POLITECNICO MILANO 1863

DIPARTIMENTO DI SCIENZE E TECNOLOGIE AEROSPAZIALI

Ariadna Study
Investigation of low energy Spiking Neural Networks based on temporal coding for scene classification

Final presentation

Paolo Lunghi [1] Stefano Silvestrini [1] Dominik Dold [2] Gabriele Meoni [2] Dario Izzo [2]

[1]Politecnico di Milano, Aerospace Science and Technology Department

[2]ESA ACT

25/09/2023 – ESA/ESTEC, Noordwijk, The Netherlands

Artificial Intelligence applications in space

The use of **Artificial Intelligence in space applications** is attractive:

- Growing number of spacecraft with **ground communication bottleneck**;
- Increasing complexity of operative scenarios non necessarily compatible with **communication delay** and scheduling in **uncertain environment**.

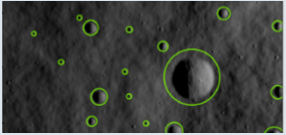
Wide spectrum of possible applications:

Earth Observation

- Onboard autonomous data preprocessing for bandwidth optimisation

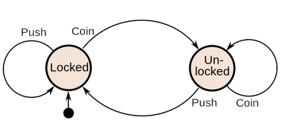
Spacecraft GNC

- Vision-based navigation
- Hazard Detection and Avoidance
- Automatic control policy adaptation



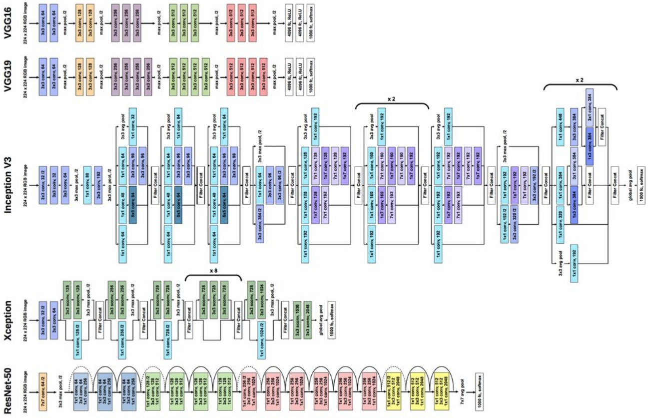
Failure detection

- Early detection of anomalies
- Onboard failure recovery without waiting for next comm window



Limitations of Artificial Neural Networks

- State-of-the-art deep Neural Networks are made up by **long stacks of layers**.
- Hardware high efficiency is based on **batched computation**.
- Memory, power, and energy requirements **limits the applicability** of such systems in space.
- **Energy** is the most limiting factor.



[0] Image credits: M.M. Leonardo, et al., "Deep Feature-Based Classifiers for Fruit Fly Identification (Diptera: Tephritidae)", 2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI).

Spiking Neural Networks – Attractiveness

- **Spiking Neural Networks** are based on neuron models that exchange information by means of discrete spikes.
- The neuron has an **internal dynamic** and accumulates presynaptic spikes in and internal state (the voltage/potential).
- As the potential reach a certain **threshold**, it resets to the initial value and the neuron emits a spike.
- The computation is inherently **sparse** (no computation is performed if there are not incoming spikes).
- Since **spikes are binary**, the required operation is just an accumulate operation instead of Multiply-and-ACcumulate.
- **Potential energy saving by orders of magnitude**.
- Even better with **neuromorphic hardware**, tailored for sparse, asynchronous computation.

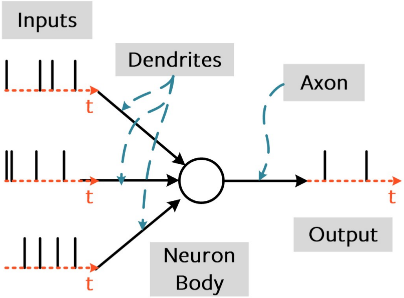
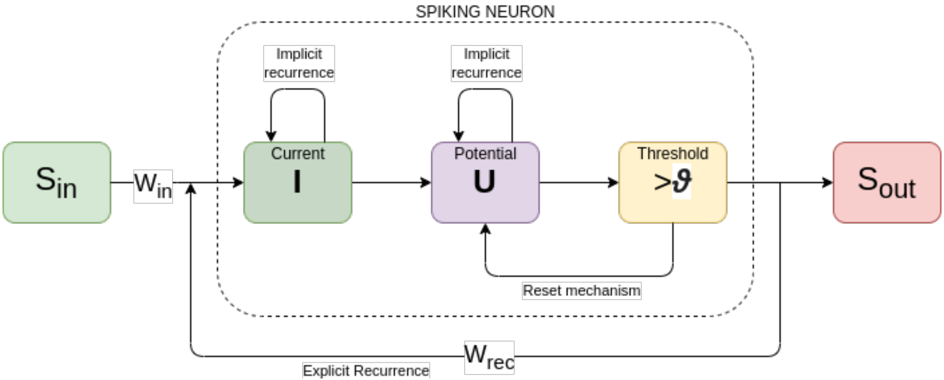


Image credits: J. K. Eshraghian et al., "Training Spiking Neural Networks using lessons from Deep Learning." arXiv [Preprint] arXiv:2019.12894 2021.

Spiking Neuron model



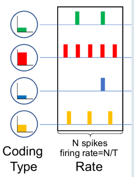
Spiking Neural Networks – Current Limitations

- There is no standard way to train SNN.
- Spikes are **non-differentiable**, making traditional backpropagation not directly applicable to SNN training.
- There is no unique method to **encode information** in spikes.
- ANN to SNN conversion methods exist, relying to **rate-based coding**, but with certain drop in performance (accuracy).
- State-of-the-art accuracy can be recovered **increasing the network latency**, but **losing part of the energy gain** (since more spikes are emitted).

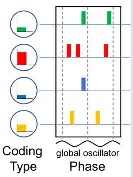
Information encoding in Spiking Neural Networks

Rate coding

- Most common encoding.
- Information is encoded in the fire rate of the neuron.



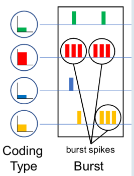
Phase coding



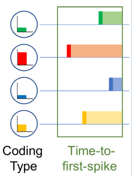
- Information is encoded in the phase of the spikes w.r.t. a global oscillator signal.

Burst coding

- Information is carried in the number of spikes and in the inter-spike interval.
- Most robust to synaptic noise.



Temporal coding



- Information is encoded in the time of the first spike arrival.
- earlier = more relevant.

[0] Image credits: S. Park, et al., "T2FSNN: Deep Spiking Neural Networks with Time-to-first-spike Coding." arXiv, 2020. doi: 10.48550/arXiv.2003.11741.

Temporal coding

- Temporal coding tends to use **less spikes** than other methods.
- It can be combined with models of neurons which **spike once at most**, further limiting the number of spikes.
- **Time-To-First-Spike (TTFS)** coding: to express numeric values, the value expressed is encoded in the time of the arrival of the first spike (the higer the value, the lower the time).
- **Rank Order Coding**: for classification purposes, only the order of the received spikes matter, not the specific time.
- Comparison on benchmark tasks shows it is the **most efficient for power consumption**.

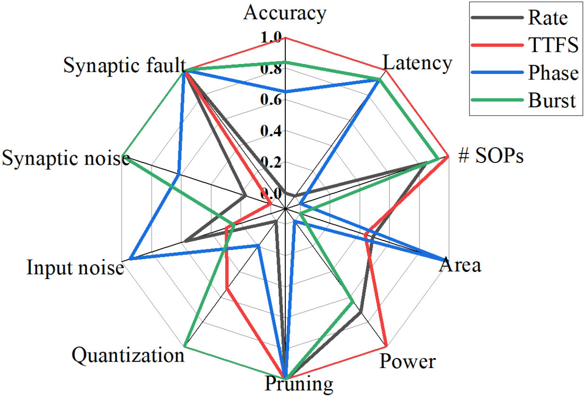


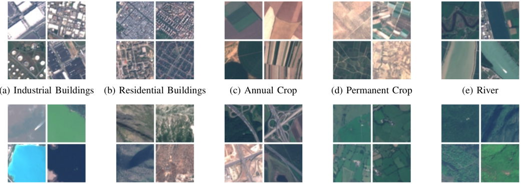
Image credits: W. Guo, et. al., "Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems," Frontiers in Neuroscience, vol. 15, p. 212, 2021, doi: 10.3389/fnins.2021.638474.

Project Objectives

- Perform a preliminary investigation of the **potential benefits of SNNs** based on **temporal coding** for onboard AI applications.
- SNN models are compared in terms of **accuracy** and **complexity**.
- Proper **training algorithms** for the SNN models evaluated and selected.
- Establish a method to perform **hardware-agnostic**, relative comparison of the **computational load** required by different architectures, both SNNs and ANNs.
- Highlight the possible **advantages** and **drawbacks** of SNN models compared to ANN.

Case study: **EuroSat** dataset (land use classification)

- Reference dataset for **scene classification** representative of a plausible use case in the Earth Observation field.
- The activity presented in this report focuses on the RGB EuroSAT dataset.
- Image format: 8 bit, $3 \times 64 \times 64$ px (c, h, w) in size.
- $27\,000$ images, divided in 10 classes each one represented by a number of samples between 2000 and 3000.
- 70/20/10 (training, validation, test) split adopted for the training and cross-validation.
- Random horizontal and vertical flip as only data augmentation at training.



(a) Industrial Buildings (b) Residential Buildings (c) Annual Crop (d) Permanent Crop (e) River
(f) Sea & Lake (g) Herbaceous Vegetation (h) Highway (i) Pasture (j) Forest

Image credits: P. Helber, et al., "EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification." arXiv, 2019 doi: 10.48550/arXiv.1709.00029.

Spiking Neural Networks training approach 1/4

ANN-to-SNN conversion:

- The training is performed on a standard ANN that is then converted to SNN.
- High precision activation function converted in spike rate or latency code.
- Leverage standard, state-of-the-art, **backpropagation** techniques.
- Maintaining high precision requires **long number of time steps**, **losing energy efficiency**.
- The result **approximate** the original ANN (unlikely to match the performance).

Local learning rules:

- Weights updates are a function of signals that are spatially and temporally local to the weight, rather than a global signal as in error backpropagation (e.g. Spike Timing Dependent Plasticity, STDP).
- Biologically inspired.
- Lightweight, **unsupervised learning**.
- Requires a **classifier** at output, or **complex reward** mechanisms.
- Currently they **struggle to achieve high accuracy**.

Spiking Neural Networks training approach 2/4

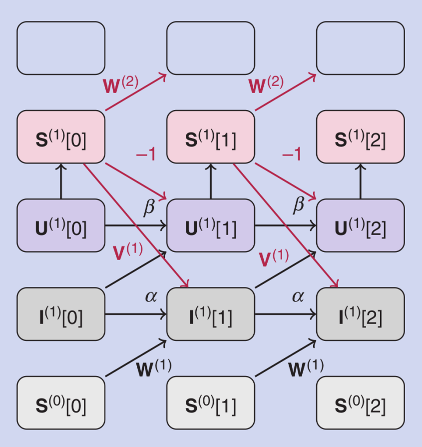
Backpropagation using spike times

- Instead the spikes, the derivative of the **spike times** is used.
- Spike times are a **continuous** variable (differently w.r.t. spikes themselves).
- Successfully overcome the discontinuity problem **without approximations**.
- Every neuron **must spike** to enter training (no solution exists if a neuron does not spike).
- Can enforce **stringent priors** to the network.
- Derivatives need to be rewritten for **every specific neuron model**.

Spiking Neural Networks training approach 3/4

Surrogate Gradient (SG):

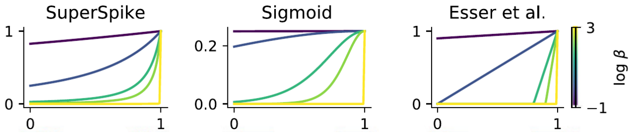
- Generalised **backpropagation** algorithm is applied to the unrolled computational graph (backpropagation through time, BPTT).
- At the **forward pass**, the Heaviside operator $H(x)$ is applied to determine whether the neuron spikes.
- At the **backward pass**, $H(x)$ is substituted by a **continuous function** whose derivative is used as substitute of the discontinuous gradient.



[0] Image credit: E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-Based Optimization to Spiking Neural Networks," IEEE Signal Processing Magazine, vol. 36, no. 6, pp. 51–63, Nov. 2019, doi: 10.1109/MSP.2019.2931595.

Spiking Neural Networks training approach 4/4

Surrogate Gradient selected for training with **SuperSpike**[1] (a fast sigmoid) as surrogate function.



- Not dependent on a specific neuron model.
- Not dependent on the type of encoding.
- Can leverage **traditional deep learning libraries** (PyTorch, Tensorflow).
- **Large memory consumption** and slow training (due to the unrolling in time).

Models are tested in PyTorch[2] with the Norse[3] library.

[1] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," Neural Computation, vol. 30, no. 6, pp. 1514–1541, Jun. 2018, doi: 10.1162/neco_a_01086.

[2] https://pytorch.org/

[3] C. Pehle and J. E. Pedersen, "Norse - A deep learning library for spiking neural networks." Oct. 2021. url: https://github.com/norse/norse

Constant current encoder

Numeric values from RGB images can be converted in binary spikes train, both rate and temporal-based, by just simply supplying them as **constant current** to the suitable neuron model.

Other encoder models exist (e.g. the Poisson encoder translates pixel intensity in a likelihood to spike by a random spiking neuron).

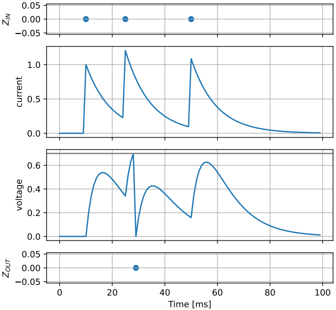
Learnable encoder

- A **convolution layer** can be placed before the conversion in spikes.
- Can be applied **to different encoder types**.
- In this way the network is capable to **learn its own encoder**.
- Such layer is appropriately taken into account **in the energy estimation**.

Looking for extremely efficient systems, bio-plausibility is not sought. Most simple neurons model are used.

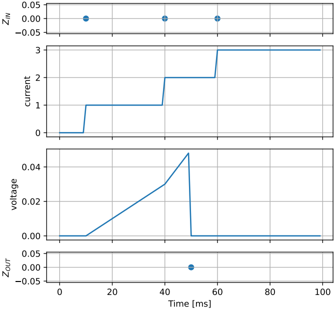
Leaky Integrate and Fire (LIF)

- Most popular neuron model.
- Exponentially decay both current and voltage.
- used for **rate-coded** test cases.



Linear Integrate and Fire

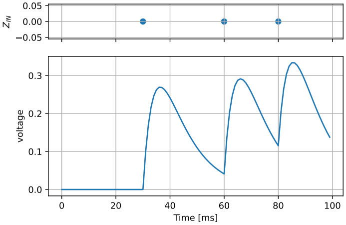
- Used for **latency-coded** test cases.
- Stepwise current, linear potential.
- Set to fire once at most.



Output layers

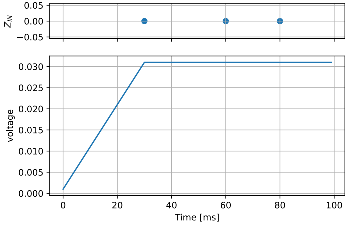
Ad hoc readout layers are used to output differentiable spike rates and times to exploit autodiff capabilities.

Leaky Integrator



- Standard for **rate based networks**.
- Accumulate incoming spikes.
- Maximum of last time step value taken as readout value.

Spike time readout layer



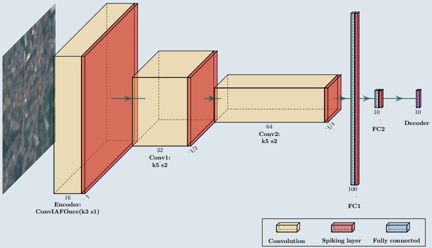
- Used for **tmporal coded networks**.
- Integrate time until a spike is received.
- Differentiable to enable backpropagation.
- Developed for Ariadna activity.

Benchmark architectures

- Benchmark architectures are established for test cases.
- Neuron models and layers parameters are varied.

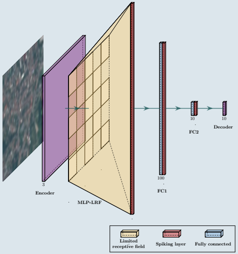
Convolutional Neural Network

- VGG style with convolutional current encoder.



Limit Receptive Field Network

- Constant current encoder.
- Mimick convolution by connecting blocks of image to fully conncedted layers.



Performance Metrics: EMAC per inference 1/3

Energy consumption is due to two factors:

$$\mathrm{E}_{\mathrm{tot}} = \boxed{s e_{\mathrm{syn}}} + \boxed{n T e_{\mathrm{upd}}}$$

Synaptic operations

- e_{syn} energy per synaptic operation.
- s number of synaptic operations.

Neuron updates

- e_{upd} energy per neuron update.
- n number of neurons in the network.
- T number of time steps.

Number of synaptic operations

s can be estimated per layer in function of the spiking rate f:

$$s_{(l)} = n_{s(l)}\, n_{n(l)}\, f_{(l-1)}$$

- $n_{s(l)}$ number of synapse per neuron
- $n_{n(l)}$ number of neuron in the layer

Recurrent layer case:

$$s_{r(l)} = n_{s(l)}\, n_{n(l)}\, f_{(l)}$$

Performance Metrics: EMAC per inference 2/3

e_{syn} and e_{upd} are evaluated on the **specific neuron model**:

$$IFL: \begin{cases} \tilde{i}_{k+1} = \tilde{i}_k + \underbrace{\boxed{\sum_{j=1}^{n_S} \tilde{w}_j S_{jk}}}_{\text{AC}} + \underbrace{\boxed{\tilde{b}}}_{\text{AC}} \\ v_{k+\frac{1}{2}} = v_k + \underbrace{\boxed{\tilde{i}_{k+1}}}_{\text{AC}} \\ v_{k+1} = v_{k+\frac{1}{2}} - \boxed{v_{\text{th}} S_{k+1}} \end{cases}$$

$$LIF; \begin{cases} i_{k+1} = i_k - \underbrace{\boxed{i_k \frac{\Delta t}{\tau_{\text{syn}}}}}_{\text{MAC}} + \underbrace{\boxed{\sum_{j=1}^{n_S} w_j S_{jk}}}_{\text{AC}} + \underbrace{\boxed{b}}_{\text{AC}} \\ v_{k+\frac{1}{2}} = v_k + \underbrace{\boxed{(i_{k+1} - v_k)\frac{\Delta t}{\tau_{\text{mem}}}}}_{\text{AC+MAC}} \\ v_{k+1} = v_{k+\frac{1}{2}} - \boxed{v_{\text{th}} S_{k+1}} \end{cases}$$

1. The **discrete-time** neuron model is written;
2. Single operations are identified **by stage**: **synaptic ops**/**neuron upd**;
3. **Type of operation** (AC/MAC) are identified;
4. Different **contributions are summed** in e_{syn} and e_{upd} assuming $1\text{MAC} = 1\text{EMAC}$ and $1\text{AC} = 0.667\text{EMAC}$.

$$\text{IFL:} \begin{cases} e_{\text{syn}} = 1\,\text{AC} = 0.667\,\text{EMAC} \\ e_{\text{upd}} = 2\,\text{AC} = 1.333\,\text{EMAC} \end{cases}$$

$$\text{LIF:} \begin{cases} e_{\text{syn}} = 1\,\text{AC} = 0.667\,\text{EMAC} \\ e_{\text{upd}} = 2\,\text{AC} + 2\,\text{MAC} = 3.333\,\text{EMAC} \end{cases}$$

Performance Metrics: EMAC per inference 3/3

Assumption

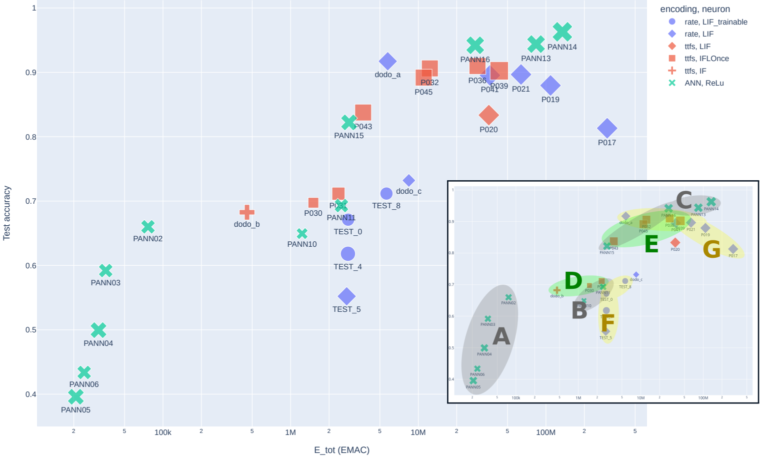
Memory operations dominate the cost of the computation.

$$1\,\text{MAC} = 3/2\text{AC}$$

- Agnostic w.r.t. the hardware (but can be tuned to target specific platform).
- Capable to compare **both ANNs and SNNs**.
- Takes into account **different neuron models**.
- Differentiate **neuron update** and **synaptic operations**.
- Finer estimation w.r.t. to raw number of emitted spikes.
- **Conservative in estimate SNN load**.

Accuracy vs Energy (EMAC/inf)

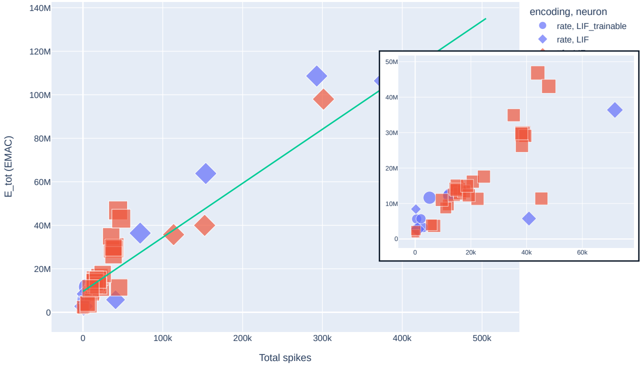
- 73 test cases, with benchmark architectures (ANNs, SNNs both time and rate based).
- **SNN are capable to reach similar accuracy to standard ANNs with a fraction ($20\,\%$ to $50\,\%$) of the EMAC/inference.**

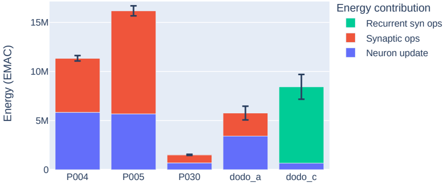


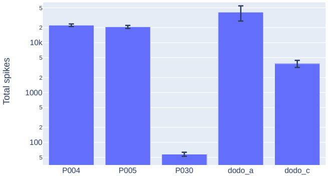
Test cases main groups:

A) ANN – MLP, limited receptive field;
B) ANN – MLP;
C) ANN – CNN;
D) SNN – MLP, TTFS encoding, IF neuron;
E) SNN – CNN, TTFS encoding, IF neuron;
F) SNN – MLP, rate encoding, LIF neuron;
G) SNN – CNN, rate encoding, LIF neuron.

EMAC/inf as proxy for energy consumption







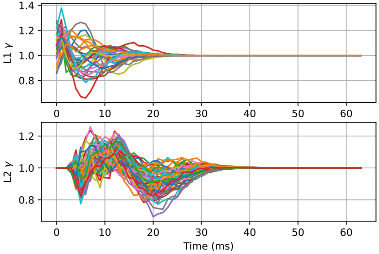
EMAC/inf increases with the number of spikes, but it is **only a general trend**.

Synaptic operations play a crucial role in determining energy consumption, even with same number of spikes.

Batch Normalisation Through Time

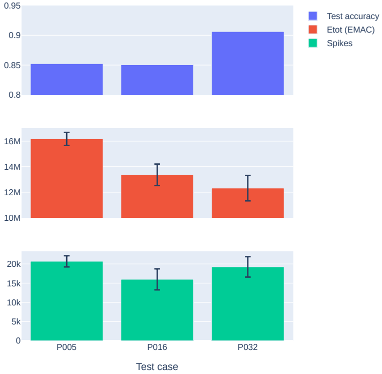
- Batch Normalisation Through Time (**BNTT**) proved effective in network regularisation.
- It is the same as standard BN, except that:
  - Mean and variance computation executed independently at each time step;
  - The hyperparameter γ is learnt at training;
  - No offset term is used (redundant with the layer bias).

$$\mathrm{BNTT}(x_i^t) = \gamma^t \hat{x}_i^t, \; \hat{x}_i^t = \frac{x_i^t - \mu_{\mathcal{B}}^t}{\sqrt{\sigma_{\mathcal{B}}^{2\,t} + \epsilon}}$$



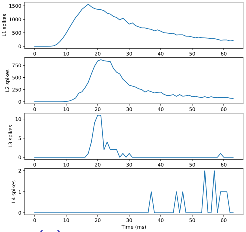
- The trend of γ after training shows that a **temporal pattern** is identified.
- The **temporal receptive field** of each layer can also be easily identified.

- Best results: **spatial BNTT**.
- Increased accuracy and more efficient network usage (less energy even with same spikes).

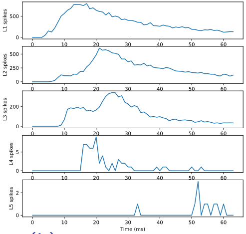


SNNs model scaling

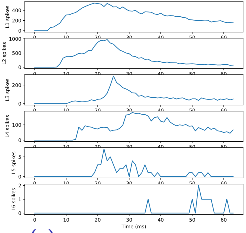
- **As network complexity increases, the overall accuracy starts to drop**.
- Information does not flow correctly between layers: late layers spike basing on **incomplete information from the previous ones**.
- Possible causes:
  - Limited **number of time steps** at training.
  - **Limited size of the batch** induce malfunctioning of regularisation methods (i.e. BNTT).
- Both factors provoked by **bad scaling of memory consumption** at training, due to the network unrolling in time required by SG.



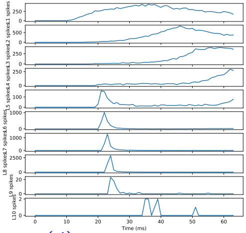
(a) **Case P036 – effective**.



(b) **Case P039 – suboptimal**.



(c) **Case P033 – suboptimal**.



(d) **Case P012 – failed**.

Conclusions

- A preliminary investigation of the potential benefits of SNNs based on temporal coding for onboard AI applications in space was carried out.
- **EMAC per inference** used to compare the computational load in a hardware-agnostic way.
- Benchmark SNN models, both latency and rate based, exhibited a **minimal loss in accuracy**, compared with their equivalent ANNs, with **significantly lower** (from $-50\,\%$ to $-80\,\%$) EMAC per inference.
- **An even larger improvement can be expected** with SNNs implemented on **neuromorphic HW**.
- SG proved effective in training SNNs, but **scaling to very deep architectures is still an issue**.

Overall, SNNs are a competitive candidate to achieve autonomy in space systems.

Future research

- A research effort is still needed, looking for **architectures**, **regularisation** techniques, and **initialisation** methods capable to exploit the peculiarities of latency-based SNNs.
- Recently proposed innovative **training algorithms**, which try to **overcome the bottleneck of BPTT** (i.e. Forward Propagation Through Time) should be investigated.
- Future works should also explore sensitivity of event-based HW to space environment, to identify **disturbance models** enabling robustness even in presence of input or synaptic noise.

POLITECNICO MILANO 1863

DIPARTIMENTO DI SCIENZE E TECNOLOGIE AEROSPAZIALI

Ariadna Study
Investigation of low energy Spiking Neural Networks based on temporal coding for scene classification

Final presentation

Paolo Lunghi [1] Stefano Silvestrini [1] Dominik Dold [2] Gabriele Meoni [2] Dario Izzo [2]

[1]Politecnico di Milano, Aerospace Science and Technology Department

[2]ESA ACT

25/09/2023 – ESA/ESTEC, Noordwijk, The Netherlands