Spiking Neural Networks for energy-efficient audio signals classification : representation matters

Noémie MARTIN, Alban MAXIMIN et **Tristan AVERTY**

Institut de Recherche et d'Études Navales (IRENav), École navale / ENSAM

Conference on Artificial Intelligence for Defence (CAID 2025)

18 novembre 2025

ÉCOLE NAVALE

IRENAV

Arts Sciences et Technologies et Métiers

Outline

Outline

Background and issues

Conventional ANNs (Artificial Neural Networks)

- ✓ Very easy to understand
- ✓ Well studied for years
- ✗ Use of real numbers as activation (dense)
- ✗ High energy, data, and computing cost consumption
- ✗ Low fidelity to actual brain function
- ✗ Initially not suitable for time-varying signals (except for the RNNs)

Background and issues

SNNs (Spiking Neural Networks)

- ✓ Inspired by the biological functioning of the brain
- ✓ High energy efficiency (when implemented on neuromorphic hardware[1])
- ✓ Use spikes (discrete signals)
- ✓ Created to be suitable for time-varying signals
- ✗ Complex learning due to the non-differentiable nature of spikes
- ✗ Very recent : `snntorch` exists only since 2021

Initially introduced to process 1D signals in their natural representation, is this the best way to take advantage of SNNs ?

1. The most common practice is to simulate the training of a SNN on a « classical » computer using the backpropagation algorithm, then implement the resulting network on neuromorphic hardware.

Background and issues

- « **Spiking Neural Networks for More Efficient AI Algorithms** », 2020
  C. Eliasmith, University of Waterloo
  `https://www.youtube.com/watch?v=PeW-TN3P1hk`

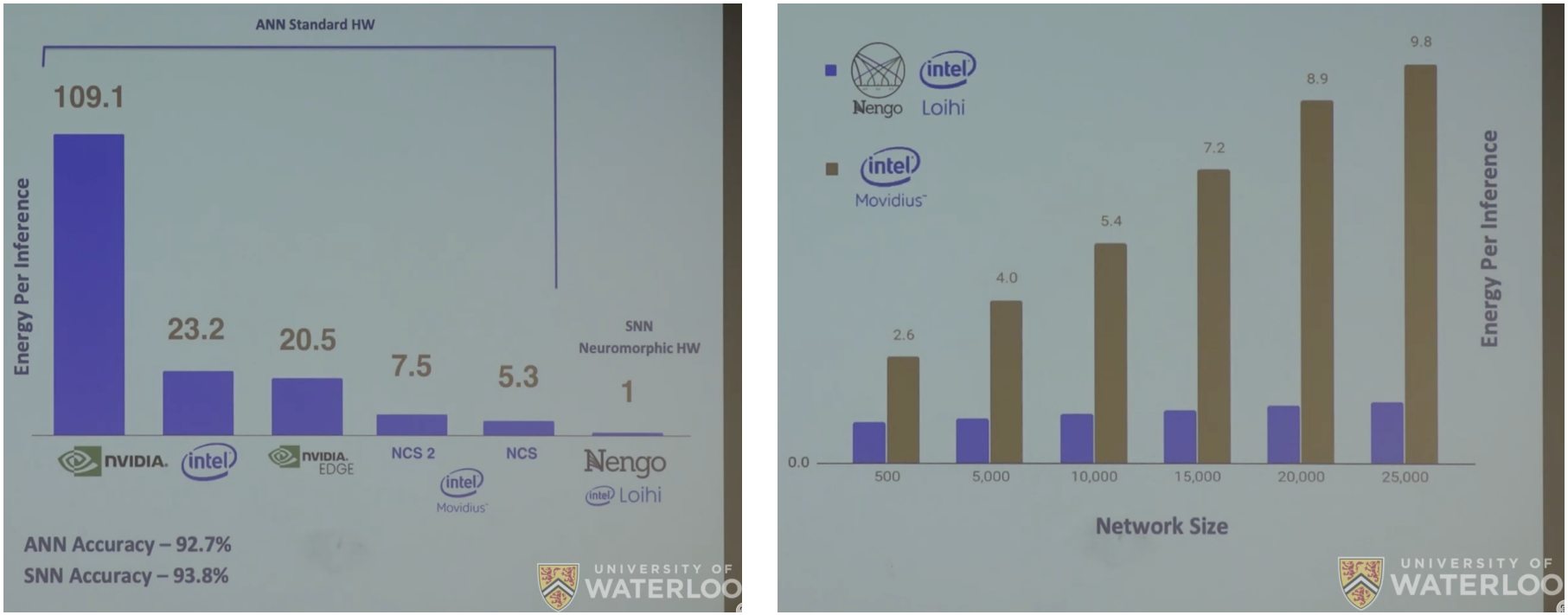


Figure – **Left** : Comparison of power consumption betwwen ANN and SNN / **Right** : Scalability of SNN

Background and issues

- « **Spiking Neural Networks : The Future of Brain-Inspired Computing** », 2025
  Sales G. Aribe Jr.

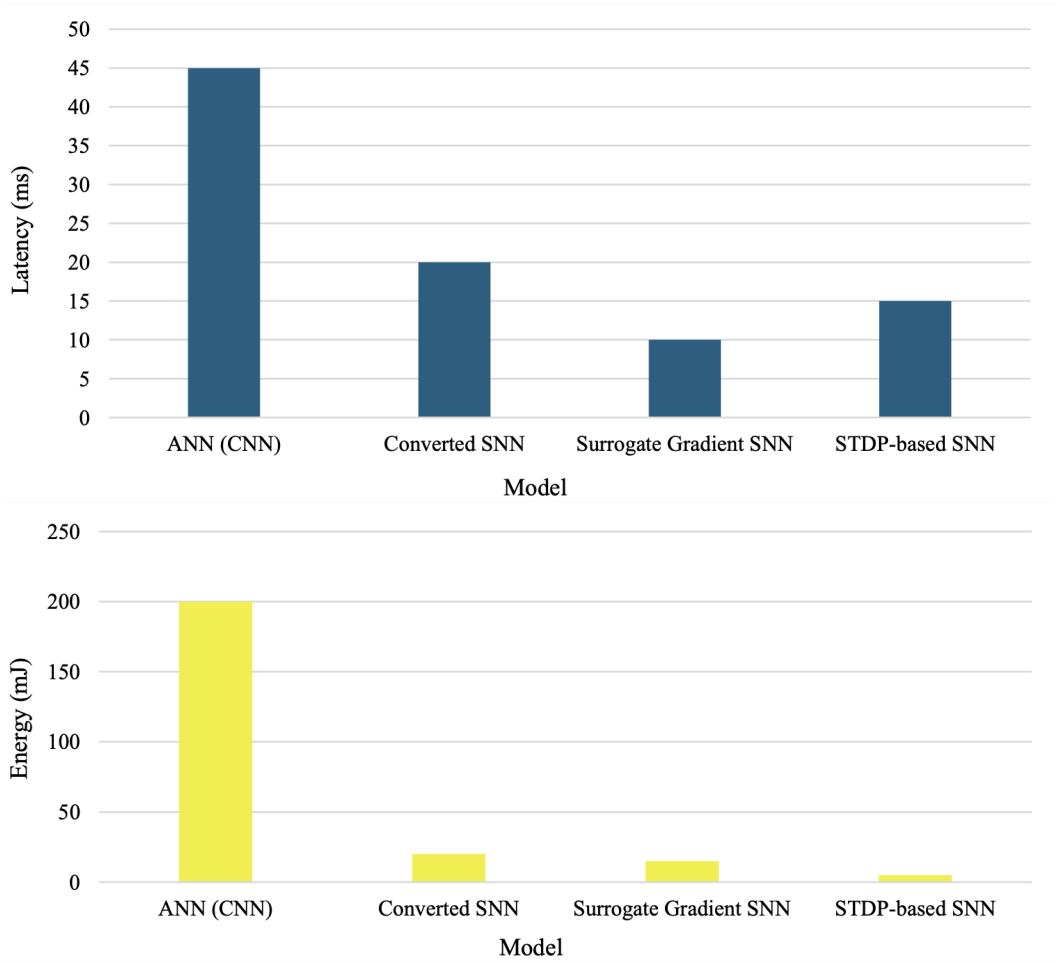
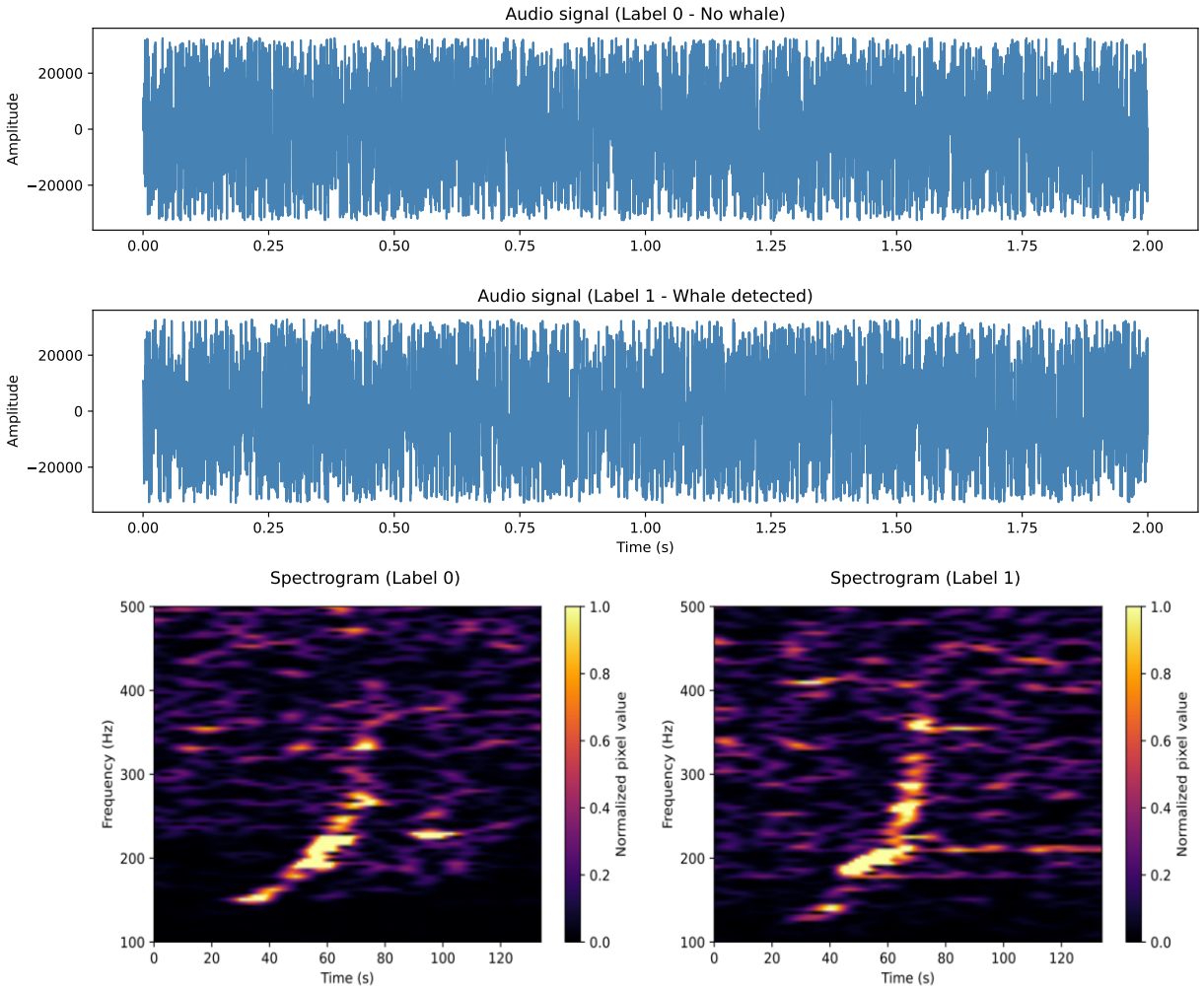


Figure – Latency (in ms) and energy (in mJ) comparison across models

The Marinexplore and Cornell University Whale dataset

- Dataset retrieved from a **Kaggle challenge**
- **Presence (1) or absence (0) of a right whale call**
- 30 000 **audio signals**
- Each lasting **2 seconds at 2 000 Hz**



Outline

Outline

Biological intuition behind Spiking Neural Networks

- In our brains, a neuron might be connected to 1 000 – 10 000 other neurons.
- If one neuron spikes, all downhill neurons might feel it.
- But what determines whether a neuron spikes in the first place ?
- If a neuron experiences sufficient stimulus at its input, then it might fire its own spike.
- Where does this stimulus come from ? Mostly from other pre-synaptic neurons.

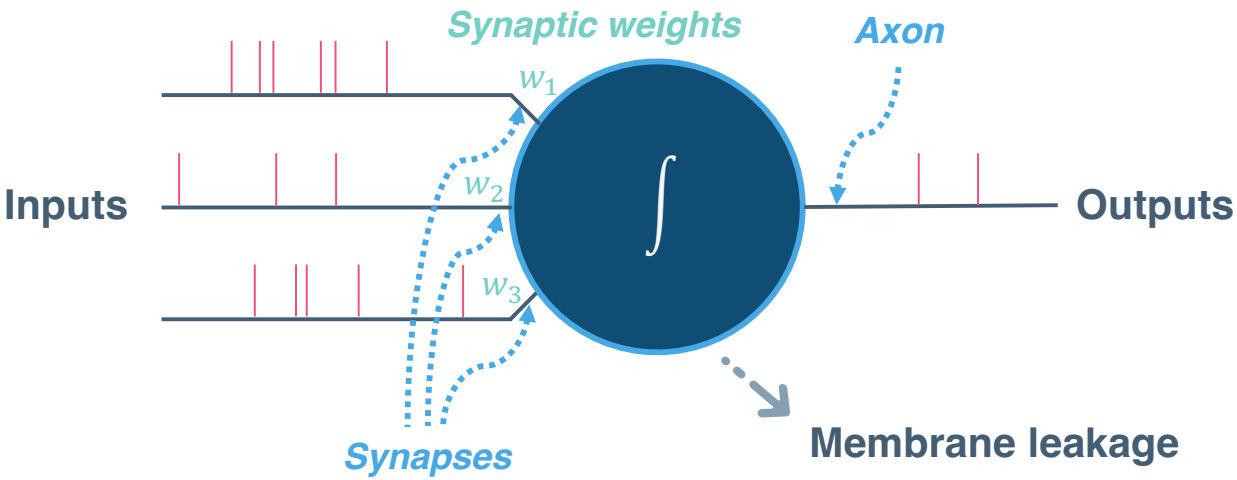


Figure – Principle of a spiking neuron

- Many neuron models : Lapicque, **Leaky Integrate-and-Fire**, Hodgkin-Huxley, etc.

Leaky Integrate-and-Fire (LIF) neuron

- The membrane discharge of a neuron is **similar to that of an RC circuit**

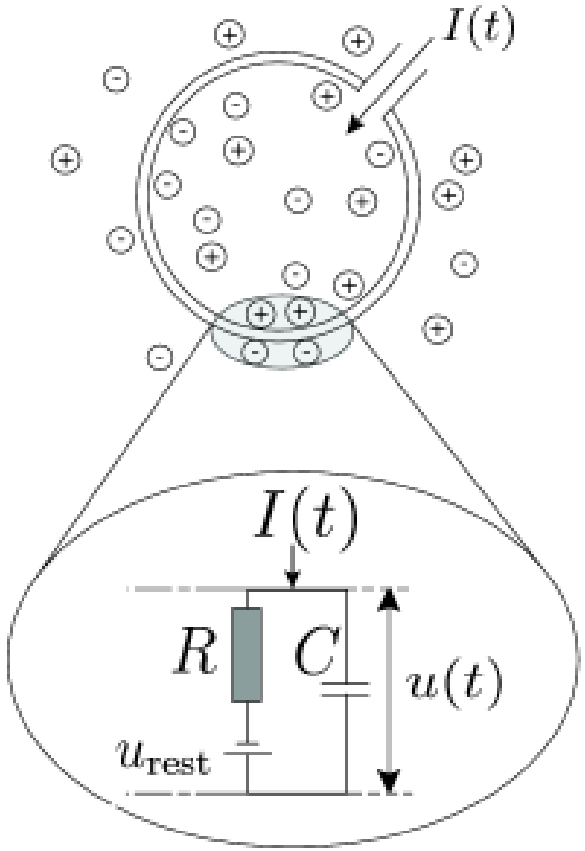


Figure – Electrical properties of neurons : the passive membrane [2]

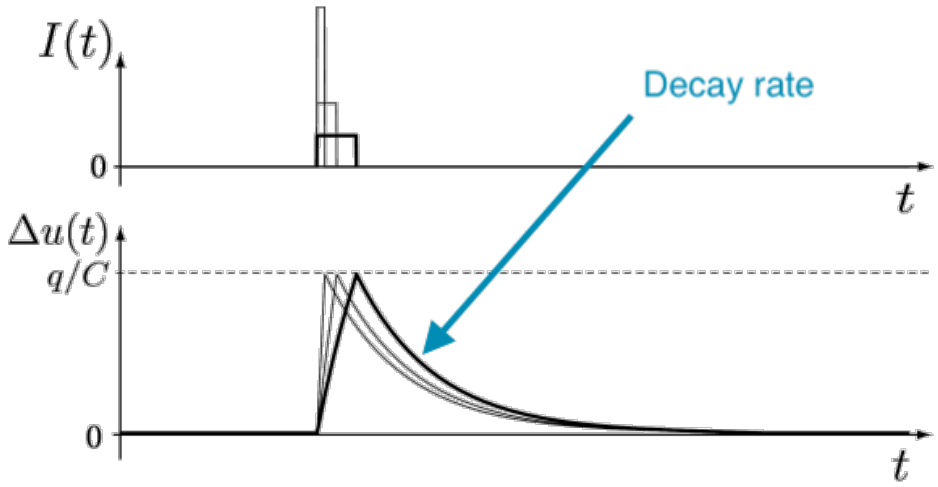


Figure – Short pulses and total charged delivered on the passive membrane [2]

2. https://neuronaldynamics.epfl.ch/online/Ch1.S3.html

Leaky Integrate-and-Fire (LIF) neuron

Simplification of the LIF model (discrete time)

Let $S(\cdot)$ be the step function :

$$S(U[t]) = \begin{cases} 1, & \text{if } U[t] > U_{\text{th}} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Then we have

$$\boxed{U[t+1] = \beta U[t] + WX[t+1] - S(U[t])U_{\text{th}}} \tag{2}$$

- $X[t]$ is an input voltage (or spike).
- W is the synaptic conductance. **This « weight » is the only learnable parameter**.
- β is the decay rate. **This is a hyperparameter**.

Leaky Integrate-and-Fire (LIF) neuron

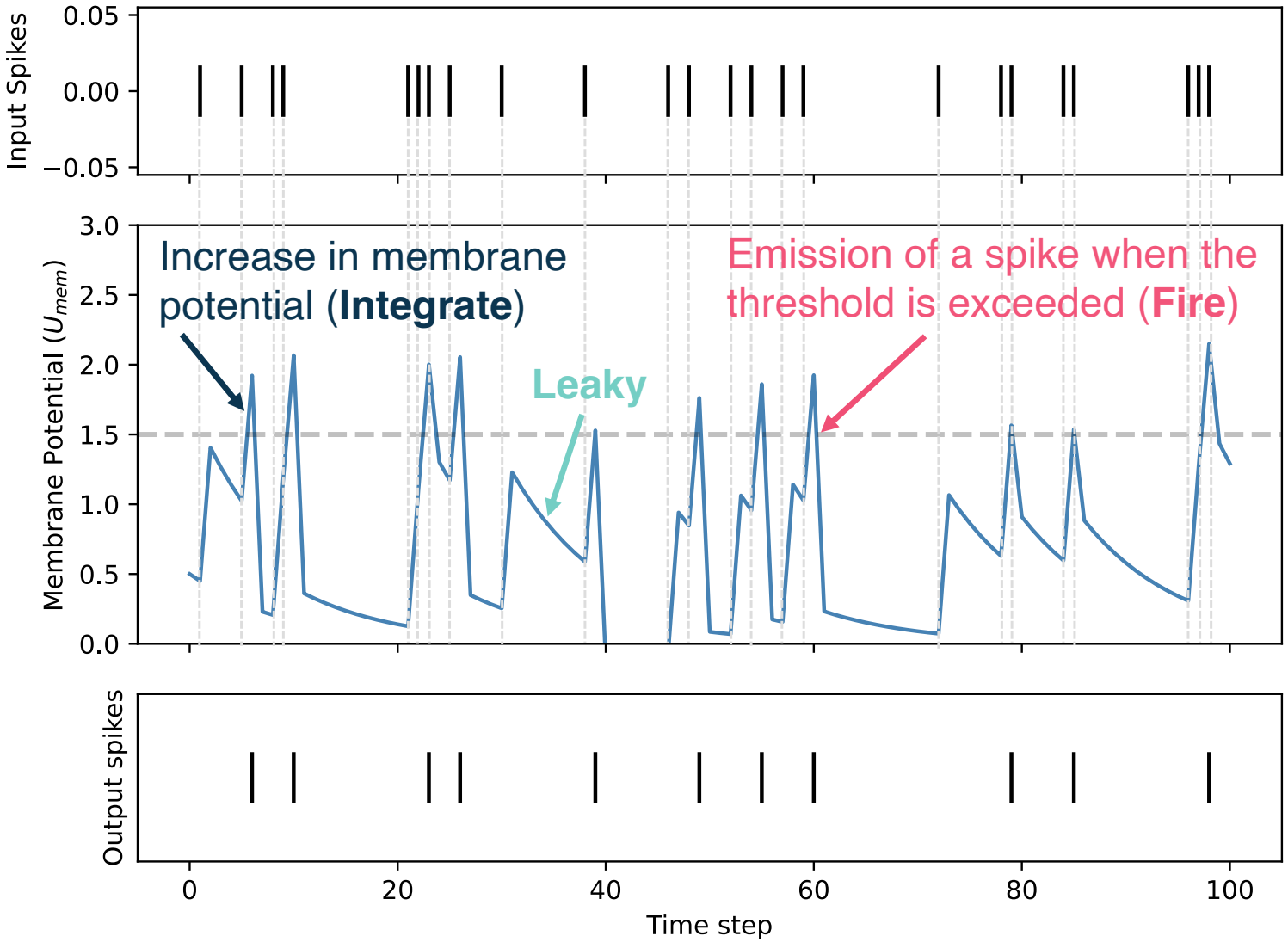


Figure – Evolution of the membrane potential when spikes are received

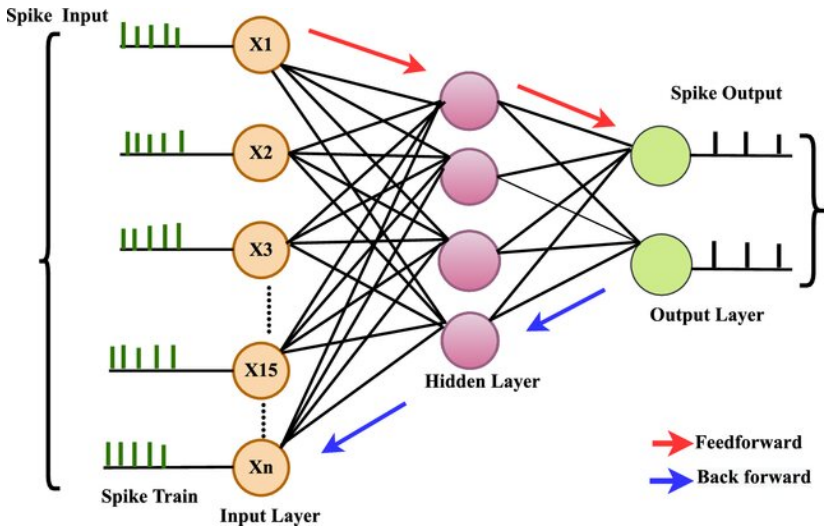
Feedforward of a SNN with LIF neurons

1 Spike encoding

- Create an initial spike sequence : so many ways
- Let the input layer neurons creating the spike sequence (the signal is thus treated as a direct current (DC))

2 Inference

- Each neuron receives weighted spikes from preceding neurons and, if sufficiently excited, fires a spike to the next neuron, and so on



3 Loss function

- In a conventional ANN, the output neuron with the highest activation is considered as the predicted class
- In a SNN, the most common way is to take the neuron with the highest spike count as the predicted class
- Computation of the conventional binary cross-entropy loss function :

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{3}$$

Backpropagation in a SNN with LIF neurons

4 Backpropagation

- Gradient descent (with learning rate η) of an arbitrary weight w :

$$w^{(k+1)} = w^{(k)} - \eta \frac{\partial \mathcal{L}}{\partial w} \tag{4}$$

- **Problem with the chain rule** :

$$\frac{\partial \mathcal{L}}{\partial w} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial S} \frac{\partial S}{\partial U} \frac{\partial U}{\partial w} \tag{5}$$

The activation function assigned to a LIF neuron is the non-derivable Heaviside step function (corresponding to spike emission).

- **Use of a surrogate gradient** :

$$\frac{\partial S}{\partial U} \approx \sigma_s'(U) \tag{6}$$

where $\sigma_s(U)$ is the *fast sigmoid* function with a variable slope (**hyperparameter**) $s > 0$:

$$\sigma_s(U) = \frac{U}{1 + |sU|} \tag{7}$$

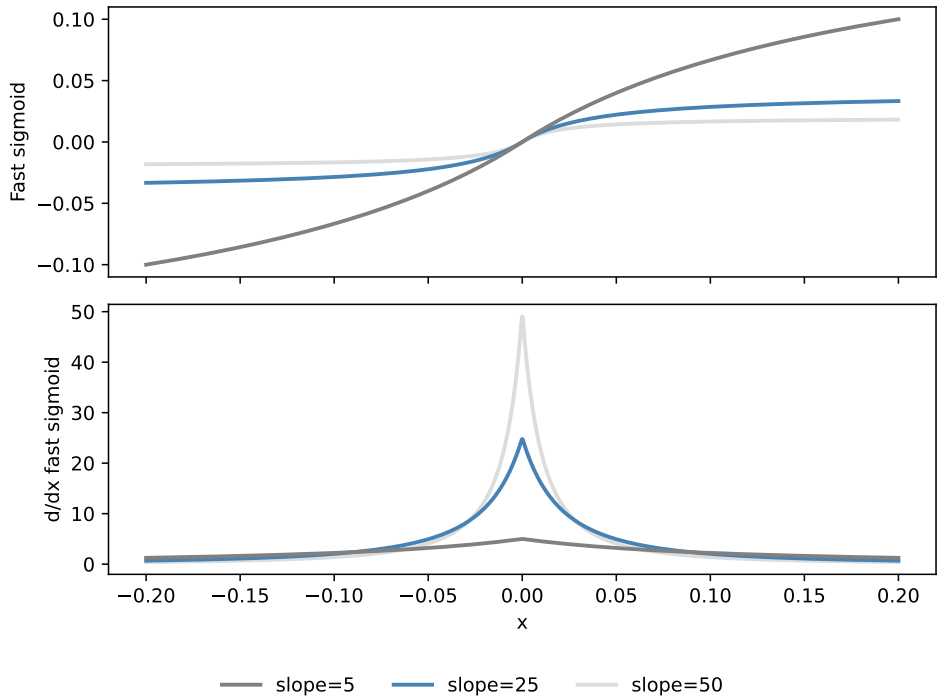


Figure – Fast sigmoid function and its derivative for different values of slope.

Outline

Short-Term Fourier Transform $\longrightarrow$ Spectrogram

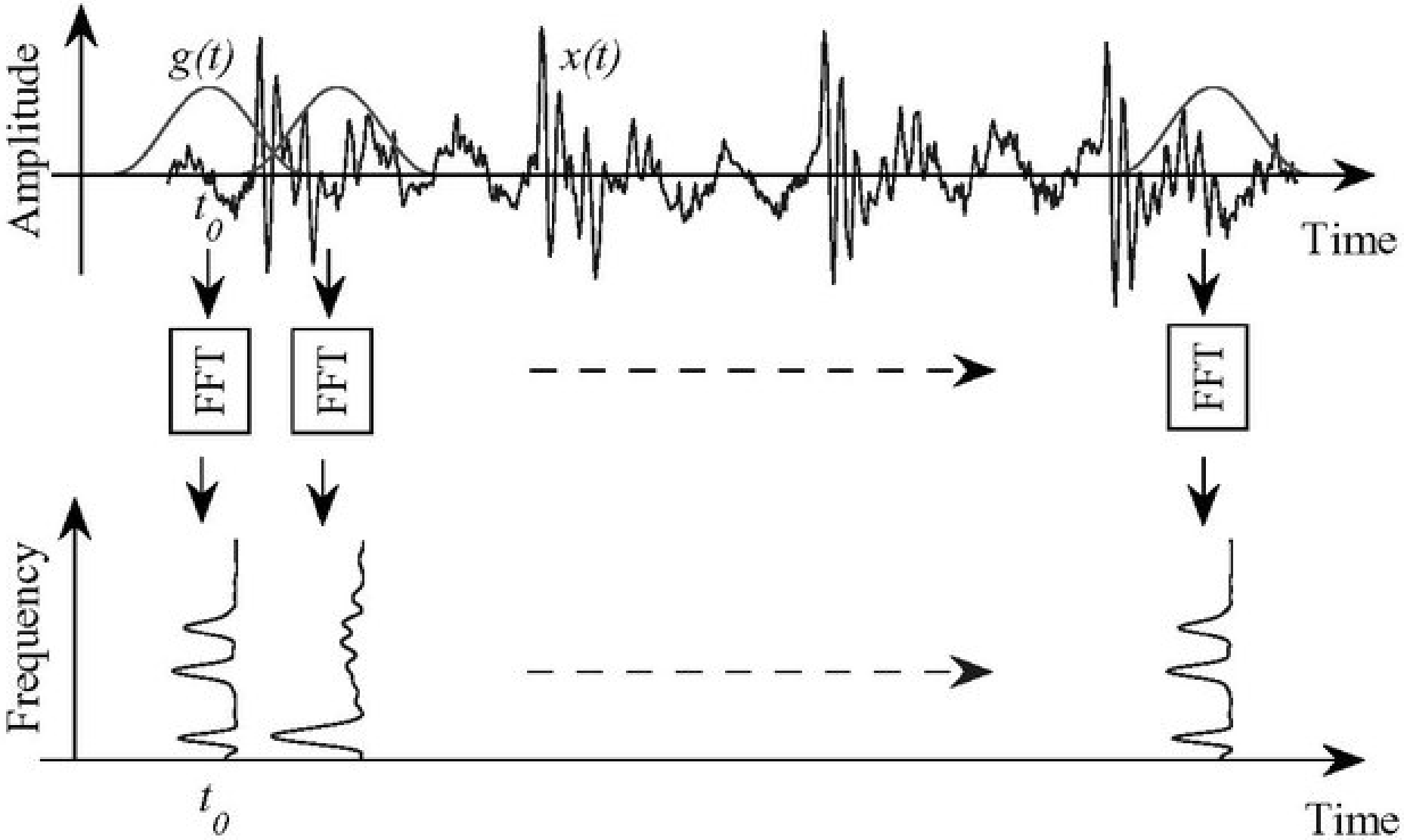


Figure – Principle of the Short-Term Fourier Transform (STFT)

Short-Term Fourier Transform $\longrightarrow$ Spectrogram

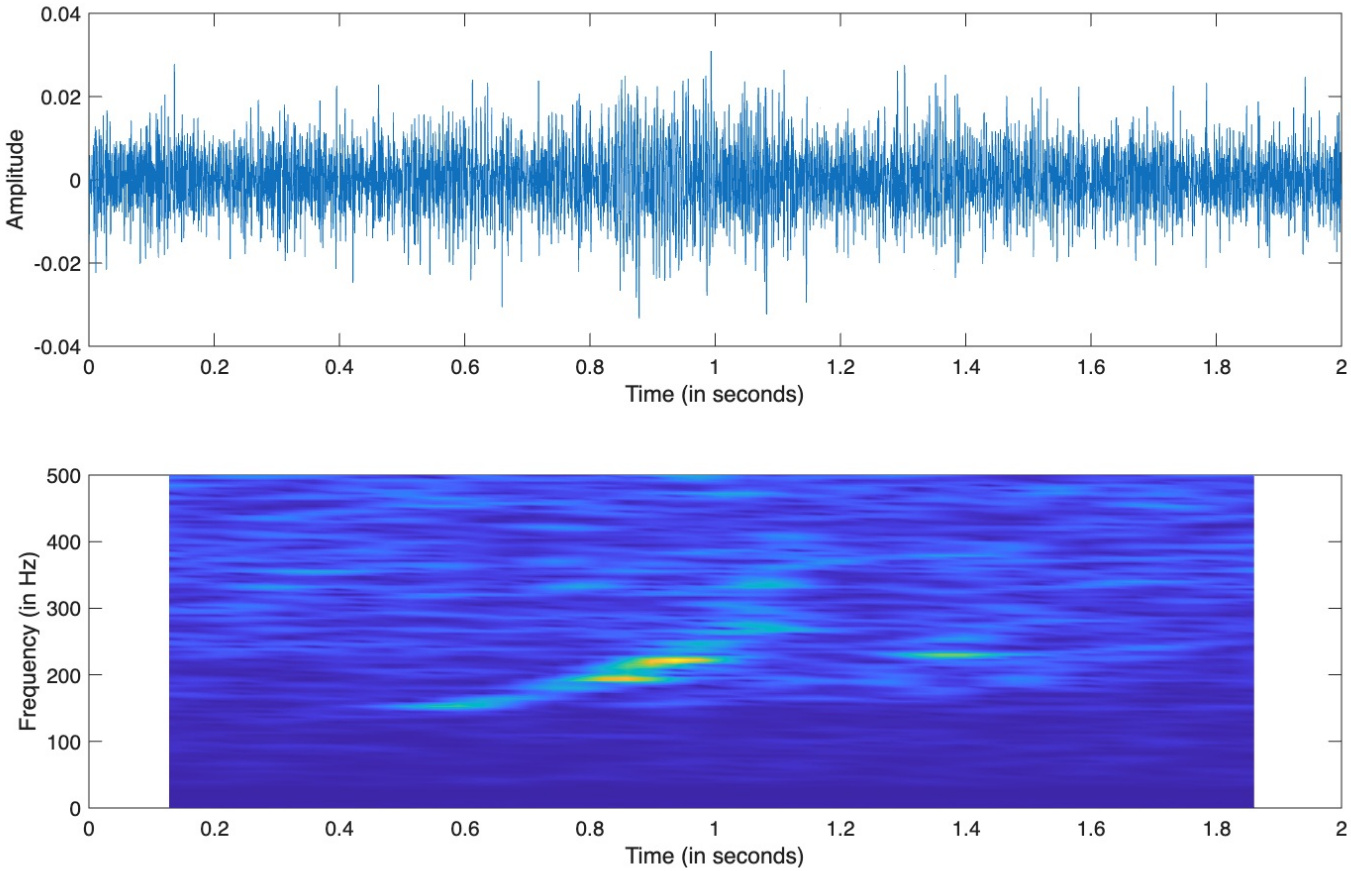


Figure – Audio signal from the dataset and its spectrogram

- **Hyperparameters** :
  - **Window** : Hamming function
  - **Number of sample frequencies** : $N_{\text{fft}} = 512$
  - **Overlap** : $N_{\text{overlap}} = 0.05 * N_{\text{fft}} = 486$

Outline

SNN architectures

- Inputs **must be normalized** between 0 and 1 (negative samples don't contribute $\longrightarrow$ useless)

	SNN with 1D inputs	SNN with 2D inputs
flatten	-	103 x 135
Input layer	8 000	14 175
Hidden layer 1	2 048	4 096
Hidden layer 2	512	512
Hidden layer 3	64	64
Output layer	2	2

- To have a model that converges and generalizes well while limiting overfitting :
  - **Learning rate scheduler**
  - K–fold **cross-validation**
  - **Grid search** to find the best hyperparameters

	Values
`beta`	0.8, 0.9, 0.95, *learnable*
`num_step`	1, 10, 20, 30, 40, 50

Classification results

- Performance metric : **Area Under the Curve** (AUC)
- Reflects the model's ability to **maximize true positives while minimizing false positives**

beta \ num_step	1	10	20	30	40	50
0.7	0.725	0.746	0.736	0.751	0.759	0.746
0.8	0.726	0.735	0.745	0.738	0.740	0.747
0.9	0.716	0.731	0.729	0.728	0.742	0.742
0.95	0.724	0.728	0.715	0.718	0.740	0.712
learnable	0.727	0.731	0.737	0.738	0.747	0.732

Table – Median AUC for the SNN with 1D inputs

beta \ num_step	1	10	20	30	40	50
0.7	0.737	0.920	0.922	0.921	0.911	0.891
0.8	0.648	0.922	0.921	0.925	0.923	0.919
0.9	0.502	0.923	0.925	0.925	0.927	0.918
0.95	0.602	0.922	0.925	0.926	0.926	0.926
learnable	0.500	0.924	0.924	0.926	0.927	0.923

Table – Median AUC for the SNN with 2D inputs

- **For two similar architectures, the input representation matters**

SNN training behavior

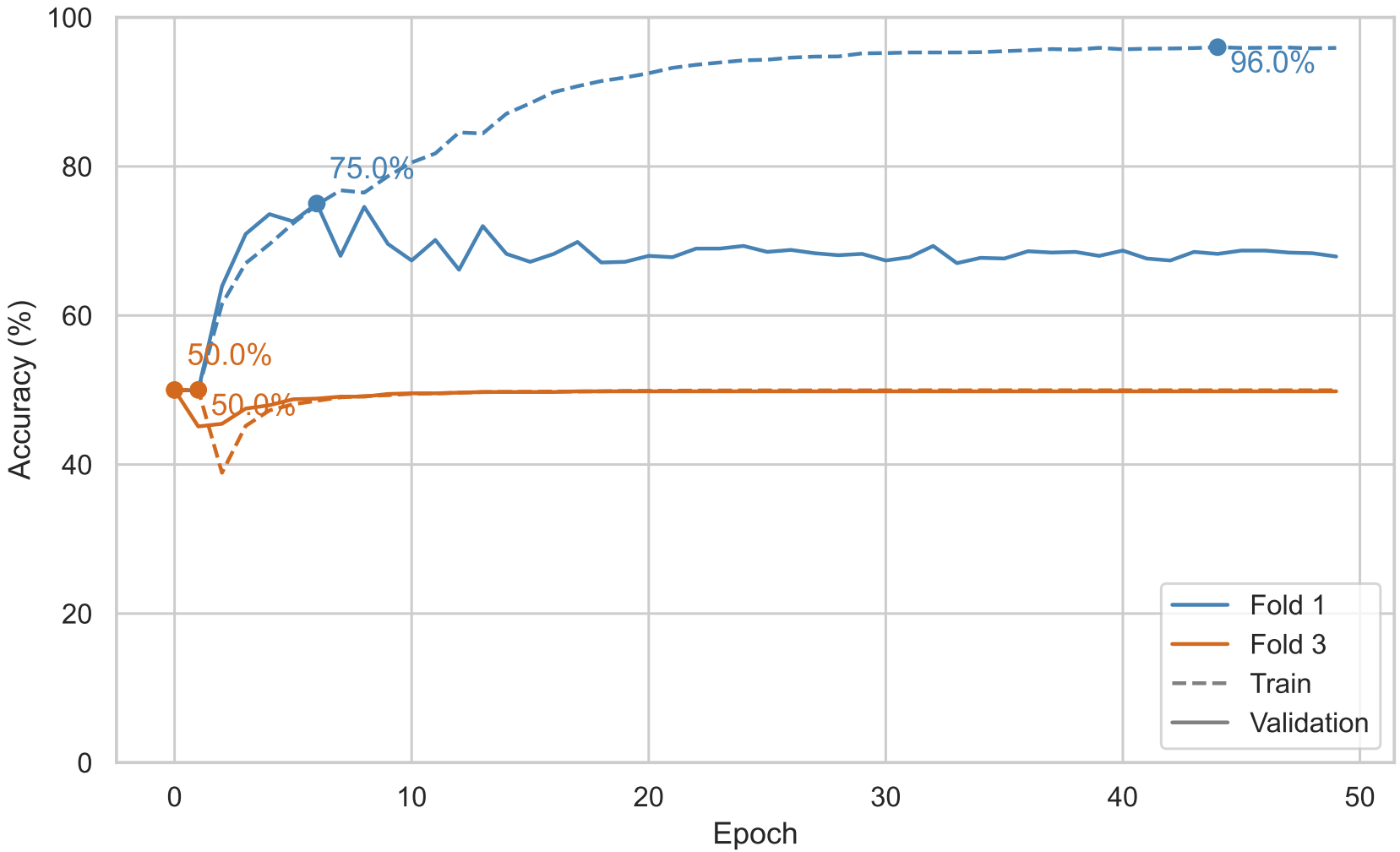


Figure – Illustration of learning failure on fold 3 when training the network with 1D inputs with identical hyperparameters (`beta` $= 0.7$ and `num_step` $= 40$).

SNN training behavior

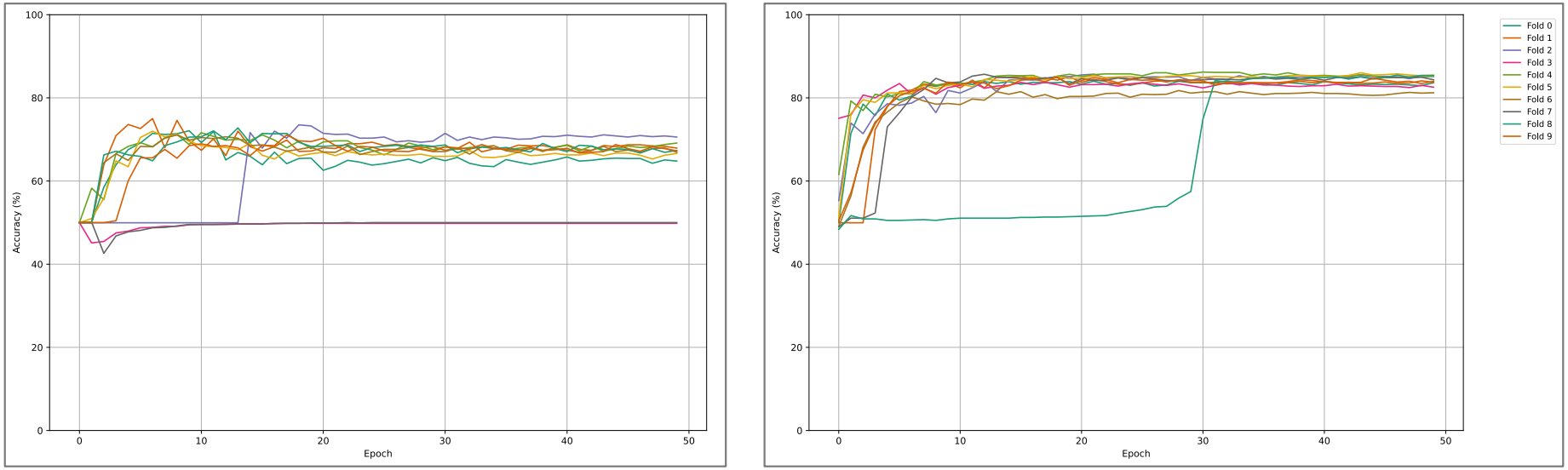


Figure – Accuracy on different validation sets for SNN with 1D inputs (left) or 2D inputs (right)

- The training dynamics reveal **clear differences between the two approaches**

Outline

Conclusion & Perspectives

✎ Getting started with a very recent type of neural network : **Spiking Neural Networks**

✎ Focus on **Leaky Integrate-and-Fire (LIF) Neuron**

✎ The **effect of the input representation** have been explored

✘ Problem to handle backpropagation due to the **non-differentiable spiking mechanism**

✘ **Difficult to understand all the training mechanisms** (presence of some learning failures)

✘ The architectures are **far from being the best**

✘ The results of the Kaggle challenge show that **traditional networks are still better**

✍ Other simpler but **more explainable architectures** could be studied.

✍ A **traditional convolution** can be done where the results are **inferred in a 1D SNN**

✔ **Efficiency** of such neural networks when implemented on neuromorphic hardware

✔ Interesting in the case of **onboard systems**

✔ Useful for **detection and classification tasks**

Thanks for your attention