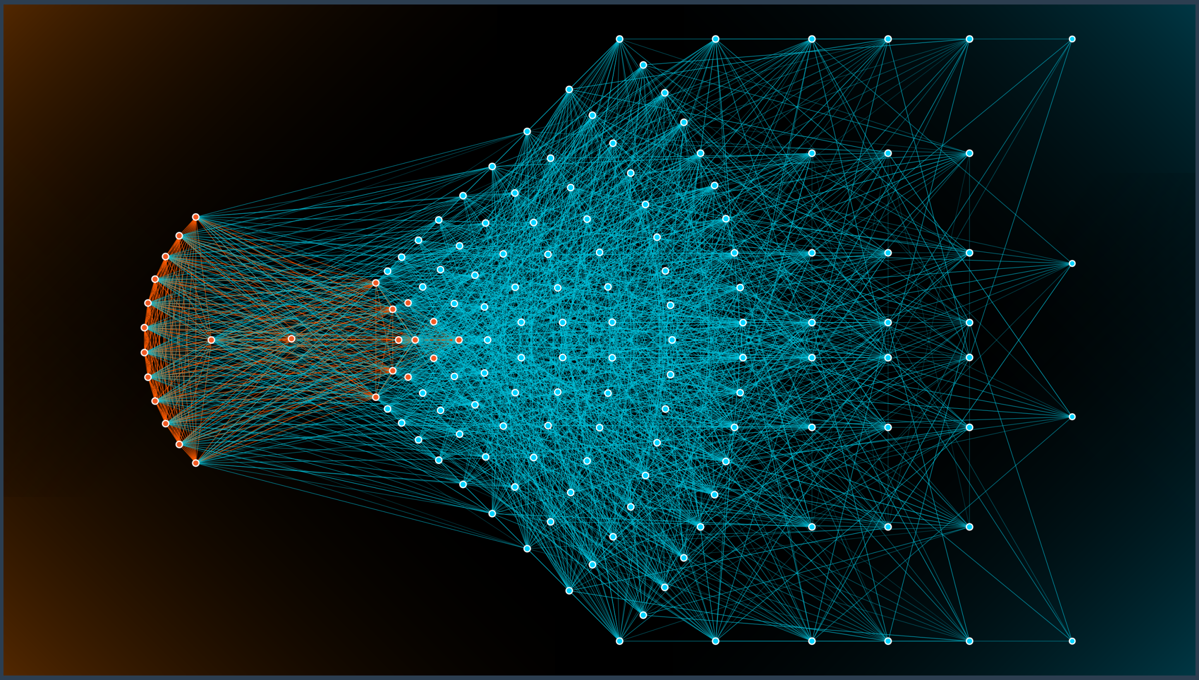
Description of Work

Sound Detection and Classification
using Spiking Neural Networks



COURREGE Téo
GANDEEL Lo'aï

Date: February 26, 2024

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.

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

2 Chosen theme

2.1 The SNN

2.1.1 Definition

Spiking Neural Network (SNN) is a variant of artificial neural networks that tries to mimic biological neural networks more accurately. More specifically, it tries to copy the way neurons and synapses work. As a result, instead of working with continuously changing time values as ANNs do, SNNs work with discrete events that occur at defined times. SNNs 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 example, in a leaky integrate-and-fire (LIF) neuron model, we have:

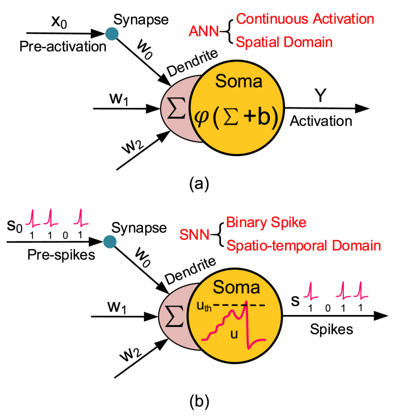


Figure 1: Difference ANN and SNN model[1]

$$\begin{cases} \tau \frac{d\,u(t)}{dt} = -[u(t)-u_{r1}] + \sum_j w_j \sum_{t_j^k \in S_i^{T_w}} K(t-t_j^k) \\ \begin{cases} s(t)=1 & u(t)u_{r_2} \text{ if } u(t)\geq u_{th} \\ s(t)=0 & \text{otherwise} \end{cases} \end{cases} \tag{1}$$

where t denotes the time step, τ is a time constant, and u and s are membrane potential and output spike, respectively. u_{r_1} and u_{r_2} are the resting potential and reset potential, respectively. w_j is the synaptic weight from the j^{th} input neuron, t_j^k is the time when the k^{th} spike of the j^{th} input neuron fires within the integration time window of T_w (a spike sequence of $S_j^{T_w}$ in total), and $K(.)$ is a kernel function describing the time decay effect. u_{th} is the firing threshold that determines whether to fire a spike or not. [1]

This model has the advantage of being easier to implement than other kinds of models like the Hodgkin-Huxley model or the Izhikevich model.

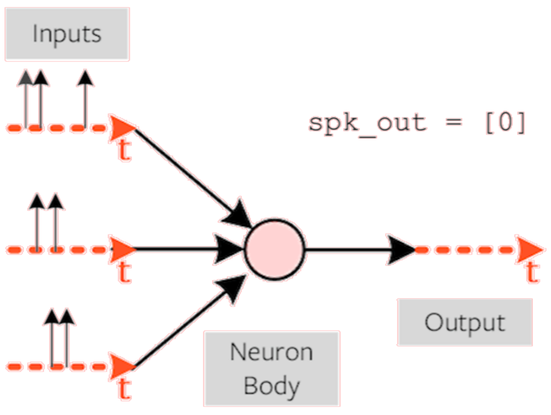


Figure 2: SNN input

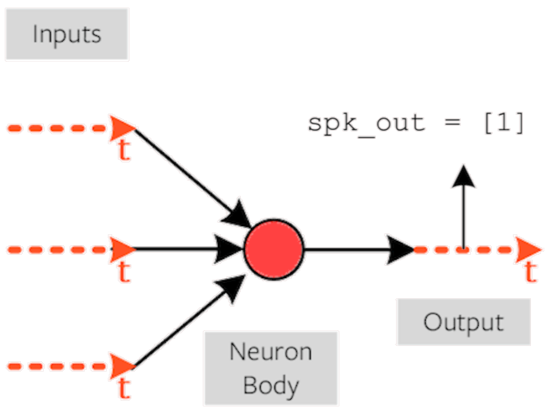


Figure 3: SNN output

Note:

We also found this kind of equation for the membrane potential for a LIF neuron model:

$$\tau \frac{d\,U_j(t)}{dt} = -U_j(t) + \sum_{i=1}^{N} w_{ij}.n_i(t)$$

where, τ is the leakage time constant, and $n_i(t)$ is the input value of ith neuron, and w_{ji} is a synaptic weight (conductance) between neuron i and j. Membrane potential $U(t)$ increases whenever post-synaptic spike is generated by input spikes and it will decay spontaneously with time constant, τ. When the potential crosses over the pre-defined threshold level, it fires a post-synaptic spike and $U(t)$ instantaneously relaxes to the resting state and maintains the level for a refractory time, tref. without responding to any received signals.[4]

Here, $V(t)$ is the membrane potential, W_i is the synaptic weight, and $\delta(t - t_i)$ is the Dirac delta function representing the occurrence of a spike at time t_i.

The output spikes Y can be obtained based on a threshold θ:

$$Y(t) = \begin{cases} 1 & \text{if } V(t) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

2.1.2 Advantages / Disadvantages

SNNs, when well designed, should provide 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 at 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, it may be a case of improperly recorded audio.
- Neuroplasticity: SNNs are able to adapt to new inputs and change their behavior 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, which are caused by the discrete, temporal nature of spikes. Backpropagation in time (BPTT) is the typical method used, but it suffers from issues such as the temporal credit assignment problem (the determination of which past events or actions contributed most significantly to the current outcome or state)

  Moreover, the memory and the compute cost can be calculated by:

$$\begin{cases} \text{ANN:} & M = M_w + M_a \\ \text{SNN:} & M = M_w + M_p + M_s \end{cases} \tag{2}$$

Where M_w, M_a, M_p and M_s denote the memory cost for weights, activations, membrane potentials, and spikes, respectively. Compared to the static values of M_w, M_a, and M_p that are determined by the network structure, M_s is dynamically determined by the maximum number of spike events at a certain time step.[1]

$$\begin{cases} \text{ANN:} \quad C = C_{mult} + C_{add} \\ \text{SNN:} \quad C = C_{add} \end{cases} \tag{3}$$

where C_{mult} and C_{add} are the compute cost for multiplications and additions, respectively (note that elements such as the activation function in ANNs and the membrane potential update and firing activity in SNNs are ignored) [1]

- There is currently no learning algorithm designed specifically for this task, SNNs lacks the "maturity" of more classic neural networks.
- Building a small SNN is impractical. Indeed, we need a sufficient number of neurons to retrieve useful informations.
- We will try to highlight these specificities during our project

2.1.3 snnTorch

We will work primarily in Python and use the snnTorch library 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. Predefined spiking neuron models are integrated into the PyTorch framework and can be treated as recurrent activation units.

2.2 Neural networks applied to audio classification

2.2.1 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 several applications:

- Speech recognition: Identifying spoken words or phrases in audio recordings.
- Music genre classification: Classify music into different genres, such as rock, pop, jazz, etc.
- Environmental Sound Analysis: Detect and classify environmental sounds such as sirens, footsteps, or car engines.
- Anomaly Detection: Identify unusual or unexpected sounds in a given context.

2.2.2 How it works

The process of audio classification is relatively similar to image classification and typically involves the following steps

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

2. Feature extraction: Transforming the raw sound 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 Coefficient, and other time or frequency domain features.
3. Model training: Use machine learning algorithms or deep learning architectures to train a model on the extracted features and their corresponding labels.
4. 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.

2.3 Relevance of this choice

2.3.1 Defining the project's pros and cons

Pros :

- SNN's can be effective for detecting and recognizing different types of audio (especially on embedded devices).
- The comparison between SNN and other types of neural networks will be particularly important to us, and we will try to show what can be the strengths and drawbacks of SNNs compared to other networks.

Cons:

- SNNs can be difficult to understand sometimes, we will need to study the snnTorch library.
- Training SNNs is more difficult, and we need to find the most efficient encodings/decodings, training method, and parameters.

2.3.2 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.

3 Data used

3.1 Choosing the type of training data

When we first read the survey about $SNNs$[2], it was understood that training this type of model would be more difficult 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 resource intensive than videos (so that we could store and train our models on many of them). This type of data also has the advantage of being more adaptable to pre-recorded and real-time processing.

3.2 Feasibility

In our case, we decided to apply our analysis to the Google AudioSet database, which contains a large number of audio samples classified into different categories. The structure of this dataset is as follows: each audio sample is associated with a label, and each label is associated with a set of audio samples. In practice, this dataset is composed of '.csv' files that contain one or more labels, a timecode (10 seconds of duration), and a part of an url that would help us find an associated Youtube video. This gave us 2 problems: a copyright problem and the fact of extracting the data and formatting it in a way that would be usable for our project.

3.2.1 Copyright

According to the AudioSet website (download page):

> The dataset is made available by Google Inc. under a Creative Commons Attribution 4.0 International (CC BY 4.0) license, while the ontology is available under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license.

3.2.2 Data collection/extraction

Fortunately for us, we found some GitHub repositories whose code aims to download, format, and crop the various sound files so that it "unifies the formats of the datasets so that it is possible to analyze them in the same pipelines". The given script also included a parallelization of the calculation.

After some slight modifications we were able to check the impact on memory resources, for example for this youtube video we get

- Resource usage (RAM, GPU, CPU, time)

  Size of 144p: 2-4 MB (depending on format)

  Size of the same video as an audio file (wav in best quality): 1-3 MB (depending on the format)

Without parallelizing the code we used, we were able to download, format, and trim 6 audio files in about 60 seconds (and our first attempts at parallelizing seemed promising, though there are still bugs to work out). At this pessimistic rate, we think we should be able to download, format and trim about 2000 10-second audio files in about 5-6 hours. By limiting the number of labels to study to about ten, we think we might get some results (to be tested with a pre-trained Pytorch model).

3.2.3 Uncertainties about data quality

While lookin up to the data, we found that a lot of these sounds are taken out of context and other sounds are mixed with them (for example, this sample is labeled with as a music sound but a person is talking at the end of the sample). Also, each sample can have one or multiple labels, which can be related to multi-clustering problems. Another problem we face is the fact that the data is always balanced (there exists a balanced version of the dataset) and that there are some Weak and Strong Label annotations.

Strong labels are precise and indicate the exact start and end times of the event in the audio file whereas the Weak labels, only indicate the presence of a sound event somewhere in the audio file, without specifying the exact time.

3.3 Open to other databases

In the case that we use for this classification purpose is not sufficient (for example when collecting all the data related to a unique label), We might try to complete it with other open-source label databases such as Freesound or Kaggle.

4 State of the art

While searching for existing work on the topic, we found that there are already some studies on the subject of audio classification using SNNs. We will present a couple of them here.

4.1 ANNs Used for Audio Classification

First of all, we made some researches to find some existing ANNs made for audio classification for comparison with the model we plan to build. We found the following models:

- A VGG model already trained of the Google AudioSet dataset (GitHub).
- Rearanged Resnet, inception, densenet pretrained models (GitHub and research paper)
- An audio classifier that uses LSTM (made in class of Deep Learning): (Google Colab)
- others: 1 2 3 4

4.2 SNNs Used for Audio Classification

Secondly, we made some researches to find some existing SNNs made for audio classification, again, for comparison with the model we plan to build. We found the following models:

- A spiking convolutional neural network (SCNN) for audio classification (GitHub).
- A multi-layer SNN for audio classification using SpiNNaker (GitHub).
- Shadow training (”conversion” of an ANN to an SNN) (GitHub and research paper).
- Spiking used for image classification (GitHub and doc).
- SNN simulators : BindsNET and NEST

The results we may get from the aforementioned networks may serve as a basis of comparison for some of them.

5 Technical objectives of the project

5.1 Technical objectives achievable with pre-existing ANNs

We should find pre-trained ANNs that can achieve good accuracy with AudioSet.

5.2 Technical objectives with data

In its raw form, sound is an analog signal that must be converted to digital. Analog signals are continuous waves. An analog wave has a discrete value at each given point in time. It’s impossible to convert every value to a digital representation because there are infinite points in a continuous wave. We will have to convert the audio signal to digital through sampling and quantization. In our case, since we are extracting the audio from YouTube, the conversion to a digital signal step has already been done.

After the conversion, the next step will be to extract useful features from the audio. Some critical features used for audio classification with ML are

- Time domain features are extracted directly from the raw audio (waveform). Examples include amplitude envelope and root mean square energy. Time domain features are not sufficient to represent the sound because they do not contain 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. They are extracted from a time domain representation using the Fourier transform. The Fourier transform converts a waveform from a function of time to a function of frequency.

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

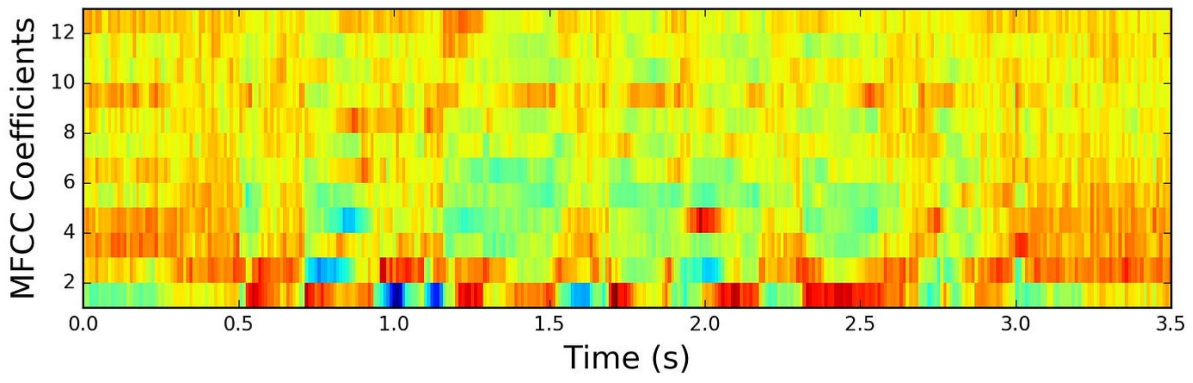


Figure 4: Mel Spectrogram

5.3 Technical objectives with SNNs

We need to either find a pre-trainedmodel our create it ourselves. We found a framework for the creation a SNN for robust sound classification.[5]

5.4 Feasibility

5.4.1 Describing Uncertainties

Uncertainties are mostly due to the dynamic nature of audio signals, varying acoustic environments, and the complexity of differentiating between various sound patterns. Additionally, challenges in precisely modeling the temporal aspects of auditory processing and understanding the dynamics of spiking neurons contribute to many uncertainties in the feasibility of the project. Furthermore, training Spiking Neural Networks to achieve good accuracy is difficult due to the inherent non-linearity and sparsity of spikes.

Regarding data uncertainties, dealing with Youtube means that some audio may not be in the right format (a 5s video from which we extract a 10s audio for example), and some may not even be available (users delete their video), which could lead to unbalanced data.

5.4.2 Finding a Working Model

Many models for sound classification already exist. In particular, there are several pre-trained deep neural networks for AudioSet, such as YamNet. However, open source SNNs are harder to find, as they are less widely used than other types of neural networks.

5.4.3 Computation Time Estimation

Data import and training are the most time-consuming tasks. We don't know yet how long a training will take. For importing audio, we estimate a little over 10 seconds on average for importing a single audio sample without parallelization, we expect to import between 1,000 and 10,000 audio samples.

6 Theoretical study

6.1 Theoretical study of SNN

There are multiple archetypes of spiking, each model as its specificity, and while it seems interestig to confine ourselves to the LIF (Leaky Fire and Integrate), which seems the more interesting, it may be relevant to analyze some types of models to evaluate their efficiency. [6]

We will need to choose a lot of parameters regarding our neural network, such as the encoding, the type of training.

Neural coding schemes are used to convert input pixels into spikes that are sent to the excitatory neurons. There are four main neural coding schemes: rate coding, time-to-first spike coding, phase coding, and burst coding.[3]

There are also different ways to train a SNN:

- Shadow training: A non-spiking ANN is trained and converted to a SNN by interpreting activations as firing rate or spike time.
- Backpropagation using spikes: The SNN is natively trained using error backpropagation, typically through time as in sequential models.
- Local learning rules: Weight updates are a function of signals that are spatially and temporally local to the weight, rather than a global signal as in error backpropagation.

7 Matching with available resources

7.1 Work time

Schedule		
Task	Duration	When
Data collection/extraction		
From csv/Youtube to mp3	2-3 days	22/11-24/11
Data preprocessing		
cleaning/enhancement/ augmentation	3-4 days	24/11-28/11
Models implementation		
understanding snnTorch basics	2 days	22/11-24/11
finding pretrained/creating and tuning models	4-5 days	26/11-01/12
Training/tuning		
training	multiple weeks	
extracting/synthetizing info	multiple weeks	
Working on the DOWs and presentation		
DOWs writing: continous work	2 hours x 2 per week	
slides for oral presentation	2-3 days	27/11-29/11

7.2 Computer resources

We will work with the following:

- Hardwares: 2 laptops
- Softwares: Google Colab (primarily), Kaggle

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 Existing models

- 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
- StereoSpike: Depth Learning With a Spiking Neural Network for image classification (CNN) : https://github.com/urancon/stereospike - https://ieeexplore.ieee.org/document/9969606
- Spiking Neural Networks Trained via Proxy (shadow training): https://ieeexplore.ieee.org/document/9810220
- VGG model trained on Google's AudioSet dataset : https://github.com/MaryamEbr/Audio-Classification-with-AudioSet-and-VGGish/tree/main - https://research.google/pubs/pub45611/
- Rethink CNNs for Audio Classification : https://github.com/kamalesh0406/Audio-Classification/tree/master - https://arxiv.org/abs/2007.11154
- An LSTM model for audio classification we made in class of Deep Learning : https://colab.research.google.com/drive/1kJ4hFfh3J13_nrtCMRrPoE_eIl5xUQWC?usp=sharing
- BindsNET, a Python package for training and simulating spiking neural networks : https://github.com/BindsNET/bindsnet
- NEST, a simulator for spiking neural network models that focuses on the dynamics, size and structure of neural systems rather than on the exact morphology of individual neurons : https://www.nest-simulator.org/

8.3 Code

- A collection of scripts to analyze, prepare for and download Google's Audioset : https://github.com/bakhtos/GoogleAudioSetScripts
- SNNtTorch, a Python package for performing gradient-based learning with spiking neural networks : https://snntorch.readthedocs.io/en/latest/

References

[1] L. Deng, Y. Wu, X. Hu, L. Liang, Y. Ding, G. Li, G. Zhao, P. Li, and Y. Xie, *Rethinking the performance comparison between SNNS and ANNS*. https://www.sciencedirect.com/science/article/pii/S0893608019302667?ref=pdf_download&fr=RR-2&rr=829f60cc982b9a09, 2020.

[2] 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.

[3] W. Guo, M. E. Fouda, A. M. Eltawil, and K. Nabil Salama, *Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems*. https://www.frontiersin.org/articles/10.3389/fnins.2021.638474/full, 2021.

[4] T. Kim, S. Hu, J. Kim, J. Y. Kwak, J. Park, S. Lee, I. Kim, Jong-Keuk, and P. Y. Jeong. https://www.frontiersin.org/articles/10.3389/fncom.2021.646125/full#B4, title = Spiking Neural Network (SNN) With Memristor Synapses Having Non-linear Weight Update, year = 2021.

[5] J. Wu, Y. Chua, M. Zhang, H. Li, and K. C. Tan, *A Spiking Neural Network Framework for Robust Sound Classification*. https://www.frontiersin.org/articles/10.3389/fnins.2018.00836/full, 2019.

[6] K. Yamazaki, V.-K. Vo-Ho, D. Bulsara, and N. Le, *Spiking Neural Networks and Their Applications: A Review*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9313413/, 2021.