

First Report

Sound Detection and Classification
using Spiking Neural Networks

`./image/spike_brain.png`

COURREGE Téo
GANDEEL Lo'aï

Date: December 22, 2023

Contents

List of Figures

1 Introduction

Our project addresses the challenge of applying spiking neural networks (SNNs) to audio classification in the field of spiking neural network research. This report provides an overview of our initial progress in this area. Our project specifically addresses the problem of audio classification within the broader context of SNNs.

Before delving into the details, we outline key aspects including preprocessing, data manipulation/augmentation, initial model implementations, and a look at preliminary results.

Using audio data primarily from the [Google AudioSet](#) database, our work involves preprocessing, which includes conversion of signals to image representations, feature extraction, and consideration of encoding schemes suitable for SNNs. Challenges related to data quality, context, and labeling complexity prompted the exploration of data augmentation strategies to improve model robustness.

In the following sections, we will elaborate on the organizational structure of our project, the technical environment utilized, a detailed account of the work accomplished, challenges faced and solutions implemented, concluding with a reflection on our achievements and future perspectives.

2 The project

2.1 Description of the project - Spiking Neural Networks

2.1.1 Reminder of the Dow

Inspired by the neural signaling patterns of the human brain, SNNs introduce a temporal element into artificial neural networks. This temporal characteristic positions SNNs as promising candidates for real-time processing and pattern recognition tasks.

A Spiking Neural Network is a variant of artificial neural networks designed to more accurately mimic biological neural networks. Unlike traditional artificial neural networks (ANNs) that work with continuously changing time values, SNNs operate with discrete events occurring at defined times. They take a set of spike values as input and produce a set of spike values as output.

The spiking behavior of a neuron in an SNN is modeled by a membrane potential equation. For instance, in a leaky integrate-and-fire (LIF) neuron model, the membrane potential equation is defined by a set of parameters including a time constant (τ), resting potential (u_{r1}), reset potential (u_{r2}), synaptic weights (w_j), and a firing threshold (u_{th}). The output spike (s) is determined based on the membrane potential (u) and various conditions. This discrete event-based approach distinguishes SNNs from other types of neural networks.[?]

$$\begin{cases} \tau \frac{du(t)}{dt} = -[u(t) - u_{r1}] + \sum_j w_j \sum_{t_j^k \in S_i^{Tw}} K(t - t_j^k) \\ \begin{cases} s(t) = 1 & u(t) = u_{r2} \text{ if } u(t) \geq u_{th} \\ s(t) = 0 & \text{otherwise} \end{cases} \end{cases} \quad (1)$$

This equation was firstly formulated as :

$$\tau \frac{du(t)}{dt} = -[u(t) - u_r] + RI \quad (2)$$

From a mathematical perspective, Equation (??) represents a linear differential equation. Alternatively, an electrical engineer may recognize it as the equation of a leaky integrator or RC -circuit with parallel resistor (R) and capacitor (C). In the realm of neuroscience, this equation is termed the equation of a passive membrane. [?]

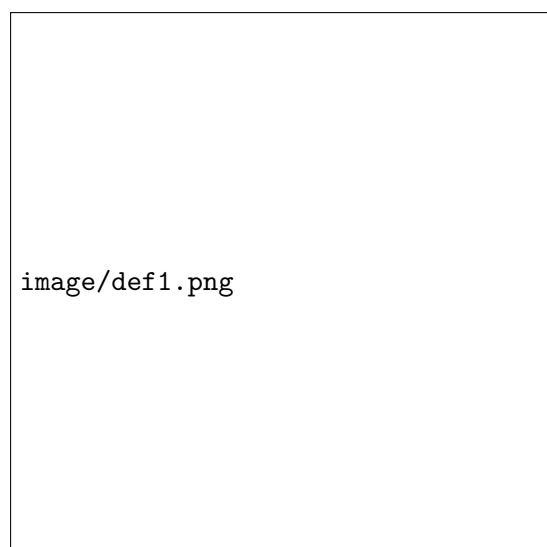


Figure 1: SNN input



Figure 2: SNN output

2.1.2 Audio classification task

Audio classification is a fundamental problem in the field of audio processing. It involves assigning a label to an audio clip based on its content. The audio classification task can be further divided into two subtasks: sound event detection (SED) and sound event classification (SEC). SED involves detecting the onset and offset times of sound events in an audio clip, while SEC involves assigning a label to each detected sound event.

In the context of audio classification, training an SNN involves working on image representations of audio data and encoding schemes suitable for SNNs.

2.1.3 Objectives of the project

The primary goal of our project is to exploit the temporal processing capabilities of SNNs for audio classification tasks. Specifically, we want to develop models capable of classifying (and possibly detecting) sound events from audio data.

Furthermore, knowing that SNNs consume less power than traditional Artificial Neural Networks (ANNs), but have lower overall accuracy, we want to perform a performance comparison of SNNs with ANNs.

The fulfillment of these objectives would allow us to determine the potential of SNNs for audio classification tasks and to identify the advantages and disadvantages of SNNs compared to other neural networks. Moreover, it is a great way for us to learn more about SNNs and audio classification.

3 Organizing the project

3.1 Main tasks

During the last full time period, we worked on:

- **A preprocessing pipeline** that allows us to download, format and segment the audio part of the Youtube videos composing the Google Audioset audio files into images. In order to be efficient, the pipeline needed to be parallelized.

After downloading these, it became also necessary to perform some verification on the data, which includes checking the audio file properties (sample rate, number of channels, etc.) and the labels related to the audio files.

- **Finding some correct data augmentation techniques** that can be used to improve the performance of the SNNs.
- **Finding a way to encode the audio data into spikes** that can be used as input for the SNNs.

- **Implementing the SNNs** that will be used for the audio classification task.

Training the SNNs on the audio data.

- **Implementing the ANNs** that we would compare to the SNNs.
- **Comparing the performance of the SNNs and the ANNs** on the audio classification task.

3.2 Planning and team organization

3.2.1 Previous work - full time period

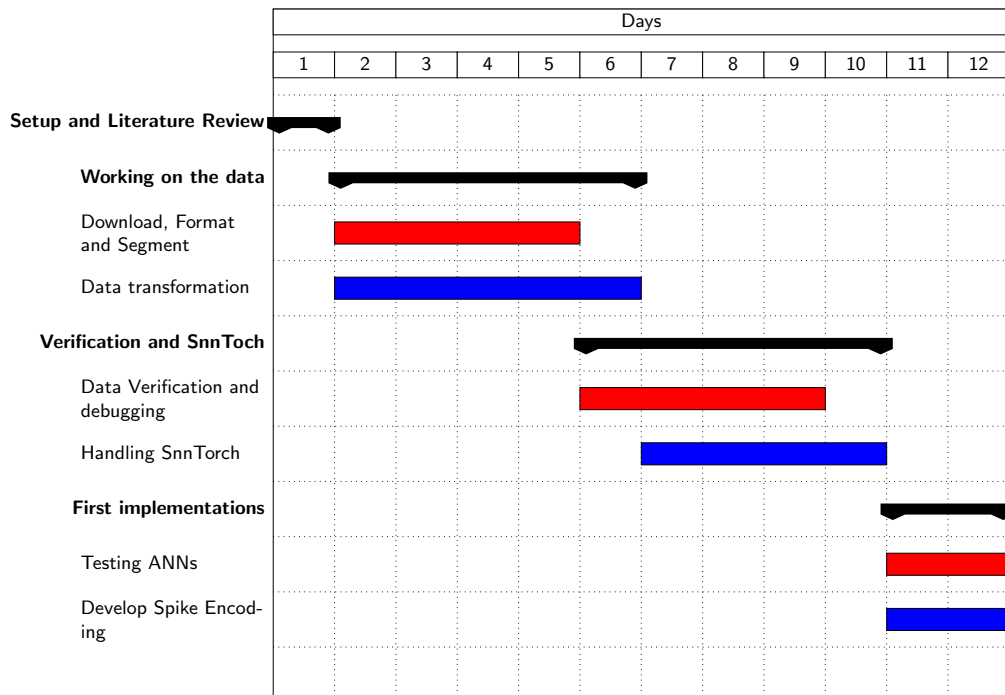


Figure 3: Planning of the full time period

- Téo : blue
- Loïc : red

So far, we have been able to encode and decode the audio data into spikes.

Since the first tasks we had to perform would not be connected until they were finished, we decided to work on them in parallel. This allowed us to be more efficient and, most importantly, to save time.

3.3 Changes in the organization

In the initial planning, there was no task related to the verification of the data nor to the debugging of part of the code. Each subpart of the preprocessing pipeline part takes a lot of time to be implemented and tested. We had to spend some time debugging the code and verifying the data.

4 Technical environment

4.1 Computational tools

We worked on our personal computers and we used the following computational tools:

Softwares	Visual Studio Code Jupyter Notebook Git Github Google Colab Anaconda Overleaf
Programming languages	Python Latex
Libraries	Pytorch Librosa Numpy Pandas Matplotlib sox Youtube-dl
Frameworks	SnnTorch

Figure 4: Computational tools

([Our Github repository](#))

4.2 Documentation

4.3 Changing

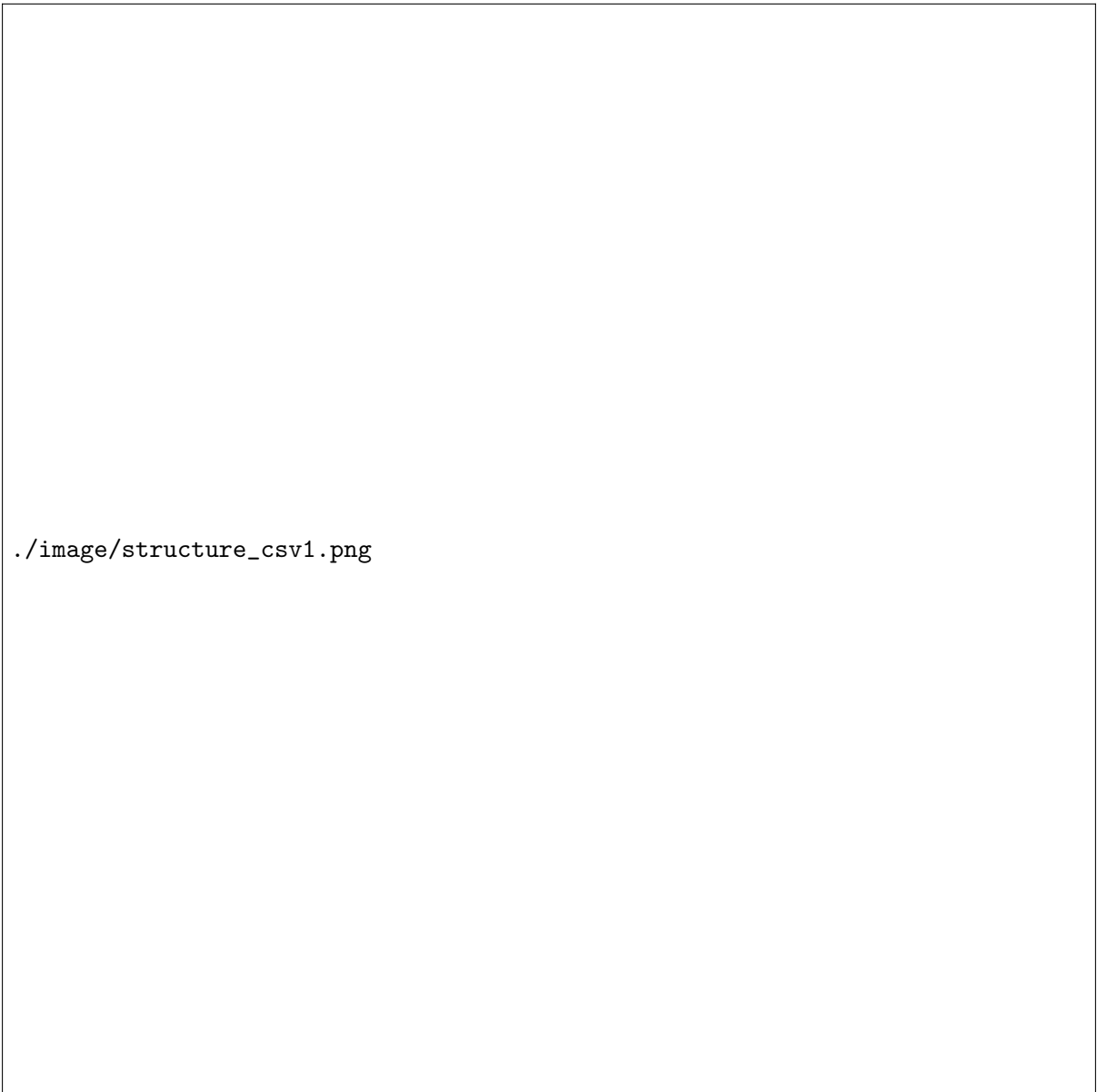
5 Work accomplished

5.1 Data

Notes: In the section below, we decided not to include some code snippets because they were not very relevant to the report.

5.1.1 The Audioset

In order to use the Google Audioset dataset, we needed to download the audio files and their associated labels. This first resulted in a file containing the links to the audio files and a file containing the label(s) of the audio files. Their structure is as follows:



`./image/structure_csv1.png`

Figure 5: Structure and small content of the csv file containing the links to the audio files

Filtering the data by labels

The first column contains the links to the YouTube video, then the start and end time of the audio segment in the video, and finally the labels of the audio segment (there can be multiple labels for one audio segment).

Since we want to train our first model on small scenarios with only 3 classes [Animals](#), [Sound of things](#), and [Music](#), we had to filter the data so that in all the rows that make up the csv file, there is at least one of the 3 classes we wanted to keep, but no repetition between the 3 classes.

Basically, we needed to iterate over the initial csv file to create 3 csv files, each containing only videos with one of the 3 classes we wanted to keep (see `reorder_to_categories.py`).

Also, there might be some repetitions between the labels associated with each audio segment, so we needed to remove those duplicates (see `reorder_bt看_categories.py`).

(see our [GitHub repository](#)).

Downloading the audio files

Once we had the csv files containing the links to the audio files, we needed to download them. To do this, we started using this [Github](#) repository as a base for our work. Since there were some problems using the `youtube-dl` library (that was changes to the `yt-dlp`), we had to modify the code to allow parallelization of the process (crucial for the efficiency of the pipeline if we don't want to wait for days to download the audio files).

We ended up with a code similar to the first one, but with some compatibility, performance and parallelization improvements (see `main_dfs.py`).

Checking the data

Once we had downloaded the audio files, we had to check that all the registered ones were not missing some information, corrupted or of duration less than 10 seconds (see `valid_with_labels`). This part ended-up being a source of problems as it consisted in a test and debug part.

5.2 Study of SNNs

5.2.1 Neurons model

1. Leaky Integrate-and-Fire (LIF) Neuron:

The LIF neuron model is a simple but commonly used model in spiking neural networks.

Equation:

$$\tau_m \frac{du}{dt} = -[u(t) - u_{\text{rest}}] + RI(t)$$

2. Hodgkin-Huxley (HH) Neuron:

The Hodgkin-Huxley model is a more complex and biophysically detailed model.

Equations:

$$\begin{aligned} C_m \frac{dV}{dt} &= -I_{\text{Na}} - I_{\text{K}} - I_{\text{L}} + I_{\text{ext}} \\ I_{\text{Na}} &= g_{\text{Na}} m^3 h (V - E_{\text{Na}}) \\ I_{\text{K}} &= g_{\text{K}} n^4 (V - E_{\text{K}}) \\ I_{\text{L}} &= g_{\text{L}} (V - E_{\text{L}}) \\ \frac{dm}{dt} &= \alpha_m (1 - m) - \beta_m m \\ \frac{dh}{dt} &= \alpha_h (1 - h) - \beta_h h \\ \frac{dn}{dt} &= \alpha_n (1 - n) - \beta_n n \end{aligned}$$

C_m : Membrane capacitance

V : Membrane potential

I_* : Sodium, Potassium, leakage, and external current

g_* : Maximum sodium, potassium, and leakage conductance

E_* : Nernst potential for sodium, potassium, and leakage (the equilibrium potential)

m, h, n : Gating variables that represent the fraction of channels in the open state for sodium, potassium, and leakage channels, respectively

$\alpha_m, \beta_m, \alpha_h, \beta_h, \alpha_n, \beta_n$: Voltage-dependent rate constants governing the kinetics of the opening and closing of ion channels

3. Izhikevich Neuron:

The Izhikevich neuron model is a two-dimensional model designed to capture a wide range of spiking behaviors.

Equations:

$$\begin{aligned} \frac{dv}{dt} &= 0.04v^2 + 5v + 140 - u + I_{\text{syn}} \\ \frac{du}{dt} &= a(bv - u) \end{aligned}$$

Parameters:

v : Membrane potential

u : Membrane recovery variable

a, ba, b : Parameters determining the neuron's behavior

I_{syn} : Synaptic current

Although it is not the one which represents reality the most, LIF model is the best compromise if we take into account computation complexity.

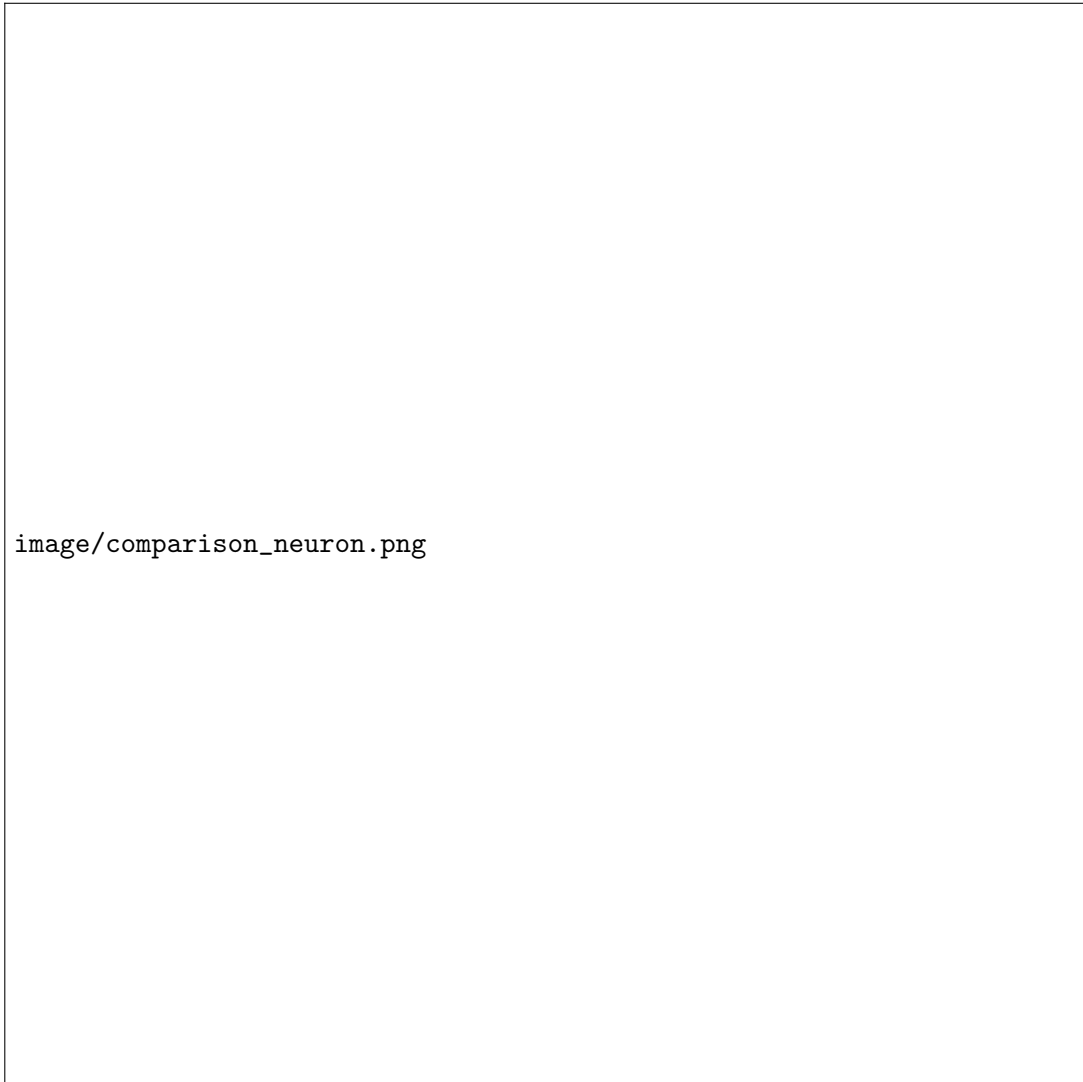


Figure 6: Comparison of different neuron model's complexity and biological similarity

5.2.2 Spike encoding

The encoding in SNNs refers to how information is represented and transmitted through the spikes of the SNN. In our study, we focus primarily on two of them:

Rate Encoding: In rate encoding, information is represented by the firing rate of neurons. The more spikes a neuron produces within a given time window, the higher the firing rate, and the stronger the encoded information.

Latency Encoding: Temporal encoding focuses on the precise timing of spikes. The timing information of individual spikes is crucial for representing the input data. The exact timing of spikes

can convey additional information, and the temporal patterns of spikes are used to encode features or events in the input signal.

5.3 Spectrograms, Mel spectrograms, MFCC

- Spectrograms

Spectrograms are essential graphic tools in audio analysis. They offer a visual representation of the frequency spectrum of a sound signal as a function of time, providing detailed information on the frequency composition and temporal dynamics of an audio signal. This report explores the use of spectrograms in various contexts and highlights their importance in the analysis and understanding of audio signals.

A spectrogram is generated by applying a short-time Fourier transform (STFT) to an audio signal. This technique divides the signal into small time windows, then applies a Fourier transform to each window to obtain the frequency distribution at that particular moment. The results of these transformations are then represented graphically, using colors to indicate the intensity of frequencies at different periods.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \quad (3)$$

Figure 7: Discrete Fourier Transform

Spectrograms enable in-depth analysis of the temporal characteristics of audio signals. Events such as transients, attacks and decays can be clearly identified, which is essential for understanding dynamic variations in music, speech and other forms of sound.

By examining the color and intensity of areas in a spectrogram, it's easy to identify the dominant frequencies present in an audio signal. This is particularly useful for detecting anomalies, characterizing musical instruments and separating sound sources.

Spectrograms are widely used in fields such as professional audio, music and linguistics. In particular, we are interested in Mel spectrograms, as well as MFCCs as input data for our neural networks.

- Mel Spectrograms

While spectrograms offer a detailed view of the frequency spectrum of an audio signal, a significant evolution in audio analysis occurred with the introduction of Mel Spectrograms.

These represent an adaptation of traditional spectrograms, using a frequency scale based on the Mel scale, which is more in line with human auditory perception. This transition to Mel Spectrograms has broadened the possibilities of analysis, offering a more faithful representation of the auditory characteristics perceived by the human ear. Let's take a closer look at this innovation and its impact on modern audio analysis.

While spectrograms offer a detailed view of the frequency spectrum of an audio signal, a significant evolution in audio analysis occurred with the introduction of Mel Spectrograms. These represent an adaptation of traditional spectrograms, using a frequency scale based on the Mel scale, which is more in line with human auditory perception.

- MFCC

MFCCs are derived directly from Mel Spectrograms and are calculated by applying a discrete cosine transform to the log-power of Mel Spectrograms. This approach captures information specific to human auditory characteristics while reducing data redundancy. MFCCs thus encapsulate frequency variations over time in a compact way, creating a set of cepstral coefficients that are widely used for automatic speech recognition and other audio signal processing tasks.

Two of the main advantages of MFCCs are compactness and information discrimination. MFCCs condense information while retaining the signal's distinctive characteristics. Compact representation

image/Spectrogram_piano.png

Figure 8: Spectrogram of a slowly ascending piano arpeggio, we can notice the new notes with higher frequencies appearing over time, where the image is brighter

facilitates the storage, transmission and processing of large amounts of data. By focusing on perceptual features rather than raw frequency, MFCCs are less sensitive to pitch variations, which improves the robustness of sound recognition. The calculation of MFCCs involves several steps, including Mel scale transformation, calculation of the logarithm of spectral powers, discrete cosine transformation and selection of relevant coefficients.

To summarize, in order to obtain an MFCC from an audio signal:

- Take the Fourier transform (STFT) of the signal.
- Map the powers of the resulting spectrum onto the mel scale, multiplying it by overlapping window functions (triangular or cosinusoidal).
- Take the logarithm of the amplitudes at each mel frequency.
- Take the discrete cosine transform (DCT) of the list of logarithmic powers of the mel frequencies, as if it were a signal. The MFCCs are the amplitudes of the resulting spectrum.

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[\frac{\pi}{N} \left(n + \frac{1}{2} \right) k \right] \quad (4)$$



Figure 9: Mel Spectrogram, one of the main differences is that the bands corresponding to a note of the arpeggio are closer than in the spectrogram, which represents better the human auditory system

5.4 Signal reconstruction

To compare most used encoding methods, we used an SNN model to encode and then decode MFCCs extracted from an audio signal. The two most used encoding approaches were compared: frequency encoding and latency encoding.

RESULTS: Reconstruction quality

The results showed that latency encoding (when normalized and linearized and with a sufficient number of time steps) led to a better reconstruction quality of the audio signal than frequency encoding, which was much more subject to small errors that propagate during reconstruction, the errors being due to the probability of emitting a pulse for each time step, which does not perfectly represent the pixel intensity. This improvement can be explained by the ability of latency encoding to capture fine temporal variations in the signal. However, due to the loss of information during signal transformation in MFCC, and more specifically during DCT, no reconstruction reaches the quality of the original signal.

Moreover, with latency encoding, one pulse carries much more information, which can lead to less robustness in the face of noise, which can be very present in our dataset. We will therefore try

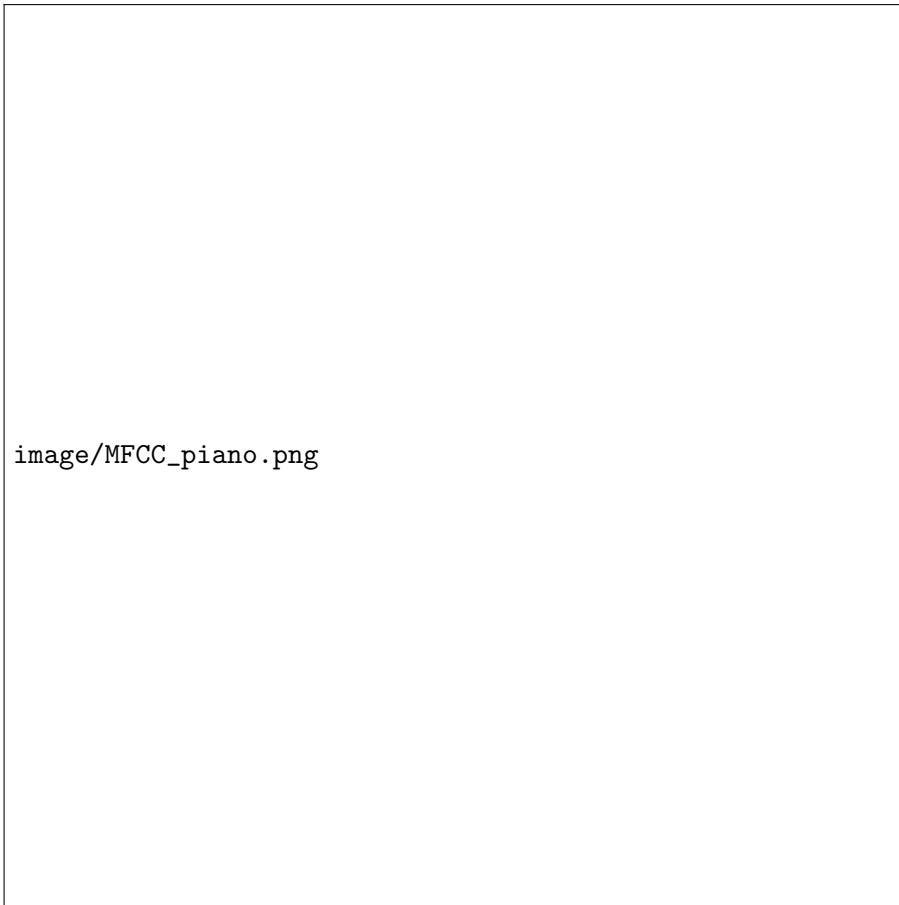


Figure 10: MFCC

to compare the two approaches in our SNN.

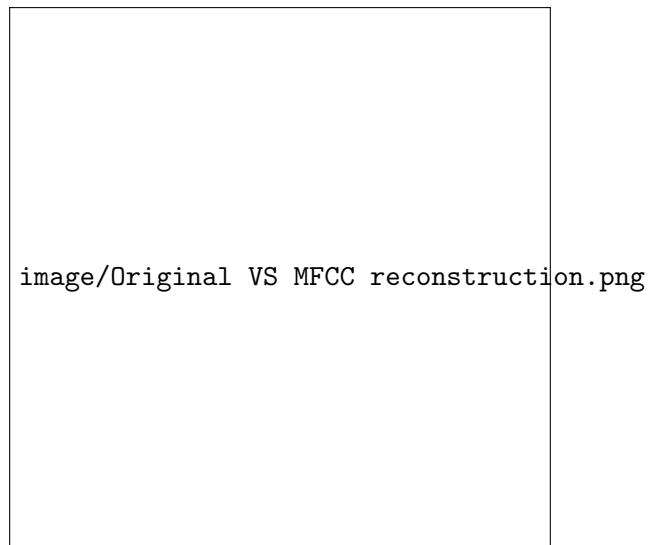


Figure 11: Original VS MFCC reconstruction

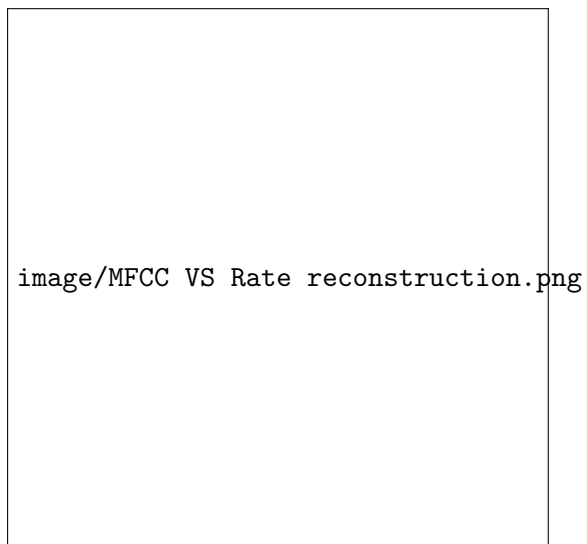


Figure 12: MFCC VS Rate reconstruction

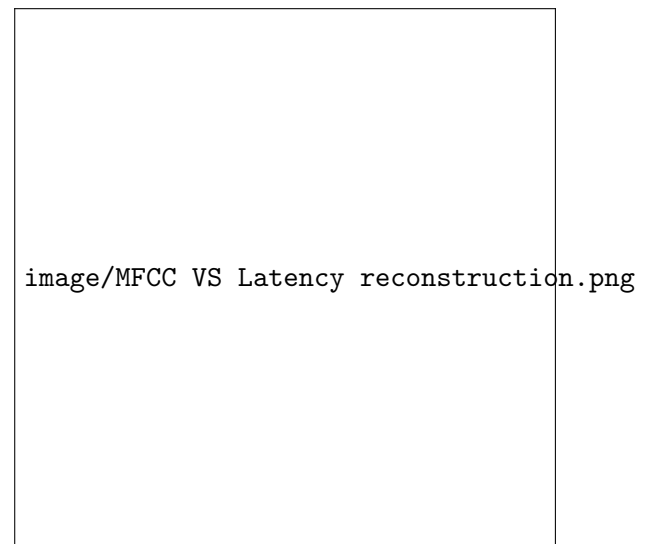


Figure 13: MFCC VS Latency reconstruction

6 Difficultés rencontrées

For the reconstruction of the audio, the data needed to be properly normalized to create the MFCC and then denormalized to extract good audio. We used the MinMaxScaler of sklearn fitted on the data to achieve our results.

7 Conclusion et perspectives

- Mettre en valeur votre travail et les domaines où vous avez progressé.
- Souligner en quoi la formation suivie à Polytech Nice Sophia (ou ailleurs) vous a aidé.
- Que comptez-vous faire dans les prochaines semaines ?

7.1 Audio samples



Figure 14: Original Audio

Figure 15: MFCC Reconstruction



Figure 16: Latency Reconstruction

Figure 17: Rate Reconstruction

Temporary page!

\LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because \LaTeX now knows how many pages to expect for this document.