ns= 5.33µs	855pJ+$E_{readout}$= 855pJ+500x1.4pJ= **1.56nJ**
Frenkel et al., ISCAS 2020	SPOON, 28 nm event-driven CNN	97.5	117µs	313nJ
Park et al., ISCC 2019	200+200+10 SNN, 28 nm	97.83	10µs	236.5nJ
Chen et al., IEEE Symp. VLSI 2018	4096-neuron, 1M synapse SNN, 10 nm	97.9	not given	1.7µJ
Stuyt et al., Front. Neurosci 2021	"µBrain", 256+64+16 SNN, 40 nm	91.7	4.2ms	308nJ

Take Home Messages



- MMVs = non-biologically inspired spiking "neurons" = simple timers

- MMV networks can be trained with PyTorch and implemented in digital hardware
  They work by setting up and testing timing conditions

- Extremely low energy/inference due OR-ing style interconnect instead of synaptic addition, at good accuracy, with current technology

- Many things still to be discovered: other types of MMVs, heterogenous MMV nets, MMV based CNNs, evolutionary optimization...

Preprint on ResearchGate

Not Biologically Inspired: On Training Networks of Monostable Multivibrator Timer Neurons

Lars Keuninckx, Matthias Hartmann, Paul Detterer, Ali Safa, Ilja Ocket

***Abstract*—Monostable multivibrators are simple timers which are easily implemented using counters in digital hardware and can be interpreted as non-biologically inspired spiking neurons. We show how fully binarized event-driven recurrent networks of monostable multivibrators can be trained to solve classification tasks. We mitigate an important bottleneck in neuromorphic hardware concepts by circumventing synaptic addition within the network. Here rather, input signals to a neuron are simply OR-ed together. Temporally overlapping input events are resolved at the neuron level. We demonstrate our approach on the MNIST handwritten digits, Google Soli radar gestures, IBM DVS128 gestures and Yin-Yang classification tasks, all with excellent results. The estimated energy consumption for the MNIST handwritten digits task, excluding the final readout layer, is 855pJ per inference for a test accuracy of 98.61% for a reconfigurable network of 500 units that was mapped to a 28nm process.**

and probably even more unknown constraints imposed by their biological reality. Biological neurons are never *only* computational units and therefore may be limited in their computational abilities in ways their artificial counterparts needn't be, but actually might become if we push the analogy too far.

In Ref. [6], the original bio-inspired question was reversed from "how can we build a system that somewhat behaves like a brain?" to "which fundamental building blocks that are easy to build *and network* in large quantities do we have at our disposal?". As said there, whether such a unit should be called a neuron or not, is merely a matter of semantics. We are only concerned with its computational or "neuronal" properties. The authors of Ref. [6] investigate the monostable

http://dx.doi.org/10.13140/RG.2.2.27417.70242

Thanks!

Ilja Ocket

Paul Detterer

Ali Safa

Matthias Hartmann

References (1)

[32] C. Frenkel, J.-D. Legat, and D. Bol, "A 28-nm convolutional neuromorphic processor enabling online learning with spike-based retinas," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[33] J. Park, J. Lee, and D. Jeon, "7.6 a 65nm 236.5nj/classification neuromorphic processor with 7.5feedback," in *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2019, pp. 140–142.

[34] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1m-synapse 3.8pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10nm finfet cmos," *2018 IEEE Symposium on VLSI Circuits*, pp. 255–256, 2018.

[10] J. Stuijt, M. Sifalakis, A. Yousefzadeh, and F. Corradi, "?brain: An event-driven and fully synthesizable architecture for spiking neural networks," *Frontiers in Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2021.664208

References (2)

[21] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *bioRxiv*, 2021. [Online]. Available: https://www.biorxiv.org/content/early/2021/03/23/2021.03.22.436372

[29] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. Di Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha, "A low power, fully event-based gesture recognition system," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7388–7397.

[30] "iniLabs," https://inivation.com/, 2022, [Online; accessed 25-May-2022].

[31] A. Samadzadeh, F. S. T. Far, A. Javadi, A. Nickabadi, and M. H. Chehreghani, "Convolutional spiking neural networks for spatio-temporal feature extraction," 2020. [Online]. Available: https://arxiv.org/abs/2003.12346

[32] A. Safa, I. Ocket, A. Bourdoux, H. Sahli, F. Catthoor, and G. Gielen, "A new look at spike-timing-dependent plasticity networks for spatio-temporal feature learning," 2021. [Online]. Available: https://arxiv.org/abs/2111.00791

References (3)

[26] S. Wang, J. Song, J. Lien, I. Poupyrev, and O. Hilliges, “Interacting with soli: Exploring fine-grained dynamic gesture recognition in the radio-frequency spectrum,” in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, ser. UIST ’16. New York, NY, USA: Association for Computing Machinery, 2016, pp. 851–860. [Online]. Available: https://doi.org/10.1145/2984511.2984565

[27] I. J. Tsang, F. Corradi, M. Sifalakis, W. Van Leekwijck, and S. Latré, “Radar-based hand gesture recognition using spiking neural networks,” *Electronics*, vol. 10, no. 12, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/12/1405

Copyright Notice

TINY ML EMEA
AMSTERDAM, NL
June 26-28,2023

www.tinyml.org