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

Noémie MARTIN
IRENav
École navale
Brest, France
martin53.eleve@ecole-navale.fr

Alban MAXIMIM
IRENav
École navale
Brest, France
maximin.eleve@ecole-navale.fr

Tristan AVERTY
IRENav
École Navale / Arts & Métiers ParisTech
Brest, France
tristan.averty@ecole-navale.fr

***Abstract*—Spiking Neural Networks (SNN) are a promising way of classifying time series, thanks to their energy efficiency and their ability to model biological temporal dynamics. The aim of this work is to study the influence of the form taken by the input data—1D raw signal vs. 2D time-frequency representation (spectrogram)—on the performance of a SNN in a binary classification task of sounds emitted by right whales. After searching for optimal hyperparameters using a 10-fold cross-validation, the results highlight that representing time series as spectrograms significantly improves time-frequency pattern discrimination and stabilizes network training, demonstrating the value of integrating a 2D representation for time series classification thanks to a SNN. These results are all the more interesting in that SNNs were originally introduced to handle one-dimensional time signals.**

***Index Terms*—Spiking neural network, binary classification, supervised learning, LIF neuron, signal processing, spectrogram.**

I. Introduction

Artificial neural networks (ANNs) such as convolutional neural networks (CNNs) or multilayer perceptrons (MLPs) have shown a great efficiency for numerous applications, among them data processing, object recognition and brain activity modelization [1]. Although designed for this purpose, these models remain only loosely inspired by the functioning of biological neurons and differ substantially from the temporal and energetic dynamics observed in the human brain [2]–[4].

In this context, a new class of so-called neuromorphic neural networks, spiking neural networks (SNNs), has been gaining increasing interest. Unlike ANNs, which use continuous functions to enable gradient backpropagation—an essential building block for training them—SNNs use spikes, adding a temporal dimension to data processing [1]. Nevertheless, SNNs remain globally complicated to train, mainly due to complex neuronal dynamics and the non-differentiable nature of neuronal discharge operations [2], [5].

Being inherently energy-efficient thanks to their event-driven impulse mode of operation, SNNs are seen as more energy-efficient than ANNs, as they replace energy-consuming weight multiplications with simpler additions [6] — making them particularly well suited to contexts where energy efficiency is crucial, such as embedded systems (e.g. UAVs, AUVs, autonomous cars) [2]. Their ability to process temporal information in real time [1], [6] also makes them relevant for onboard intelligence in operational environments, such as those encountered in naval or aerial systems. To fully exploit these advantages, SNNs must be implemented on neuromorphic hardware and coupled with event-based sensors [5], [7], [8], where computation occurs only upon spike events. Thus, their efficiency depends more on reducing firing spikes than on shrinking network size [6], allowing a new generation of energy-efficient and resilient AI systems.

As the underlying philosophy of SNNs is to model biological neural processes, they were initially developed for time series processing [8], in particular to save computational resources. Nevertheless, recent work in the literature shows the possible application of SNNs to image classification [9], thanks to hybrid architectures combining convolution layers with spiking neurons [2]. This dynamic is now being extended to more complex tasks, such as object detection, traditionally performed by architectures like YOLO (*You Only Look Once*), a convolutional neural network widely used for real-time object identification and localization. To this end, the "spiking-YOLO" model [10] has recently been proposed, combining the advantages of YOLO with the energy efficiency of SNNs, demonstrating the feasibility and relevance of SNNs in performance-constrained computer vision tasks.

These considerations position neuromorphic architectures as promising candidates for onboard signal analysis in unmanned maritime, aerial, or land systems. The detection of whale vocalizations, used here as a representative acoustic task, illustrates how such architectures can enable real-time and low-power signal processing in complex and uncertain environments, including passive acoustic detection, environmental awareness, or threat identification scenarios.

The aim of this study is to compare the performance of a spiking neural network in the context of a task involving the binary classification of audio recordings. The data used are 2-second time series, sampled at 2 000 Hz, containing or not right whale vocalizations, from a contest organized on the *Kaggle* platform. More specifically, this study aims to assess whether SNN performance is influenced by the nature of the input data: the raw signal (time series) or its spectrogram (image representation of the signal's time-frequency content).

The outline of this article is as follows: section II explains how a spectrogram is built and how an impulse neural network works, and in particular how the gradient is back-propagated in this very specific context. Section III then presents the database used and the preprocessing applied to it. Section IV discusses the different spiking neural network architectures tested. Section V describes and discusses the classification results obtained. Finally, a conclusion opens the way to a few perspectives.

II. Spiking neural network & spectrogram

A. Leaky Integrate-and-Fire (LIF) neuron

The *Leaky Integrate-and-Fire* (LIF) neuron model is one of the most common model in SNNs [6] mainly due to its simplicity and its low calculation costs [2], [4], [11]. It modelizes the membrane potential of a neuron that integrates inputs (excitatory or inhibitory) over time while gradually losing energy (hence the term *leaky*) to reach a resting value of the membrane potential, like an RC circuit [3] [12]:

$$\tau_m \frac{dU(t)}{dt} = -(U(t) - U_{\text{rest}}) + R_m I(t), \tag{1}$$

$$\tau_m = R_m C_m. \tag{2}$$

Equations (1) and (2) governs the LIF model. Equation (1) describes the temporal dynamic of the membrane potential $U(t)$ where

- τ_m is the membrane time constant (in seconds), defined by the relation (2) as the product of the membrane resistance R_m (in ohms) and the membrane capacitance C_m (in farads);
- $U(t)$ (in volts) is the electric potential at time t;
- U_{rest} (in volts) is the resting potential, towards which the membrane potential tends in the absence of stimulation;
- $I(t)$ (in amperes) represents the injected input current received by the neuron at time t;
- $R_m I(t)$ is the contribution of the input current to the membrane potential, translated into voltage by Ohm's law.

The equation (1) says that, in the absence of an injected current, the membrane potential decreases exponentially towards the resting potential with a time constant τ_m. When a current $I(t)$ is injected, it contributes to modifying the membrane potential in proportion to the membrane resistance. When a threshold is reached, the neuron emits a spike and the amplitude of the emitted spike is subtracted from the potential (1 in general) [13]. Figure 1 illustrates the evolution of membrane potential by aggregating input spikes. When this potential exceeds the threshold set at 0.5, an output spike is emitted.

This model therefore illustrates the essentials of neuronal dynamics, without however being as complex as detailed biophysical models such as Hodgkin-Huxley (HH) [3], [4]. The HH model improves upon the LIF model by providing a biophysically detailed description of how action potentials are generated through voltage-gated ion channels, offering a much closer match to real neuronal behavior [9]. However, this realism comes at a high computational cost. In contrast, the Izhikevich model offers a computationally efficient compromise between the LIF and HH models [8], [11].

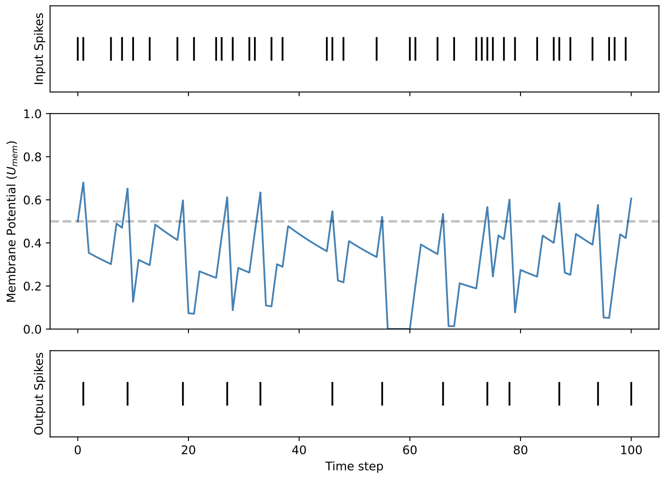

Figure 1. Evolution of membrane potential leading to the emission of 13 spikes.

B. Supervised learning

SNNs fall into the category of supervised learning algorithms and pose specific challenges due to the non-differentiable nature of the spikes, which are in a way binary activation functions [14]. To overcome this, the temporal backpropagation used for ANNs is adapted, allowing the error on the network weights to be backpropagated. The result is called surrogate gradient learning or spike-based backpropagation and replaces the non-derivable spike function with an approximate continuous function during learning [1] [2] [5]. These techniques make gradient descent possible in SNNs.

Let $\mathbf{y} = (y_i)_{1\leq i\leq N}$ be the vector of N true labels ($y_i \in \{0,1\}$ for all i) and $\hat{\mathbf{y}} = (\hat{y}_i)_{1\leq i\leq N}$ be the vector of N probabilities predicted by the SNN ($\hat{y}_i \in [0,1]$ for all i). The cost function used in supervised learning for a classification task is generally the cross-entropy :

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

As the spikes are non-differentiable functions, to apply optimization by gradient descent (*i.e.* backpropagation), the Dirac distribution $S(U)$ modeling a spike is replaced by a continuous function $\sigma(U)$, called surrogate, with a locally non-zero derivative (*e.g.* sigmoïd, bounded ReLU, ...) :

$$\frac{\partial S(U)}{\partial U} \approx \frac{\partial \sigma(U)}{\partial U}. \tag{4}$$

The approximate backpropagation of the gradient then becomes possible using the chain rule [6]

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

where w is an arbitrary weight of the SNN.

The activation function plays a crucial role in a neural network, as it introduces the non-linearity required for learning. To make gradient backpropagation possible, the surrogate function σ used here is the variable-slope fast sigmoid function defined by

$$\sigma_s(x) = \frac{x}{1 + |sx|} \tag{6}$$

where $s > 0$ is an adjustable slope factor. Figure 2 illustrates the fast sigmoid function $\sigma_s(x)$ for different values of s, as well as its derivative. The greater the slope, the more the function resembles an impulse. However, this increase in slope is associated with an increasingly steep and localized derivative, which can make learning unstable. This trade-off between expressiveness and stability must therefore be carefully considered when choosing the slope [15].

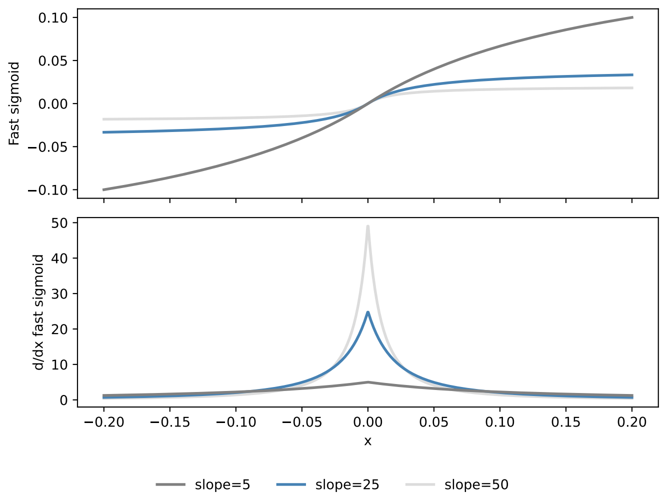

Figure 2. Fast sigmoid function and its derivative for different values of slope.

C. Spectrogram

In order to evaluate whether, in the context of a binary classification task, the performance of an SNN is influenced by the nature of the input data, we need to obtain a 2D representation of the audio signal. For this reason, we use in this article the spectrogram, the most widely used time-frequency representation. The spectrogram visualizes the evolution of a signal's frequency components over time. It is based on the Short-Time Fourier Transform (STFT), which consists of slicing the signal into short-duration segments using a sliding window that weights the signal to reduce the appearance of secondary lobes in the Fourier transform [16]. The Fourier transform is then applied to each of these segments. Formally, for a discrete signal $(x[n])_{0 \leq n \leq T-1}$ of T samples, the STFT is given by

$$X[n,k] = \sum_{m=0}^{T-1} x[m] \times h[n-m] \times \mathrm{e}^{-\mathrm{j}2\pi km/N_{\text{fft}}}, \tag{7}$$

where $h[\cdot]$ is the analysis window (the Hamming window is used in this work), N_{fft} is the total number of points (*i.e.* the number of frequencies) of the Fourier transform and k is the frequential index. This transform calculates the spectral content of the signal around time n. Spectrogram resolution depends on several parameters : the number of frequencies considered N_{fft} determines the frequency resolution and for better temporal resolution, it is also possible to overlap the sliding windows by a number of points N_{overlap}.

III. Database & preprocessing

A. Database source and structure

The database used comes from the "Whale Detection Challenge" hosted on the Kaggle[1] platform. The data were provided by Marinexplore and Cornell University and consist of audio recordings captured via a network of buoys along the North American east coast. The data present a wide range of acoustic diversity, including anthropogenic (*e.g.* marine traffic) and biological noise, making the detection task particularly complex.

The database contains 30 000 two-second AIFF recordings, sampled at 2 000 Hz and annotated in a CSV file according to the presence (label 1) or absence (label 0) of right whale calls. The distribution is as follows: 76.6% of recordings are labeled 0 and 23.4% of recordings are labeled 1. By comparing a signal labeled 0 with one labeled 1, Figure 3 illustrates the complexity of the right whale call detection task.

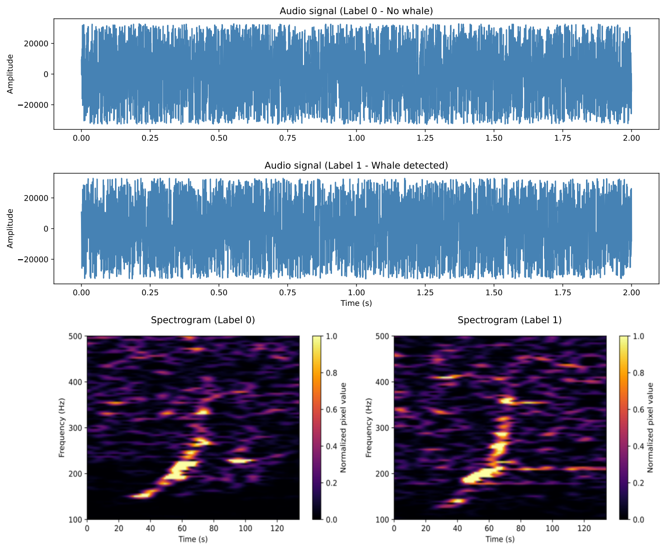

Figure 3. 1D (raw signal) and 2D (time-frequency) visualization of signals labeled 0 or 1.

B. Signal processing

Firstly, to improve learning, the database was processed in such a way as to balance the distribution between the two labels. Hence, 23.4% of the signals that are labeled 0 are randomly selected. It corresponds to 7027 time series.

[1]https://www.kaggle.com/c/whale-detection-challenge

The processing applied to the raw audio data begins with 4th order bandpass filtering to retain only the frequency components between 100 Hz and 500 Hz (a choice motivated by biological reasons). Next, a threshold at 1 and 99th percentiles is applied to limit the impact of extreme values. The signal is then decomposed into two distinct channels: one containing positive values, the other negative values. These two vectors are concatenated and the absolute values are taken to obtain a positive representation of the signal. This transformation is essential, as SNNs require inputs between 0 and 1. A simple normalization between 0 and 1 would distort the energy dynamics of the signal by assigning an average energy around 0.5 even to a silent signal, leading to saturation of the network with spikes. Therefore, this method preserves the energy coherence of the original signal, while partially preserving the phase information. Each signal is processed individually, taking into account its own extreme values. Figure 4 depicts the above-mentioned processing on a synthetic time series.

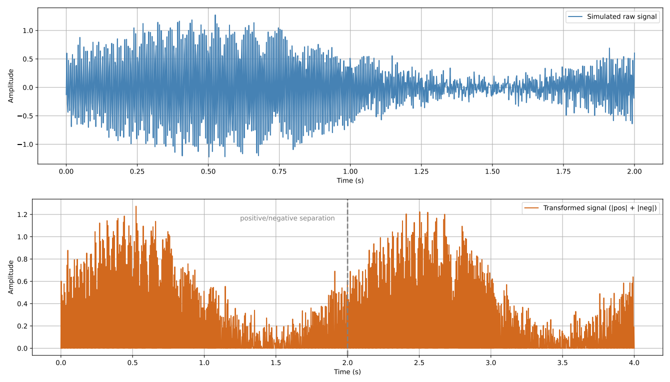

Figure 4. Up : Raw synthetic time series / Down : Preprocessed synthetic time series (separation and concatenation of components)

The processing applied to the spectrograms also includes a thresholding step at the 99th percentile. This is followed by filtering of frequencies between 100 Hz and 500 Hz, then normalization between 0 and 1. Each spectrogram is processed individually, taking into account its own extreme values.

IV. Methodology

A. Tools & libraries

The general construction of the neural networks is implemented using the `pytorch` library. To integrate the spiking character specific to this study's approach, we used `snntorch`, a library specially designed to be used as an extension of `pytorch` for spiking neural networks. It provides adapted modules (spiking neurons, time-coding mechanisms, etc.) to simulate the discrete-time behavior of this type of neuron. The `scikit-learn` library is used for performance evaluation, notably using the AUC-ROC metric, as well as for cross-validation. Finally, some processing operations require signal operations (filtering, spectrogram generation), which are performed with the `scipy` module.

B. Spiking neural network architecture

Table I summarizes the architectures of the two considered networks. The encoding of the data in spikes is made by the input layers. The static data is thus treated as a direct current (DC), whose characteristics are transmitted to the first layer of the network at each time step [3]. The neuron model chosen is the LIF neuron as presented in subsection II-A.

	1D input	2D input
Input layer	8 000	14 175 (105 $\times$ 135)
Hidden layer 1	2 048	4 096
Hidden layer 2	512	512
Hidden layer 3	64	64
Output layer	2	2

Table I
Number of neurons per layer for the two networks.

C. Hyperparameters optimization

Firstly, a K-fold cross-validation is performed to train the networks and find the best hyperparameters. As a reminder, cross-validation is a statistical evaluation method used to measure a model's ability to generalize and thus optimize hyperparameters while limiting overfitting. Formally, the database, composed of 14 054 elements, is divided into K subsets and the model is trained K times: at each time, $K-1$ subsets are used to train the neural network and the K^{th} is used for validation. Finally, the scores are averaged over the K validations to obtain the global metric. The following hyperparameters are set:

- Batch size : 512
- Spike emission threshold : 1
- Initial learning rate : 10^{-5}
- Slope of the surrogate fast sigmoïd function : 25

The use of a scheduler such as `ReduceLROnPlateau` for training allows to dynamically adjust the learning rate (an essential parameter of a gradient descent) in response to performance stagnation on the validation set, measured via the loss function. Given the complexity of optimization in SNNs, this adaptation stabilizes learning, facilitates convergence and achieves better performance while reducing the risk of overfitting. Concretely, the patience of the scheduler corresponds to the number of epochs where the value of the loss function can stagnate (*i.e.* vary by a value below the set threshold) before the learning step decreases proportionally to the decrease factor. Thus, the list of set hyperparameters is completed by those of the scheduler:

- `patience` : 5 for the SNN with 1D inputs and 3 for the SNN with 2D inputs
- `factor` : 0.5
- `threshold` : 10^{-4}

Finally, a grid-search is performed to find the best combination of hyperparameters `beta` (leakage rate of LIF neurons) and `num_step` (number of time steps):

- `num_step` : search among $\{1, 10, 20, 30, 40, 50\}$
- `beta` : search among $\{0.8, 0.9, 0.98, \textit{learnable}\}$

The *learnable* option means that the `beta` parameter is initially set to 0.9 and then optimized during training as a network parameter.

V. Results & discussions

A. Performance score

The ROC curve (for Receiver Operating Characteristic) is a curve used to evaluate the performance of a binary classifier. It consists of plotting the true positive rate against the false positive rate for different threshold values, the threshold being set to decide whether the neuron output predicts 1 or 0. Calculating the area under the curve (AUC) summarizes the quality of the neural network across all thresholds [17]. The AUC is particularly interesting because it is independent of class distribution and the arbitrary choice of a classification threshold. What's more, unlike a simple accuracy measure, it reflects the model's ability to maximize true positives while minimizing false positives, which is crucial in sensitive applications where the cost of errors differs according to their nature [18].

B. Results

To compare the different configurations tested, the median of the AUC score on the K-folds was preferred to the mean. This is because the SNN model has an instability intrinsic to training that can lead to learning failures on certain folds, as illustrated in Figure 5. In this context, the mean is sensitive to these extreme values, while the median provides a more robust and representative estimate of actual performance.

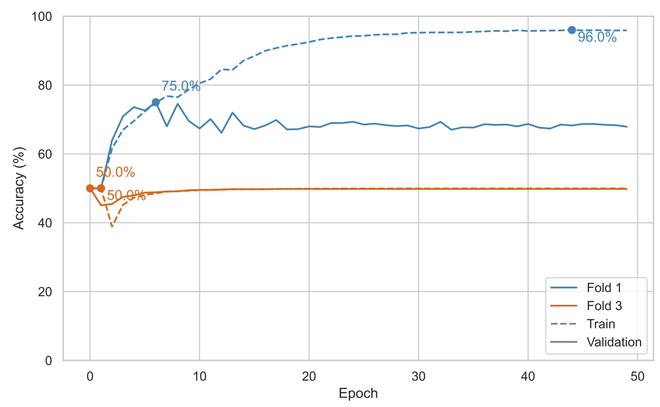

Figure 5. Illustration of learning failure on fold 3 when training the network with 1D inputs with identical hyperparameters (`beta` $= 0.7$ and `num_step` $= 40$).

Tables II and III summarize the median AUC scores obtained on the different cross-validation datasets according to the values of parameters `beta` and `num_step`. The best results are highlighted.

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 II
Median AUC scores according to `beta` and `num_step` parameters for the network 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 III
Median AUC scores according to `beta` and `num_step` parameters for the network with 2D inputs.

C. Discussion

Performance evaluation was based on the best median AUC scores obtained from K-fold cross-validation, for each architecture tested. The best SNN model receiving raw 1D data (temporal audio signal) achieved a median score of 0.759 with the optimal hyperparameters `beta` $= 0.7$ and `num_step` $= 40$. In comparison, the best SNN model taking 2D inputs (spectrograms) achieves a significantly higher score of 0.927 for `beta` $= 0.9$ and `num_step` $= 40$. These results clearly show that the representation of input data has a major impact on the performance of an SNN. Exploiting the time-frequency structure of the signal through 2D spectrograms enables the network to extract discriminating patterns more efficiently, particularly in an audio classification context. Conversely, 1D raw data appear to be insufficiently informative for efficient learning with an SNN.

Beyond performance, learning dynamics reveal clear differences between the two approaches. The network which takes 1D inputs shows a tendency towards overfitting: validation accuracy peaks before decreasing slightly, while training accuracy continues to rise. Learning is also unstable, with two out of ten folds showing no progression (the *accuracy* stagnate around 50%), indicating a lack of convergence. In comparison, the network which takes 2D inputs, shows stable and regular learning: the accuracy curves for training and validation follow parallel trajectories, with no observable overfitting. This stability is consistent both within and between folds. These results confirm that spectrogram (2D input) is better suited to SNNs, offering both better performance and more reliable learning for audio signal classification.

Finally, it should be noted that the test dataset was only provided to contest participants, so we are unable to accurately compare ourselves with their models, the results of which are published on the contest website[2] and, in some cases,

[2]https://www.kaggle.com/c/whale-detection-challenge/leaderboard

have been published [19], [20]. However, we must remain realistic: the results obtained using their methods will certainly be much better than those obtained with our "simple" SNN architectures. Nevertheless, the real objective of this work was to focus specifically on the influence of representations on performance and training behavior.

VI. Conclusion

This work confirms that, for the binary classification of audio signals by an SNN, a 2D time-frequency input (spectrogram) provides both better performance and higher learning stability than the 1D raw signal. Switching to a time-frequency representation enables discriminative features to be extracted more efficiently, leading to an AUC increase of over 0.15. The performance obtained on the train set can be complemented by results obtained on a test set.

Although this representation is naturally two-dimensional, it is vectorized before being injected into the SNN, and thus treated as a 1D vector. This means that no convolution operation is currently used to explicitly exploit the local spatial correlations present in the spectrograms. This limitation raises a potential improvement: the integration of spatio-temporal convolutional layers could improve network performance while remaining compatible with the constraints of deployment on neuromorphic hardware.

In parallel, sample-wise SNNs have been developed and tested, capable of processing each audio sample step by step, as well as neuromorphic recurrent loop architectures to better capture fine temporal dynamics. These approaches, combined with vectorized 2D representation, open up promising perspectives for enhancing both the robustness and energy efficiency of SNNs for audio signal classification tasks. Such properties—real-time operation, low energy consumption, and temporal adaptability [21]—make these architectures particularly relevant for future embedded applications requiring autonomous and resilient perception in complex environments. Moreover, it should be noted that some recent studies show that it might be more beneficial to use hybrid ANN (for parallel processing) / SNN (for event-based approach) architectures for further energy savings [22].

References

[1] Guillaume Marthe. *Neurones à impulsion pour les communications sans fil*. PhD thesis, Institut National des Sciences Appliquées de Lyon, 2024.

[2] Kashu Yamazaki, Viet-Khoa Vo-Ho, Darshan Bulsara, and Ngan Le. Spiking Neural Networks and Their Applications: A Review. *Brain Sciences*, 12(7):863, 2022.

[3] Jason K Eshraghian, Max Ward, Emre O Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE*, 111(9):1016–1054, 2023.

[4] Veïs Oudjail. *Réseaux de neurones impulsionnels appliqués à la vision par ordinateur*. PhD thesis, Université de Lille, 2022.

[5] João D. Nunes, Marcelo Carvalho, Diogo Carneiro, and Jaime S. Cardoso. Spiking neural networks: A survey. *IEEE Access*, 10:60738–60764, 2022.

[6] Yufei Guo, Xuhui Huang, and Zhe Ma. Direct Learning-Based Deep Spiking Neural Networks: A Review. *Frontiers in Neuroscience*, 17:1209795, 2023.

[7] Michael Pfeiffer and Thomas Pfeil. Deep Learning With Spiking Neurons: Opportunities and Challenges. *Frontiers in Neuroscience*, 12:774, 2018.

[8] Hagar Hendy and Cory Merkel. Review of spike-based neuromorphic computing for brain-inspired vision: biology, algorithms, and hardware. *Journal of Electronic Imaging*, 31, 2022.

[9] Y. Dan, Z. Wang, H. Li, and J. Wei. Sa-snn: Spiking attention neural network for image classification. *PeerJ Computer Science*, 10, 2024.

[10] Seijoon Kim, Seongsik Park, Byunggook Na, and Sungroh Yoon. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11270–11277, 2020.

[11] Yin Bojian. *Efficient and Accurate Spiking Neural Networks*. PhD thesis, Eindhoven University of Technology, 2022.

[12] Jiankun Chen, Xiaolan Qiu, Chibiao Ding, and Yirong Wu. SAR Image Classification Based on Spiking Neural Network through Spike-Time Dependent Plasticity and Gradient Descent. *ISPRS Journal of Photogrammetry and Remote Sensing*, 188:109–124, 2022.

[13] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks: Bringing the Power of Gradient-based optimization to spiking neural networks. *IEEE Signal Processing Magazine*, 36(6):51–63, 2019.

[14] Hyeryung Jang, Osvaldo Simeone, Brian Gardner, and Andre Gruning. An introduction to probabilistic spiking neural networks: Probabilistic models, learning rules, and applications. *IEEE Signal Processing Magazine*, 36(6):64–77, 2019.

[15] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks. *Neural Computation*, 30(6):1514–1541, 2018.

[16] A. O. Boudraa. Traitement du signal avancé. https://sites.google.com/view/aboudra/teaching, 2024. Accessed in may 2025.

[17] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[18] Christopher M. Bishop and Nasser M. Nasrabadi. *Pattern Recognition and Machine Learning*. Springer, 2006.

[19] Evgeny Smirnov. North atlantic right whale call detection with convolutional neural networks. In *Proc. Int. Conf. on Machine Learning, Atlanta, USA. Citeseer*, pages 78–79, 2013.

[20] Kostiantyn Pylypenko. Right whale detection using artificial neural network and principal component analysis. In *2015 IEEE 35th International Conference on Electronics and Nanotechnology (ELNANO)*, pages 370–373. IEEE, 2015.

[21] Sales G Aribe Jr. Spiking Neural Networks: The Future of Brain-Inspired Computing. *arXiv preprint arXiv:2510.27379*, 2025.

[22] Manon Dampfhoffer, Thomas Mesquida, Alexandre Valentian, and Lorena Anghel. Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 7(3):731–741, 2022.