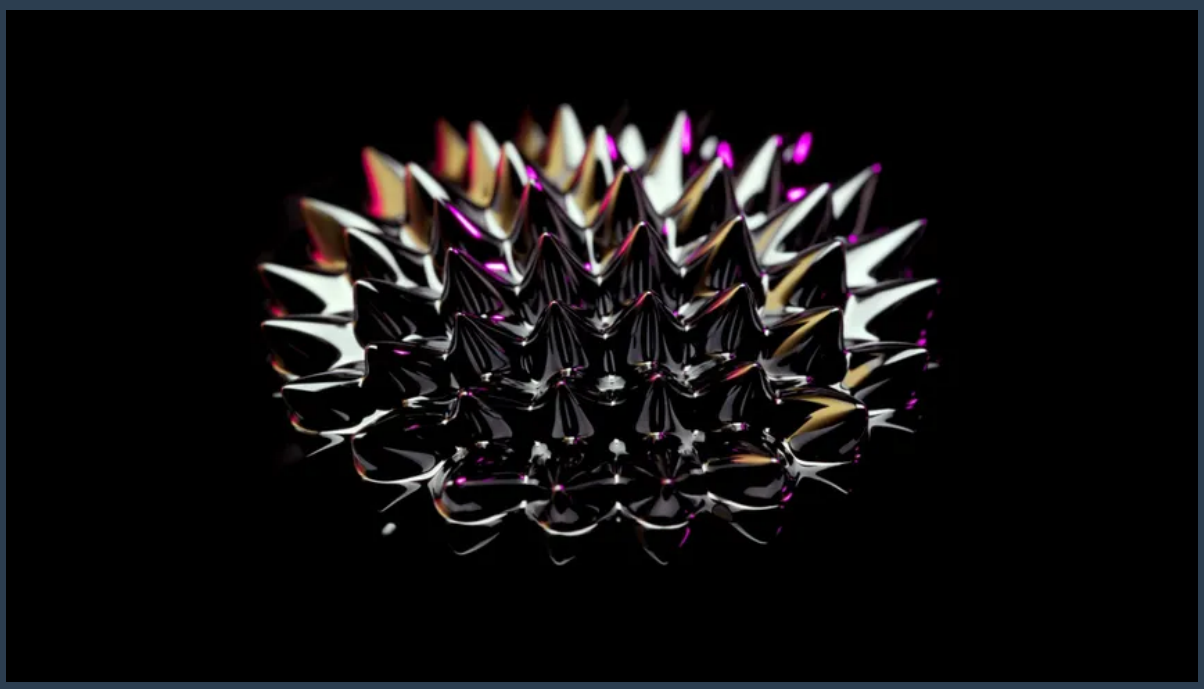
Final Report

Spiking Neural Networks
Sound Detection and Classification



COURREGE Téo
GANDEEL Lo'aï

Date: February 22, 2024

Contents

List of Figures

List of Tables

1 Introduction

In the last report, we explored the foundational aspects of Spiking Neural Networks (SNNs) and their potential applications, focusing on audio classification tasks. We delved into the theoretical underpinnings of SNNs, studying various neuron models and encoding methods such as rate, latency, and frequency coding. Additionally, we addressed challenges related to data preprocessing, verification, and the reconstruction of audio signals using Mel-frequency cepstral coefficients (MFCCs).

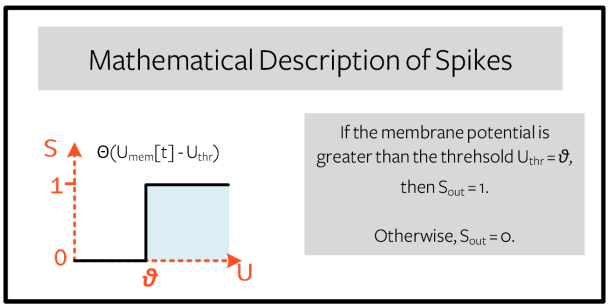


Figure 1: Mathematical description of a Spike

Building upon this foundation, the current report will delve deeper into the intricacies of SNNs, specifically addressing the non-differentiability issue inherent in these networks. Non-differentiability poses a challenge in applying traditional gradient-based optimization techniques commonly used in training Artificial Neural Networks (ANNs). We will explore strategies to tackle this issue and optimize SNNs effectively.

Furthermore, we will introduce the concept of Convolutional Spiking Neural Networks (CSNNs), extending the discussion beyond simple SNN architectures. CSNNs leverage the spatial hierarchies present in convolutional neural networks (CNNs) and integrate them with the temporal dynamics of SNNs. This fusion holds promise for tasks like image recognition, where both spatial and temporal features play crucial roles.

In the practical implementation section, we will present a small program showcasing our results. This program will include the application of CSNNs in a specific task, demonstrating the capabilities and potential advantages of this hybrid architecture.

Throughout this report, our aim is to provide a comprehensive understanding of the advancements and challenges in the realm of Spiking Neural Networks, offering insights into their unique characteristics and applications.

2 Reminder on Spiking Neural Networks

2.1 Spikes encoding

The idea behind spikes is simple: we need to encode some data (let's say an image) into some other kind of data that has a temporal dependency. To do so, we will need to define time steps, a number of steps, and a threshold. In a rate encoding scheme, we take the values of the pixels of the image and make them pass through a function that will output or not a spike (for example, a Bernoulli function).

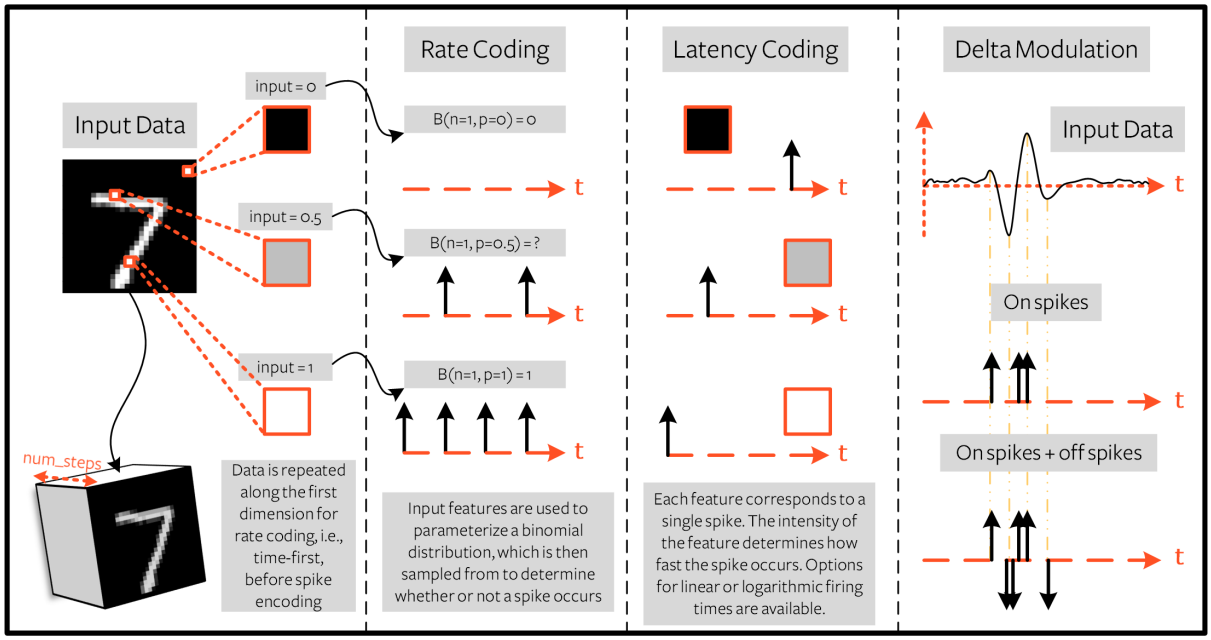


Figure 2: Spike encodings of an image

Let's say we have a white pixel (value close to 1) and a black pixel (value close to 0). We will then pass these values through a Bernoulli function with a threshold of 0.5. If the value is above 0.5, we will output a spike (1); if not, we will output 0. In doing this same procedure for all the pixels of the image, for all the time steps until the number of steps, we will have a temporal dependency of the image that, when combined, can approximate the original image.

3 Non-differentiability issue

To train a spiking neural network, we aim to adjust the weights based on the loss gradient, minimizing the overall loss. Backpropagation achieves this through a chain of derivatives:

$$\frac{\partial \mathcal{L}}{\partial W} = \frac{\partial \mathcal{L}}{\partial S} \underbrace{\frac{\partial S}{\partial U}}_{\{0,\infty\}} \frac{\partial U}{\partial I} \frac{\partial I}{\partial W}$$

Here, $\mathcal{L}$ is the loss, W represents weights, S is the output, U is the activation function, and I is the input.

The challenge lies in the term $\frac{\partial S}{\partial U}$, which takes values between 0 and ∞. The derivative of the Heaviside step function from the input (U) is the Dirac Delta function. This function is 0 everywhere except at the threshold θ, where it tends to infinity. Consequently, the gradient is often nullified to zero (or saturated if θ precisely aligns with the threshold), hindering learning. This issue is commonly known as the **dead neuron problem**. There are multiple ways to address this issue.

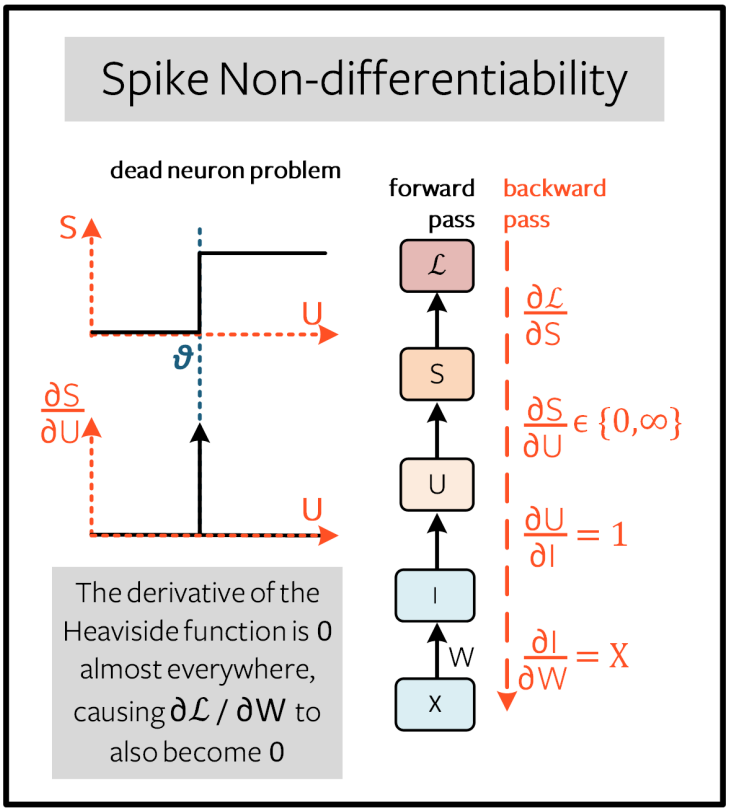


Figure 3: Dead neuron problem

3.1 Surrogate Gradient

One way to address this non-differentiability issue would be to compute a gradient on a "relaxed" version of the non-differentiable function. This would mean approximating the Heaviside step function $S(U)$ with a differentiable function $f(U)$, such as the sigmoid function or the $\arctan$ function. For example:

$$S(U) \approx \frac{1}{1 + e^{-\beta(U-\theta)}}$$

Or alternatively

$$S(U) \approx \frac{1}{\beta} \arctan(\beta(U - \theta))$$

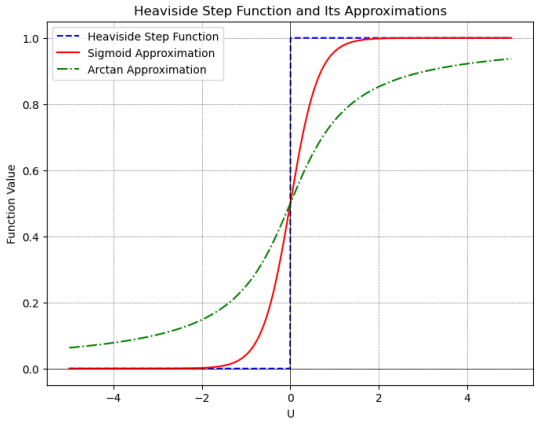


Figure 4: Visualization of the Heaviside step function approximation

3.1.1 Loss function treatment

In order for this to be taken into account in the loss function (and because we now have to take the time as a parameter), we have to perform an operation on the loss function for all its time steps.

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_t \frac{\partial \mathcal{L}[t]}{\partial W} = \sum_t \sum_{s \leq t} \frac{\partial \mathcal{L}[t]}{\partial W[s]} \frac{\partial W[s]}{\partial W}$$

4 Convolutional Spiking Neural Networks

5 Practical implementation