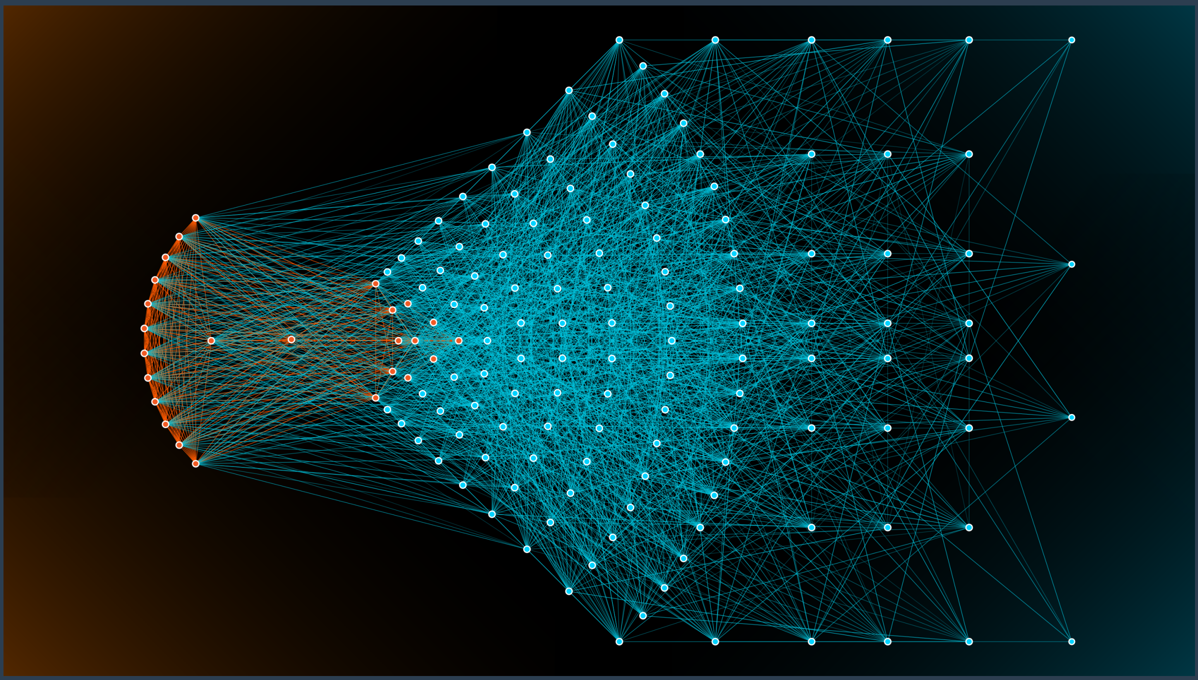
Description of Work

Sound Detection and Classification
using Spiking Neural Networks



COURREGE Téo
GANDEEL Lo'aï

Date: November 22, 2023

Contents

1 Introduction

In the constantly evolving landscape of neural network studies, the project outlined in this Description of Work (DoW) proposes to explore the field of spiking neural networks (SNNs). This effort will at term combine both a study and a prospective implementation, with a focus on the emerging field of neuromorphic computing.

The overall goal of this project is to perform a thorough investigation and subsequent implementation of spiking neural networks. SNNs, which are inspired by the neural signaling patterns of the human brain, show promising potential in various applications, especially in real-time processing and pattern recognition.

Before getting into the specifics of our project, let us first give a brief overview of what spiking neural networks are and how they work:

A Spiking Neural Network (or SNN) is a type of artificial neural network that mimics the functionality of biological neural networks. Unlike traditional neural networks (ANNs - Artificial Neural Networks), SNNs incorporate the concept of time into their operating model. The neurons in SNNs generate spikes of activity and communicate through these spikes (like brain neurons stimulating each other with electrical impulses), allowing them to process information in a more complex and potentially more efficient manner.

This type of neural network is theoretically less energy consuming than some ANNs and potentially more efficient in terms of processing power (in practice, it will be necessary to create an adapted hardware architecture to take advantage of this potential). Therefore, it seems that the use of this type of neural network could be a viable solution to a classification processing problem (especially in real time ones).

Our project focuses on a specific problem within the broader domain of audio classification.

Adopting an exploratory approach, we will begin this DoW with a more detailed presentation of the dataset used and its construction. After a brief introduction of our topic, we will review the state of the art in the field of SNNs and audio classification. We will then present the technical goals of our project and a simple theoretical study of SNNs. Finally, we will present the resources we have at our disposal to carry out this project, as well as a description of the work to be done. Finally, we will propose a schedule for the project.

2 Data used

2.1 Choosing the type of training data

When we first read the survey about $SNNs$[1], it was understood that training this type of model would be more difficult to train than some ANNs due to the temporal dynamics involved in SNNs. This made us want to work on a type of data that would be less computationally or ressource intensive than images or videos, which is why we decided to work on audio data. This type of data also offers the advantage of being more adaptable to pre-recorded and real-time processing.

In our case, we decided to apply our analysis on the Google AudioSet database, which contains a large number of audio samples classified into different categories (structure defined later).

2.2 Feasibility (audio $>$ image - simpler)

- Ressource consumption (RAM, GPU, CPU, time)

  Size of this youtube video in 360p: 8.68 MB

  Size of the same video as an mp3

2.3 Choose database

2.3.1 Copyright

2.3.2 Time for pre-processing, cleaning ...

2.3.3 Uncertainties about data quality, ...

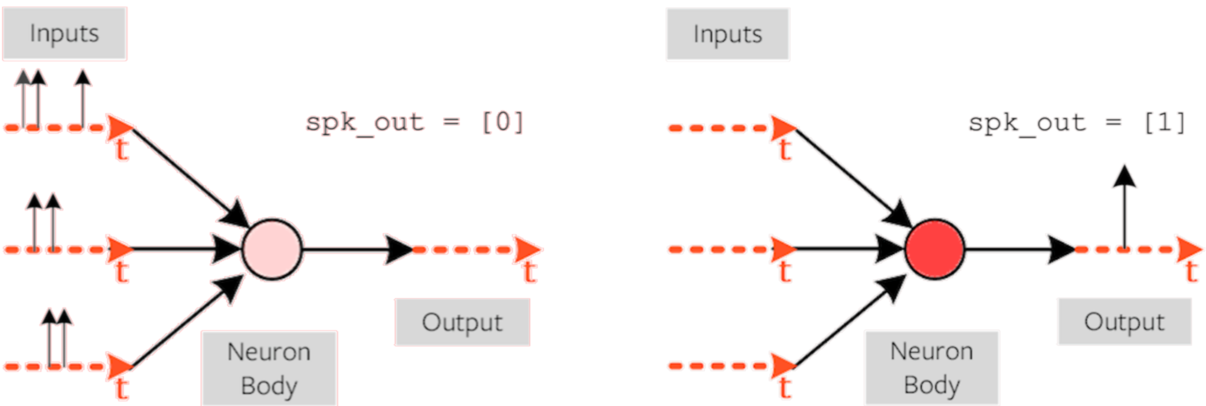
2.4 Open to other databases

3 Chosen theme

3.1 The SNN

3.1.1 Definition

Spiking Neural Network (SNN) is a variant of artificial neural network that try to mimic more closely the biological neural networks. More precisely, it tries to copy the functioning of neurons and synapses. As a result, rather than working with continually changing time values as ANNs do, SNNs work with discrete events that happen at defined times. SNNs take a set of spikes as input and produces a set of spikes as output



3.1.2 Advantages / Disadvantages

SNNs, when well constructed, offer the following benefits

- Energy efficiency: SNNs are highly energy efficient because they only consume energy when a spike is generated, unlike "classical" neural networks which consume energy continuously. This makes them ideal for low-power embedded devices.

- Event-driven processing: While ANNs typically operate on a fixed time interval, SNNs only process information when there is an input spike. This makes them highly efficient at processing information in an event-driven environment, such as processing visual or audio data.

- Temporal coding: SNNs are able to process information based on the timing of spikes, allowing them to encode temporal information and be temporally accurate. This is particularly useful for tasks that require the processing of sequential data, such as speech or gesture recognition.

- Robustness to noise: SNNs are inherently robust to noise and can effectively filter out irrelevant information. This makes them useful in environments where noise is a common problem, in our case when the audio is not well recorded.

- Neuroplasticity: SNNs are able to adapt to new inputs and change their behaviour over time, much like the brain. This allows them to learn and adapt to new tasks and environments, making them highly versatile.

Disadvantages:

- SNNs are difficult to train, mainly due to backpropagation issues
- There is currently no learning algorithm designed specifically for this task.
- Building a small SNN is impractical.

3.1.3 snnTorch

We will primarily work in Python and use the snnTorch package to work on SNNs. snnTorch is a Python package for performing gradient-based learning with spiking neural networks. snnTorch is built on top of Pytorch and takes advantage of its GPU-accelerated tensor computation. Pre-defined spiking neuron models are integrated into the PyTorch framework and can be treated as recurrent activation units.

3.2 Neural networks applied to audio classification

3.2.1 Definition

Audio classification is a machine learning and signal processing task that involves categorizing or labeling audio data into different predefined classes or categories. The goal is to automatically assign a label to an audio segment based on its content or characteristics. This can be useful in various applications such as speech recognition, music genre classification, environmental sound analysis, and more.

Applications:

Speech Recognition: Identifying spoken words or phrases in audio recordings. Music Genre Classification: Categorizing music into different genres, such as rock, pop, jazz, etc. Environmental Sound Analysis: Detecting and classifying sounds in the environment, such as sirens, footsteps, or car engines. Anomaly Detection: Identifying unusual or unexpected sounds in a given context.

3.2.2 How it works

The process of audio classification typically involves the following steps

Data collection: Collecting a dataset of audio samples, each labeled with its corresponding class or category.

Feature extraction: Transforming the raw audio data into a set of relevant features that can be used to represent the content of the audio. Common features include spectrogram representations, Mel Frequency Cepstral Coefficients (MFCCs), and other time or frequency domain features.

Model training: Using machine learning algorithms or deep learning architectures to train a model on the extracted features and their corresponding labels. Apart from neural networks, others algorithms can be used for audio classification such as support vector machines, decision trees, or random forests for example.

Validation and testing: Evaluate the performance of the trained model on a separate set of data that it has not seen before. This helps evaluate the model's ability to generalize to new, unseen examples.

Deployment: Integrate the trained model into applications or systems that require real-time or batch audio classification.

3.2.3 Advantages / Disadvantages

PAS NECESSAIRE,

3.3 Relevance of this choice

3.3.1 Defining the project's pros and cons

Pros : - SNN's can be effective to detect and recognize different types of audio (especially on embedded devices).

Cons: - SNNs are new to us, we need to understand the package snnTorch. - SNNs might not be the best method to process audio data.

3.3.2 Cases of use to define the relevance of the project

Our final goal is primarily to compare the efficiency and energy use of SNNs compared to others networks (CNNs, RNNs, LSTMs...) in audio classification, and to evaluate if SNNs are a good solution in our case.

4 State of the art

4.1 Finding what already exists

4.1.1 In a simple neural network

Trained on the same data set or not

4.1.2 In SNN

Trained on the same data set or not

4.1.3 Existing libraries, existing models

4.2 Results obtained as a basis for comparison

5 Technical objectives of the project

5.1 Technical objectives achievable with pre-existing ANNs

5.2 Technical objectives with data

How AI and ML Understand and Classify Audio

In its raw form, a sound is an analog signal — to process it, we have to convert it to digital. Analog signals are continuous waves. An analog wave has a discrete value at any given point in time. It's impossible to convert every value to a digital representation because there are infinite points in a continuous wave. You must convert the audio signal to digital through sampling and quantization.

Sampling means recoding these points at specific time intervals or frequencies. This interval is the sample rate. The higher the sample rate, the less information lost and the fewer errors. However, if the sample rate is too high, it increases the audio file size without significantly improving the audio.

Quantization means rounding the amplitude sampled at each interval to the nearest bit. The greater the bit depth, the lower the quantization. The larger the bit depth, the more memory it takes, so what bit depth to use depends on the amount of available RAM. If you want to learn more, this is a good explanation of analog to digital conversion.

After conversion, the next step is to extract useful features from the audio. You can extract many features, so you must focus on the specific problem you want to solve. Here are some critical features used for audio classification with ML:

Time domain features are extracted directly from the raw audio (waveform). Examples are the amplitude envelope, and the root mean square energy. Time domain features are not enough to represent the sound because they do not include frequency information.

Frequency domain features are also called the spectrum. Examples include band energy ratio, spectral flux, and spectral bandwidth. These features lack time representation. You extract them from a time domain representation through Fourier transform. Fourier transform converts a waveform from a function of time to a function of frequency. Learn more about Fourier Transforms here.

Time and frequency domain features extract a spectrogram from the waveform using a short-time Fourier transform, like Spectrogram or Mel Spectrogram. The spectrogram shows both the frequency and time domain. Spectrograms represent frequency linearly, but humans perceive sounds logarithmically. This means that humans can tell the difference between lower frequencies like 500-100 Hertz (Hz) but find it harder to differentiate between sounds at a higher frequency, from 10,000 Hz to 15,000 Hz. Learn more about how humans respond to various sound ranges here. This difficulty is why the Mel-scale was introduced. A Mel Spectrogram is a spectrogram on the Mel-scale, which measures how different pitches sound to the listener. It maps pitches that, to listeners, sound equidistant.

5.3 Technical objectives with SNNs

5.4 Feasibility

5.4.1 Describing uncertainties

Uncertainty are due to the dynamic nature of audio signals, varying acoustic environments, and the complexity of distinguishing between diverse sound patterns. Additionally, challenges in precisely modeling the temporal aspects of auditory processing and the understanding of spiking neuron dynamics contribute to many uncertainties in the feasibility of the project. Moreover the training of Spiking Neural Networks in order to attain a good accuracy is difficult due to the inherent non-linearity and sparsity of spikes.

5.4.2 Looking for a working model

Many model for sound classification already exist. In particular, for AudioSet, there are already multiple pre-trained deep neural network, for example YamNet. However, it is more difficult to find open source SNNs, as they are less widespread than other types of neural networks.

5.4.3 Estimate computation time

6 Theoretical study

6.1 Theoretical study on SNN

Key concepts

What distinguishes a traditional ANN from a SNN is the information propagation approach.

The general idea is as follows;

Each neuron has a value that corresponds to the electrical potential of biological neurons at any given time. A neuron's value can change according to its mathematical model; for example, if a neuron receives a spike from an upstream neuron, its value can increase or decrease. If a neuron's value exceeds a certain threshold, the neuron will send a single impulse to each downstream neuron connected to the first, and the neuron's value will immediately drop below its average. As a result, the neuron goes through a refractory period similar to that of a biological neuron. The neuron's value will gradually return to its mean value over time.

Spike Based Neural Codes

Artificial spiking neural networks are designed to do neural computation. This necessitates that neural spiking is given meaning: the variables important to the computation must be defined in terms of the spikes with which spiking neurons communicate. A variety of neuronal information encodings have been proposed based on biological knowledge:

Binary Coding:

Binary coding is an all-or-nothing encoding in which a neuron is either active or inactive within a specific time interval, firing one or more spikes throughout that time frame. The finding that physiological neurons tend to activate when they receive input (a sensory stimulus such as light or external electrical inputs) encouraged this encoding.

Individual neurons can benefit from this binary abstraction because they are portrayed as binary units that can only accept two on/off values. It can also be applied to the interpretation of spike trains from current spiking neural networks, where a binary interpretation of the output spike trains is employed in spike train classification.

Rate Coding:

Only the rate of spikes in an interval is employed as a metric for the information communicated in rate coding, which is an abstraction from the timed nature of spikes. The fact that physiological neurons fire more frequently for stronger (sensory or artificial) stimuli motivates rate encoding.

It can be used at the single-neuron level or in the interpretation of spike trains once more. In the first scenario, neurons are directly described as rate neurons, which convert real-valued input numbers "rates" into an output "rate" at each time step. In technical contexts and cognitive research, rate coding has been the concept behind conventional artificial "sigmoidal" neurons.

Fully Temporal Codes

The encoding of a fully temporal code is dependent on the precise timing of all spikes. Evidence from neuroscience suggests that spike-timing can be incredibly precise and repeatable. Timings are related to a certain (internal or external) event in a fully temporal code (such as the onset of a stimulus or spike of a reference neuron).

Latency Coding

The timing of spikes is used in latency coding, but not the number of spikes. The latency between a specific (internal or external) event and the first spike is used to encode information. This is based on the finding that significant sensory events cause upstream neurons to spike earlier.

This encoding has been employed in both unsupervised and supervised learning approaches, such as SpikeProp and the Chronotron, among others. Information about a stimulus is encoded in the order in which neurons within a group generate their first spikes, which is closely connected to rank-order coding.

There are multiple ways to train a SNN:

• Shadow training: A non-spiking ANN is trained and converted into an SNN by interpreting the activations as a firing rate or spike time

• Backpropagation using spikes: The SNN is natively trained using error backpropagation, typically through time as is done with sequential models

• Local learning rules: Weight updates are a function of signals that are spatially and temporally local to the weight, rather than from a global signal as in error backpropagation

7 Matching with available resources

7.1 Work time

data collection/extraction:

- from csv/Youtube to mp3 : 1-2 days

data preprocessing:

-data cleaning/enhancement/augmentation: 3-4

models implementation:

-understanding snnTorch basic: 1-2 days
-finding pretrained/creating and tuning models: 4 days
training/tuning:
- training: weeks
- extracting information: weeks
working on the DOWs and presentation:
- DOWs writing: continous work, 1-2 hours each day
- slides for oral presentation: 2-3 days

7.2 Computer resources

Hardware: 2 laptops
Software: Google Colab (primarily)

7.3 Human resources

We will work in parallel most of the time, each one focusing on a different aspect to gain time when it is possible.

8 Appendix

8.1 Data

Data Set used for the project

- Google AudioSet : https://research.google.com/audioset/

8.2 Code

- Sparsity through Spiking Convolutional Neural Network (SCNN) for Audio Classification at the Edge : https://github.com/CongSheng/SpikingConvNN-AudioClassification/tree/main
- Multilayer Spiking Neural Network for audio samples classification using SpiNNaker : https://github.com/jpdominguez/Multilayer-SNN-for-audio-samples-classification-using-SpiNNak

8.3 Links

References

[1] J. K. Eshraghian, M. Ward, E. Neftci, X. Wang, G. Lenz, G. Dwivedi, M. Bennamoun, D. S. Jeong, and W. D. Lu, *Training Spiking Neural Networks Using Lessons From Deep Learning*. https://arxiv.org/abs/2109.12894, 2021.