POLITECNICO DI TORINO

Master's Degree in Physics of Complex Systems

Master's Degree Thesis

Impact of Encoding Techniques on the Classification of Raw Time-Variant Signals with Spiking Neural Networks

Supervisors
Gianvito Urgese
Evelina Forno
Vittorio Fra

Candidate
Riccardo Pignari

April 2022

Abstract

Spiking Neural Networks (SNNs) represent the third generation of neural networks, whose technology is inspired by the human brain. Preliminary analysis carried out in state of the art neuromorphic works, highlight a number of advantages in the use of these technologies compared to the modern techniques used for data analytics. Since the information is processed in the form of an electrical impulse also referred to as spike, and by exploiting a sparsely connected neural structure and asynchronous processing of the neurons, these networks promise to be the solution to the enormous energy consumption of the ANNs, while maintaining competitive accuracy. Therefore, proceeding with a complete analysis of a machine learning pipeline on the analysis of sensor samples, I analyzed the preprocessing, coding, modeling and classification steps in a neuromorphic key.

As first I selected two type of dataset of time-varying signals, to be encoded in the spike domain, the Free Spoken Digit (FSD) and the WISDM. By carrying out an analysis of the types of data processed, I outlined a first classification, based on the organization of the information, identifying the temporal, spatial or a combination of the two, defining the classes of Temporal data, Spatial Data and Spatial-Temporal Data. This allowed me to identify the characteristics of a signal in order to choose the processing technique.

Referring to bio-inspired techniques, I was able to identify a plethora of preprocessing and coding techniques capable of encoding audio, video, olfactory, tactile signals, etc., which can be divided into classes, such as the Rate Encoding class where the information is encoded in the number of spikes per unit of time, and the Temporal Encoding where the information is included in the number of spikes, the interval between two spikes, the spike time, etc. By applying these techniques on temporal and spatial data, I identified specific criteria for selecting the most suitable technique.

Then as third phase I implemented a SNN construction and training process, based on the Transfer Learning method, performed by training an artificial neural network ANN and then inject the trained weights in a SNN twin. Specifically, in this work, this procedure was adopted to switch from a convolutional neural network CNN to a spiking CNN. The data to train the CNN architecture, achieved through the production of the sonogram, a reprocessing of the spike data converted

into an image. Finally, the last step implemented is the encoding the sonogram through the rate coding, in order to make the information suitable for SNN processing. This method might seem laborious, however these steps allowed me to define a unique evaluation criterion of the various coding techniques. Observing test accuracy sets, 98% accuracy was recorded for some types of encoding algorithms, while for others up to 8%, this confirms that some encoding techniques are specifically implementable for Temporal data while others on Spatial data.

Once the best performing techniques were identified, I decided to conduct further investigations on synapse reduction techniques, which allow to reduce the size of the network making it suitable for use on embedded devices, thanks to a reduced memory footprint and a smaller computational cost. In some cases I have found that the accuracy of the reduced network exceeds the classification capabilities compared to the complete one, as in the case of the WISDM for which I observed an increase in accuracy starting from 86.7% up to 95%

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Machine Learning

Machine learning represents a branch of artificial intelligence, if with the latter we mean the abilities of a machine in showing human abilities such as reasoning, learning, planning and creativity, for machine learning we mean the capabilities by a machine to automatically extract information without the aid of an external agent, from the surrounding environment. These tasks are developed through mathematical models or more sophisticated methods such as artificial neural networks, to solve tasks that are very difficult or otherwise impossible to achieve with traditional programming techniques, such as image classification [1], image processing [2], audio segmentation [3].

In this thesis the main focus is to combine the concepts and models developed in the field of state of the art Machine learning, with the most recent studies addressed in the field of Neuroscience and Neuromorphic Engineering, highlighting advantages and disadvantages in the use of methodologies and bio-inspired models.

But before diving into this study, let's try to describe what Machine Learning are, and the key elements that we are going to use during our study.

There are several criteria that can be used to classify the Machine learning models, given the vastness of method, the possible tasks that can be performed and their versatility.

Analyzing the criteria individually with reference to fig. 1.1 we have:

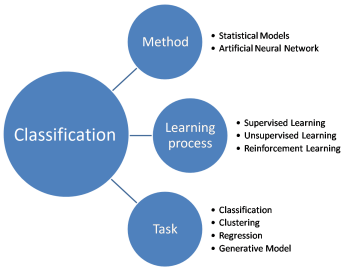


Figure 1.1: Classification methods

- Classification based on the method [4], by method we mean the approaches from the modeling point of view to extract information from raw data:
  - Statistical models: historically this is first type of model, that use the classical statistical tools, for the identification of a probabilistic model and a set of parameters necessary to describe the data analyzed;
  - Artificial neural networks: used in more recent applications, these models instead are adaptive systems capable of modifying their behavior based on the internal interactions between the elements that constitute it, and the data that are used during the training phase.
- Classification based on the learning process. The learning process represents the set of operations carried out in order to obtain the value parameters capable of describing the data under examination, obtaining:
  - Supervised learning: this learning process consists in providing to the algorithm the data to be analyzed such as an image [5], or in less direct cases quantities collected by a set of sensors [6] [7], and the corresponding outputs, i.e. the class to which it belongs of the data in a classification problem, the action to be carried out in the control of systems or the output value of a continuous quantity as in regression problems. This couple of quantity define the pair datum and label;
  - Unsupervised learning: in this case, on the other hand, the algorithm is supplied only with the data to be analyzed without providing a corresponding label;
  - Reinforcement learning: this particular approach, unlike the previous ones, is used in a dynamic environment for which data are provided during the temporal evolution of the system, such as learning a game. Recent examples have been applied in the case of Chess [8], Go [9], Atari [10].
- Classification based on the task to be performed, we have:

- Classification/Clustering problem: although these two tasks seem very different from each other, in reality they represent the same problem but seen from two different perspectives. In both cases an attempt is made to make a prediction of the class to which the data analyzed belongs, but in the first case in classification problem, the model used derives from a supervised learning process, in which the number of classes is known,and the target is to define the class. In the case of clustering the number of classes is not known a priori, consequently the goal is to identify the number of classes most suitable for data modeling and define the class they belong to through an unsupervised learning;
- Regression analysis: this method as opposed to the previous ones tries to model outputs in a continuous space, an example of this model can be applied in the field of finance for the prediction of the stock market [11], the prediction of the position of a vehicle based on the GPS data [12];
- Generative Models: these models are more recent implementations, in which the aim is ti generate new data starting from an ensemble of pre-existing data. An example of this application is the generation of a new dataset of faces starting from a dataset of real faces [13].

1.2 Neuromorphic Engineering

Focusing now on the definition and fields covered by Neuromorphic Engineering; the concept of Neuromorphic Engineering was first introduced by Carver Mead [14] in 1980 to describe a new field of engineering which focuses attention to the study and realization of models and devices, whose architecture and functioning try to mimic the same functionalities of a nervous system human or animal. Examples of devices made through these studies using biological systems as inspiration are silicon cochlea [15], silicon retina [16], devices whose operation tries to reproduce the operations carried out by the ear and the human eye respectively. Obviously the biggest challenge is to develop models and devices capable of replicating the cognitive abilities of the human brain in order to exploit its great potential in terms of cognitive abilities, great computing power and low energy consumption.

Research conducted in this new branch of the science has numerous advantages in various fields, such as in the medical environment for the creation of prostheses which can interface directly with the nervous system, creating stand-alone systems, very compact in terms of computing and energy consumption, or the study of diseases that affect the brain such as schizophrenia, dementia, parkinson, in the IT field through the development of low energy consumption processors but with performance comparable to the most advanced chips made in a classical way [17], sensors, and so on.

1.3 Spiking Neural Network (SNN)

The union of the Machine Learning and Neuromorphic Engineering has given rise to the Spiking Neural Network (SNN) defined as the third generation of natural networks [18]. This new typology of neural networks has shown complementary advantages compared to the classic Artificial Neural Networks (ANN), demonstrating a reduction of several orders of magnitude in terms of energy consumption, and very high computation speed and other possible advantages inherited from biological systems, while maintaining accuracy comparable with the state of the art in ANN.

Through the use of SNNs we will study the impact of the different encoding techniques on time-variant signals, obtained through the analysis of the datasets Free Spoken Digits (FSD) composed of audio signals of spoken digits and the Wireless Sensor Data Mining (WISDM) a dataset a dataset aimed at collecting sensory signals of human daily activities.

In order to proceed we analyze the all machine learning pipeline from the neuromorphic point of view, using the classical structure present in the case of state of the art machine learning procedure, referring to the diagram show in fig. 1.2:

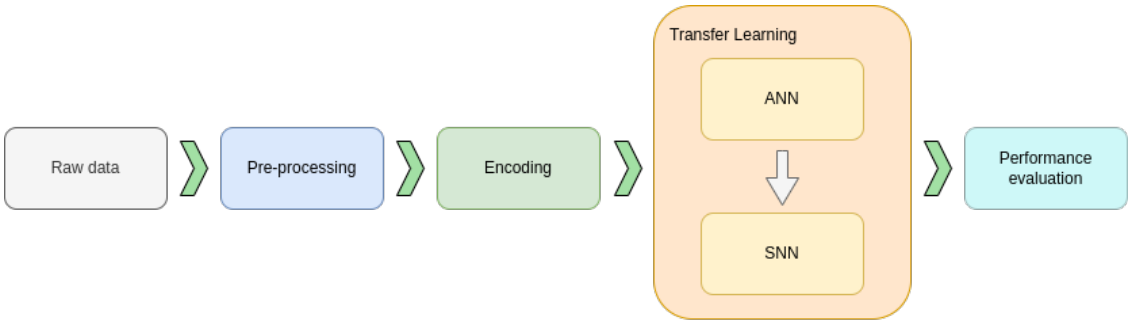


Figure 1.2: Machine learning pipeline

- Raw data: represent the set of samples collected through and their respective labels, since in this case a supervised training process is carried out;
- Pre-processing: represent the techniques used to elaborate the raw data in order to make the network training process easier, such as the normalization process, filter decomposition, whitening and so on;
- Encoding: are the ensemble of techniques used to translate the analyzed sample in a neural coding structure, allowing to be processed by the SNNs;
- ANN/SNN: this block describes the structure of the networks used and the learning process implemented to train the Spiking Neural Network. Since to date no efficient algorithms have been found to train an SNN, a hybrid

method called transfer learning will be used to proceed in this step. In which the learning process of the network is divided into 2 phases:

1. the training process is carried out on the ANN, which constitutes the mother network from a structural point of view and the parameters;
2. once the learning process is performed, a structurally identical SNN is generated, and the parameters used for the description of the model are inherited from the mother ANN.

- Performance Evaluation: in this last step in which the performance of the different encoding techniques is evaluated, and we try to define the main fields of application.

Chapter 2

Background

> The human brain, (...) is the most complicated organization of matter that we know.
>
> *Isaac Asimov (1987). "Past, Present, and Future"*

In order to understand the structure and functioning of SNNs, in this section we will introduce the key concepts related to the biological systems to which these networks draw inspiration, describing the nervous system from a generic point of view and describing in detail the neuron and its models in order to describe the basic element of SNN artificial neuron, which tries to mimic the behavior of the real neuron as faithfully as possible.

2.1 Nervous System

By neuroscience we mean the set of studies carried out on the nervous system, through which we can determine the morphology and its functioning. Focusing the attention on the nervous system of vertebrate organisms, we can identify two types of nervous system. The Peripheral Nervous System (PNS) consists of the set of nerve cells capable of collecting information from the external environment, such as the collection of light signals by the eyes, or audio signals through the auditory system system and so on and the Central Nervous System (CNS) that has the task of combining, controlling and processing the information coming from the PNS, and provide a response stimulus. Although these nervous systems have a large heterogeneous cell typology, from the point of view of structure and specialization, the common element is how these cells treat the information to be processed, that is, in the form of an electrical impulse.

2.1.1 Neuron

The human brain fig. 2.1 represents the main organ of the CNS, whose purpose is to collect, process, send and store information. Observing its functioning in detail, it can be seen that it is made up of highly specialized areas, capable of carrying out well-defined tasks. Although the constituent element of the brain are neurons, the following degree of specialization is obtained based on the different structure that neurons have within the region, defining neural networks, such as feedforward, recursive, convolutional or hybrid, whose element base we repeat is the neuron.

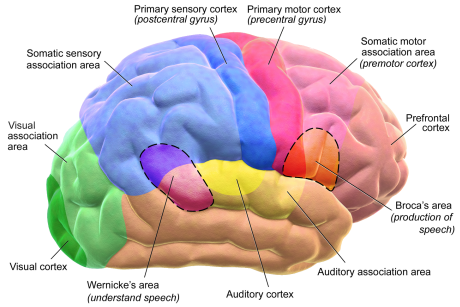


Figure 2.1: Human Brain; Medical gallery of Blausen Medical 2014 [19] (CC BY 3.0)

The neuron is the building block of every neural network in the brain, so knowing how it works, is necessary to understand how information is processed, and referred to this the only elements of our interest in this case are reported in fig. 2.2:

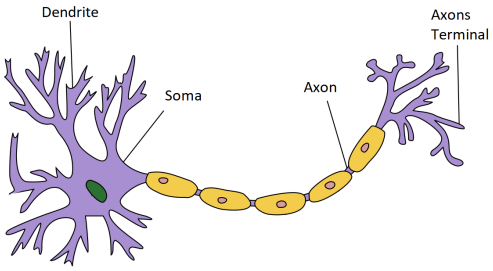


Figure 2.2: Minimal neuron structure; Quasar Jarosz (CC BY-SA 3.0)

By describing the role and structure of each element we have:

- Soma: the central body of the neuron where all stimuli are collected in the form of electrical impulses;
- Dendrites: they are a dense network of ramifications, that starting from the soma extend allowing to communicate with the synaptic terminals of other neurons, thus defining the input terminals of the neuron;

- Axon: it is the only branch that has the task of carrying to all the connected neurons the electrical impulse also called Action potential produced through stimulation of the soma;
- Axonal terminals: it defines the second network of branches that allow the output communication of the neuron with those to which it is connected, thus defining the output terminals of the neuron.

2.1.2 Synapse

The communication between two or more neuron is obtained through the couple dendrite and axonal terminal represent the synapses (fig. 2.3), this connection allows neurons to exchange information in the form of an electrical impulse.

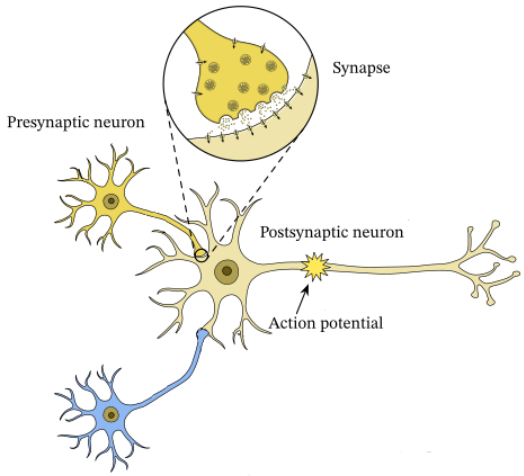


Figure 2.3: In this case is reported a little neural network composed by two Pre-synaptic neuron that interact with one Post-synaptic neuron

In order to observe how stimuli are collected, and used to communicate with other neurons we analyze in detail the physical structure and the communication process of the synapse between two neurons.

From the point of view of the nomenclature it is indicated with:

- Pre-synaptic neuron: neuron that sends the action potential;
- Post-synaptic neuron: neuron that receives the action potential;

The parameter that allows us to describe the state of a neuron is the Membrane Potential V_m, through the study of this parameter we can deduce the behavior of the neuron and how it is influenced by the stimuli received. In order to analyze the communication process, the diagram of fig. 2.4, identify three main communication steps are 3:

1. starting from rest condition of the synapse, inside the pre-synaptic terminal there are vesicles that contain receptors (proteins) capable of interacting with the ion channels of the post-synaptic terminal, opening them or asking them by changing the membrane potential of the post-synaptic neuron. If an action potential is produced by the pre-synaptic neuron, these vesicles are expelled releasing the receptors towards the post-synaptic terminal;

2. the newly released receptors reach the active sites of the ion channels favoring their opening or closure, based to the type of synapse;

3. once opened, these ion channels allow the passage of positive ions outside the neuron, causing a polarization of the membrane potential, producing an action potential.

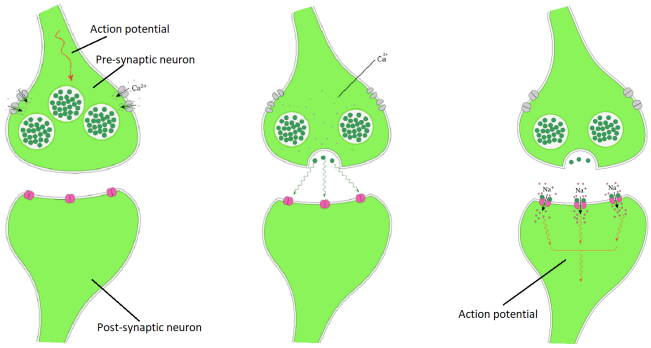


Figure 2.4: Here are reported the three fundamental step of the propagation of the action potential from the Pre-synaptic neuron to the Post-synaptic neuron

Analyzing in detail the production of the action potential from the point of view of the membrane potential V_m we have:

1. starting from a rest level V_{rest} it takes a value $-60mV \leq V_m \leq -70mV$, the arrival of an action potential by the pre-synaptic neuron causes progressively polarization of the membrane increasing its potential up to a threshold value of $V_{th} \simeq -50mV$;

2. a further increase in the level causes a sudden increase generating our action potential reaching about $V_{ap} = 40mV$, generating the action potential that propagates along the axon up to axonal terminals;

3. subsequently the production of the action potential cause the closure of the ion channel, and the ion pump trying to restore the correct membrane potential level, this re-polarization process however causes a hyperpolarization of the membrane reaching potential around $-80mV \leq V_{rest} \leq -100mV$, preventing

in this region an emission of another action potential, this condition is quantified over a period of time called the refractory period $1ms \leq \tau_{refra} \leq 3ms$.

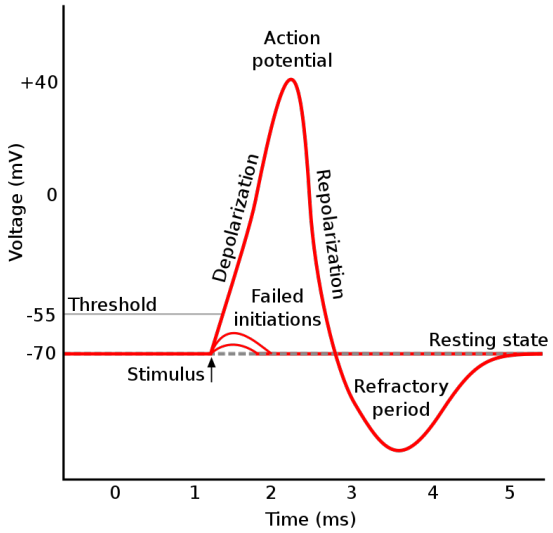


Figure 2.5: Action Potential; Jarvisa (CC BY-SA 3.0)

Experimentally it has been observed that the duration and intensity of the various action potentials emitted by the neurons are similar, however other observations have shown that although the action potential received by the post-synaptic neurons in a network are the same, some neurons in a network emit more action potentials than others. This discrepancy is justified by the degree of excitability of a neuron based on what is called "synaptic weight", linked to the number of ion channels, which can increase or decrease depending on the degree of interaction between the neurons involved. During the learning process in the field of machine learning it is precisely the "synaptic weight" that we try to extrapolate.

2.2 Neuron model

There are numerous models that try to describe the behavior of a single neuron, such as the Hodgkin-Huxley model, Perfect Integrate-and-fire, Leaky integrate-and-fire, and so on, or models that take into account stochastic signals coming from the contribution of the neurons inside a network, Noisy input model (diffusive noise), Noisy output model (escape noise), and more accurate model.

Focusing on a model that tries to describe the interaction process described in the previous chapter we will analyze the Hodgkin-Huxley model and a simpler version of it used for future purposes the Leaky integrate-and-fire models.

2.2.1 Hodgkin-Huxley model

Developed in 1952 by Alan Hodgkin and Andrew Huxley, through the study of the action potential that propagated in the squid giant axon [20], the value of the membrane potential V_m as a function of the ion channels. Since the ion channels allow to carry out a charge transport, these channels can therefore be modeled through current generators, consequently we can model the neuron as an equivalent circuit of the type fig. 2.6,

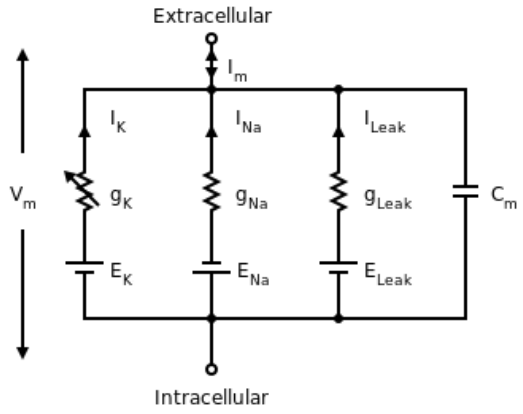


Figure 2.6: Equivalent circuit Hodgkin-Huxley model

from this circuit we deduce that V_m depends on:

- C_m: a capacity due to the presence of a different concentration of ions between the inside and outside of the membrane;
- I_{Na}, I_K: they model the charge transport due to the sodium and potassium ion channels respectively,; since these terms are modeled as a current generator, using Ohm's law, this terms can be interpreted as a voltage generator E_k, E_{Na} and corresponding admittance $g_{Na}(V_m)$, $g_K(V_m)$, through experimental observations it was observed that this parameter depends on the membrane potential:

$$I_{Na} = E_{Na} \cdot g_{Na}(V_m), \quad I_K = E_K \cdot g_K(V_m) \tag{2.1}$$

  focusing attention on potassium channel K, the same conclusion it can be applied in other channels. Analyzing the relations that link the various parameters we have that the voltage value E_K is related through the Nernst equation obtained from the results of statistical mechanics through the Boltzmann weight [21], which expresses the potential that is established in a solution in presence of a charge imbalance:

$$n_1^K \propto e^{-\beta\mu_1^K}, \quad n_2^K \propto e^{-\beta\mu_2^K} \quad | \quad \beta = \frac{1}{k_B T} \tag{2.2}$$

where n_1^K, n_2^K are the ionic charge concentration, defined respectively inside and outside the membrane; μ_1^K, μ_2^K the respective chemical potential; $q = Z \cdot e$ ionic charge that allow to convert the chemical potentials into electrode potentials $E_1^K = \mu_1^K/q$, $E_2^K = \mu_2^K/q$:

$$\begin{aligned}
\frac{n_1^K}{n_2^K} &= \frac{e^{-\beta\mu_1^K}}{e^{-\beta\mu_2^K}} \\
\frac{n_1^K}{n_2^K} &= e^{\beta(\mu_2^K - \mu_1^K)} \\
\beta(\mu_2^K - \mu_1^K) &= \ln\left(\frac{n_1^K}{n_2^K}\right) \\
\mu_2^K - \mu_1^K &= k_B T \ln\left(\frac{n_1^K}{n_2^K}\right) \\
\frac{\mu_2^K - \mu_1^K}{q} &= \frac{k_B T}{q} \ln\left(\frac{n_1^K}{n_2^K}\right) \\
E_2^K - E_1^K &= \frac{k_B T}{q} \ln\left(\frac{n_1^K}{n_2^K}\right)
\end{aligned} \tag{2.3}$$

$$E_K = E_2^K - E_1^K = \frac{k_B T}{q} \ln\left(\frac{n_1^K}{n_2^K}\right) \tag{2.4}$$

the term $g_K(V_m)$ it allows to define the equivalent admittance of the ion channel whose value depends on the equilibrium value g_k that is, in the case of $V_m = V_{rest}$ and from the membrane potential through a dimensionless parameter n which is between 0 and 1, which respectively indicate a completely closed and completely open channel.

$$g_K(V_m) = g_K \cdot n^4 \tag{2.5}$$

The temporal evolution as function of V_m is obtained through a master equation approach:

$$\frac{dn}{dt} = \alpha_n(V_m)(1 - n) - \beta_n(V_m)n \tag{2.6}$$

where $\alpha_n(V_m)$ refers to the probability of passing from an open channel starting from a closed channel, while $\beta_n(V_m)$ is the opposite. Consequently the value of g_K is expressed as the quantity in the rest condition:

- I_l: defined as leakage current a constant value, this term takes into account the continuous current that passes through the membrane due to a diffusive phenomenon, caused by the different concentration of ions.

The set of differential equations that allow to describe the output action potentials produced by a neuron, as a function of an external current $I(t)$ that takes into account, the action potential received are:

$$\begin{aligned}
\frac{dV_m}{dt} &= \frac{I(t) - I_{Na}(V_m, m, h) - I_K(V_m, n) - I_l(V_m)}{C_m} \\
\frac{dm}{dt} &= \alpha_m(V_m)(1-m) - \beta_m(V_m)m \\
\frac{dn}{dt} &= \alpha_n(V_m)(1-n) - \beta_n(V_m)n \\
\frac{dh}{dt} &= \alpha_h(V_m)(1-h) - \beta_h(V_m)h
\end{aligned} \tag{2.7}$$

$$\begin{aligned}
I_{Na} &= g_{Na} m^3 h \cdot (V_m - E_{Na}) \\
I_K &= g_k n^4 \cdot (V_m - E_K) \\
I_l &= g_l \cdot (V_m - E_l)
\end{aligned} \tag{2.8}$$

$$\begin{aligned}
\alpha_m &= \frac{0.1(25-V)}{e^{\frac{25-V}{10}} - 1} & \beta_m &= 4e^{-\frac{V}{18}} \\
\alpha_n &= \frac{0.01(10-V)}{e^{\frac{10-V}{10}} - 1} & \beta_n &= 0.125e^{-\frac{V}{80}} \\
\alpha_h &= 0.07e^{-\frac{V}{20}} & \beta_h &= \frac{1}{e^{\frac{30-V}{10}} + 1}
\end{aligned} \tag{2.9}$$

with $V = V_m - V_{rest}$.

Parameter	Value
C_m	1.0 μF cm^{-2}
g_{Na}	120.0 mS cm^{-2}
g_K	36.0 mS cm^{-2}
g_l	0.3 mS cm^{-2}
E_{Na}	50.0 mV
E_K	-77.0 mV
E_l	-54.387 mV
V_{rest}	-65.0 mV

Table 2.1: Hodgkin-Huxley model parameters

2.2.2 Leaky integrate-and-fire model

The leaky integrate-and-fire model (LIF) described by Louis Édouard Lapicque [22], represents a simplified version of the model described above, in which the individual

ion channels are not considered, but only the admittances related to the global behavior of the channels. As can be seen in fig. 2.7 within this model we can identify two voltage generators with different signs, this means that the presence of an admittance other than 0 for the different generators allows to describe a process of membrane polarization or depolarization, favoring or not the production of an action potential. This allows us to identify two types of synapses:

- excitatory synapses: these types of synapses if stimulated through the action potentials favor the depolarization of the membrane, allowing the emission of a action potential;
- inhibitory synapses: in this case the opposite behavior occurs;

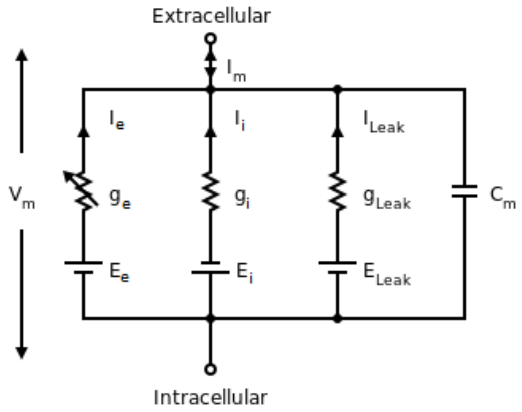


Figure 2.7: Equivalent circuit Leaky integrate-and-fire model

In order to obtain the equations of this LIF model, we refer to the master equation relation obtained in the Hodgkin-Huxley model, and we reorganize the terms as follow:

$$\begin{aligned}
\frac{dn}{dt} &= \alpha_n(V_m)(1-n) - \beta_n(V_m)n \\
\frac{dn}{dt} &= \alpha_n(V_m) - \alpha_n(V_m)n - \beta_n(V_m)n \\
\frac{dn}{dt} &= \alpha_n(V_m) - [\alpha_n(V_m) + \beta_n(V_m)]n \\
\frac{1}{\alpha_n(V_m) + \beta_n(V_m)}\frac{dn}{dt} &= \frac{\alpha_n(V_m)}{\alpha_n(V_m) + \beta_n(V_m)} - n \\
\tau_n(V_m)\frac{dn}{dt} &= n_\infty(V_m) - n \\
\frac{dn}{dt} &= \frac{n_\infty(V_m)}{\tau_n(V_m)} - \frac{n}{\tau_n(V_m)}
\end{aligned} \tag{2.10}$$

$n_\infty(V_m)$ indicates the value assumed at $t \to \infty$, while $\tau_n(V_m)$ indicates the time needed to reach the $n_\infty(V_m)$ value after a perturbation.

Making a comparison between the equation obtained in the Hodgkin-Huxley model and the Leaky integrate-and-fire model:

$$\begin{aligned} \tau_m \frac{dV_m}{dt} &= g_e(E_e - V) + g_i(E_i - V) - (V_m - E_l) \\ \frac{dg_e}{dt} &= w_e \sum_{i=1}^{I} \delta(t - t_i) - \frac{g_e}{\tau_e} \\ \frac{dg_i}{dt} &= w_i \sum_{j=1}^{J} \delta(t - t_j) - \frac{g_i}{\tau_i} \\ V &= V_m - V_{rest} \end{aligned} \tag{2.11}$$

the parameter τ_m indicates the time necessary for the membrane potential V_m to return to the equilibrium value V_{rest} after a perturbation, in a similar way we interpret τ_i, τ_e for admittances.

The E_e, E_i indicates the electrical potential relating to the excitatory and inhibitory synapses, while g_e, g_i the admittances valued that are function of the weight of the synapses w_e, w_i, these parameters indicate the effect of excitability or inhibition of the neuron after receiving an action potential, the greater they are the greater the degree of excitability or inhibition of the synapse. Finally the Dirac delta function takes into account the instant the action potential is received.

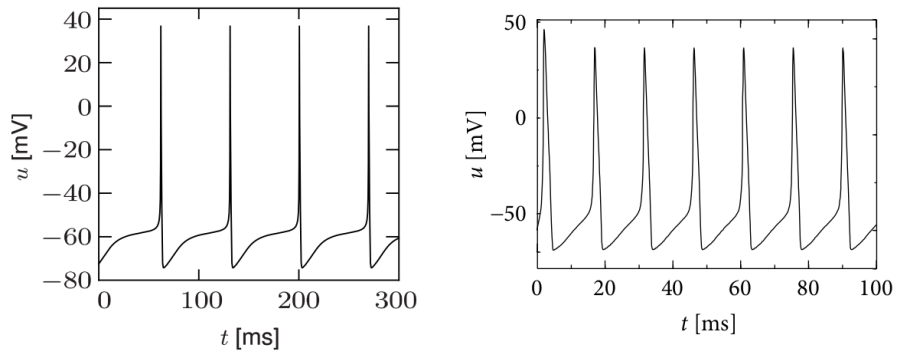
$$\delta(t - t_i) = \begin{cases} 1 & t = t_i \\ 0 & t \neq t_i \end{cases} \tag{2.12}$$

2.3 Neural code

By neural code we mean the way in which the stimuli received from a biological system, are encoded in the form of a series of action potentials, called spikes in neuromorphic engineering. By analyzing the behavior of different neurons subjected to external and internal stimuli, a variety of response can be observed, which can be enclosed in two encoding techniques, the Rate Coding and Temporal coding.

Analysing separately the cases we have:

- Rate Coding: in this case the information is encoded in the number of spikes per unit of time. As can be seen in fig. 2.8 two different quantities are encoded, and this is reflected in having a number of spikes. This kind of behavior can be obtained by through a constant stimulus of different intensity, from which is easily deduced that the information of this stimulus is contained in the number of spikes per unit of time. Examples of this behavior have been observed in muscle control process [23].



(a) Rate Coding (b) Rate Coding

Figure 2.8: Representation of spike train in the Rate coding case, adapted from [24]

- Temporal coding: in this second category of coding, the information is contained in different elements of the series of spikes [25] such as:
  - the time a spike was emitted, respect to the source;
  - the time interval between the various spikes called Inter-Spike Interval (ISI);
  - the maximum or the minimum number of spike;
  - the combination by reinterpreting the spikes as a sort of binary code.

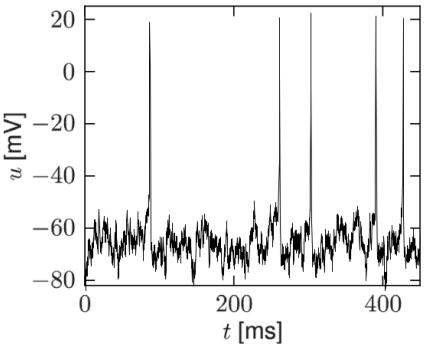


Figure 2.9: Representation of spike train in the Temporal coding case, adapted from [24]

Chapter 3

Materials and methods

SNNs represent the new generation of neural networks, whose functioning and structure are inspired by the networks of organisms present in nature. During the evolutionary process the various organisms have developed different biological systems specialized in collecting, coding and processing based on the nature of the stimuli received. As described in section 2.3, the information inside the neurons is encoded in the form of electrical impulses, so to be correctly processed it is necessary to identify the most suitable coding method based on the data. Below will discuss using the diagram shown in fig. 3.1: the properties of the three categories of data, the pre-processing techniques, the coding algorithms and the pros and cons of using them based on the data and finally the evaluation of the use of such methods as coding for SNNs.

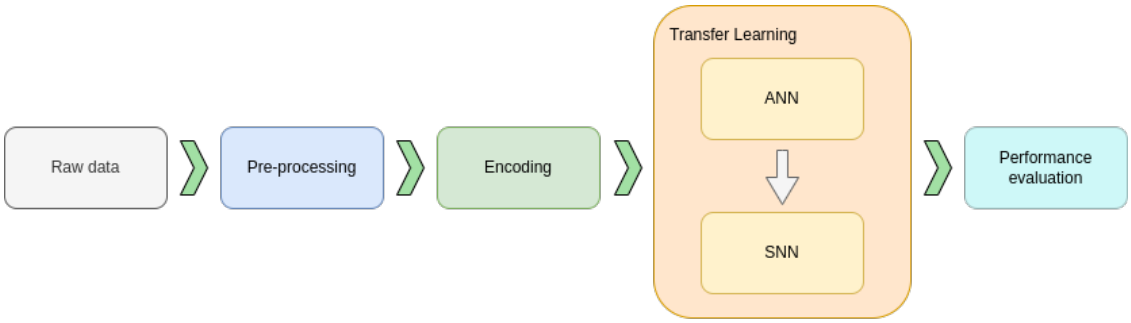


Figure 3.1: Machine learning pipeline

3.1 Temporal data and Spatial data

A first criterion that can be used to determine the most suitable type of encoding, that can be used to treat the information can be extracted through the type of data, being able to identify 3 classes: temporal data as in the case of audio signals,

spatial data for example of the images and the spatio-temporal data in the case of the data that is a combination of the two, which however will not be treated here.

3.1.1 Temporal data

With temporal data we refer to signals whose information content is coded in sequence, i.e. in order to understand the information contained in an audio signal, it is necessary to carry out a serial analysis of the data, analyzing in sequence the various instants.

Obviously, there could be objections to this, given the multitude of techniques inside and outside the field of machine learning that allow to elaborate a parallel analysis using different methods like: wave form analysis, Fourier transform, deep neural network or other kind of methodology that permit to perform a change of space reorganization of the data in a different perspective respect to the temporal organization.

However, this class separation between temporal and spatial data, is based on a bio-inspired analysis procedure. An example of temporal data can be defined by an audio signal as shown in the fig. 3.2.

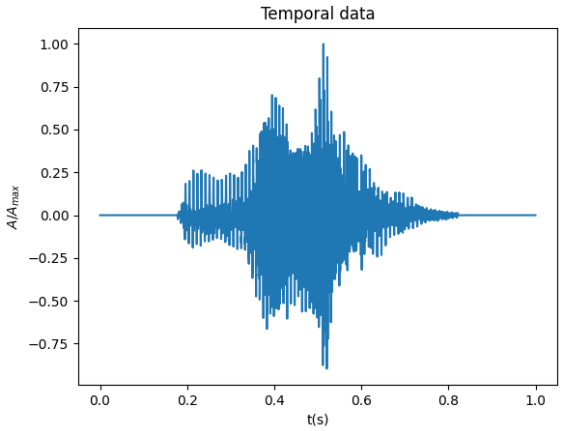
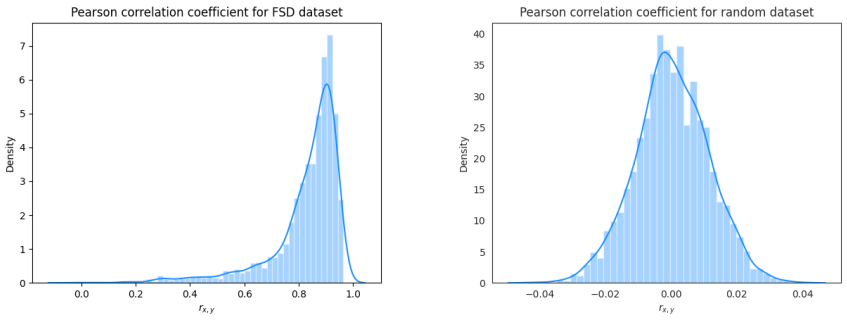


Figure 3.2: FSD audio sample

To verify the temporal structure of the analyzed data it can be used a quantity called in physics temporal correlation, this quantity allows to define how much the value of the data in an instant t is affected by the previous instants $t-1, t-2, \ldots$. A suitable approach is to employ the Pearson correlation coefficient r_{xy}, which in this case gives us that exists a correlation between two consecutive temporal instants:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{3.1}$$

where x represents the vector of our sample, while y is obtained through a shift of the original data, and the $\bar{x}$, $\bar{y}$, is the expected value, being a coefficient of correlation this factor can assume values between $-1 \leq r_{xy} \leq 1$, with $+1$ corresponding to a positive correlation, -1 defined as negative correlation or anti-correlation and 0 indicating non-correlation.



(a) FSD dataset (b) Randomly generated dataset

Figure 3.3: Pearson Correlation Coefficient for Temporal Data, this type of coefficient allow to determine the correlation of the data between two consecutive instant of the audio sample

Graphically representing the distributions of the Pearson correlation coefficient fig. 3.3, obtained through a dataset of audio signals the Free Spoken Digits (FSD), and a dataset of the same size of samples built through a random distribution. In the fig. 3.3(a) shows the distribution relative to the FSD where the maximum peak is at 0.9, index that there is a strong direct correlation between two instants in the samples index, that these data turn out to be temporal data. But in the second case fig. 3.3(b), related to random dataset the peak is around 0 which corresponds to non-correlation.

The comparison between these two cases allows us to demonstrate that a sample in order to be classified as a temporal data, it must possess a fundamental property, there must be a temporal correlation between the successive instants of the signal.

In identifying the coding techniques that can be used to process these types of data, it is necessary to identify algorithms that allow to preserve the temporal link of the information.

3.1.2 Spatial data

In the case of Spatial data, the information appears to have a spatial organization. This means that considering a point in the sample under examination, it presents links with other points based on the dimension of the data. Obviously the following definition is applicable in generic cases such as in case of chemical bonds in crystals that bind atoms, edge in graphs that connect the vertex and so on.

In the case under consideration, however, we will limit ourselves to simple spatial data such as images, to obtain information from this type of data, one has to analyze the structure as a whole, or by analyzing the various clusters, as in the case of image segmentation [26]. A classic example is obtained from the MNIST dataset.

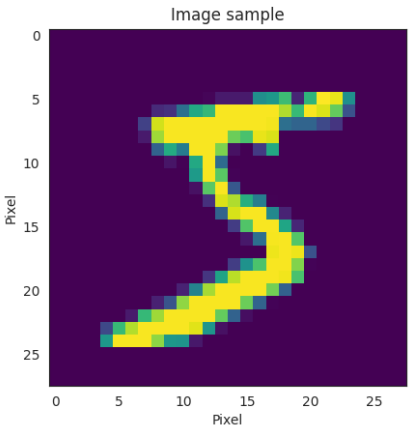
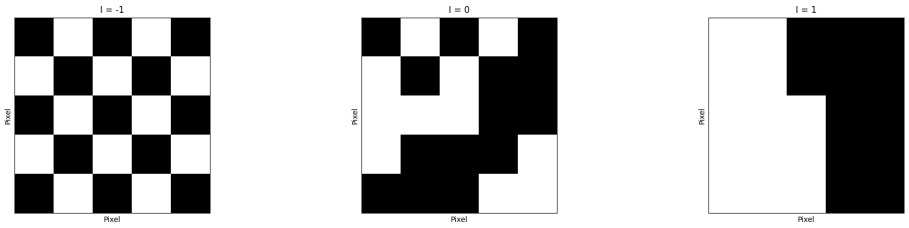


Figure 3.4: MNIST sample, the MNIST dataset is a collection of handwritten digit composed by 60000 training image and 10000 testing image, with 28×28 pixel

As in the case of temporal data, we can identify a parameter that allows us to determine the degree of correlation between the different points in the sample, by defining a spatial correlation. This parameter is defined as Moran's I, which allows to define the degree of clustering within a connected graph.

$$I = \frac{N}{W} \frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2} \tag{3.2}$$

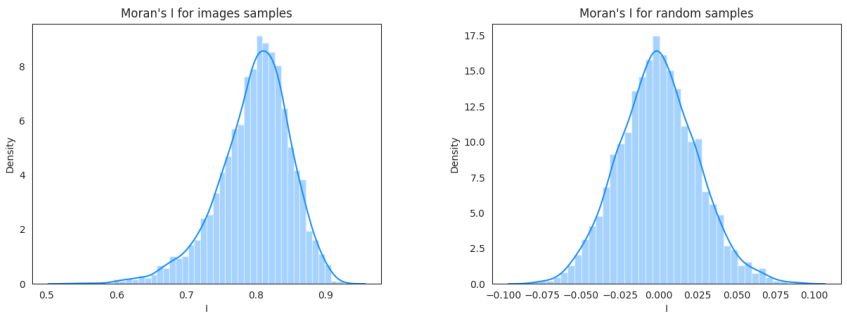
Analyzing the various parameters, we have that N represents the size of the image; x_i defines the point of the image; $\bar{x}$ indicates the expected value; w_{ij} indicates the map value of the first nearest neighbors, while $W = \sum_{i=1}^{\infty} w_{ij}$. Also in this case, since this parameter is a correlation coefficient, its value can be $-1 \leq I \leq 1$, an toy example of a spatial data and the corresponding Moran's I, can be appreciated in fig. 3.5:



(a) Negative correlation (b) Non correlation (c) Positive correlation

Figure 3.5: Moran's I example

As in the previous case, also in this case we try to determine the value of this index, in order to understand if a particular dataset fall into the category of Spatial data. To do this we compare the value of the Moran's I, between the MNIST dataset composet of a set of written digits and one generated through a random distribution.



(a) MNIST dataset (b) Randomly generated dataset

Figure 3.6: Moran's I for Spatial Data

As in the previous case in fig.3.3, also in this case we can observe the same behaviour presenting in the fig. 3.6(a), the distribution of the Moran's I of the MNIST with peak at 0.8, as you can easily guess the presence of such a high value indicates a correlation between the various points in the image, obtaining a spatial correlation of the data. In the second case fig. 3.6(b) the same index is evaluated but in the case of a dataset composed by sample generated randomically, and as can be seen the peak is at 0, this means that no spatial information appears to be contained in these samples.

Therefore, so that such a sample can belong to the set of Spatial data, it is necessary that there is a correlation from the spatial point of view between the various points, of which it is composed.

In this data instead it is not necessary to use coding techniques that preserve the data structure, delegating the task of extracting this link to the neural network, rather an appropriate coding algorithm must be identified, on the contrary must be identified a suitable encoding algorithm, in order to associate a spike code for each pixel as faithfully as possible.

3.2 Preprocessing

In the field of machine learning with pre-processing we mean the set of techniques that allow you to adapt the information to be processed to the system used, making the training process simpler. Examples of such procedures are: the analysis of the samples and their labels to verify possible class imbalance, standardization of the feature having zero mean, the adoption of feature reduction techniques to identify the most relevant characteristics of the data, and all the possible process that applied to the data improve the learning process.

In our case, since time-varying signals are analyzed, the pre-processing step applied is that of decomposition the signal into frequency channels through the use of a filter bank, the application of this method it is founded on the way the human ear processes audio information, furthermore this step not only useful from the point of view of network learning but also makes encoding techniques more effective, because performing such operation results in two main advantages:

1. by decomposing the signal into various channels, we obtain a sufficiently large number of features from which our neural network can extract a more robust model;

2. using this method, Fourier series development is exploited, describing the signal as a combination of n sinusoidal signals. This allows to standardize the criterion by which the various parameters used in the encoding algorithms are defined, since each component has a pseudo-sinusoidal structure.

3.2.1 Filter bank

By filter bank we mean an array of filters that allow you to separate the various frequency components of an input signal. There are different architectures with which the various filters at different frequencies can be connected in order to develop this decomposition based on the type of filter which can be: low-pass filter, high-pass filter or band-pass filter.

Assuming to use a band-pass filter type, the architecture used is reported in fig. 3.7, the next steps to define the characteristics of our filter bank, is to choose how many channels do we need, the technical design specifications of the single filter, that allow to define the gain, bandwidth and the slope of the Transfer function.

Through the theory of signal analysis there are numerous types of filters, for example the Butterworth, Chebyshev, Bessel, and so on, each of these types of filters has different characteristics, which make them suitable for very specific purposes, respectively, such as the introduction of low distortion for audio applications, or the presence of ripple for mechanical applications or phase shifts in phase components.

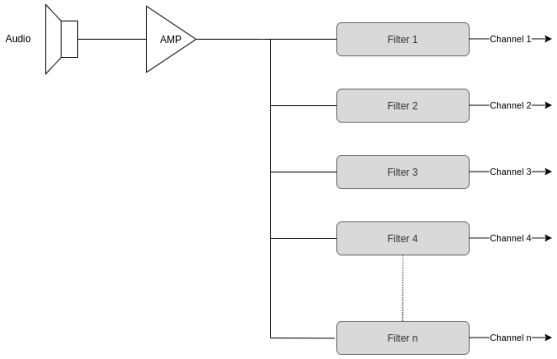


Figure 3.7: Filter bank example scheme

Consequently, since we want to use bio-inspired techniques, for the choice of the type of filter most suitable for the purpose, the characteristics of the biological systems analyzed must be taken into consideration, which in this case refer to the auditory system of the human ear. By this analogy the sensory organ capable of carrying out this filter decomposition is the cochlea [15] whose structure is shown in fig. 3.8.

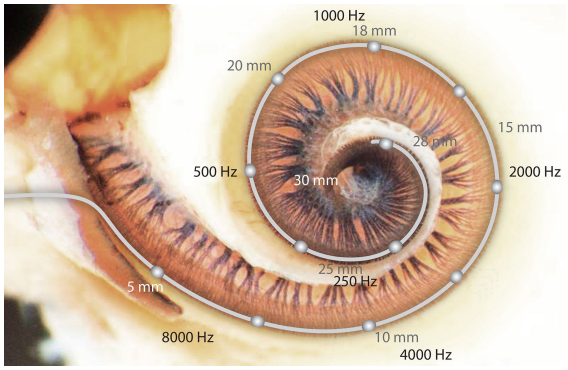


Figure 3.8: The ciliated structure of the cochlea and the relative resonance frequencies based on their length, adapted from [27]

The cochlea represents the terminal part of the auditory apparatus, consisting of a spiral structure very reminiscent of a snail. Inside this spiral there are a series of lashes of gradually increasing length [28], this allows each lash to have its own oscillation frequency, triggered by the resonance phenomenon through external

stimuli. Since an audio signal is made up of different frequency components, each component stimulates a specific cilia based on its intensity, subsequently these oscillations are encoded into electrical impulses which are transposed and finally processed [29]. Through the following analogy it can be observed that the cochlea essentially represents a filter-bank where each cilia represents a specific band-pass filter.

Through the study of the acoustic spectrum and psychoacoustics, it is hypothesized that the decomposition performed by the cochlea can be modeled as a filter bank consisting of Butterworth [30] or Gammatone [31] [32] filters.

Butterworth Filter

By Butterworth filter we mean a filter whose transfer function is defined by means of a Butterworth polynomial $B^{(n)}(s/\omega_0)$ where n define the degree, the corresponding transfer function is:

$$H(s) = \frac{B^{(n)}(\frac{s}{\omega_0})}{B^{(m)}(\frac{s}{\omega_0})} \quad | \quad m > n \tag{3.3}$$

analyzing in detail the Butterworth polynomial:

$$B_n(s) = \sum_{k=0}^{n} a_k s^k \quad | \quad a_k = \prod_{l=1}^{k} \frac{\cos((l-1)\gamma)}{\sin(l\gamma)}; \gamma = \frac{\pi}{2n}; a_0 = 1 \tag{3.4}$$

A graphic example of a Butterworth band-pass filter bank with is shown below:

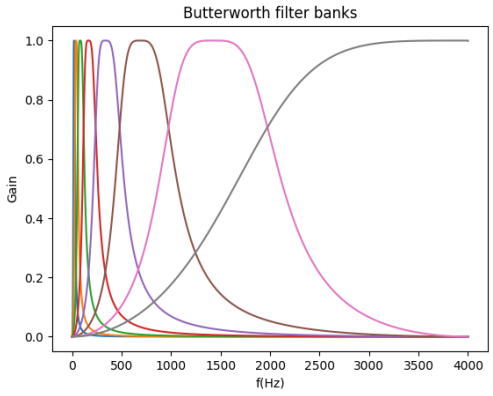


Figure 3.9: Butterworth filter bank

Gammatone Filter

The Gammatone filter represents another type of filter that tries to model the acoustic spectrum of the human ear; it assumes a slightly more complex structure

compared to that of Butterworth from the point of view of the transfer function $H(s)$ eq. 3.5 [32].

$$H(s) = \frac{e^{i\phi}[s + \frac{\omega_0}{2Q} + i\omega_0\sqrt{1 - \frac{1}{4Q^2}}]^N + e^{-i\phi}[s + \frac{\omega_0}{2Q} - i\omega_0\sqrt{1 - \frac{1}{4Q^2}}]^N}{[s^2 + \frac{\omega_0}{Q}s + \omega_0^2]^N} \tag{3.5}$$

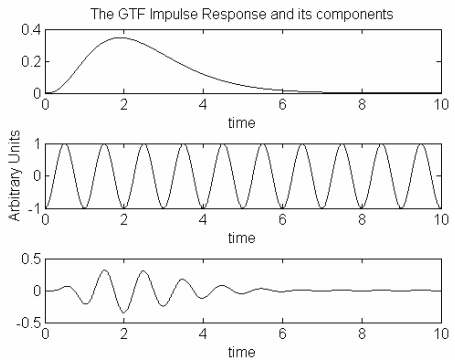
A more informative analysis is obtained through the Gammatone impulse response function $f(t)$ eq. 3.6 [32], which allows to describe the response over time by the Gammatone in response to a stimulus described by a Dirac delta $\delta(t = 0)$

$$f(t) = At^{N-1}e^{-bt}\cos(\omega_r t + \phi) \tag{3.6}$$

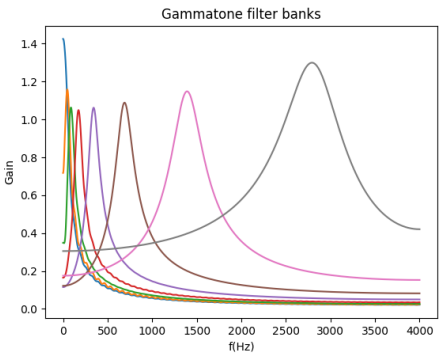
analyzing this function we can easily understand where the term Gammatone comes from:

- $At^{N-1}e^{-bt}$: gamma-distribution, where A represents the amplification factor, while N indicates the filter grade and b the bandwidth;
- $\cos(\omega_r t + \phi)$: tone-function ω_r is the oscillation pulsation, ϕ the phase shift.

A graphic example of a band-pass filter bank with Gammatone is shown below fig. 3.10.



(a) Gammatone impulse response, adapted from [32]



(b) Gammatone filter bank

Figure 3.10: Gammatone Time and Frequency response

A notable difference between the two types is the overlap of the bands, which is greater in the case of Gammatone for a high number of filters; this will prove to be an important aspect in the encoding performance.

An example of the application of the application of the filter bank on a sample of the FSD is shown in fig. 3.11:

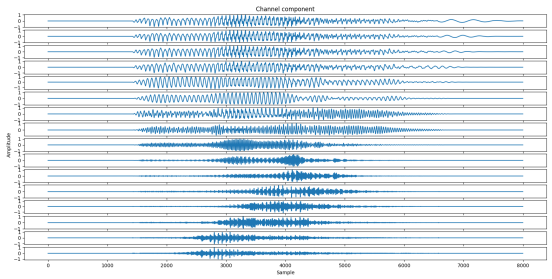


Figure 3.11: Here is an example of splitting into 16 channels through a filter bank consisting of butterworth filters of a sample of the FSD

3.3 Encoding Algorithms

In this section we will explore the various coding techniques that can be used to encode information, in order to be properly processed by SNNs.

As mentioned in Section 2.3 during the evolutionary process organisms have developed biological systems specialized in the collection of various external stimuli and in the encoding in the form of electrical impulses, also called spike trains. Based on the way the information is encoded, two coding techniques can be identified, Rate Coding and Temporal Coding, each of which has different classes to which different methods are associated.

3.3.1 Rate Coding

Rate coding was one of the first classes of algorithms hypothesized through experimental observations, performed in the nervous system specialized in the transport of information [33]; in this case the information is encoded within the number of spikes per unit of time. Further studies on the analysis of the reaction time, for the visual [34], olfactory [35], tactile [36] and auditory [37] systems, have shown that this coding method it would take a very long time of analysis to collect the stimuli necessary to obtain an exhaustive response [38], for this reason, the use of this method in time-varying signals has a limited application.

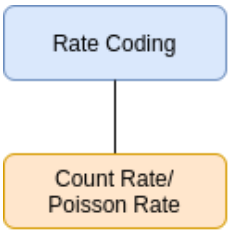


Figure 3.12: Algorithms that are part of Rate coding

Count Rate/Poisson Rate

The algorithm that is part of Rate Coding is called Count Rate or Poisson Rate, the name of this technique derives from the probability distribution used to model the experimental results, i.e. the Poisson distribution.

This coding method was experimentally observed for the first time in nerve cells specialized in the transport of information. The experimental procedure employed [24] to observe this behavior consisted in recording the spikes in time of a set of neurons. Subsequently, in order to extract information from the spike train, the probability distribution of the time intervals between one spike and the next was analyzed, obtaining the graph shown in fig. 3.13

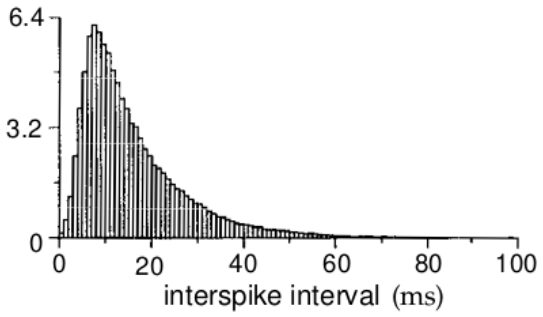


Figure 3.13: ISI distribution, adapted from [39]

Through the following experimental analysis it can be concluded that the Poisson distribution allows to describe this time interval between two spikes, also referred to Inter-Spike Interval (ISI) [40], consequently the information inside this encoding is obtained from the number of spikes per unit time.

In order to use this method for our purposes we develop the inverse process, i.e. starting from the information we want to encode, a spike train will be generated whose interval between the two spikes is generated using the Poisson distribution.

Assuming that the value to be encoded is equal to the value $r \in \mathbb{R}$, to determine the various intervals, we start from the classical definition of Poisson probability distribution (eq. 3.7):

$$P_n(\Delta t) = \frac{(r\Delta t)^n}{n!} e^{-r\Delta t} \tag{3.7}$$

this function allows to define probability of obtaining $n \in \mathbb{N}$ spikes in the time interval $[t, t + \Delta t]$ with a production rate of r.

In order to determine the time interval between only two spikes we impose $n = 0$ and start from the complement of the probability distribution, we reverse

the function by isolating Δt:

$$\begin{aligned} P(\Delta t) &= 1 - e^{-r\Delta t} \\ e^{-r\Delta t} &= 1 - P(\Delta t) \\ r\Delta t &= -\log\left(1 - P(\Delta t)\right) \\ \Delta t &= \frac{-\log\left(1 - P(\Delta t)\right)}{r} \end{aligned} \tag{3.8}$$

$$ISI = \frac{-\log\left(1 - P(\Delta t)\right)}{r} \tag{3.9}$$

The function described by eq. 3.9 it allows to define the time interval between two pulses whose spike rate is r, consequently ISI is the key parameter for the encoding of the information of our interest.

The procedure to be used in order to obtain a rate encoding of a value $x \in \mathbb{R}$ which in the following coding corresponds to the spike rate $x = r$, is described in the following points:

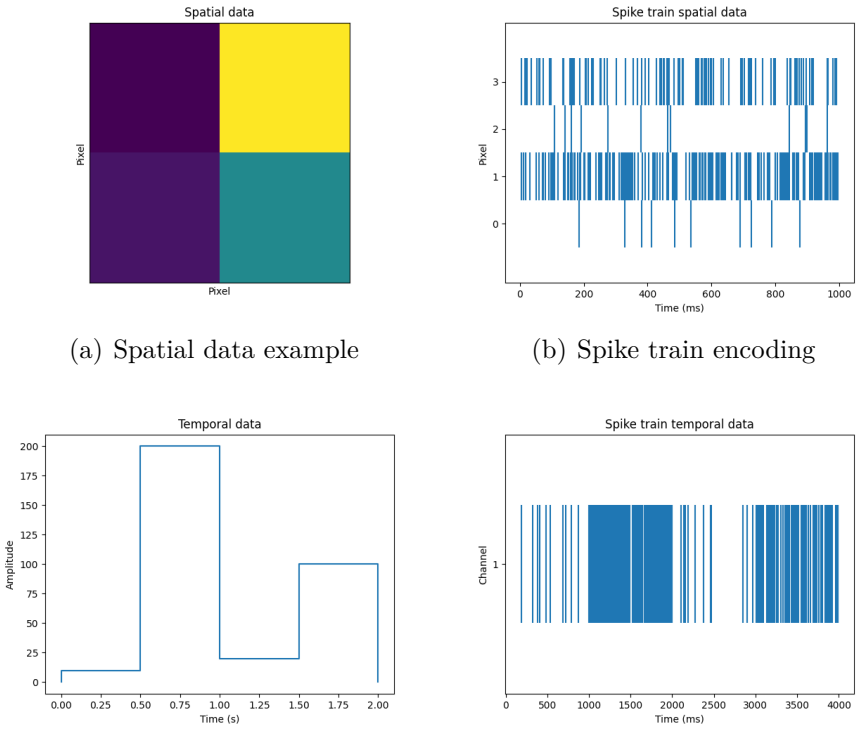
1. first of all it is necessary to define the duration of the stimulus Δt, i.e. how long we want the information to be encoded according to the method used for processing, in the following cases it will be used as stimulus time $\Delta t = 1s$;

2. in order to describe the interval of the first spike with respect to time $t = 0$, eq. 3.9 is used, where the value $P(\Delta t)$ is a value generated according to a random distribution in an interval of $0 \leq P(\Delta t) < 1$, obtaining the first spike at time $t_1 = ISI$;

3. the subsequent spikes are obtained using the same procedure as the previous point by defining the time of the nth spike by means of the relation $t_n = t_{n-1} + ISI$;

4. spike production stops when the time of the next spike is greater than the total duration of the stimulus if $t_{n+1} = t_n + ISI > \Delta t$;

Conclusion As we can see, in fig. 3.14 the application of the Rate encoding technique is shown on the two types of data belonging to the class of Spatial data fig. 3.14(a), which shows an image formed by 4 pixels with a different intensity and Temporal data fig. 3.14(c), consisting of a signal lasting 2 seconds at different levels of intensity.



(a) Spatial data example

(b) Spike train encoding

(c) Temporal data example

(d) Spike train encoding

Figure 3.14: Comparisons between Spatial data and Temporal Data with Rate Encoding algorithm

As can be clearly observed, a better coding performance is obtained in the case of the Spatial data class (fig. 3.14(b)), for which each pixel is treated individually, whose spike production is proportional to the intensity of the same, allowing to encode in the time interval by $1s$ the whole image.

On the other hand, in the case of temporal data (fig. 3.14(c)), one can immediately notice some great disadvantages. Although in this particular case the different intensity produces a well-defined region in which it can be identified the amount of spike related to the level, in the case of weakly variable signal one would have an almost equal distribution of spikes, making it impossible to recover the correct level. Furthermore in order to have an correct encoding as shown turns out to be much longer than the duration of the signal itself. For this reasons the use of this algorithm is inapplicable in the case of rapidly varying signals, since a very long time interval is required to obtain correct encoding [25].

From this analysis it can be concluded that rate encoding is particularly suitable for the encoding of data belonging to the class of Spatial data.

3.3.2 Temporal Coding

Temporal Coding represents the second type of logic that can be used to encode the data, presenting a great variety of classes of algorithms each having advantages and disadvantages depending on the field of application. Compared to the case of Rate encoding, however, they have greater advantages in terms of the number of degrees of freedom that can be used to encode the information, including it in four elements:

- number of spikes;
- the time interval between two spikes;
- time to spike;
- the spike sequence, by treating this combination as some sort of binary code.

through the following criteria we can identify four Classes of encoding techniques as reported in the fig. 3.15, Temporal Contrast, Filter & Optimizer, Global Referenced, Latency/ISI and Correlation & Synchrony.

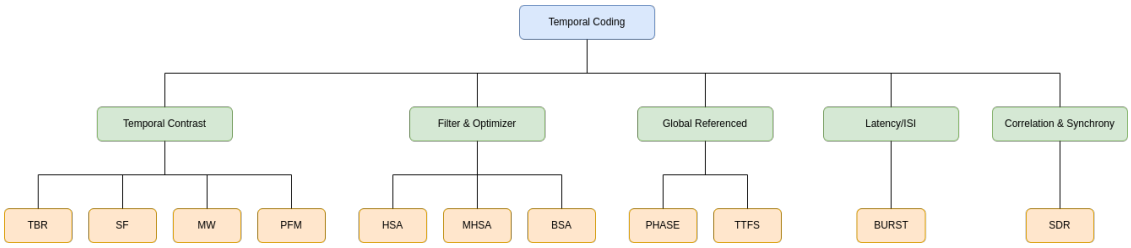


Figure 3.15: Algorithms that are part of Temporal coding

Temporal Contrast

The algorithms in this class are suitable for the coding of time-varying signals, as they are able to detect the variations of between two or more instants of time, making them suitable for the analysis of audio signals [15], measurements for electrocardiogram [41], vibrations in a failure prediction of machine [42], and other type of problem that involve time varying signals.

Belonging to this class of algorithms are Threshold-Based Representation (TBR), Moving Window (MW), Step-Forward (SF), Pulse-Frequency Modulation (PFM).

Threshold-Based Representation (TBR)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $SpikeTime \leftarrow \infty$
  $Variation \leftarrow zeros(1, L)$
  for $i = 0 : L - 1$ **do**
    $Variation[i+1] \leftarrow Sample[c][i+1] - Sample[c][i]$
  end for
  $Threshold = mean(Variation) + factor \cdot std(Variation)$
  $Variation[0] \leftarrow Variation[1]$
  for $i = 0 : L$ **do**
    if $Variation[i] > Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow 1$
    else if $Variation[i] < -Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow -1$
    end if
  end for
end for

Using as reference the signals in fig. 3.11 which shows the 16 frequency channels of a sample of the FSD, let us briefly describe the key steps for the implementation of this coding method.

1. treating each channel individually, first of all a parameter called $Variation$ is defined, which allows to determine the variation of the signal amplitude of the signal between the instant considered and the previous one as shown in fig. 3.16;

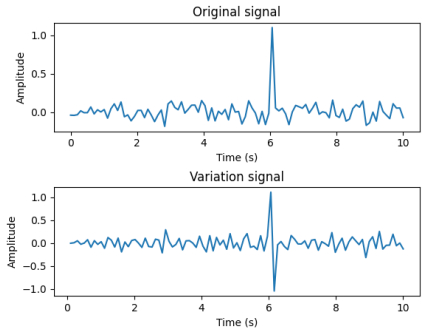


Figure 3.16: Example of signal and the content in the variable $Variation$

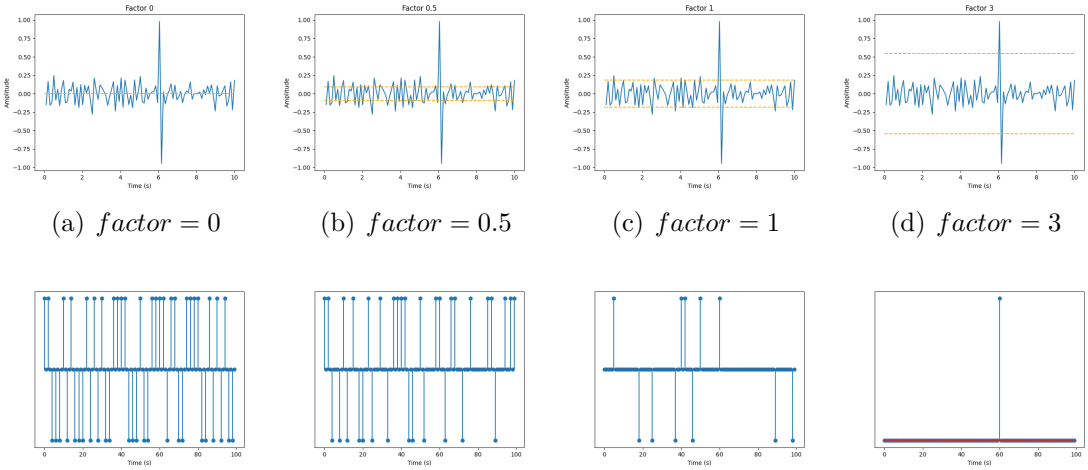
2. through the previous parameter, we can obtain another very important quantity, the $Threshold$, which allows to define the band beyond which, if the signal intensity is greater, a spike will be emitted:

$$Threshold = mean(Variation) + factor \cdot std(Variation) \tag{3.10}$$

the width of this band is regulated by the parameter $factor$, the greater this value, the greater the threshold band, this parameter also plays a more important role from the point of view of noise suppression based on the assumed value:

- $factor = 0$: encode all variation that is different from the mean;
- $0 < factor < 1$: used for signal without noise, but take also low variation that produce high number of spike;
- $factor = 1$: used for signal without noise, but suppress a low variation;
- $factor > 1$: used to suppress noisy signal.

the different encoding performances based on the range of values described above are shown in fig. 3.17;



(a) $factor = 0$ (b) $factor = 0.5$ (c) $factor = 1$ (d) $factor = 3$

(e) Spike train encoding (f) Spike train encoding (g) Spike train encoding (h) Spike train encoding

Figure 3.17: TBR encoding parameters comparisons

3. finally, for the production of spikes it is sufficient to compare the level of the input signal with the threshold values, and in the event that this value is greater in a positive or negative sense than the threshold, a spike is emitted.

The fig. 3.17(e)-(f)-(g)-(h) represent the coded information, in a series of spikes $+1$ and -1. Even if the information in the brain is encoded through positive impulses, this does not represent a contradiction, as two types of synapses can be identified within the connections:

- excitatory synapse: in this case the spike is translated by the neuron as a $+1$ stimulus bringing it towards a state of depolarization;
- inhibitory synapse: in this case the spike is translated by the neuron as a -1 stimulus bringing it towards a state of hyperpolarization.

Moving Window (MW)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $Variation \leftarrow zeros(1, L)$
  for $i = 0 : L - 1$ **do**
    $Variation[i + 1] \leftarrow abs(Sample[c][i + 1] - Sample[c][i])$
  end for
  $Threshold = mean(Variation)$
  $Base = mean(Sample[c][0 : Window])$
  for $i = 0 : Window$ **do**
    if $Sample[c][i] > Base + Threshold$ **then**
      $SpikeTrain[c][i] \leftarrow 1$
    else if $Variation[i] < Base - Threshold$ **then**
      $SpikeTrain[c][i] \leftarrow -1$
    end if
  end for
  $SpikeTime \leftarrow \infty$
  for $i = 0 : Window : L$ **do**
    $Base = mean(Sample[c][i : i + Window])$
    if $Sample[c][i] > Base + Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow 1$
    else if $Variation[i] < Base - Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow -1$
    end if
  end for
end for

MW encoding uses more or less the same logic as TBR, also in this case using a variant of the $Threshold$ value, to define the range of values beyond which a spike is emitted. Although in the case of the TBR the threshold band is fixed for the

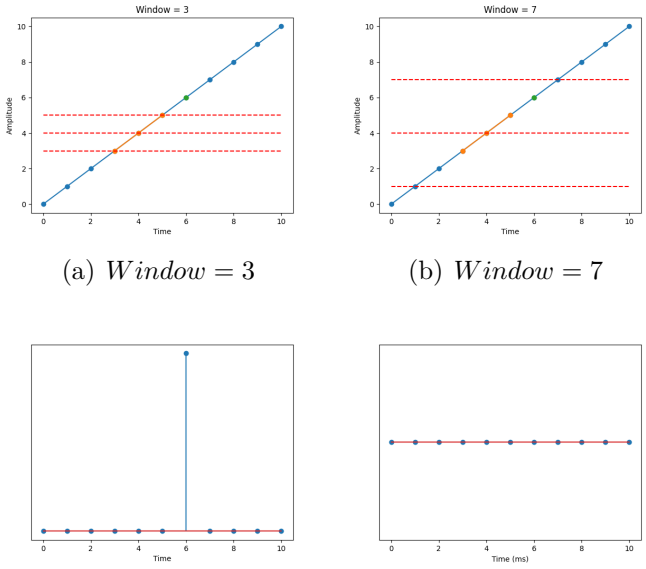
entire duration of the signal, in the case of the MW the reference position depends on the instant of the signal to be encoded, while the amplitude derive from the $Windows$ parameter, that define how many how many instants before and after compared to the one to be encoded and the $Base$ with in a $Window$ indicating the central point of the range.

Regarding the size of the window, we can use the expansion series of sine truncated to first order, to understand the logic behind the definition of the amplitude of the $Window$, since according to the Fourier decomposition, the signal is obtained as a linear combination of sinusoidal signals:

$$f(x) = \sin x \approx x + O(x) \tag{3.11}$$

as you can see $f(x)$ is a monotonically increasing function, this means that an increase in the signal appears to be present in the immediately following instants, consequently to capture even small variations it is sufficient to fix the value of the $Window$ between 3 and 4.

By exploiting this property, a sort of filtering of the signal from noise is also obtained in this case.



(a) $Window = 3$ (b) $Window = 7$

(c) Spike train encoding (d) Spike train encoding

Figure 3.18: MW encoding parameters comparisons

In fig. 3.18 (a)-(b) the construction of the threshold of values is obtained in the following way: the central line represents the expected value of the signal in the dimension of the considered window, while the ranges are obtained through the sum and difference between the mean and the $Threshold$. From here it can be deduced

that if the parameter $Window$ is sufficiently small and the level of signal exceeds the range described previously, a spike is produced fig. 3.18(c); on the contrary if the window is too large, the data for signals whose variation is too small will always be contained in the range and this prevents correct conversion fig. 3.18(d), from this example we can conclude that also in this case it is possible to obtain a noise suppression effect, based on a careful choice of the parameter value, which in this case is represented by the $Window$ parameter. In order to obtain the desired noise suppression, an optimization process is carried out on the $Window$ starting from very low values such as 3 and progressively increases to the desired level.

Step-Forward (SF)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $Jump \leftarrow max(Sample[c]) - min(Sample[c])$
  $Threshold \leftarrow mean(Jump)/factor$
  $SpikeTime \leftarrow \infty$
  $Base \leftarrow Sample[c][0]$
  for $i = 0 : L$ **do**
    if $Sample[c][i] > Base + Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow 1$
    else if $Sample[c][i] < Base - Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow -1$
    end if
  end for
end for

The SF [43] algorithm also known as Delta Encoding, has a completely different coding logic than the TBR or MW, in which the implementation procedure is inspired by digital analog conversion techniques and vice versa.

Applying this coding method for each channel individually, the basic idea for the realization of such encoding is based on the reconstruction of the signal through a step function whose jump is defined through the maximum excursion of the signal in terms of amplitude, an example of the reconstruction is shown in fig. 3.19:

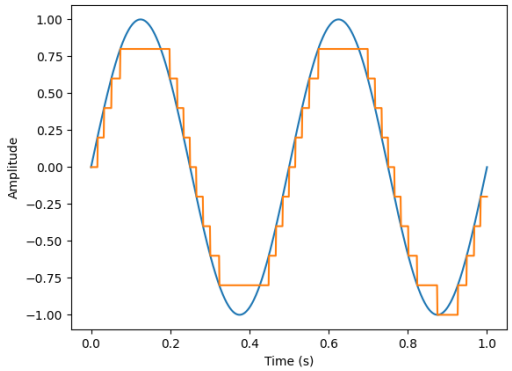


Figure 3.19: Example of reconstruction (orange curve) of the sinusoidal signal (blue curve) through the SF algorithm

The procedure that allows you to define the spikes associated with the signal to be coded are described below:

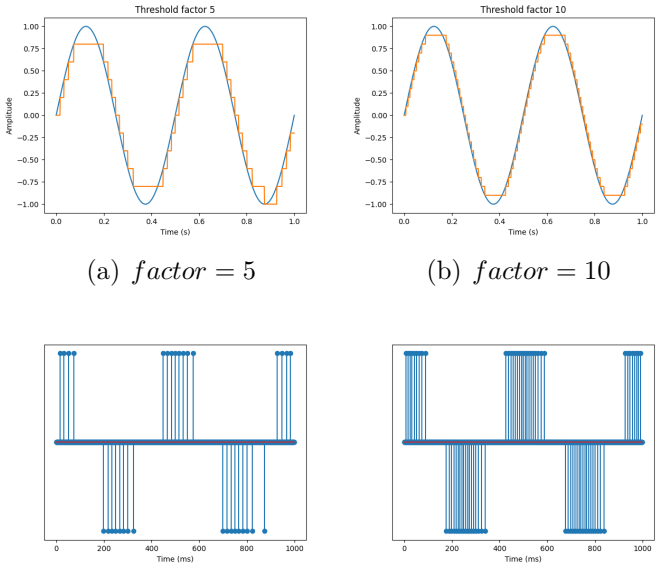
1. the first step is to define the amplitude of the step between one instant and the next used to reconstruct the signal, that is defined by the $Threshold$ parameter, as function of the maximum excursion of the signal known as $Jump$ and the parameter $factor$ whose value can be equal to 5 or 10 or greater depending on the number of spikes we want, the greater is this $factor$ and the greater is the number of spikes;

$$\begin{aligned} Jump &= max(Sample[c]) - min(Sample[c]) \\ Threshold &= mean(Jump)/factor \end{aligned} \tag{3.12}$$

2. in the second step the value of $Base$ is defined, which essentially represents the value of the reconstruction signal at a certain instant based on the value of the original signal, whose initial value coincides with the value at time $t = 0$ of the signal to be encoded;

3. in the following steps the value of the original signal is compared with the reconstructed signal value $Base$ and the value of $Threshold$, defining an upper threshold is lower. The emission of a spike is obtained if the value of the original signal differs by the value of $Base$ with respect to the $Threshold$ in a positive or negative sense.

Examples of SF encoding for different values in fig. 3.20



(a) $factor = 5$ (b) $factor = 10$

(c) Spike train encoding (d) Spike train encoding

Figure 3.20: SF encoding parameters comparisons

Pulse-Frequency Modulation (PFM)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $Jump \leftarrow max(Sample[c]) - min(Sample[c])$
  $Threshold = mean(Jump)/factor$
  $SpikeTime \leftarrow \infty$
  for $i = 0 : L$ **do**
    if $Sample[c][i] > Threshold$ and $SpikeTime > \tau_{refra}$ **then**
      $SpikeTrain[c][i] \leftarrow 1$
    end if
  end for
end for

The PFM [44] [45] [46] encoding algorithm appears to have an implementation logic very close to that of the SF, however carrying out a further pre-processing operation which in this case consists in eliminating the negative half-wave of the signal, exploiting the property of the Fourier decomposition stating that information for a sinusoidal signal can be extracted by considering only the positive half wave [15]:

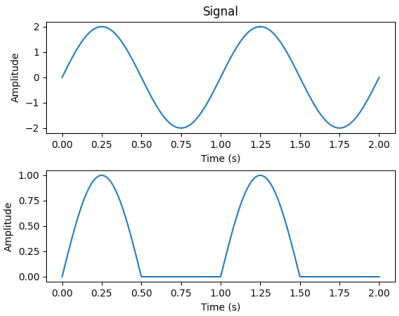
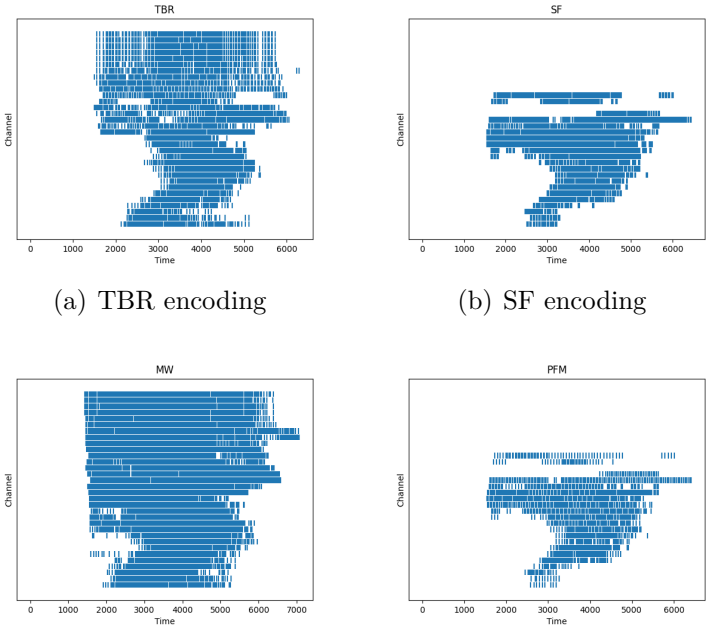


Figure 3.21: PFM pre-processing step

unlike SF, the spike emission occurs if the value of the original signal is greater than the *Threshold* value.

Conclusions

As regards data belonging to the temporal data class, this class of algorithms are ideal for describing time-varying signals, as they are able to encode the variations of the signal, allowing to reduce the number of spikes to encode signals that do not contain relevant information, such as small variation.



(a) TBR encoding

(b) SF encoding

(c) MW encoding

(d) PFM encoding

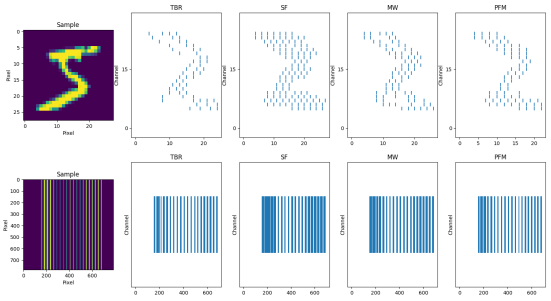
Figure 3.22: Representation of the same sample with different encoding algorithms

This fig. 3.22 shows an example of a spike train, for which the sample time is shown on the x axis, while the channels obtained as pre-processing of the filter bank

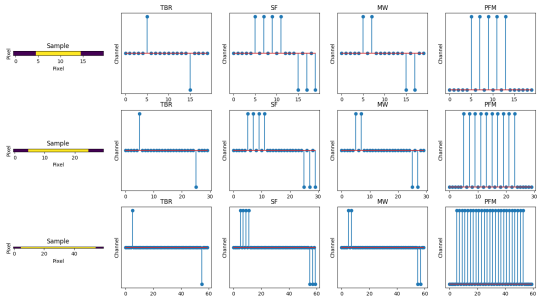
are shown on the y axis, obtaining an increase in frequency moving away from the x axis, while the blue segments indicate the spikes that are produced.

On the contrary in the case of spatial data given the use of such algorithms, it has several disadvantages:

- the dimensionality with which the data can be treated is not clear, whether in an original 2D structure or in a 1D structure;
- it is not useful to decompose the 1D signal into channels since there is no sampling frequency and there are large variations, such as the presence of steps;
- for images that have large invariant sections, in most cases no conversion is performed in those sections, since such classes of algorithms are designed to capture variations in signals;
- loss of spatial correlation in the converted data.



(a) MNIST Sample



(b) Image with invariant central section

Figure 3.23: The first image is an example of MNIST sample, while in the second row the same sample is treated but in a linear array form, in the second figure instead is generated an image with invariant central section at different length

Filter & Optimizer

It describes a different class of encoding algorithms which exploits the use of a convolution-like operation through the use of a filter to determine spikes. The basic idea used for the emission of a spike is through the use of the comparison of the area obtained from the convolution operation and a certain threshold value, consequently a major disadvantage is linked to the fact that it is necessary to have a priori knowledge of the signal, in order to determine the structure of the filter to be used, that allow to obtain a signal coding, as selecting an unsuitable filter could lead to the non-emission of any spike, or an emission for each instant, obtaining a coding without information.

Part of this class are the algorithm Hough Spiker Algorithm (HSA), Modified Hough Spiker Algorithm (MHSA) and Ben's Spiker Algorithm (BSA).

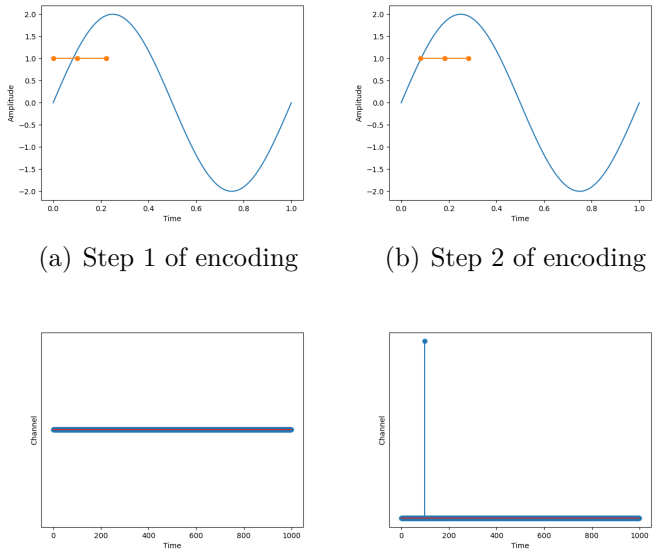
Hough Spiker Algorithm (HSA)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$LF \leftarrow length(Filter)$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $SpikeTime \leftarrow \infty$
  for $i = 0 : L$ **do**
    $Counter \leftarrow 0$
    $SpikeTime+ = 1$
    for $f = 0 : LF$ **do**
      if $i + f < L$ and $Sample[c][i + f] \geq Filter[f]$ **then**
        $Counter+ = 1$
      else
        break
      end if
    end for
    if $Counter == LF$ **then**
      for $f = 0 : LF$ **do**
        if $i + f < L$ **then**
          $Sample[c][i + f] \leftarrow Sample[c][i + f] - Filter[f]$
        end if
      end for
      if $SpikeTime > \tau_{refra}$ **then**
        $SpikeTime \leftarrow 0$
      end if
    end if
  end for
end for

In the case of HSA algorithm [47], the quantities that must be fixed to define the quantities used during the encoding are type and structure of the filter. By type of filter we mean the structure, which can be assumed as for example boxcar, cosine, hamming, gaussian and so on, while by structure we mean its width and amplitude, these parameters are extremely important to obtain a correct encoding, which must be fixed in relation to the signal to be analyzed.

The choice of these parameters must be very careful, because in order for a spike to be emitted, the filter area must be completely included in the subtended area of the signal to be encoded.

Finally, since the filter can assume only a positive or negative value, only part of the signal can be coded, but this does not represent a problem as by combining two conversion steps, one with a positive filter and one with a negative filter, both parts can be encoded.



(a) Step 1 of encoding (b) Step 2 of encoding

(c) Spike train (d) Spike train

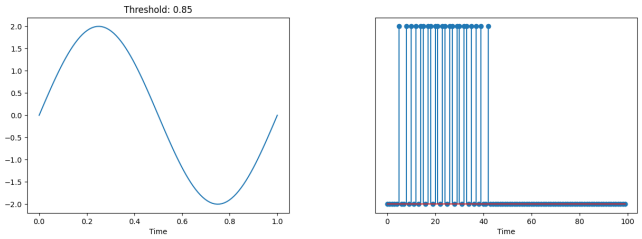
Figure 3.24: HSA example encoding procedure

In the fig. 3.24 two conversion steps are represented, where in the first case, in fig. 3.24(a), part of the filter is above the signal, and this violates the first condition defined previously, while in the second fig. 3.24(b) both the are respected and so we have the emission of a spike fig. 3.24(d).

Modified Hough Spiker Algorithm (MHSA)

```
Channels ← length(Sample)
L ← length(Sample[0])
LF ← length(Filter)
SpikeTrain ← zeros(Channels, L)
for c = 0 : Channels do
    SpikeTime ← ∞
    for i = 0 : L do
        Error ← 0
        SpikeTime+ = 1
        for f = 0 : LF do
            if i + f < L and Sample[c][i + f] < Filter[f] then
                Error = Error + Filter[c] − Sample[c][i + f]
            end if
        end for
        if Error ≤ Threshold then
            for f = 0 : LF do
                if i + f < L then
                    Sample[c][i + f] ← Sample[c][i + f] − Filter[f]
                end if
            end for
            if SpikeTime > τ_refra then
                SpikeTime ← 0
            end if
        end if
    end for
end for
```

MHSA [48] which uses the same logic of the HSA, however to obtain the emission of a spike, it is sufficient that the convolution-like area is greater than a threshold value defined by the user. One of the problems related to this algorithm is the non-emission of spikes for too small $Threshold$ values or an emission at every instant for too large $Thresholds$. An example of coding using the MHSA algorithm is in fig. 3.25



(a) $Threshold = 0.85$ (b) Spike train

Figure 3.25: MHSA encoding example

Ben's Spiker Algorithm (BSA)

$Channels \leftarrow length(Sample)$
$L \leftarrow length(Sample[0])$
$LF \leftarrow length(Filter)$
$SpikeTrain \leftarrow zeros(Channels, L)$
for $c = 0 : Channels$ **do**
  $SpikeTime \leftarrow \infty$
  for $i = 0 : L - LF + 1$ **do**
    $Error1 \leftarrow 0$
    $Error2 \leftarrow 0$
    $SpikeTime+ = 1$
    for $f = 0 : LF$ **do**
      $Error1 \leftarrow Error1 + abs(Sample[c][i + f] - Filter[f])$
      $Error2 \leftarrow Error2 + abs(Sample[c][i + f])$
    end for
    if $Error1 \leq Error2 * Threshold$ **then**
      for $f = 1 : LF$ **do**
        if $i + f + 1 < LF$ **then**
          $Sample[c][i + f] - = Filters[c]$
        end if
      end for
      if $SpikeTime > \tau_{refra}$ **then**
        $SpikeTime \leftarrow 0$
      end if
    end if
  end for
end for

Although the considerations made in the description of the HSA algorithm, for the definition of the filter properties are the same, contrary to the previous algorithms, in the BSA [48] coding technique the production of the spike is linked

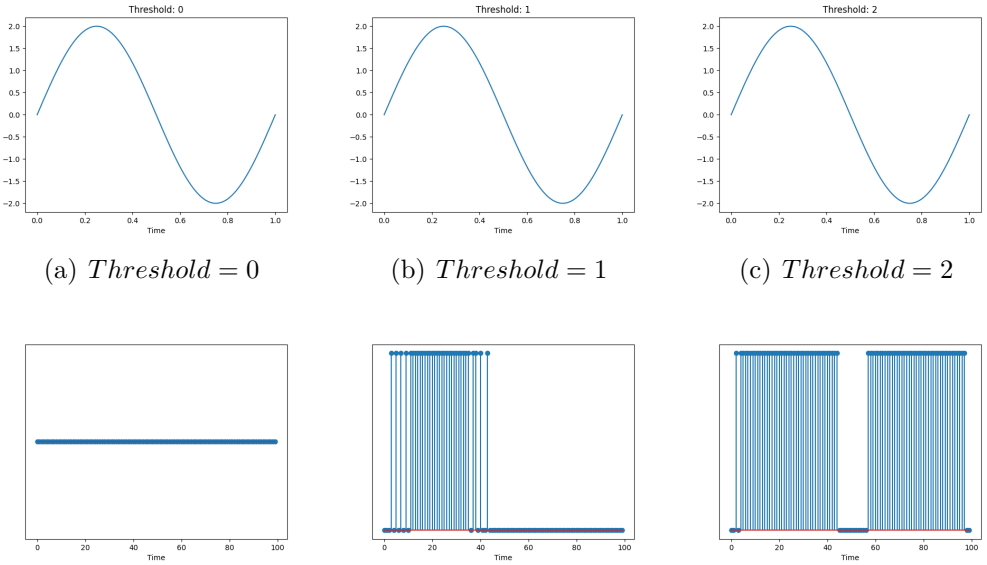
to the comparison of two parameters named $Error1$ and $Error2$, through quantity $Threshold$. Individually describing the role of the three parameters:

- $Threshold$: represents a degree of freedom that can be set by the user in order to obtain an optimal number of spikes for encoding;
- $Error1$: is linked to a sort of cumulative function of the signal to be coded;
- $Error2$: it represents more or less the same quantity as $Error1$, except for the link with the filter used for the encoding

Once these quantities have been evaluated, a spike is emitted if the following relation is satisfied:

$$Error1 = Error2 \cdot Threshold \tag{3.13}$$

As in the previous cases, the THreshold parameter is defined through an optimization process according to the analyzed signal. In fig. 3.26 some cases used to identify the most suitable value have been reported. Starting from an initial value of $Threshold = 0$ and this value has been progressively increased until the correct encoding of the positive half-wave is identified (having chosen a suitable filter for this purpose) reaching a value of $Threshold = 1$, further increasing the continuous emission of spikes, indicating that this condition does not allow the transport of information



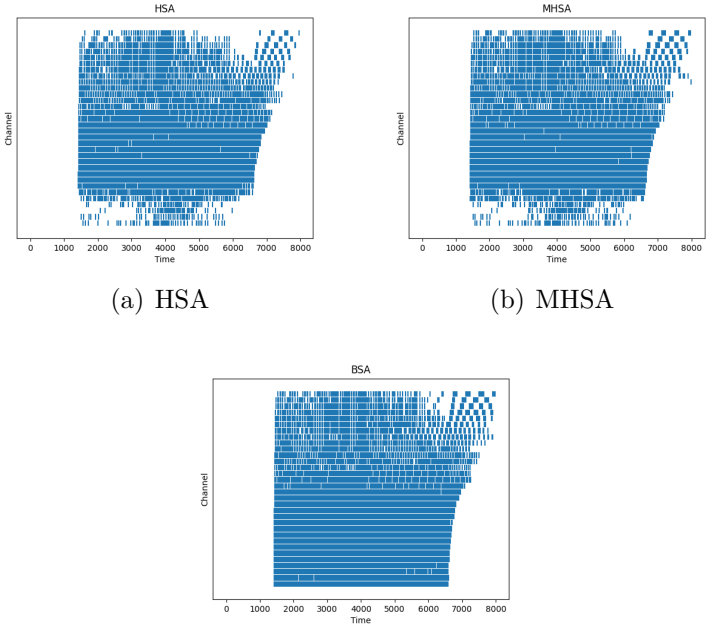
(a) $Threshold = 0$ (b) $Threshold = 1$ (c) $Threshold = 2$

(d) Spike train (e) Spike train (f) Spike train

Figure 3.26: BSA encoding parameters comparisons

Conclusion

In the case of temporal data, these types of algorithms require a more accurate evaluation of the parameters that allow the encoding, but are still able to encode the time-variant signal, although with a much greater number of spikes than the algorithms of the previous class.



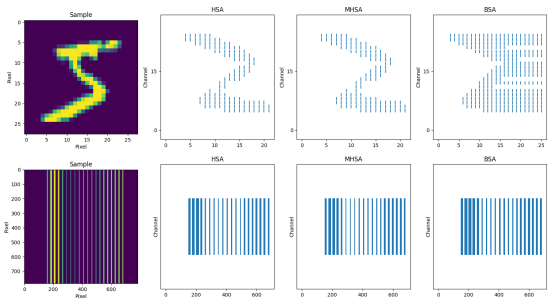
(a) HSA

(b) MHSA

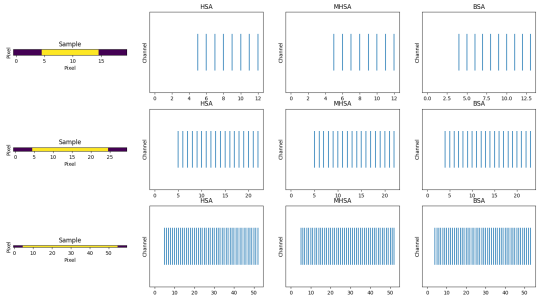
(c) BSA

Figure 3.27: Representation of the same sample with different encoding algorithms

In the case of spatial data these types of algorithms present more or less the same problems discussed in the precedent section, however there is a better conversion for data that have invariant sections, through a careful choice of the filter fig. 3.28(b).



(a) MNIST Sample



(b) Image with invariant central section

Figure 3.28: The first image is an example of MNIST sample, while in the second row the same sample is treated but in a linear array form, in the second is generated an image with invariant central section at different length

Global Referenced

The following coding class, appears to have a completely different logic from those seen so far, defining the spike train as a sort of binary code; this class includes Phase encoding (PHASE) and Time-to-first spike (TTFS) encoding.

Phase encoding (PHASE)

This coding method draws inspiration from experimental observations conducted on the auditory cortex of animals subjected to natural sounds, which could produce a state of alarm in the subjects [49] [50]. What has been observed is that the production of the spike trains turns out to be connected to the phase of the audio signal, generating a sort of binary code associated with each specific value of phase.

Using this logic we can implement an algorithmic version of this process through the following steps:

1. first of all it is necessary to carry out some pre-processing steps, allowing to develop the following. The two main operations are the elimination of the negative half-wave and subsequently the normalization of the signal to be encoded;

2. then we move on to the definition of the phase of the signal, associated with each instant through the trigonometric function:

$$\phi = \arcsin\left(f(t)\right) \tag{3.14}$$

3. finally using a process very similar to the analog-digital conversion, we identify the number of spikes that we want to use for the conversion, and we proceed with the subdivision into levels as in quantization step, associating a code to each level, in the example below the code used are 00, 01, 10, 11.

The graphical representation of the previously mentioned steps are shown in fig. 3.29

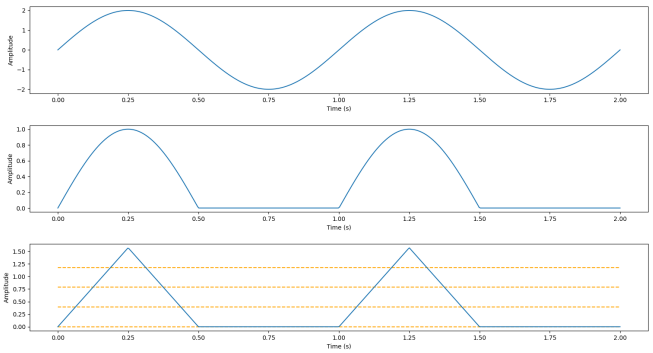


Figure 3.29: Phase encoding step

Unfortunately by applying this method to each point of the original signal, the coded signal turns out to have a greater length than the original, to avoid this the signal is divided into small segments of the length of the code, whose representative value is obtained from the average, as in the sampling process.

Time-to-first spike (TTFS)

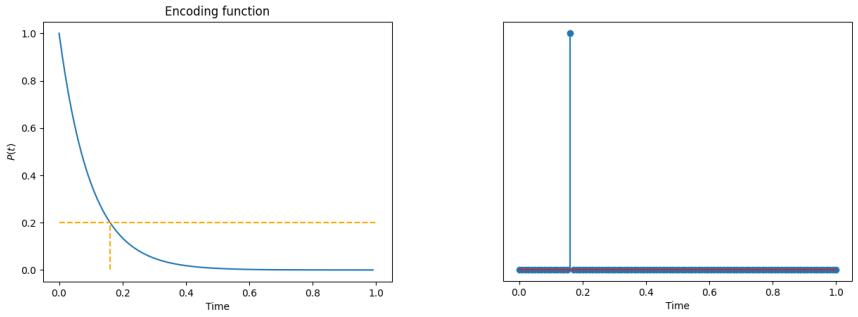
In TTFS [51], the information is encoded through a single spike, whose delay of the production of spike respect to the stimulus is related to the intensity of the signal, greater is the signal smaller is the delay, the opposite behaviour occur for small signal. Several mapping functions can be used to carry out this encoding. The one used in this case is defined by an exponential function:

$$P(t) = \theta_0 e^{-t/\tau_{th}} \tag{3.15}$$

Analyzing the function we have that θ_0 represents a normalization variable generically set to one $\theta_0 = 1$; while the τ_{th} parameter is related to the maximum spike time for very weak signals; t identify the time of the spike. Assuming to normalize the data and reinterpreting this value as the probability distribution $P(t)$ whose value is between $0 \leq P(t) \leq 1$, by inverting eq. 3.15, we obtain that t is equal to:

$$t = \tau_{th} \cdot \log\left(\frac{\theta_0}{P(t)}\right) \tag{3.16}$$

supposing that the value to be encoded is $x = 0.2$ assuming that $P(t) = x$, we can extract the spike time from eq. 3.16 is around $t = 0.16$, with $\tau_{th} = 0.1$ and $\theta_0 = 1$. Graphically represented we have in fig. 3.30 the spike time:



(a) This curve represents the function that allows the intensity of the signal to be encoded to be passed to the spike time

(b) Spike train

Figure 3.30: TTFS encoding example

Using an approach very similar to the previous case in the case of pre-processing step, in order to encode the data, the intensity curve of the signal/spike time is

divided into regular intervals, allowing to define the position of the spike in the code used for encode the information, in which the number spike that you want to use for encoding correspond to the number of interval used for the division of the curve, in this particular example in fig. 3.31 the spike time is divided into 5 ranges.

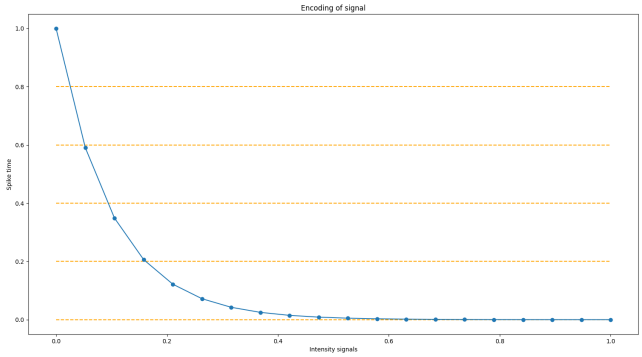


Figure 3.31: Encoding distribution

In this case for signals of great intensity for an interval between 0.2 and 1.0 the corresponding spike code is 10000, for 0.1, 0.2 it is 01000 and so on up to 00000. An important aspect of this coding method is that one spike is used at a time to encode the information. However, this method presents the same problems as PHASE encoding regarding the size of the converted data, for this reason if you want to keep the same size, it is necessary to divide the signal into segments, also in this case calculate the average value as reference to the segment and proceed with the conversion.

Latency/ISI

This type of coding allows to encode the information through the use of two elements, the amount of spikes and the time interval between the spikes, thus defining the BURST encoding method.

Burst encoding (BURST)

Also in the case of BURST encoding [51], the first step is to set the number of spikes used as code, but next to this parameter it is also necessary to set the maximum number N_{max} of spikes produced, and the minimum t_{min} and maximum ISI interval t_{max}. Also in this case the same pre-processing approach defined for PHASE and TTFS encoding is applied. The steps for generating the coding sequence are described in 2 phases:

1. the number of spikes is obtained through the product of the signal intensity that thank to the pre-processing step is included in 0,1 interval and the

maximum number of spikes:

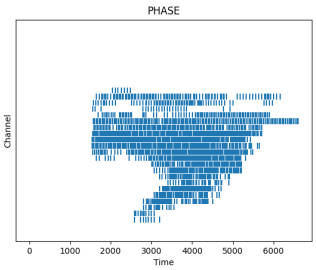
$$SpikeNumber = \lceil rate \cdot N_{max} \rceil \tag{3.17}$$

2. for the definition of the ISI it is necessary to divide two cases:

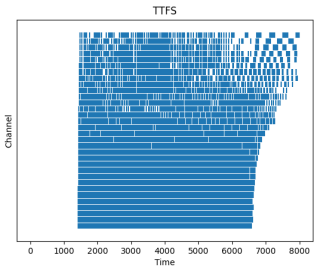
$$ISI = \begin{cases} \lceil t_{max} - rate(t_{max} - t_{min}) \rceil & SpikeNumber > 1 \\ t_{max} & Otherwise \end{cases} \tag{3.18}$$

Conclusion In order to comment the encoding performances, the Global Referenced and Latency/ISI classes are considered together, due to the affinity in the pre-processing of time-varying data, by carrying out a subdivision into bins, and the coding that can be interpreted as a binary code.

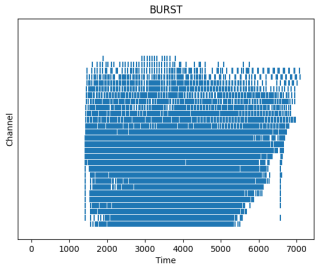
The use of Global Referenced-Latency/ISI encoding methods in the Temporal data, have greater disadvantages with respect to the other techniques, since in this case as a series of pre-processing operations are required to have a consistent encoding between the raw data and the spike code, that produce a data loss directly proportional to the resolution desired, or if you want to avoid data loss, you will get a encoded data whose size is much larger than the original.



(a) PHASE



(b) TTFS



(c) BURST

Figure 3.32: Representation of the same sample with different encoding algorithms

Unlike the previous types of data in the case of Spatial data, this type of encoding is much more suitable, as it allows to encode without loss of information the data treated.

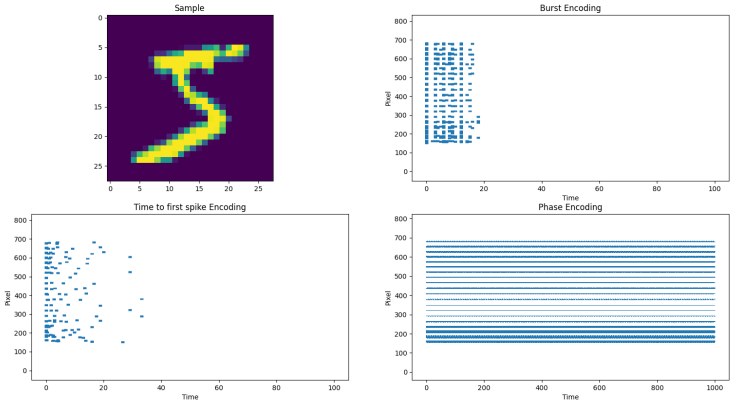


Figure 3.33: Encoding comparison in Spatial Data

Correlation & Synchrony

Correlation and Synchrony is a completely different logic of encoding structure respect the types already seen. In this case the information is elaborated in a way to have a sparse representation, allowing to reduce the amount of spike. Therefore the technique that is part this method is Sparse Distributed Representation (SDR).

Sparse Distributed Representation (SDR)

This type of encoding, described in the work of Zihan Pan et al. [52], tries to translate the functioning of the human ear cochlea as faithfully as possible, through the use of two main elements the spectrogram and the masking effect. The principal step implemented to produce this encoding are:

1. spectrogram: the spectrogram is produced using the classic procedure, dividing the signal to be analyzed in regular intervals, and then the filter decomposition is applied, this allow to extrapolate the intensity of the signal at given time. In order to use the masking effect, it's necessary to calculate the intensity of the input audio in dB obtained thanks to eq. 3.19:

$$I = 10 \cdot log_{10}\left(\sum_{n=1}^{l} x_i^2\right) \tag{3.19}$$

in which n indicate the length for the interval;

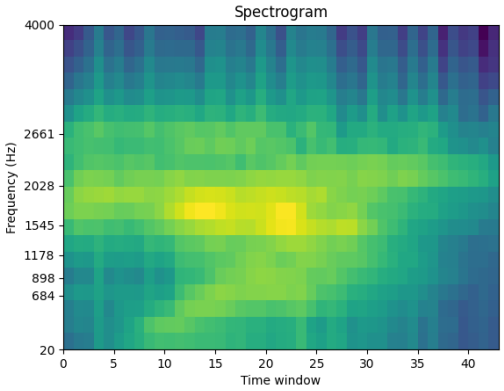


Figure 3.34: Spectrogram obtained from one sample of the FSD dataset

2. masking: the masking effect is the result of modeling the phenomenon of the human ear's inability to distinguish signals in the presence of loud noises. The effect of the masking appears to be due to two contributions:

    - Simultaneous Masking: this masking derives from studies conducted on the acoustic spectrum which allows to define the minimum threshold in dB perceivable by the human ear that produces a stimulus, described from function 3.20, where define the threshold in function of frequency, that allow the emission of spike or not:

$$T(f) = 3.64\left(\frac{f}{1000}\right)^{-0.8} - 6.5e^{-0.6\left(\frac{f}{1000}-3.3\right)^2} + 0.001\left(\frac{f}{1000}\right)^4 \quad (3.20)$$

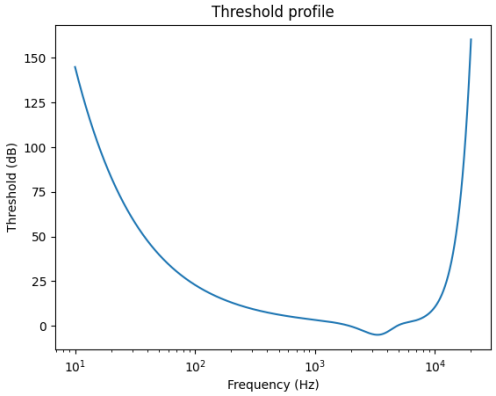


Figure 3.35: Threshold Profile used to generate the mask in the case of Simultaneous Masking

From the spectrogram and the threshold equation, a Spatial masking is generated, a matrix of the same size as the spectrogram composed of 0 and 1, defined according to the conditions 3.21:

$$S_{th}(t,f) = \begin{cases} 1 & Spectrogram(t,f) > T(f) \\ 0 & Otherwise \end{cases} \tag{3.21}$$

An example of a spatial map obtained from eq. 3.21 is shown in the fig. 3.36:

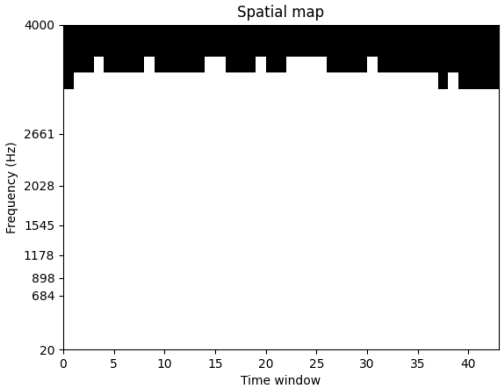


Figure 3.36: Spatial map generated by the comparisons between the spectrogram and the threshold function

the white point correspond to the site in which the intensity is higher than the threshold, allowing the emission of the spike, in the black point the opposite behavior occurs.

- Temporal Masking: have more or less the same role of the refractory period described in the section 2.1.2, although in this case the interval in which the spike isn't emitted is also related to the intensity of the stimulus, defined to by eq. 3.22:

$$T_{refra}(c,t) = e^{-p(c)t} \tag{3.22}$$

the parameter $T_{refra}(c,t)$ define the threshold level based two parameter: $p(c)$ defined by the curve in fig. 3.37 that identify the constant factor related to the channel frequency and the time of the signals. Obviously as it can be seen from the eq. 3.22, the value of this threshold decreases in time, and when a stimulus with lower intensity arrived, this type of information are suppress and no spike are emitted.

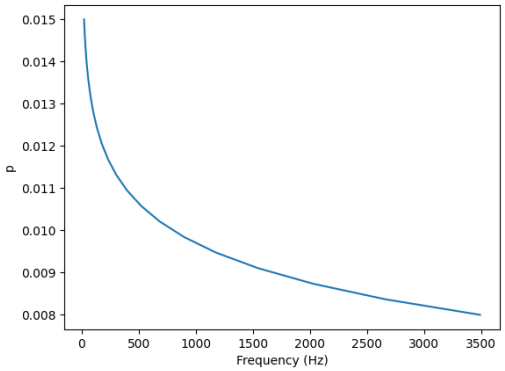


Figure 3.37: Spatial map generated by the comparisons between the spectrogram and the threshold function

An example of generation of the temporal map is described by the following step: each row of the spectrogram is considered individually, starting from time 0, the first spike is always emitted and the threshold level is generated $T_{refra}(c,0)$, if the intensity of the signals in the next time are bigger that $T_{refra}(c,0)$ a spike can be produced by setting the map value to 1 otherwise it is fixed to 0, and so on, as described by the condition below:

$$T_{refra}(c,f) = \begin{cases} 1 & t = 0 \\ 1 & Spectrogram(t,f) > T_{refra}(c,t-1) \\ 0 & Otherwise \end{cases} \tag{3.23}$$

An example of a temporal map is shown in the fig. 3.38:

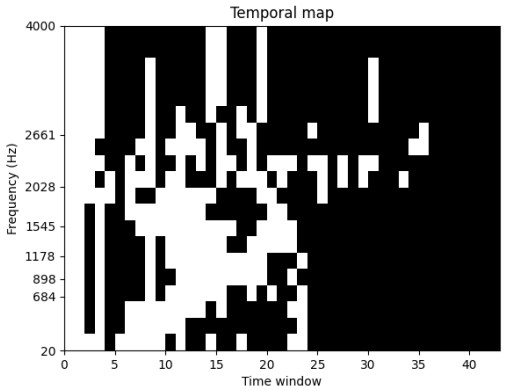


Figure 3.38: Temporal map generated by the comparisons between the spectrogram and the threshold function

The combination of this two map allow to generate the complete map (fig. 3.39) masking defined of the dot product between the spatial and the temporal map:

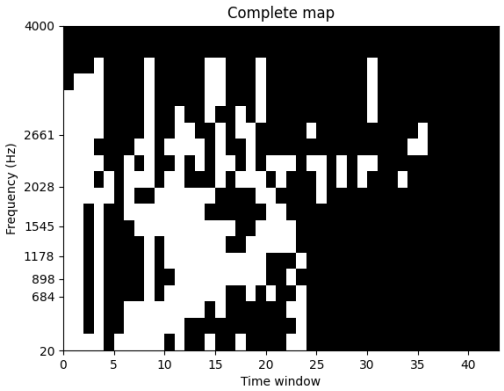


Figure 3.39: Complete map generated by the dot product between the Spatial map and the Temporal map

Finally in order to obtain the spectrogram that is used for the encoding, the original spectrogram is combined through the dot product between the two.An example based on what has been said so far is reported in fig. 3.40

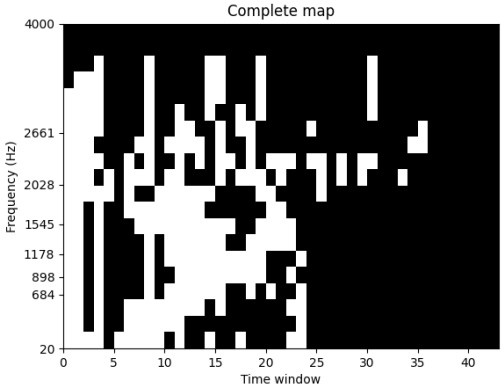


Figure 3.40: Complete map generated by the dot product between the Spatial map and the Temporal map

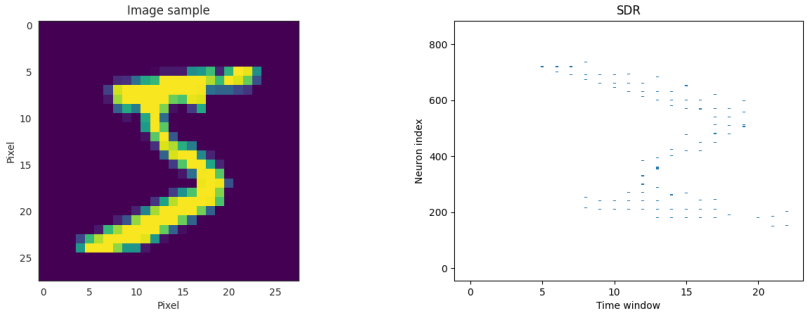
3. encoding: the encoding step is achieved by applying an ensemble of neurons with different threshold level. The neuron that emit the spike is the neuron with the threshold level closest to the intensity of the spectrogram. An example obtained with 30 neurons is in fig. 3.41.

Figure 3.41: Example of SDR encoding performed with 30 neurons per channel

Conclusion

The use of this encoding technique immediately presents a great advantage regarding the number of spikes generated, as being very small it is possible to analyze very complex signals with great variability, with a reduced number of spikes, against the use of this This technique also has a disadvantage in terms of information loss related to the generation of the spectrogram in the case of temporal data as shown in fig. 3.41.

However, the use of this technique in the case of spatial data in the contrary turns out to be more suitable as the spectrogram is identified by the image itself.



(a) MNIST Sample (b) SDR encoding

Figure 3.42: Example of SDR encoding with 30 neurons per channel in the case of MNIST

3.4 Datasets

The datasets selected to verify the impact of encoding techniques in time-varying signal, was defined based on the range of frequency to be investigated. Referring to the subdivision of the audio spectrum for the audible frequency range of the human ear, four ranges can be identified: Very-Low frequency starting virtually from 0Hz up to 20Hz, Low frequency from 20Hz to 500Hz, Middle frequency from 500Hz and 2000Hz and High frequency from 2kHz to 20kHz. By focusing on the Very-Low frequency and Middle frequency ranges, the selected datasets are classified based to the maximum frequency that can be resolved defined through the Nyquist-Shannon sampling theorem, corresponding to half of the sampling frequency $f_{Max} = f_s/2$. The two datasets that are suitable for this purpose are the Wireless Sensor Data Mining (WISDM) dataset and the Free Spoken Digit (FSD) dataset.

3.4.1 The Free Spoken Digit Dataset (FSD)

The Free Spoken Digit Dataset (FSD) [53] represents the contribution of several users in the collection of audio samples of the spoken digits with English pronunciation, in different accent, from speakers coming around the world. The first version was published in the 2017 by Zohar Jackson et al, the version used in this work was released in of the dataset count a contribution of 6 speakers, each of them providing 50 audio sample per digits, obtaining 3,000 samples in all, each recorded in mono mode with sample-rate of $f_s = 8kHz$ in wav format. During the analysis carried out in this work, however, only the portion of the FSD dataset relating to the audio samples produced by the Jackson speaker was selected, which count in total 500 samples, divided into training sets and test sets according to the percentages 90% and 10%. Analyzing the data contained in the dataset, it can be seen that the duration of all samples is variable and in any case less than $1s$. In order to make sure that the structure of each sample is uniform, a standardization process is performed inserting at the top and at the end of the audio signal intervals of silence in order to make each sample with time duration exactly $t_{sample} = 1s$. An example of audio sample is shown in fig. 3.43.

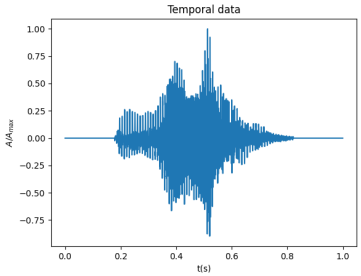


Figure 3.43: FSD audio sample, standardization process referred to the time duration

3.4.2 Wireless Sensor Data Mining dataset (WISDM)

The Wireless Sensor Data Mining dataset (WISDM) [6] [54] represents a dataset obtained from the collection of 18 human activities, carried out by 51 subjects, using devices such as smartphones and smartwatches through the accelerometer and gyroscope for a total duration of 3 minutes. Each sample is composed of 6 signals: 3 belonging to the accelerometer, 3 to the gyroscope, collected with a sampling frequency of $f_s = 20Hz$. In order to observe the effects of the encoding techniques based on the frequency of the analyzed sample, two subsets were selected, composed of the activities shown in the table 3.1.

subset1	subset2
Walking	Brushing Teeth
Jogging	Eating Soup
Brushing Teeth	Eating Chips
Eating Soup	Eating Pasta
Dribblinlg (Basketball)	Drinking from Cup
Clapping	Eating Sandwich
	Kicking (Soccer Ball)

Table 3.1: This table shows the classes associated with subset1 and subset2, isolated from the WISDM dataset

Furthermore, as in the case of the FSD, also in this case a standardization step of the samples in the dataset was performed, inserting silence intervals at the top and at the end of the sample obtaining samples with a length of exactly $t_{sample} = 240s$. An example of sample is shown in fig. 3.44.

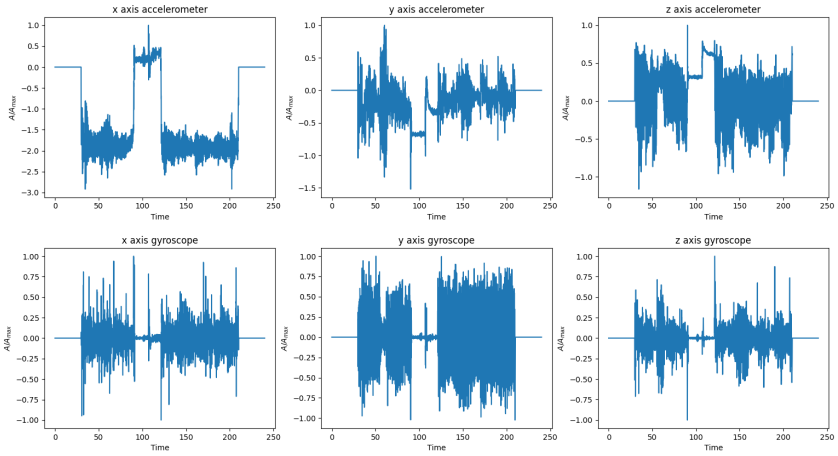


Figure 3.44: WISDM sample

3.5 Learning process

The learning process is a key step in the use of neural networks, thanks to which the network is able to automatically extrapolate the most suitable model for the description of the analyzed data. To date, unfortunately, no established learining methods such as the gradient descent for ANNs suitable for SNN have yet been identified, therefore there are numerous methods that can be used. Comparing now the tasks to be developed in the field of the neuromorphic state of the art, a particular method has been identified that makes the learning process very suitable for a traditional ANN approach called Transfer Learing, consider the work developed by Shih-Chii Liu et al. [55] [56] [44] and Juan P. Dominguez-Morales et al. [57], which analyzes the classification of audio signals after the encoding process. In these works, in order to apply the learning process on the SNN, the learning process is performed by training an artificial neural network (ANN) and then employing the resulting weights for a spiking counterpart. Specifically, in this work, this procedure was adopted to pass from a traditional convolutional neural network (CNN) to a spiking CNN. However, using this method requires an additional step to adapt the spike-encoded data to the CNN architecture, and this is achieved through the production of a sonogram.

3.5.1 Sonogram

The sonogram is obtained with a feature extraction carried out by separating the encoded sample into temporal frames, thus employing the Binning process. There are several Binning processes that can be employed, described in the work developed by Shih-Chii Liu et al. [56]; it has been shown that the Time-Binned Spike Count features (TBSC) allows to reach a better accuracy about 96%.

The TBSC process is carried out by first defining the number of Bins that you want to use in order to represent the sonogram, then the data obtained through the encoding algorithms is divided into a number of bins equal to those chosen from the temporal point of view, and a Spike Count is performed for each channel and bins. An example of this procedure is shown in fig. 3.45:

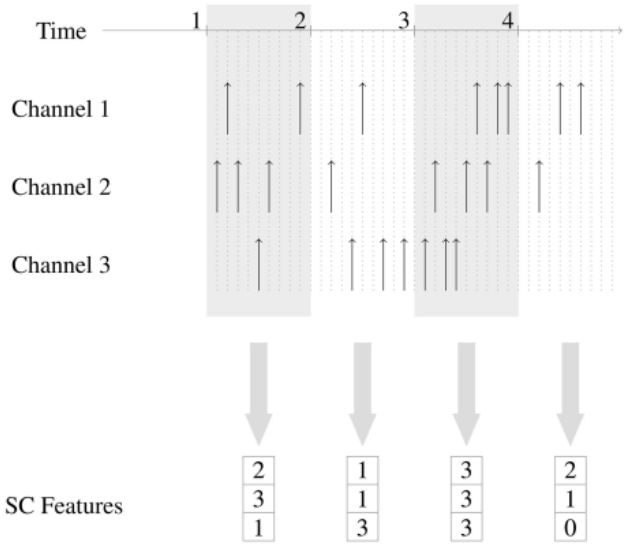
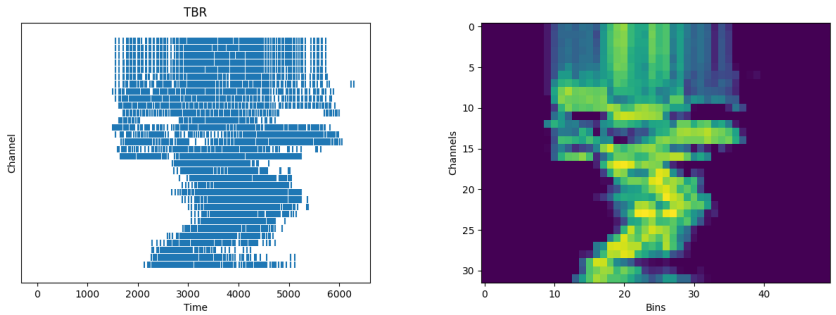


Figure 3.45: Binning procedure, adapted from [55]

The subdivision into bins and the definition of the number of spikes within a channel per bin are used to generate the sonogram. The sonogram is essentially an image in which the number of pixels along x axis correspond to the bins, the pixels along y axis are defined by the channels, while the intensity of the pixel is defined by the Spike Count procedure fig. 3.46, an example related to the FSD dataset.



(a) Sample encoded with TBR algorithm

(b) Sonogram generated from spike train

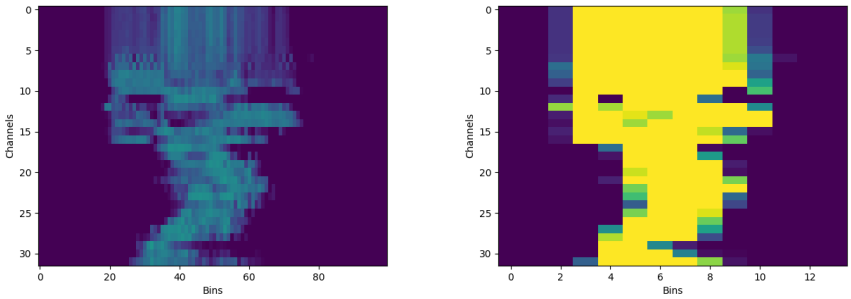
Figure 3.46: Feature extraction with Binning methods

In this process, the subdivision into bins is a very important step as it is directly linked to the classification performance in the SNN, since the sonogram will be used in two pivotal steps:

1. during the learning process by CNN, that treats the sonogram in a classical way as an image, such as in the case of MNIST, Fashion MNIST;

2. during the classification process since the last step consists in a re-encoding of the sonogram by rate encoding in order to make the information suitable for SNN.

This therefore leads to the problem of defining the number of bins, since if this number is too large, the intensity of the pixels of the sonogram are too low, in which from the global point of view the resulting sonogram present more or less the same structure in all the positions, obtaining a uniform pattern, in the same way, if the number of bins is too large, the resulting sonogram has pixels with high intensity, also generating a uniform pattern in this case, fig. 3.47.



(a) (b)

Figure 3.47: This fig. shows an example of a number of bins that is too large (a), and too small (b) respect to the spike per channel

Consequently, it can be said that within the sonogram the information is contained in 2 main components:

- in the edge of the sonogram;

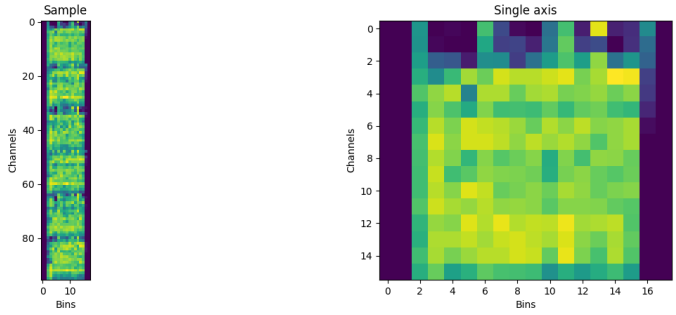
- the different intensity of the pixel in different region showing the richness of the information enclosed.

This process seems very laborious, but this has two big advantages:

- it allows to take advantage of the Transfer Learning process for which the SNN weights can be directly extracted from the corresponding CNN, exploiting the state of the art in terms of accuracy, hardware and software for the training of these networks;

- it allows to have a single evaluation criterion of the various encoding techniques by exploiting a single architecture and encoding process in input to the SNN.

In order to have a similar treatment between the FSD and WISDM samples, in the second case the axis signals are treated as if each signal were an independent signal generating a sonogram per axis, and finally a complete sonogram composed of the single sonograms is generated, fig.3.48.



(a) Sonogram for all signal (b) Single axis signal

Figure 3.48: In this image is reported the sonogram of the various axis of sample

3.6 Neural Network Architecture

As mentioned in the previous section, in order to apply the learning step in the case of SNNs, the transfer learning technique is used which uses the weights of an artificial neural network within an SNN architecture. The criterion by which the architecture to be used is chosen is defined by the type of data to be analyzed, which in this case is represented by the sonogram. Having therefore ascertained that the information within this data is encoded in the edge and in the internal structure of the image, a natural choice for the analysis of a dataset of this type is given by the Convolutional Neural Network (CNN).

3.6.1 Convolutional Neural Network (CNN)

Before being able to define the architecture of the CNNs used, it necessary to consider how this architecture will be used and which operations cannot be carried out in the case spiking counterpart, being necessary to use different Activation Functions and other tricks. By analyzing the various types of layers we have:

- Convolutional layer: referring to the classical structure the number of filters, the kernel size and the strides can be defined in a classical way through the

trial and error process, the only major differences are related to the bias parameter of the neurons which being absent in SNNs used in this work, so this quantity must be set equal to 0 for all neurons in the CNN. Furthermore it is necessary to use an Activation Function that allows to model the behavior of neurons in the CNN with a functioning analogous to those of the SNN. This function is a variant of the ReLU called in this work with the name of Modified ReLU (MReLU) developed in the work by Qian Liu et al. [58], which assumes the structure shown below:

$$y_i = \max\left\{0, \sum_{j=0}^{m} w_{ij}x_j\right\} S\tau_{syn} \tag{3.24}$$

y_i is the output of the neuron, $\sum_{j=0}^{m} w_{ij}x_j$ the summation term represents the weighted input of the neuron, finally the term $S = 201$ and $\tau_{syn} = 1/200Hz$ represents a multiplicative factor whose value is defined based on the parameters of the neuron model present in the SNN, that are related to the firing rate of the input layers in the SNN in accordance with the Rate Encoding procedure implemented in the sonogram step encoding.

- Pooling Layer: generally in the case of CNNs, a Max Pooling operation is used within the pooling layer, unfortunately the neurons inside the SNNs are not able to perform this kind of operation. However this does not represent a problem as it is possible to carry out an Average Pooling operation in the SNNs in a very simple way, by imposing the weight of the synapses equal to $1/N$ where N indicates the number of input synapses, to have a correct learning process in the CNN structure we select the Average Pooling layer;

- Fully connected Layer: in the classification layers it is necessary to use the same criterion as regards the biases. Regarding the activation function it is necessary to analyze two cases. If the fully connected layer is the last layer and since the task that we want treat is a classification problem the activation function that must be used is the Softmax function, instead if there is a inner fully connected layer the same consideration must be involved as in the convolutional layer using the MReLU.

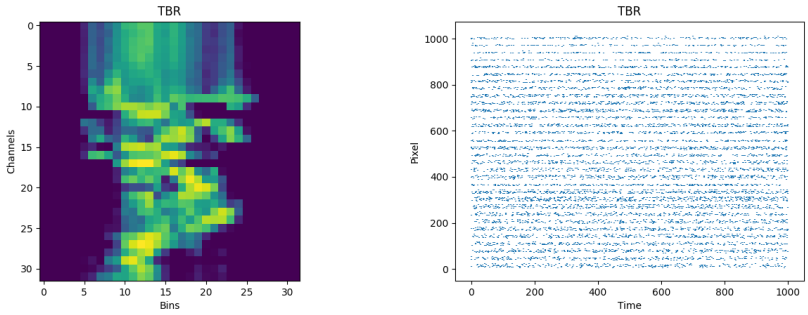
By applying these small changes in the CNN, this allows to perform classical training methods implementing the gradient descent process to extract the model from the dataset, but also this allow to use the weights as synaptic weights in the SNN twin.

3.6.2 Spiking Neural Network (SNN)

Once the CNN training has been carried out, we move on to the construction of the SNN, whose connection of the synapses must respect the structure of the CNN.

The available frameworks that can be used for the realization of this network are different such as NEURON, NEST, and Brian, however for an easier implementation in this work a Python API, called PyNN was employed.

In the case of SNNs the input of the network is always composed of the sonograms used during the training and testing of the CNN network, however in order to obtain a dataset composed of the sonograms, a further step of rate encoding is applied.

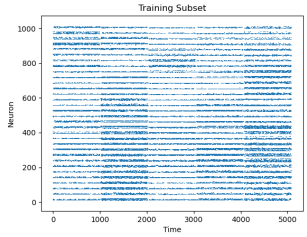


(a) Sonogram example (b) Rate encoding of sonogram

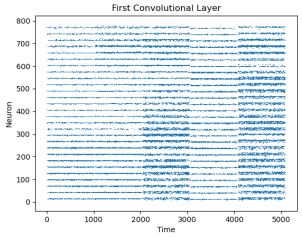
Figure 3.49: Rate encoding of the sonogram for SNN classification

In fig. 3.49 it is reported an example of rate encoding of the sonogram, that is used in a SNN input layer. In order to have a robust coding, a stimulus time is fixed $t_s = 1000ms$, this time corresponds to the duration for which the sample is submitted to the network and also is used as parameter in the encoding algorithms to define the time between two spike. In this time period each pixel is encoded separately by generating a number of spike based on pixel intensity, as reported in Encoding Algorithms Section 3.3.1 related to the rate encoding, as can be seen in fig. 3.49(b), from which the row with high number of spike is related to high intensity, and in the contrary the row without spike correspond to the darker region, so it is easy to deduce that each line y-axis of fig. 3.49(b) corresponds to a pixel of the sonogram. The second time period fixed is the $t_p = 20ms$, this time interval is used to have a separation between two sample, in which no spike is emitted.

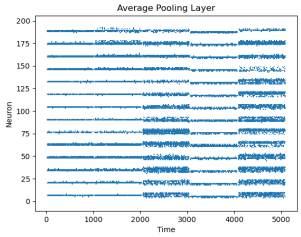
Now supposing to analyze a small portion of the training set consisting of 5 samples, and the behavior between the various layers, from fig. 3.50(a) it is possible to identify 5 regions corresponding to the various samples of the subset encoded with poisson rate, while in fig. 3.50(b)-(c)-(d)-(e)-(f) the behavior of neurons in the various layers is shown, and finally in the fig. 3.50(g) the behaviour of the classification layers for which in order to determine the predicted label of the sample it is sufficient to identify the neuron with the highest number of spikes, in the related time interval.



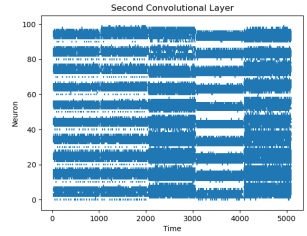
(a) Input layer



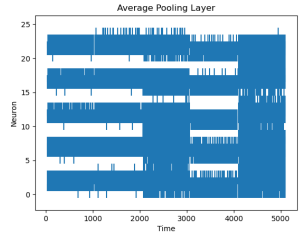
(b) First convolutional layer



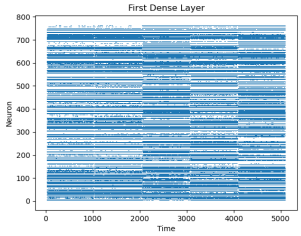
(c) First average pooling layer



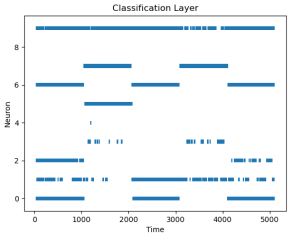
(d) Second convolutional layer



(e) Second average pooling layer



(f) First fully connected layer



(g) Classification layer

Figure 3.50: Spike Train in the various layer

3.7 Model Compression

By model compression we mean all those techniques that allow to reduce the structure of the network. In this case, this process is developed in two phases, through synapse reduction and pruning process. The advantages in applying this method are many such as: obtaining a much smaller network, suitable for embedded thanks to a reduced memory footprint and a smaller computational cost, making the simulation of the network in non-neuromorphic hardware faster and in some cases a better level of accuracy is reached thanks to the reduction of the stimulus carried out by the synapses, that can introduce an high level of noise during the classification process.

3.7.1 Synapses reduction

The synapse reduction process is carried out by selectively eliminating the specific synapses, but maintaining the same structure from the amount of the neuron and the relation through the layer. The criterion used to select the synapses to be eliminated is defined through weight, because this parameter defines how easily a neuron can generate a spike or not, as mentioned in the Section 2.1.2. Using the results from statistical analysis, the method for determining the elimination is through the probability distribution of the weights, obtained through the Box Plot.

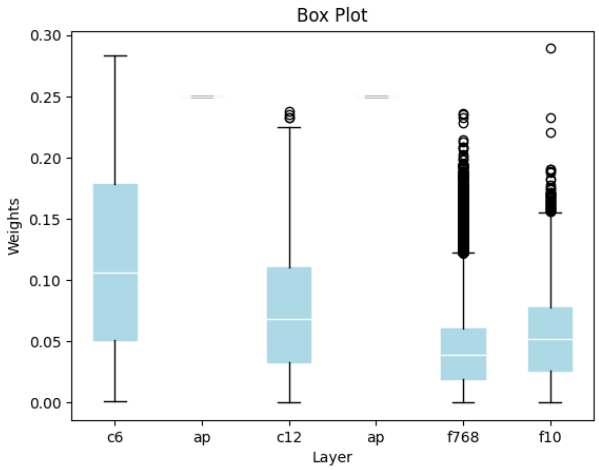


Figure 3.51: Box plot for the SNN c6c12f2

The box plot in fig. 3.51 is obtained considering only the modulus of the weights, this procedure is used to define the distribution of the quantities from the point of view of magnitude, as the greater the weight associated with a synapse, the greater the excitability of the neuron in the event that it is stimulated, in the case of excitatory synapses, the reverse behavior occurs in inhibitory synapses.

Starting from the portion of weights to be eliminated, by selecting for example the first quartile, the median or the second quartile, we proceed by eliminating all the synapses with weights below the chosen value, and then an evaluation is made on the accuracy parameter. Clearly this procedure is implemented in a convolutional and fully connected layer, but not in the Average Pooling layers since the weights are defined specifically in order to have an average action.

3.7.2 Pruning

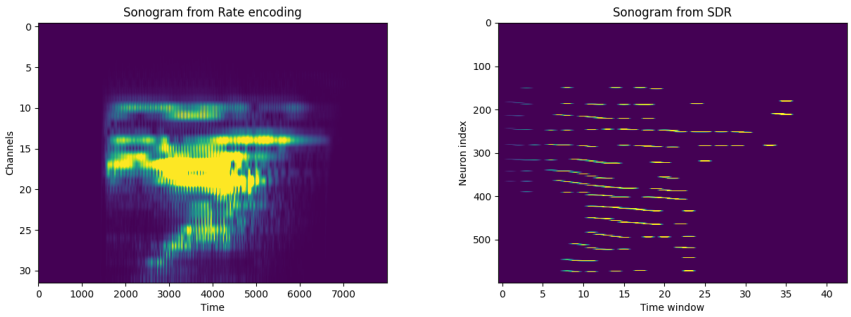
With term Pruning we mean the same procedure implemented in a synapses reduction followed by a fine tuning step, because once the Synapse reduction process has

been applied a worsening of the classification performance is observed in the network, when the synapses with very large weights are eliminated, such as in the case of upper quartile. This behavior is to be attributed to the reduction in the number of spikes in the various convolutional layers, making the classification process in fully connected layers very difficult. So in order to restore the original or even better accuracy, a fine tuning process is applied in the CNN version, constraining to 0 the weights previously eliminated in the phase of synapses reduction, carrying out a training of the weights left in the network for about 10-20 epochs, and finally re-apply the weight transfer in the spike version.

3.8 Inapplicable encoding cases

Analyzing all the steps of the pipeline that allow to carry out a classification of the data, I have treated, the type of data, the pre-processing techniques, the encoding algorithms, how to perform the feature extraction, the identification of the suitable type of network to be used were treated and the learning technique and finally the compression models. This therefore leads to the identification of a small subcategory of algorithms that cannot be used for performance evaluation, given their nature and the classification method selected in this case.

The algorithms in this case are rate encoding, implemented directly on the raw data, and SDR. The reason for this is related to the size of the sample in terms of pixels, as having a high number of pixels 32×8000 for rate encoding and 43×600 in the case of SDR.



(a) Rate encoding (b) SDR encoding

Figure 3.52: The left panel shows a sonogram relating to rate encoding, made up of a number of very large pixels, while on the left is the sonogram relating to the SDR presenting the same problem

Given the size of the songrams, the resulting datasets are very large and this produces networks with an enormous number of parameters, obtaining networks of the order of GB.

Chapter 4

Results and discussion

In this section we will discuss the impact of coding techniques on the classification of raw time-varying signals, obtained using the two datasets FSD and WISDM, and how the different Degrees of Freedom available affect such performances.

With Degrees of Freedom we indicate the set of all the possible choices that can be made at each step of the pipeline analyzed so far, of which they belong:

- Density of spike;
- Architecture CNN/SNN;
- Encoding;
- Feature Extraction;
- Synapse reduction;
- Pruning.

and the structure of the network influences the way in which the compression models act on that network, or the spike density influences the maximum number of layers of the network and so on, making these points dependent on each other. During our analysis we will describe what the various degrees of freedom represent, how to determine the parameters and finally how each of these points affects the final performance.

4.1 Performance evaluation for FSD dataset

4.1.1 Density of Spike

By density of spike we mean the number of spikes produced per unit of time, this quantity appears to be influenced by two elements:

- encoding algorithm: as described in Section 3.3, each encoding algorithm has a different implementation logic, treating the same information differently, consequently this is reflected in having a different number of spikes. In the fig. 4.1 the distributions of the number of spike per coding algorithm are reported in relation to the different pre-processing and feature extraction techniques, but what is always noticed is that regardless of the choices used, there is an identical trend in the four cases, identifying the class of Temporal contrast, Global Referenced and Latency/ISI, as the classes with the lowest number of spikes;

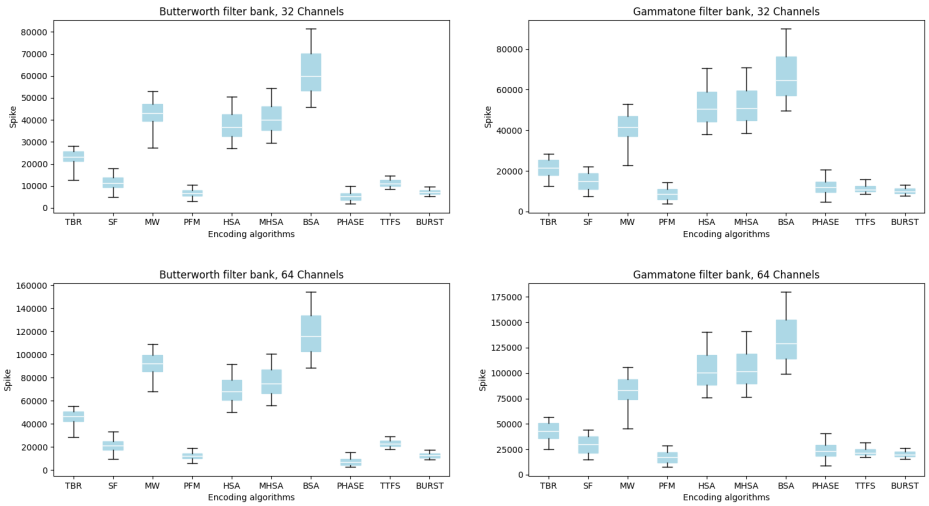


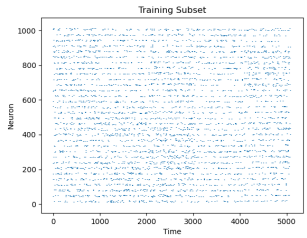
Figure 4.1: In this box plot the distribution of the number of spike for different algorithms necessary to encode the entire FSD dataset are reported, having applied different pre-processing filter bank

- refractory period: in physiology the refractory period is the minimum amount of time you need to wait after one spike is emitted before another spike is emitted. This delay is due to the membrane hyperpolarization event, making the neuron transparent to stimuli.

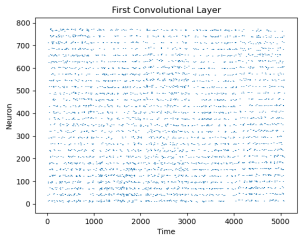
  In the case of SNN this quantity it is an external parameter respect to the encoding algorithm, which can be easily implemented by preventing the conversion of information after a spike is emitted, for a period of time equal to the refractory time. However, the presence of this parameter can significantly affect the classification accuracy; this is due to the fact that the amount of spikes in the presence of a refractory period is not sufficient to stimulate the neurons between the various layers, obtaining a progressive reduction of

the spikes, especially in dense architectures, since each neuron absorbs more spikes than are produced.

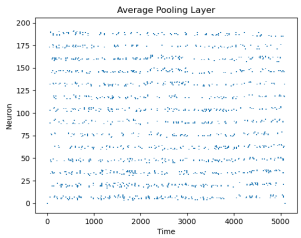
As shown in fig. 4.2 the first Convolutional layer receives a sufficient number of spikes from the input layer allowing the excitation of the neurons in this layer, whose spikes are sent in the first Pooling layer. In turn those neurons absorb the spikes coming from the previous layer, causing an excitation that produce further spikes, unfortunately, however, the amount of spikes produced is not enough to produce an excitation of the second Convolutional layer, having no more excitation of the neurons in the remaining layer.



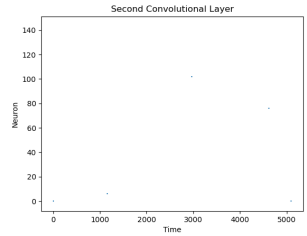
(a) Input layer



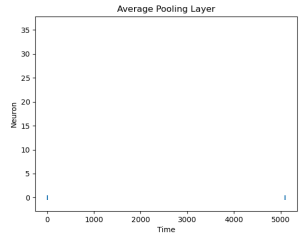
(b) First convolutional layer



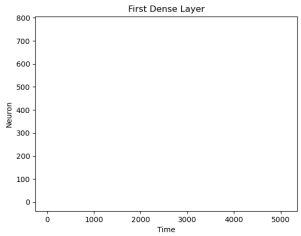
(c) First average pooling layer



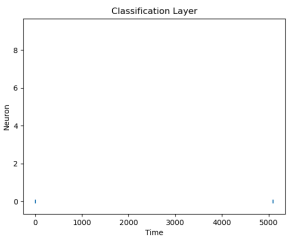
(d) Second convolutional layer



(e) Second average pooling layer



(f) First fully connected layer



(g) Classification layer

Figure 4.2: Spike Train in the various layer in the presence of refractory period in the encoding step

In fig. 4.3 it is possible to observe the encoding performances in the SNN network

as a function of the refractory period τ_{refra}. On the basis of what was previously ascertained, the worst accuracy was recorded in the case in which the refractory period turns out to be $\tau_{refra} \neq 0$, as the number of spikes produced in this case is not able to excite the neurons in the different layers; on the other hand a better classification performance is achieved in the case $\tau_{refra} = 0$. This leads us to conclude that the use of the refractory period in these cases it turns out to be disadvantageous.

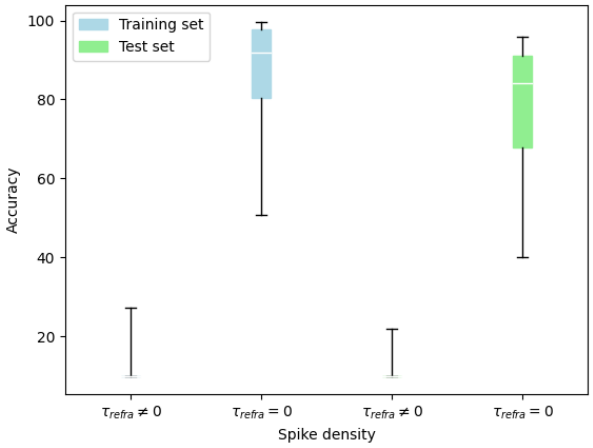


Figure 4.3: Box plot evaluating the refractory period performance in the SNN c12c24f2

4.1.2 Architecture CNN/SNN

The type and structure of the network clearly represents another element that can affect accuracy performance, since it is the networks through the learning process that extract the model that allows to describe the data. As reported in Section 3.6 the learning process used is the transfer learning, this procedure consists in carrying out the learning process in an ANN and using the weights obtained in the twin SNN version. For this reasons the Neural Network Architecture represents the second degree of freedom. Obviously the type of network must be defined on the basis of the type of data analyzed, and given the affinity of the sonogram to the images, a natural choice is the Convolutional Neural Network (CNN) structure.

The Convolutional Neural Network can be analyzed as two subsequent sub-networks: the convolutional part whose purpose is to extract features from the analyzed samples, focusing each layer in recognizing a particular pattern and improving this level of abstraction by increasing the number of convolutional layers; and the fully connected part, responsible for the classification process through the subdivision of the space of characteristics into the various desired classes fig. 4.4.

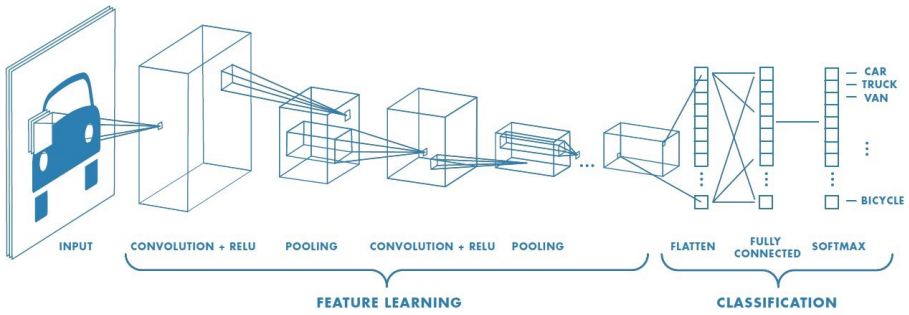
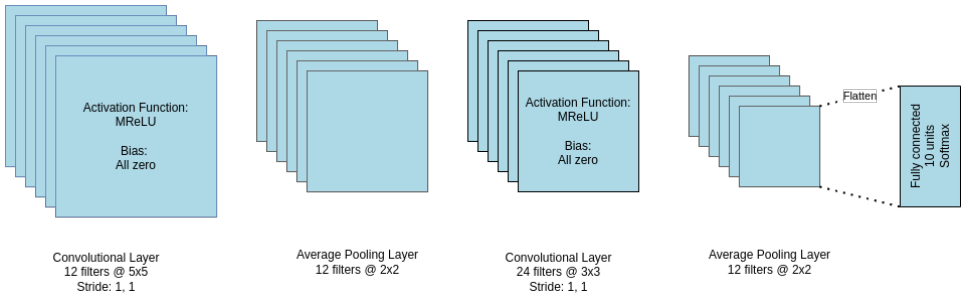
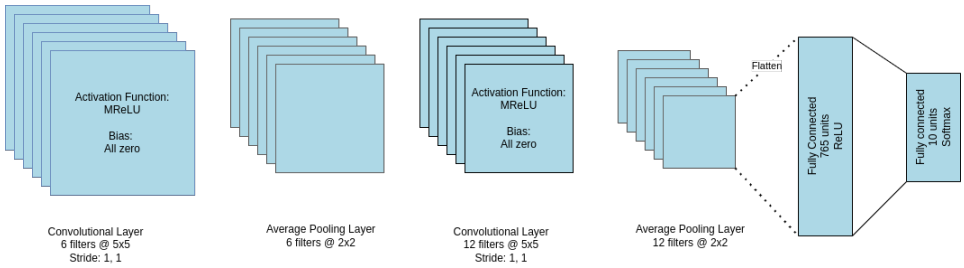


Figure 4.4: Separation of feature extraction and classification layers in CNNs

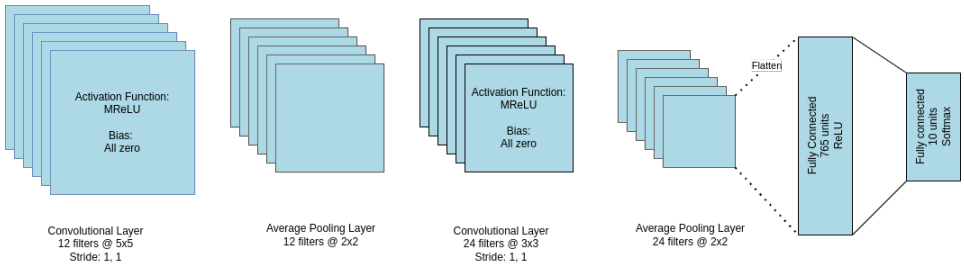
The main architectures tested that reported the highest level of accuracy are showed in fig. 4.4(a)-(b)-(c):



(a) c12c6f1



(b) c6c12f2



(c) c12c24f2

Figure 4.5: Tested Architectures

Once the training and testing process in CNN is complete and its SNN version is built, the classification process of the complete dataset is carried out again in the SNN to evaluate the accuracies performances. By making a comparison between the various structures, it was observed that the part of the network which mostly affects the accuracy is the fully connected one, achieving better accuracy in case there are two fully connected layers, as these layers are responsible for decision boundaries in the feature space. The more fully connected layers you have greater is degree of complexity you get in the decision boundary structure, however having too many layers reduces the number of possible spikes useful for classification reducing their performance, as mentioned in the previous chapter regarding the refractory period. For this reason the choice of this number is very important and does not have to be very large. The behavior described can be observed in fig. 4.6, in which a worse accuracy is recorded in the first type of network which takes into account only one fully connected layer, while a big improvement is obtained in the other two cases where the classification is performed by two fully connected layers.

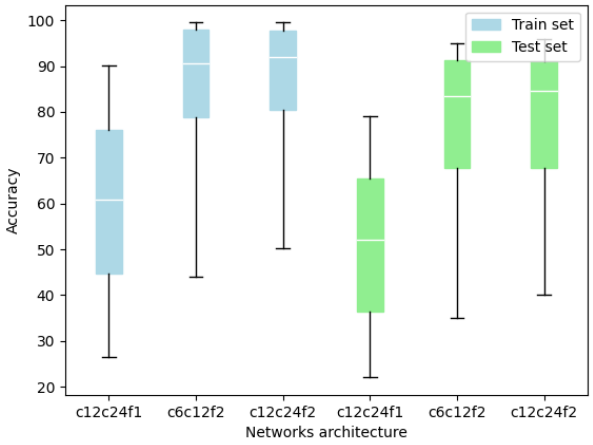


Figure 4.6: Comparison of accuracy performance in different SNN architectures

4.1.3 Encoding

Encoding is the fundamental step that allows to use the SNN, since the input must be encoded in a series of spikes in order to be correctly elaborated in this type of networks. Remembering briefly the two steps used during the encoding procedure we have the:

- Filtering decomposition: this first step allows to separate the input signals into different components using a bank of filters. This step it's performed to increase the amount of extractable features in the phase for the coding, thus favouring a generation of the sonogram with a richer amount of information. The two elements that influence this process are: the structure of the filter bank defined on the basis of the signal we are going to analyze, and the quantity of channels.

  The type of filter bank not only influences the separation between two consecutive frequency bands but also the encoding performance, as in the class of Global Referenced and Latency/ISI algorithm. The reason of this is strictly related to the amount of overlap between different bands. In the case of the Butterworth filter bank, the separation between the bands is clearer with very little overlap, but in the Gammatone an overlap between the bands are present. These two different behaviors allow to have in the Gammatone case a redundancy of information in frequency components in a single channel; this permits to have a greater production of spikes in the class Global Referenced and Latency/ISI algorithm, increasing the total accuracy.

  Finally to have a good extraction of information from the original sample, the number of channels must not be too small or too large. Analyzing the two cases separately we have that if the amount of channel is too small the component of frequency in one channel is large, and for this reason we can't use the standardization of parameters using the fact that all component have sine-like structure. On the other hand having a lot of channel reduces the amount of variation in the signals, having too little information to encode on each single channel.

- Algorithm encoding:
  - Temporal Contrast;
  - Filter & Optimizer;
  - Global Reference;
  - Latency/ISI.

The class that is most influenced by the type of filter used is the Global Referenced, since in this case the amount of spike is directly related to the intensity of the

signal. In the case of the Gammatone filter bank there are redundant components, consequently the intensity of the frequency channels is greater, and this produce more spike in this class of algorithms, allowing to transport more information, noting an increase in accuracy as shown in fig. 4.7.

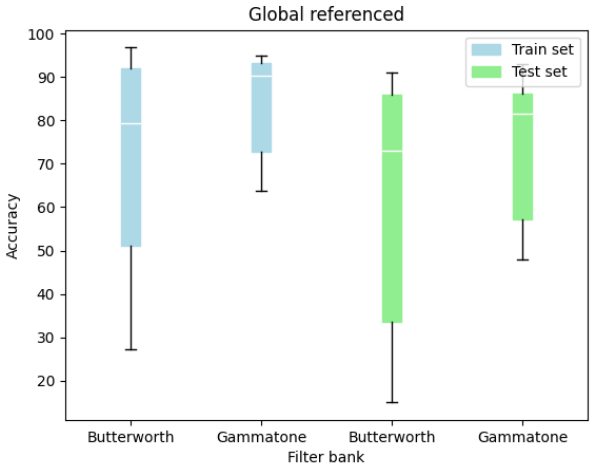


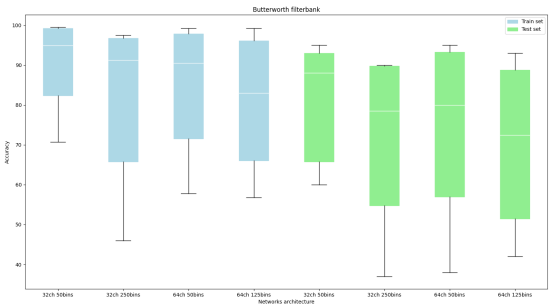
Figure 4.7: Comparison between Butterworth and Gammatone filter bank for Global Referenced class

4.1.4 Feature Extraction

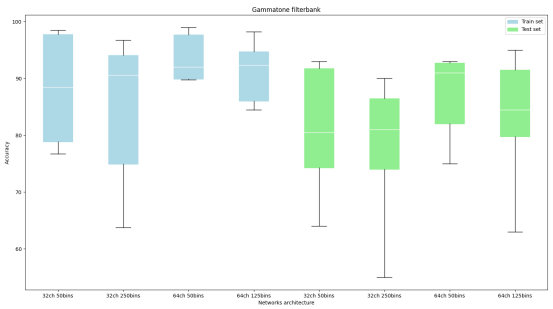
The feature extract represents the step that allows you to use the transfer learning method, thanks to the production of the sonogram, described in detail in the Section 3.5.1, which allows you to pass the information encoded in the spike domain into the spatial domain data, particularly suitable for CNN analysis.

During the production of the sonogram the parameter that allows the degree of feature extraction is the number of bins, this parameter allows to describe the number of intervals in which the spike coded signal must be divided, and as mentioned in Section 3.5.1 this value must not being too large or too small otherwise resulting in a uniform pattern with very little information, as the entropic contribution between the various positions is very small.

This behavior can be observed in fig. 4.8 for which, regardless of the number of channels selected in the pre-processing step, overall worst performances are obtained for a high number of bins.



(a) Butterworth filter bank



(b) Gammatone filter bank

Figure 4.8: Comparisons between different type of binning dimension for Butterworth and Gammatone filter bank

4.1.5 Visual analysis through feature reduction techniques

A first investigation on the separability of the classes that can be carried out on the datasets made up of sonograms is through a Data Visualization approach, which consists of a visual exploration of the data.

In the field of big data, we are dealing with samples described through a very large number of elements, generically called features in the machine learning. In order to obtain a graphical analysis of these samples, a representation in a space with n-dimensions would be necessary making such visualization impossible, in order to make this analysis accessible it is necessary to reduce the number of features to a space of 2 or 3 dimensions, through feature reduction techniques. Examples of application of feature reduction techniques are shown below.

The most used feature reduction technique is the Principal Component Analysis (PCA), whose result is reported in fig. 4.9.

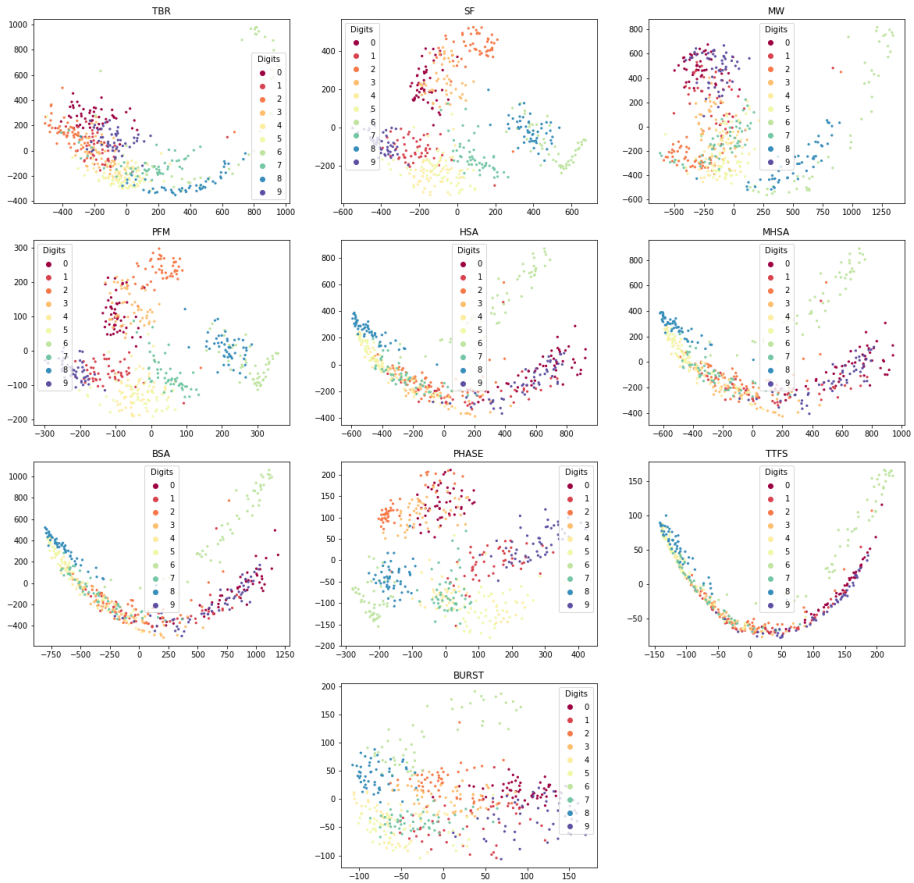


Figure 4.9: Results of the PCA feature reduction algorithm for the FSD dataset in which the sample is pre-processed with Butterworth filter bank with 32 channel, with 18 bins partition for TBSC feature extraction process

Unfortunately the application of this technique is not able to provide sufficient separation between classes. However moving on to more advanced feature reduction techniques, T-distributed Stochastic Neighbor Embedding (TSNE) and Uniform Manifold Approximation and Projection for Dimension Reduction (UMAP), a better separation can be observed in some specific types of algorithms.

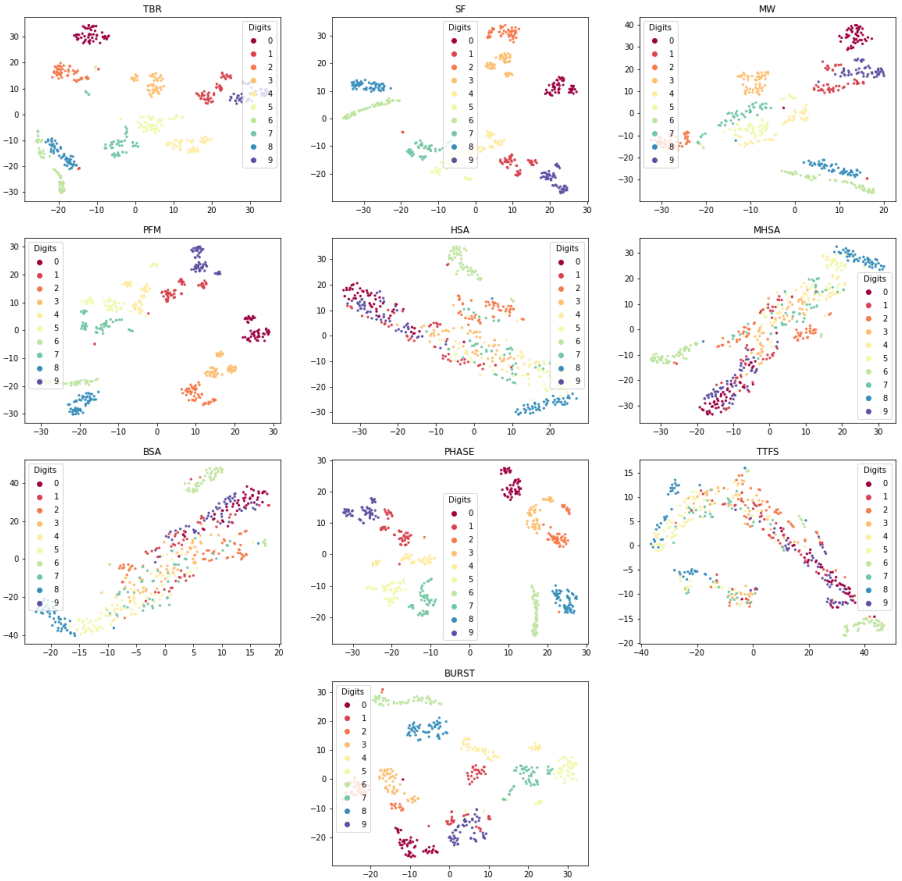


Figure 4.10: Results of the TSNE feature reduction algorithm for the FSD dataset in which the sample is pre-processed with Butterworth filter bank with 32 channel, with 18 bins partition for TBSC feature extraction process

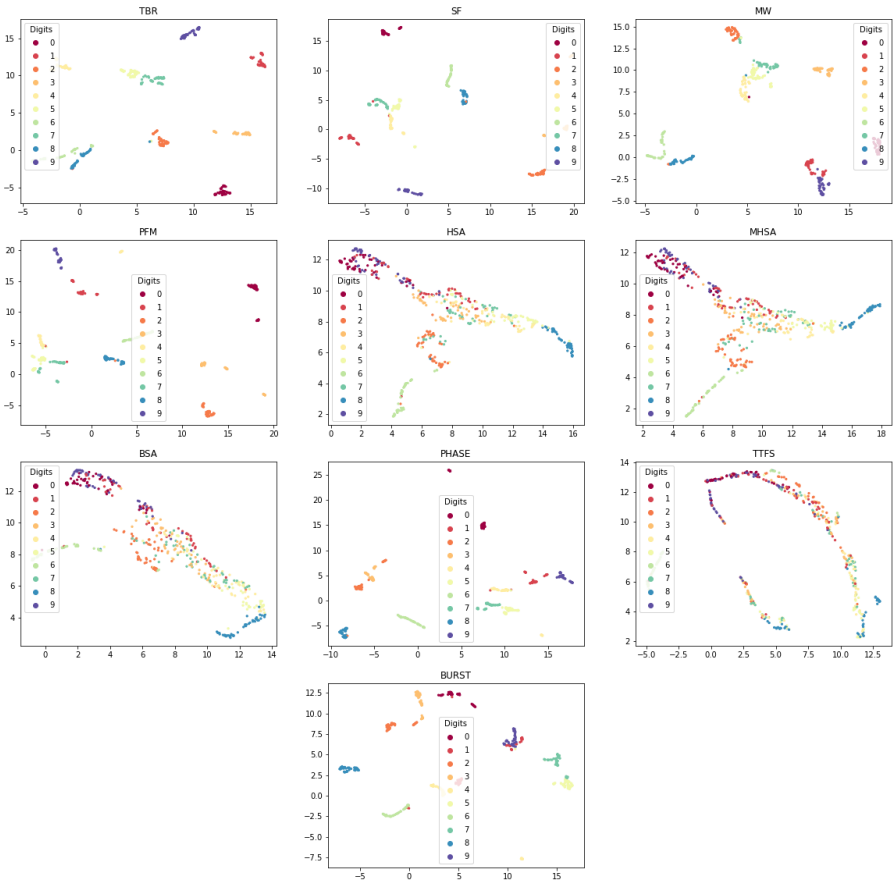


Figure 4.11: Results of the UMAP feature reduction algorithm for the FSD dataset in which the sample is pre-processed with Butterworth filter bank with 32 channel, with 18 bins partition for TBSC feature extraction process

As can be seen in fig. 4.10 and fig. 4.11, all the algorithms belonging to the Temporal Contrast class have the best separability, this is reasonable since algorithms are specialized in the coding of time-varying signals, but also algorithms such as PHASE and BURST algorithms. The worst separabilities have been recorded instead in the case of the Filter & Optimizer class as the need to find suitable filters for the analyzed signal makes the optimization of such algorithms very difficult, and in the TTFS encoding algorithm as the number of spikes generated in this case are very small, carrying very little information.

4.1.6 Comparisons between different classes of algorithms

Now carrying out an assessment on the classification capabilities of the SNN based on the coding technique, in fig. 4.12 are reported the performances of the various classes of algorithms in which the network c6c12f2 is chosen, for different types of feature extraction. As we can see the Temporal Contrast class present the best accuracy with respect to the amount of spike. As already observed in feature reduction techniques, the behavior is justified by the fact that this particular class of algorithms is dedicated to detecting the variation of signals, and since the processed signal is an audio signal from FSD, this method is particularly suitable for the purpose.

While in the case of the Filter & Optimizer, since it is necessary to define the filter used for the encoding, and given the large frequency components of the signal and the large differences in amplitudes of the various channel the encoding capabilities of the filter are not the same for each components, obtaining a reduction in the accuracy performance.

On the contrary, the Global Referenced class and the Latency/ISI one have a reduced classification capability, since such types of coding produce a loss of information directly proportional to the desired resolution as explained in Section 3.3.2, only in the case of time-varying signals.

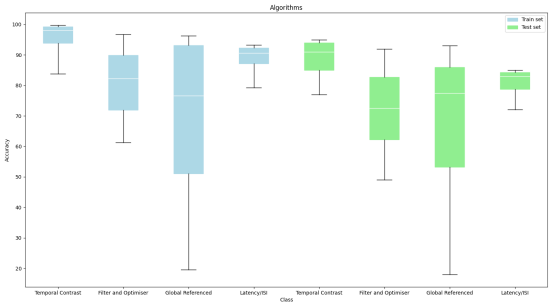
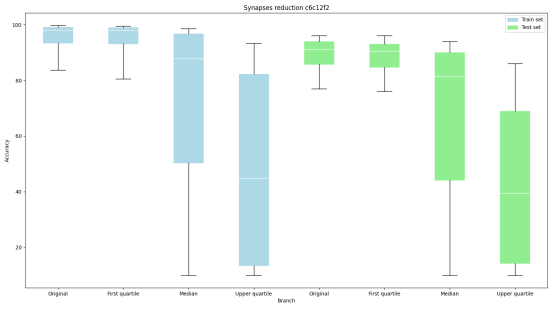


Figure 4.12: Comparisons between different classes of encoding algorithm

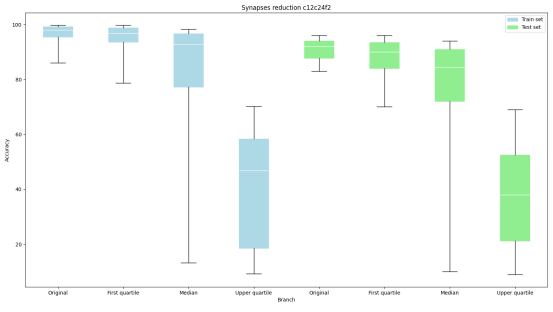
4.1.7 Synapse reduction

The Synapse reduction technique allows to extrapolate important information on the ability of the network to classify the samples, after the elimination of synapses with increasing weight. This process allows to obtain a series of advantages such as reducing the size of the network by reducing the memory footprint making them ideal for embedded applications, speeding up the classification process during simulation on non-neuromorphic devices and in some cases increasing accuracy performance.

Since the Temporal Contrast is the best performing class, we use this set of encoding algorithms to carry out further investigation regarding this process, by applying this method on two networks that have the best performances so far, identified by the names c6c12f2, c12c24f2 (fig. 4.13). Applying the method described in Section 4.1.6, as we expected as synapses with greater weights are eliminated, the accuracy of the networks decreases, reaching values around 40% test accuracies or even less. This behavior is justified by the fact that the amount of spikes produced by a neuron is proportional to the weight of the synapses connected to this neuron, by reducing the number of synapses the amount of spikes received by the neurons is reduced, preventing the proper functioning of classification neurons in the fully connected sub-networks.



(a) c6c12f2

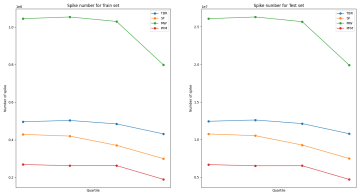


(b) c12c24f2

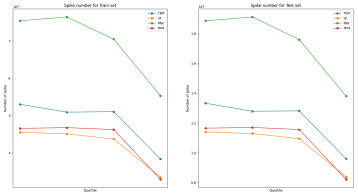
Figure 4.13: Accuracy for Synapse reduction in the architecture c6c12f2 and c12c24f2 for Temporal Contrast class

Through the reduction of the number of synapses, a decrease in the number of spikes is expected, and this is what happens in most cases. However, in some particular cases, such as the Moving Window algorithm, an increase was recorded. This behavior can be justified in the following way: if in the network there is a large number of inhibitory synapses with a low or medium weight, which are

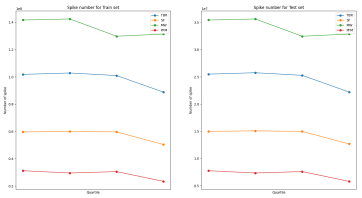
eliminated through the synapse reduction process, an accentuated effect of the excitatory synapses is obtained, this causes an overall increase in the number of spikes.



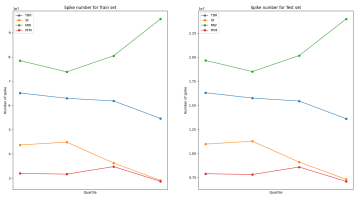
(a) Butterworth filter bank, 32 Channels, 50 Bins



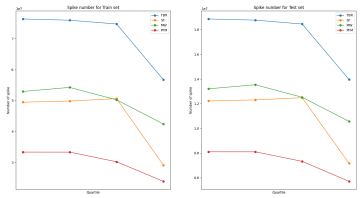
(b) Butterworth filter bank, 32 Channels, 250 Bins



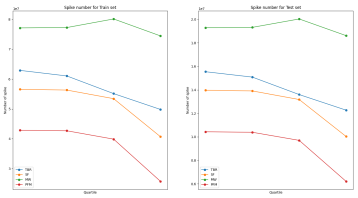
(c) Butterworth filter bank, 64 Channels, 50 Bins



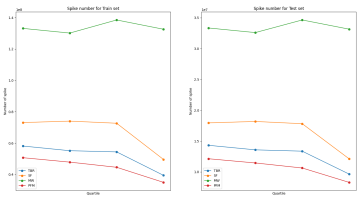
(d) Butterworth filter bank, 64 Channels, 125 Bins



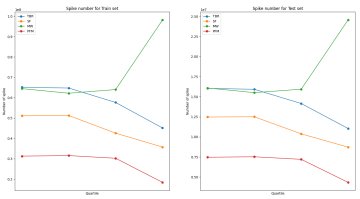
(e) Gammatone filter bank, 32 Channels, 50 Bins



(f) Gammatone filter bank, 32 Channels, 250 Bins



(g) Gammatone filter bank, 64 Channels, 50 Bins



(h) Gammatone filter bank, 64 Channels, 125 Bins

Figure 4.14: Spike Count for Temporal Contrast algorithm after synapses reduction in c6c12f2

4.1.8 Pruning

Finally, the Pruning process allows to "refine" the model of the new network obtained after Synapse reduction, through a fine tuning process. Since our goal is to reduce the size and computational cost of the network as much as possible, further analyses are focused on the structurally minimal network identified by the c6c12f2, a clear improvement in accuracy is observed, reaching a value of around 99% in the training set and 98% (fig. 4.15).

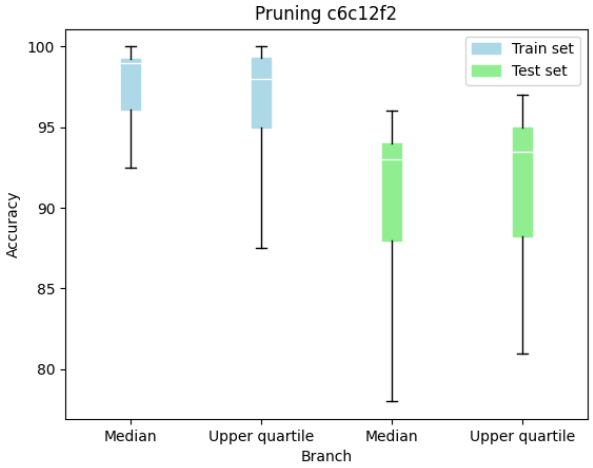
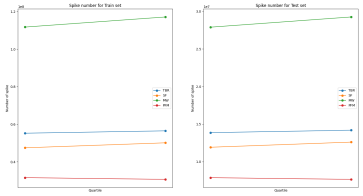
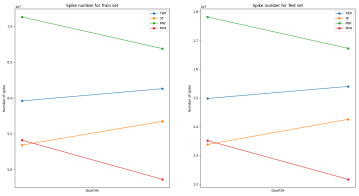


Figure 4.15: Pruning performance in c6c12f2

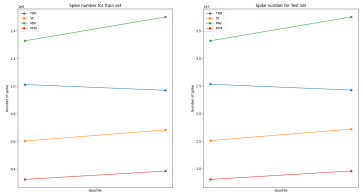
Furthermore, evaluating the amount of spike recorded in the network, necessary to classify training set and test set separately, it can be observed that in most cases the amount of spikes is preserved even in the smallest versions which count only 25% of the original network, allowing to reach a competitive accuracy compared to the larger one.



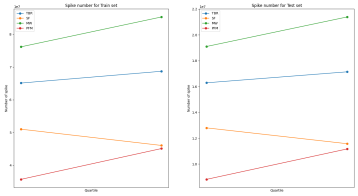
(a) Butterworth filter bank, 32 Channels, 50 Bins



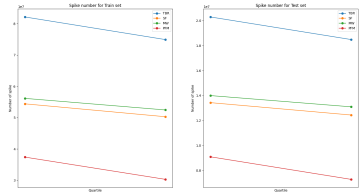
(b) Butterworth filter bank, 32 Channels, 250 Bins



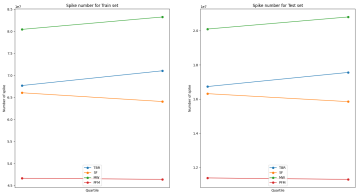
(c) Butterworth filter bank, 64 Channels, 50 Bins



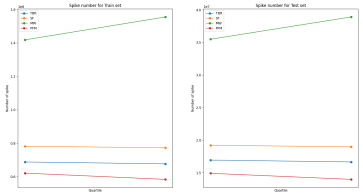
(d) Butterworth filter bank, 64 Channels, 125 Bins



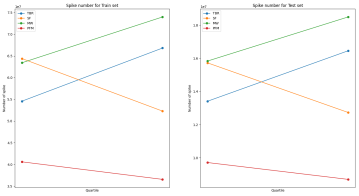
(e) Gammatone filter bank, 32 Channels, 50 Bins



(f) Gammatone filter bank, 32 Channels, 250 Bins



(g) Gammatone filter bank, 64 Channels, 50 Bins



(h) Gammatone filter bank, 64 Channels, 125 Bins

Figure 4.16: Spike Count for Temporal Contrast algorithm after pruning in c6c12f2

4.2 Performance evaluation for WISDM dataset

Compared to the FSD, the WISDM dataset has a completely different structure and property, as described in the Section 3.4.2, summing up we have that the number of components related to the raw data per sample are 6; the sampling rate is much lower $f_s = 20Hz$, that according to Nyquist-Shannon sampling theorem, the maximum frequency that can be solved is $f_{max} = f_s/2 = 10Hz$; the intensity of variation of the signal is much smaller, making more difficult to define the parameters in encoding techniques, even if the signal is amplified.

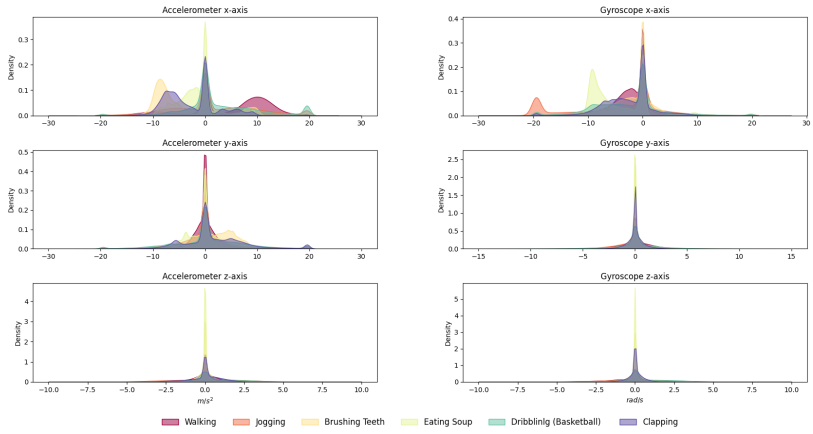
In order to verify how the presence of these differences affects all the classification steps, two subsets have been extracted from the original dataset: the subset1 and the subset2, whose activities are shown in the tab. 4.1.

subset1	subset2
Walking	Brushing Teeth
Jogging	Eating Soup
Brushing Teeth	Eating Chips
Eating Soup	Eating Pasta
Dribbling (Basketball)	Drinking from Cup
Clapping	Eating Sandwich
	Kicking (Soccer Ball)

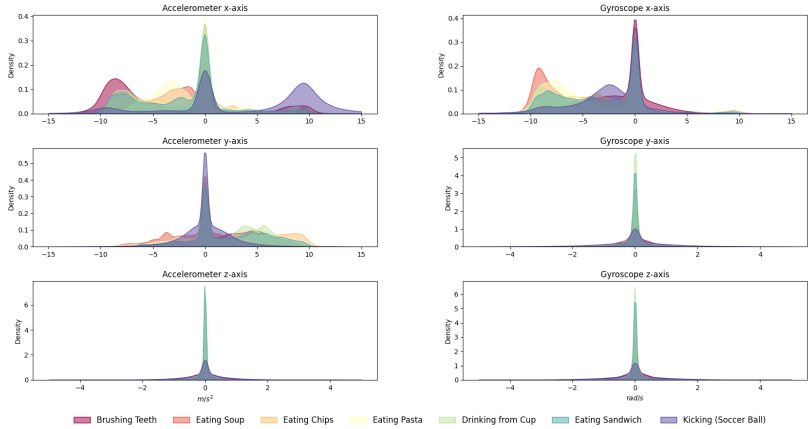
Table 4.1: This table shows the classes associated with subset1 and subset2, isolated from the WISDM dataset

4.2.1 Visual analysis through statistical methods and feature reduction techniques

In order to carry out a first analysis on the separability of the classes we use the kernel density estimation, a statistical method that allows to give an estimate of density in metric spaces, for the recognition of patterns and classification, a method used in the work carried out by Fra Vittorio et al. [59], on a differently pre-processed version of subset2.



(a) subset1



(b) subset2

Figure 4.17: Kernel density estimation for the two subset

As shown in fig. 4.17, in both cases there are overlaps between the distributions, making it very difficult to separate them through the subsequent analysis. For this reason in order to extrapolate a better information regard to the class separation from this type of subsets, we move on to the feature reduction techniques, after applying the pre-processing and encoding methods.

Unfortunately, also in this case the use of Principal Component Analysis (PCA) in both subsets respectively fig. 4.18-4.19, does not allow to have a good visualization on the separability of the classes.

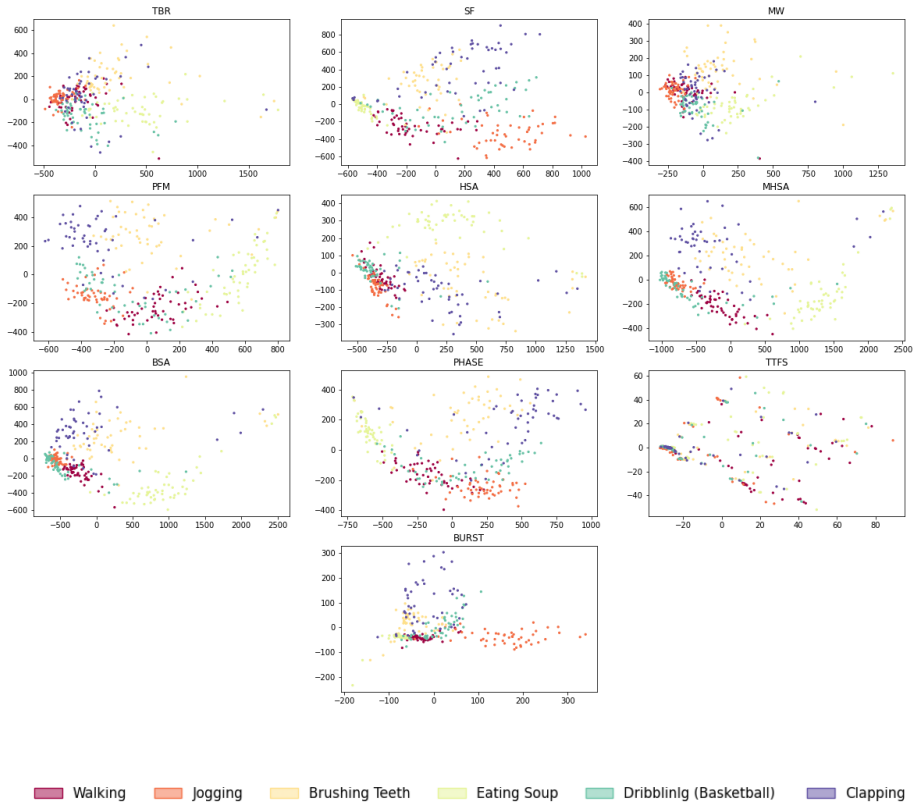


Figure 4.18: In this fig. is reported the PCA feature reduction algorithm for the subsets1 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

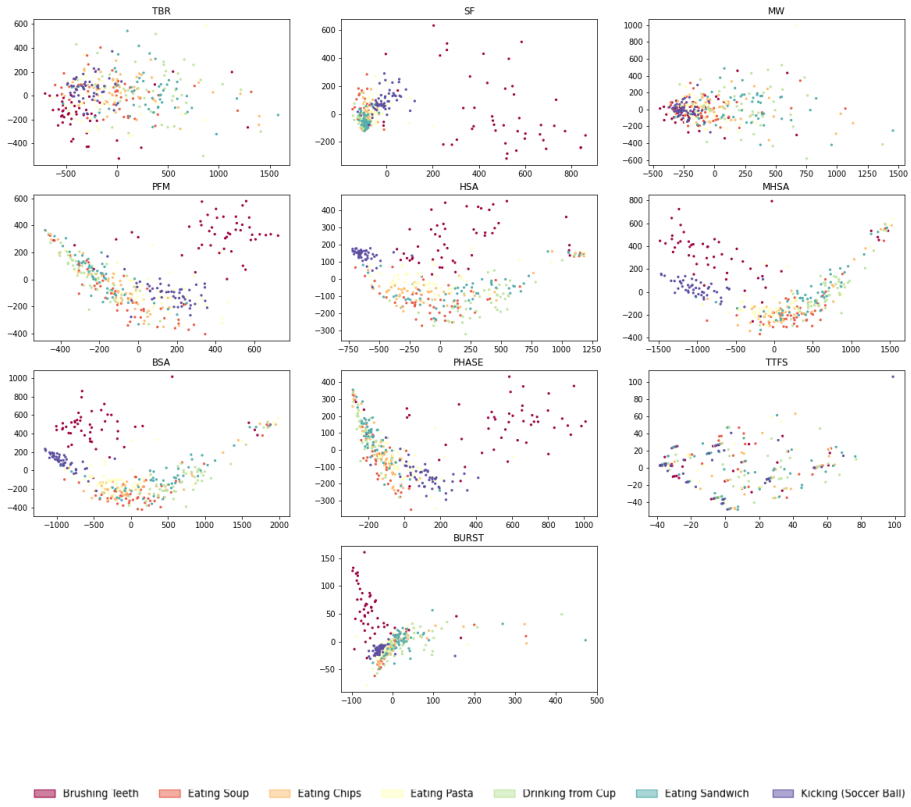


Figure 4.19: In this fig. is reported the PCA feature reduction algorithm for the subsets2 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

Shifting our attention to more advanced methods the TSNE and UMAP, as can be seen from both feature reduction methods in fig. 4.20, 4.21, 4.22 and 4.23, a better separability of classes can be identified in the case of subset1, respect to the subset2 in which no regions can be identified in which a class is prevalent.

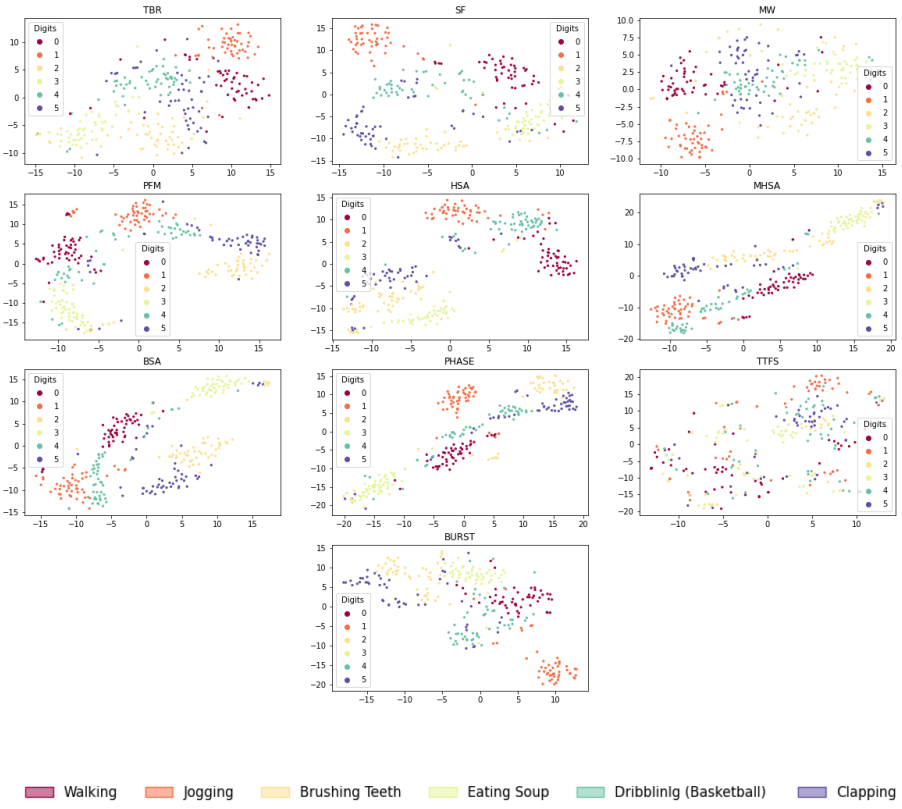


Figure 4.20: In this fig. is reported the TSNE feature reduction algorithm for the subsets1 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

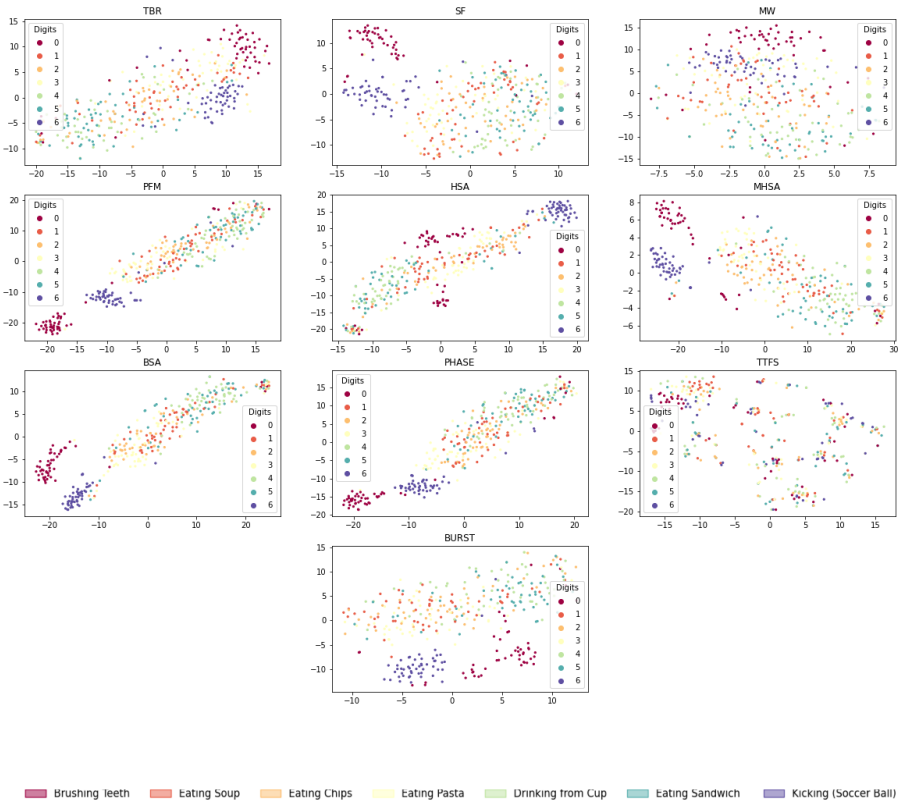


Figure 4.21: In this fig. is reported the TSNE feature reduction algorithm for the subsets2 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

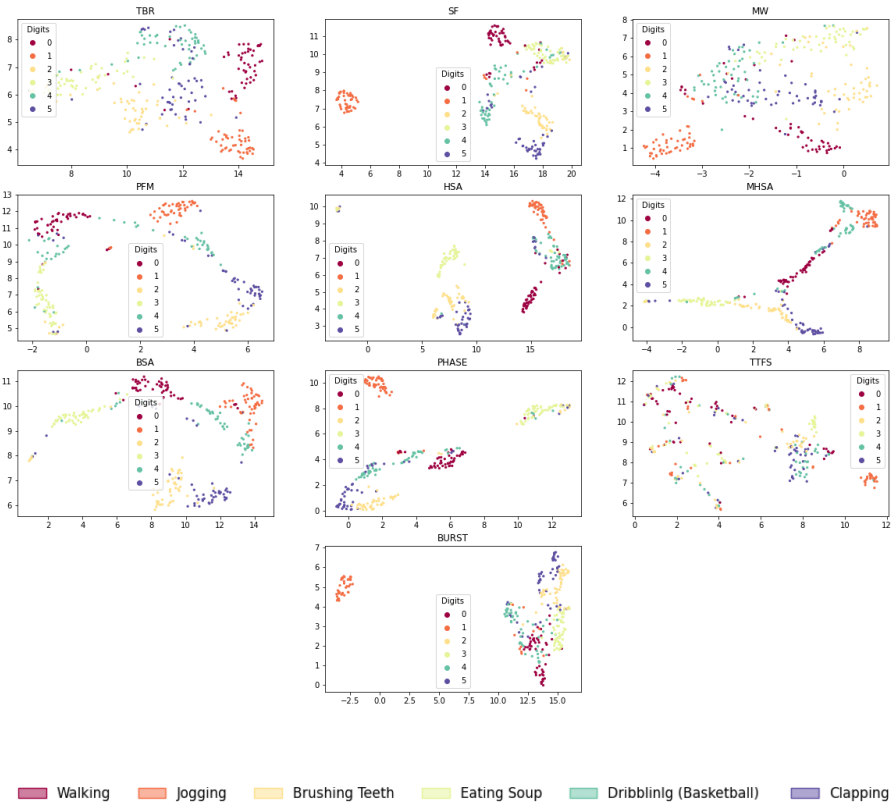


Figure 4.22: In this fig. is reported the UMAP feature reduction algorithm for the subsets1 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

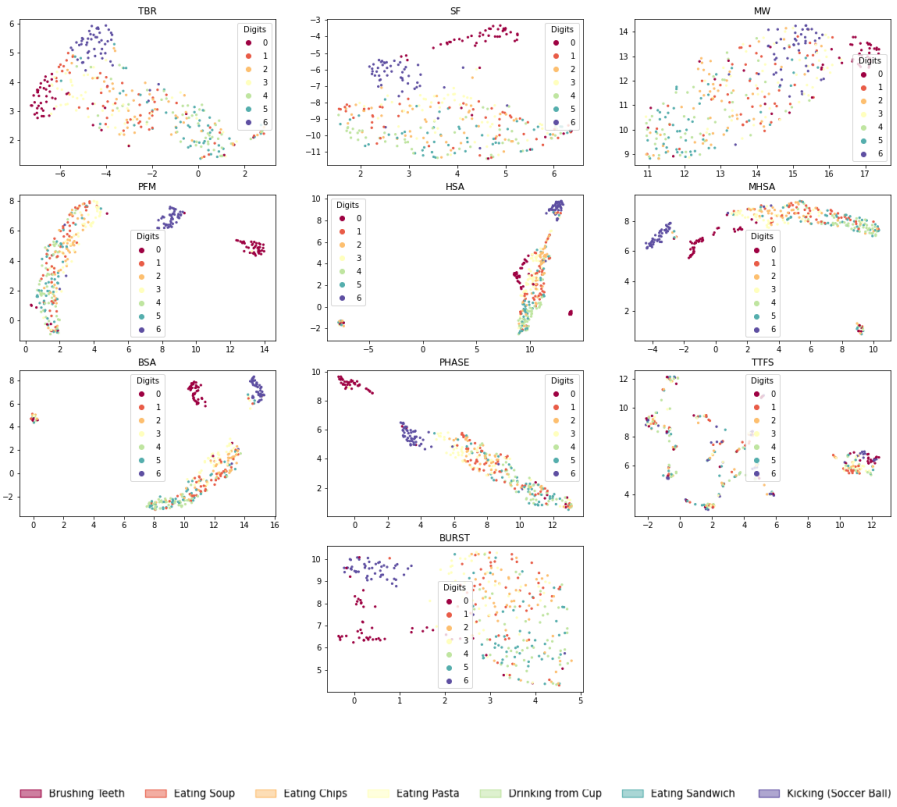


Figure 4.23: In this fig. is reported the UMAP feature reduction algorithm for the subsets2 in which the sample is pre-processed with Butterworth filter bank with 8 channel, with 18 bins partition for TBSC feature extraction process

The reasons for which in the case of subset1 a better separability of classes is obtained, after applying the pre-processing and feature extraction steps are given below:

- having a low sampling frequency, the amount of information that can be captured in the sampling stage turns out to be very small, obtaining very low informative features by carrying out a direct analysis as in the case of Kernel density estimation, and this behavior can be observed in fig. 4.17 not being able to observe a separability;

- once the pre-processing and feature extraction has been applied and specifically having carried out a separation in frequency channels, the amount of information that can be obtained is greater, no longer observing the signal as the superposition of multiple components, but analyzing the individual contributions at a different frequency.

However, the same behavior is not observed in the case of subset2, even after having performed the pre-processing, as the information content of the various channels on the classes is too similar to be discriminated with the following techniques.

4.2.2 Density of Spike

Also in this case clearly the number of spikes per unit of time appears to be dependent on the type of encoding used, as shown in fig. 4.24 but this is not true in the case of the refractory period. To understand this difference in behavior, let's start with the definition of this quantity referred to eq. 4.1:

$$\begin{aligned} t_r &= f_s \tau_{refra} \\ &= 20 \cdot 0.003 \\ &= 0.06s \end{aligned} \tag{4.1}$$

The value t_r indicates the amount of time due to the refractory period between two spikes even if the stimulus is present, since this value is larger with respect to the sampling interval time that is $t_r > t_s$ giving $t_s = 1/f_s = 0.05$, this means that once the spike is emitted, the neuron has a much faster recovery before the next stimulus is reached.

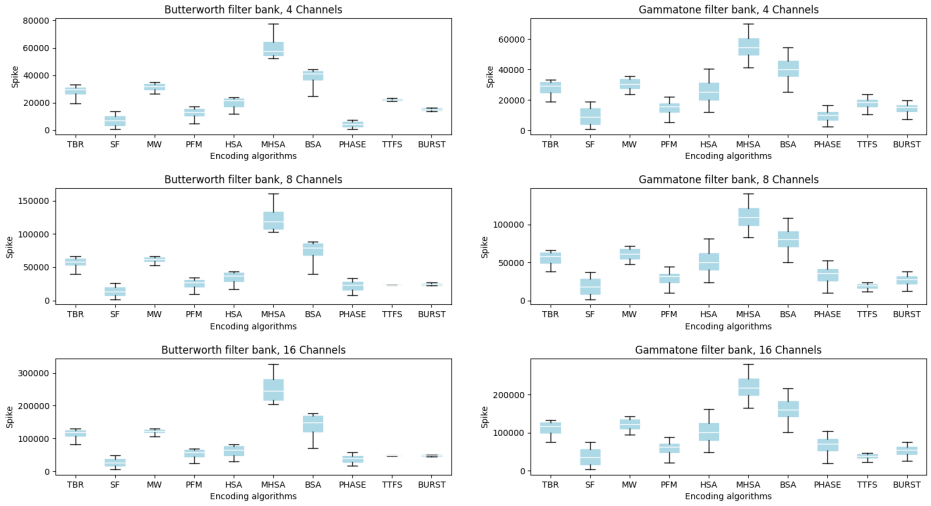


Figure 4.24: In this box plot is reported the distribution of the number of spike for different algorithms encoding the entire subset1 dataset, having applied different pre-processing filter bank

4.2.3 Architecture CNN/SNN

Comparing the results obtained in the case of the FSD in the Section 4.1.2, also in this case the architectures that have the best accuracy are the configurations c6c12f2 and c12c24f2, unfortunately, as can be seen in fig. 4.25, the accuracy in the best case is very low. The reason for these performances is not to be attributed to

the intrinsic structure of the network, but to the inability of the encoding algorithms for these subsets to provide correct encoding.

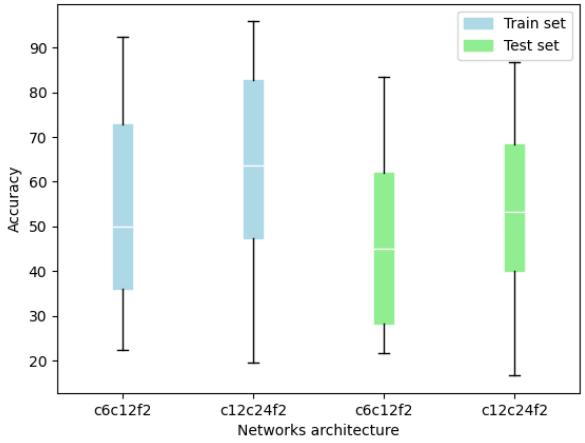


Figure 4.25: Comparisons of accuracy performance in different SNN architecture

4.2.4 Encoding

Contrary to the FSD case, in this type of subset there is not a particular class of algorithm that has better performances than the others, rather we are talking about specific algorithms, which have better performances, which are influenced by various parameters during the pre-processing, feature extraction, network structure, training, synapses reduction and pruning, making it impossible to identify a pattern. In fig. 4.26 is shown the comparisons with the PFM encoding, that in this case show the best behaviour respect to the other encoding algorithms.

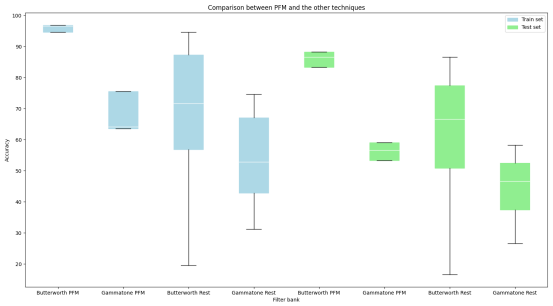
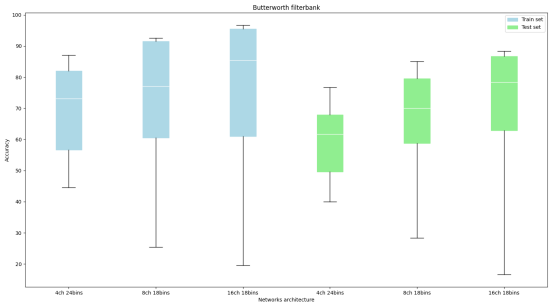


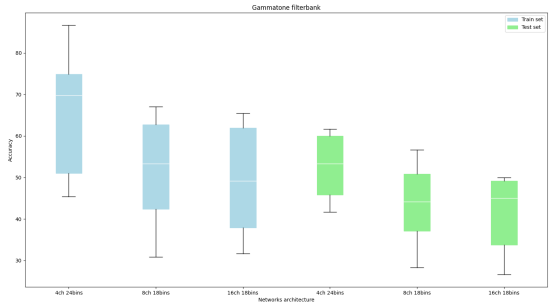
Figure 4.26: Comparisons between PFM and the other of encoding algorithms

4.2.5 Feature Extraction

Regarding the feature extraction process, a different behavior was observed than that defined in the Section 4.1.4, although in the case of the FSD as the number of bins increased, a reduction in accuracy was obtained, in this case we observe a specular behavior based on the type of filter and the channel used in the pre-processing step. Having that in the case of the Butterworth filter bank as the number of channels increases, the precision performance has an increasing trend; the opposite behavior occurs in the case of the Gammatone filter bank, the reason for this decrease could be related to the distortion introduced in the use of this filter (fig. 4.27).



(a) Butterworth filter bank



(b) Gammatone filter bank

Figure 4.27: Comparisons between different type of binning dimension for Butterworth and Gammatone filter bank

4.2.6 Comparisons between different class of algorithm

As already described in Section 4.2.4, in these cases it is not possible to identify a class of algorithms whose performance is better, since as shown in fig. 4.28 the same trend for the accuracy in the various encoding classes can be appreciated.

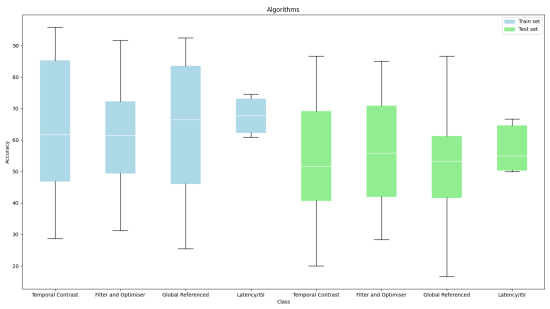
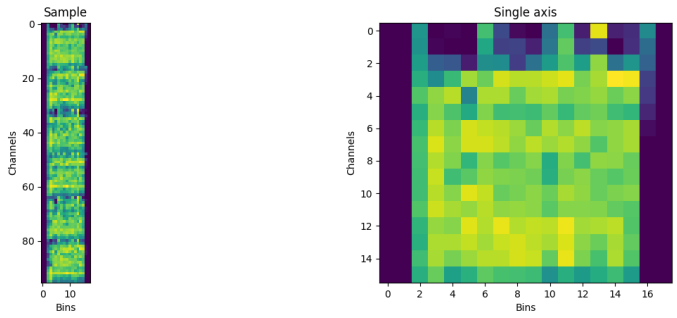


Figure 4.28: Comparisons between different class of encoding algorithm

The reason for this behavior is related to the amount of information that can be extracted during pre-processing and encoding, which in turn is related to the information contained in the raw signal. However, given the low level of sampling, the original signal is not sufficiently rich in terms of frequency and amplitude components. These characteristics are easily deducible from the sonogram of a sample examined in fig. 4.29, for which all the signals collected by the 6 sensors have the same structure from the global point of view. By analyzing in more detail the partial sonogram of the single sensor, unlike the case of the FSD, the two elements from which information can be extracted, namely: the conformation of the edge; and the internal patter; here they are almost practically absent.



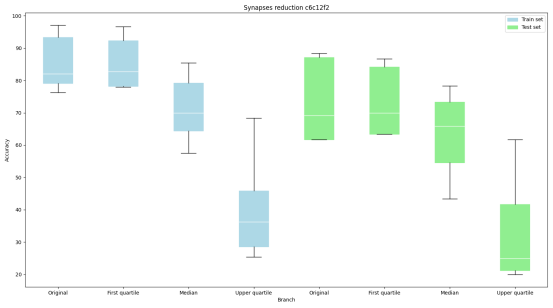
(a) Sonogram for all signal (b) Single axis signal

Figure 4.29: The sonogram of the various axis of sample

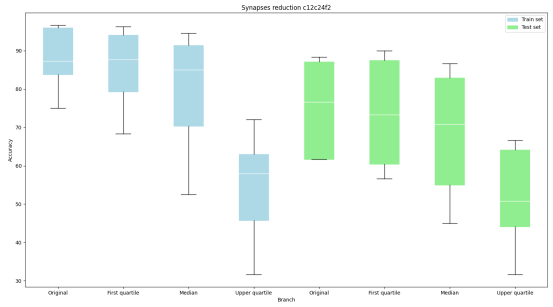
4.2.7 Synapse reduction

The applied Synapse reduction process turns out to be the same as that described in Section 4.1.7. As can be easily guessed given the nature of the process, a progressive deterioration in accuracy is expected, and this is confirmed by the experimental observations, as reported in fig. 4.30. For which as the synapses with greater

weight are eliminated, the ability of the network to carry out a correct classification is reduced, as already extensively described in the section mentioned.



(a) c6c12f2



(b) c12c24f2

Figure 4.30: Accuracy for Synapse reduction in the architecture c6c12f2 and c12c24f2

4.2.8 Pruning

The application of the pruning method in this case turns out to be extremely advantageous in a few lucky cases referred to the dataset that is well separable, as in the case of subset1, achieving test accuracies around 95% at maximum. This behaviour is justified, thanks to the combined effect of synapses reduction and fine tuning, allowing to reduce the number of synapses in the network while maintaining a model suitable for the description of the dataset; consequently there is a reduction in the number of synapses that can introduce noise making the classification process easier.

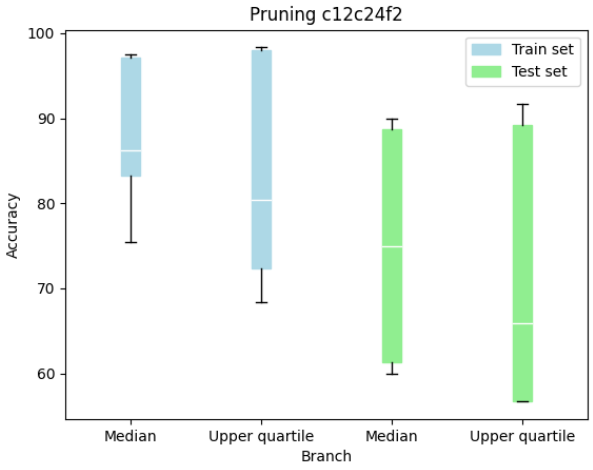


Figure 4.31: Pruning performance in c12c24f2

Performance evaluation for subset2

By applying the same analysis in the case of subset2, the same trends already described in the section of subset1 were observed, with the only difference that in this case there was a reduction in the maximum level of accuracies that can be achieved. The reason for this, as already mentioned in Section 4.2.1, is due to the reduced separability of the classes.

Density of Spike

With reference to the density of spike, the use of the refractory period is also irrelevant in this case, having the same sampling frequency. Recording the same behavior with a spike peak in the case of MHSA encoding as shown in fig. 4.32.

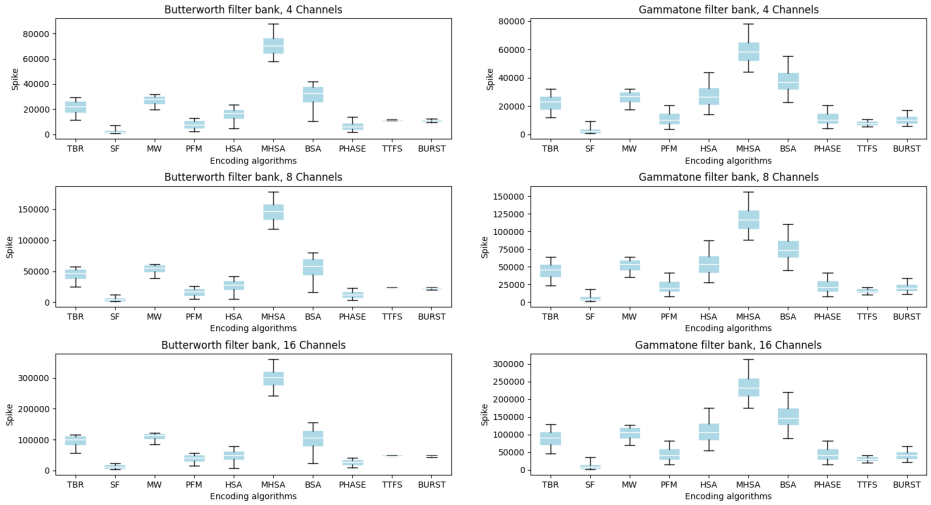


Figure 4.32: In this box plot is reported the distribution of the number of spike for different algorithms necessary to encode the entire subset2 dataset, having applied different pre-processing filter bank

Architecture CNN/SNN

As regards the performances in relation to the type of network architecture used, the best performances were recorded in this case also in the c6c12f2 and c12c24f2 networks, reporting very similar accuracies in relation to the subset1 in the case of the training sets, but noting a deterioration in the case of testing as reported in fig. 4.33.

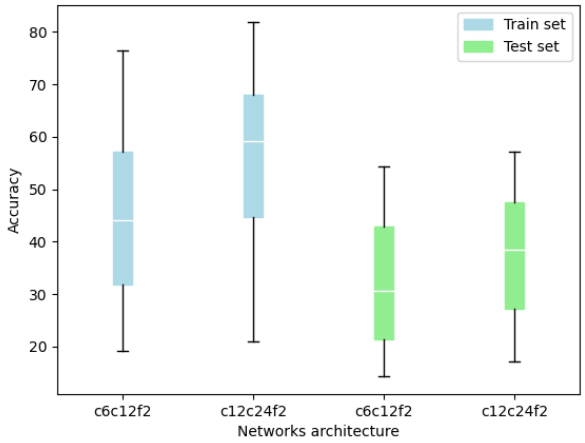


Figure 4.33: Comparisons of accuracy performance in different SNN architecture

Encoding

Also in this case it is not possible to identify a class of algorithms that appear to have better performance, but we always talk about specific encoding algorithms, which are able to capture more information than others. As can be seen in fig. 4.34 the best accuracies in this case are achieved in the case of the PFM algorithm.

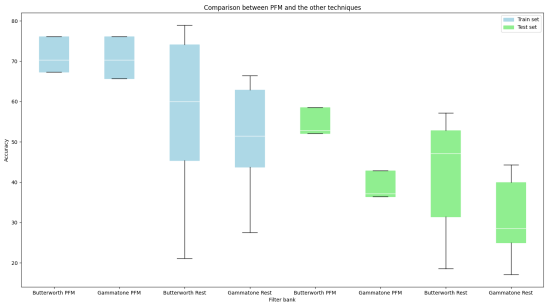
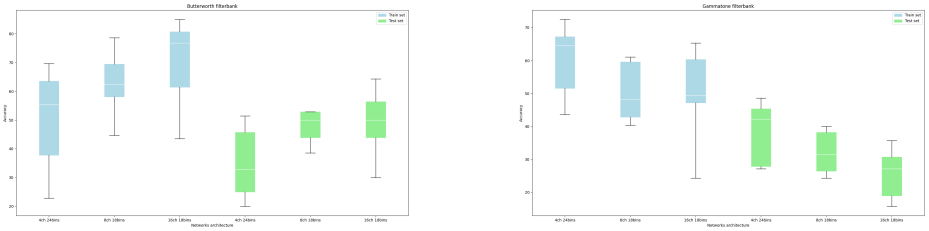


Figure 4.34: Comparisons between PFM and the other of encoding algorithms

Feature Extraction

In relation to the feature extraction in the case of subset2, the same trend already commented for subset1 is noted, with the difference that in this case the accuracy in the worst case is much greater (fig. 4.36).



(a) Butterworth filter bank (b) Gammatone filter bank

Figure 4.35: Comparisons between different type of binning dimension for Butterworth and Gammatone filter bank

Comparisons between different class of algorithm

A more detailed analysis of the performances for the various encoding classes is shown in fig. 4.36, for which it is not possible to identify a class of algorithms more suitable than others, but always talking about single algorithms that perform better than others, as already mentioned in the previous section of Feature extraction.

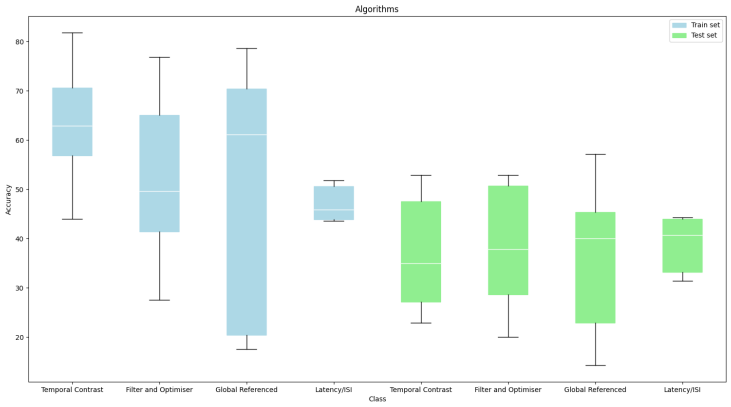
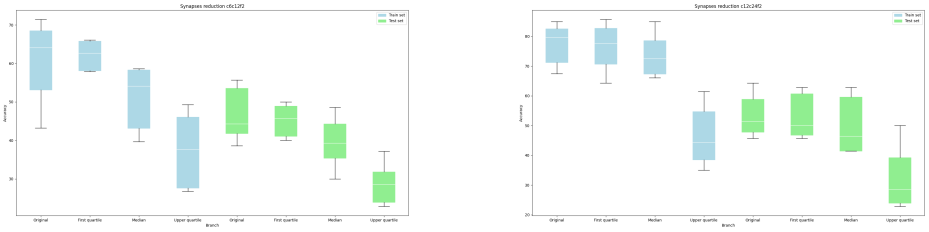


Figure 4.36: Comparisons between different class of encoding algorithm

Synapse reduction

In relation to the Synapse reduction we observe the same behavior already described in Section 4.1.7 for FSD dataset, in which a progressive reduction of the accuracy, as soon as the synapses with higher weights as indicated in fig. 4.37



(a) c6c12f2 (b) c12c24f2

Figure 4.37: Accuracy for Synapse reduction in the architecture c6c12f2 and c12c24f2

Pruning

Unfortunately, contrary to what was reported for subset1 during the pruning process, in the case of subset2 no significant improvement in accuracy was recorded, but only a slight improvement as reported in fig. 4.38. Reaching 87.1% in the best case for the training set in PHASE encoding and 60.0% in the case of the test set in the MHSA algorithm.

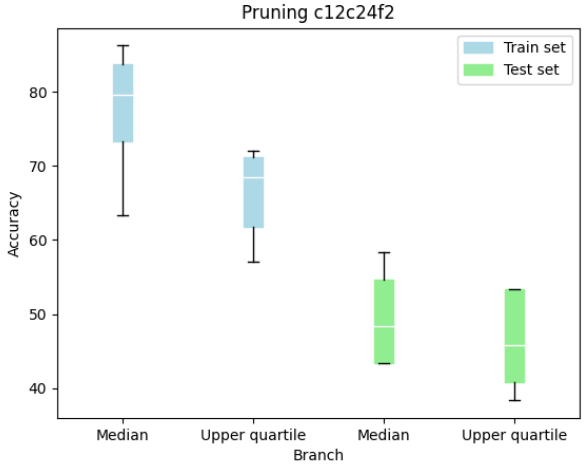


Figure 4.38: Pruning performance in c12c24f2

As previously described, the poor separability of the classes and the reduced information in terms of frequency components make it difficult the learning method used in this analysis for the classification of samples originating from subset1 and

subset2. However, in a work we recently published by Fra Vittorio, Pignari Riccardo et al.[59] it is shown that the use of a new classification method implemented through the Legendre Memory Units (LMU), a new kind of memory cell used in a recurrent neural networks described from A. Volker et al.[60], on subset2 by applying a decomposition of the signal into channels through a filter bank has allowed to achieve high accuracy comparable with the state of the art, outclassing the non-spiking and spiking CNNs.

Chapter 5

Conclusion

From the results presented in this work, it can be concluded that neuromorphic technologies and in particular SNNs represent a valid alternative to classical ANN architectures, presenting accuracies comparable to the state of the art, and at the same time they allow you to take advantage related to you bio-inspired systems, such as reduced energy consumption and computational cost.

By analyzing the nature of the samples that can be treated, a first classification of the raw information was outlined, identifying three categories: Temporal data, Spatial data and Spatio-temporal data. In order to determine the most suitable coding algorithm and to verify the impact of coding techniques on the processing and the classification performance for the various signals, the implementation steps of a neuromorphic machine learning pipeline were retraced.

The use of datasets composed of variable temporal signals, such as the FSD and WISDM, made it possible to test the pre-processing operation, i.e. the subdivision into frequency channels through a filter bank, identified in the human auditory system implemented through the cochlea, and applying the various encoding models, identified during the experimental analyzes of nervous biological systems, also divided into classes: Rate Coding, Temporal Contrast, Filter & Optimizers, Global Referenced, Latency / ISI and Corralation & Syncrony.

In order to evaluate the coding of the various algorithms and their ability to process and extract information, a classification process has been implemented through the SNNs by testing amount of spikes, different architectures, encoding, and model compression, in which the learning process is implemented through transfer learning a method that allows you to take advantage of consolidated training techniques in the field of ANNs.

Through the observations made on the available datasets and by the works developed by other authors, the classification performances were outlined that allow to determine the choice of the most suitable coding method for the type of data, based on the nature of the algorithm:

- Temporal Contrast class: if it is oriented to the detection of temporal variations;
- Filter & Optimizer: if the information is treated through a filtering process;
- Global Referenced, Latency / ISI: if the encoding is comparable to a binary code.

However, a small set of algorithms such as Correlation & Synchrony appear to be inapplicable in the cases listed, confirming the presence of algorithms capable of being applied only for specific types of data.

Summary encoding technique			Temporal data		Spatial data
			Very-Low frequency	Middle frequency	
Rate coding	Poisson encoding		✗	✗	✓
Temporal Coding	Temporal Contrast	TBR	✓	✓	✗
		SF	✓	✓	✗
		MW	✓	✓	✗
		PFM	✓	✓	✗
	Filter & Oprimizer	HSA	–	–	✗
		MHSA	–	–	✗
		BSA	✓	–	✗
	Global Referenced	PHASE	✗	✓	✓
		TTFS	✗	✓	✓
	Latency/ISI	BURST	✗	✓	✓

Table 5.1: Summary encoding technique based on the best performances in relation to the type of data, indicating with ✓ the technique is particularly suitable for the purpose, – these techniques can be used even if they have some disadvantages and ✗ they are not suitable for the purpose

Appendix A

Accuracy tables for Free Spoken Digits (FSD) dataset

A.1 Complete network

In this section is reported the train and test accuracies of the various network in the case of the FSD, in which different encoding algorithms and feature extraction is applied.

Table A.1: Network structure c12c24f1 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.8	95.0	100.0	96.0	99.5	94.0	99.5	94.0	98.0	87.0	93.3	76.0	96.0	76.0	100.0	92.0	98.0	69.0	99.8	95.0
	250 Bins	100.0	95.0	98.8	96.0	99.3	94.0	99.8	93.0	89.5	78.0	94.7	82.0	92.5	77.0	99.8	95.0	85.8	66.0	99.0	96.0
64 Channels	50 Bins	100.0	97.0	100.0	96.0	100.0	96.0	100.0	96.0	98.8	77.0	99.0	74.0	98.0	82.0	100.0	97.0	94.5	56.0	98.8	91.0
	125 Bins	100.0	96.0	100.0	96.0	99.5	95.0	99.8	97.0	98.8	83.0	99.3	82.0	99.3	87.0	100.0	98.0	98.5	73.0	99.8	94.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	100.0	92.0	99.0	94.0	99.0	94.0	99.5	92.0	98.8	93.0	98.8	90.0	97.5	68.0	98.0	91.0	94.7	79.0	99.3	90.0
	250 Bins	99.5	94.0	98.5	91.0	98.3	91.0	99.0	93.0	98.3	93.0	98.5	95.0	94.5	77.0	99.3	92.0	98.0	79.0	99.0	87.0
64 Channels	50 Bins	100.0	94.0	99.8	95.0	99.5	92.0	100.0	93.0	100.0	97.0	99.5	95.0	97.0	75.0	99.0	92.0	99.0	81.0	99.0	87.0
	125 Bins	99.8	93.0	99.8	95.0	99.5	91.0	99.5	92.0	99.5	97.0	98.8	94.0	97.8	76.0	100.0	95.0	98.8	79.0	99.5	91.0

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	66.2	55.0	69.5	59.0	20.8	23.0	81.2	67.0	29.2	26.0	35.8	31.0	54.5	45.0	82.0	70.0	42.0	35.0	74.8	64.0
	250 Bins	26.5	24.0	45.0	37.0	43.2	35.0	48.8	44.0	27.0	21.0	23.8	20.0	35.0	30.0	57.5	50.0	35.5	22.0	24.2	23.0
64 Channels	50 Bins	59.2	40.0	74.5	53.0	67.2	56.0	84.2	69.0	57.0	40.0	71.2	51.0	49.0	34.0	80.8	72.0	30.5	24.0	74.8	65.0
	125 Bins	90.8	72.0	90.2	72.0	98.8	88.0	90.0	79.0	53.2	42.0	82.2	61.0	44.0	32.0	75.0	60.0	42.8	33.0	73.2	61.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	70.5	62.0	34.2	25.0	55.5	44.0	72.2	73.0	81.0	74.0	83.0	79.0	58.2	45.0	66.8	63.0	61.3	50.0	73.0	63.0
	250 Bins	29.0	29.0	30.5	32.0	20.2	21.0	54.0	55.0	79.0	72.0	59.8	60.0	48.0	40.0	42.8	39.0	54.8	46.0	48.2	41.0
64 Channels	50 Bins	62.3	64.0	27.3	22.0	52.2	44.0	76.5	71.0	62.0	60.0	60.2	51.0	79.0	61.0	46.0	47.0	65.0	54.0	93.5	77.0
	125 Bins	85.2	74.0	74.0	69.0	52.8	46.0	88.8	83.0	76.0	76.0	85.2	83.0	78.8	62.0	45.0	43.0	65.5	53.0	96.2	88.0

Table A.2: Network structure c6c12f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	100.0	96.0	99.8	95.0	99.8	95.0	99.8	95.0	95.5	83.0	98.8	90.0	98.3	84.0	99.8	94.0	94.5	65.0	99.8	96.0
	250 Bins	99.8	96.0	99.8	96.0	100.0	97.0	100.0	96.0	97.8	92.0	97.3	82.0	97.8	84.0	100.0	94.0	94.5	67.0	99.5	94.0
64 Channels	50 Bins	99.8	98.0	100.0	98.0	100.0	98.0	99.8	96.0	100.0	80.0	97.8	71.0	99.8	90.0	100.0	97.0	99.3	65.0	99.8	97.0
	125 Bins	100.0	98.0	100.0	98.0	100.0	98.0	99.8	97.0	99.5	82.0	97.8	78.0	99.5	87.0	100.0	97.0	98.0	67.0	100.0	92.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.5	96.0	100.0	96.0	99.8	90.0	100.0	94.0	94.7	92.0	98.8	91.0	97.0	73.0	99.5	97.0	96.5	77.0	99.3	90.0
	250 Bins	100.0	91.0	100.0	94.0	99.3	94.0	99.5	94.0	99.3	95.0	99.3	91.0	92.3	73.0	99.8	95.0	91.8	69.0	98.3	88.0
64 Channels	50 Bins	100.0	96.0	99.8	94.0	99.8	93.0	99.8	94.0	99.3	94.0	99.5	97.0	97.5	80.0	100.0	96.0	99.5	81.0	99.5	83.0
	125 Bins	100.0	93.0	99.3	93.0	99.5	87.0	100.0	94.0	99.5	98.0	99.8	97.0	97.8	81.0	100.0	97.0	94.7	81.0	100.0	90.0

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.8	95.0	99.5	95.0	99.2	93.0	99.2	93.0	89.5	74.0	70.8	60.0	80.0	63.0	98.2	91.0	52.8	29.0	91.5	85.0
	250 Bins	96.0	91.0	96.8	83.0	97.5	90.0	98.8	90.0	79.2	69.0	44.0	37.0	61.3	50.0	86.5	74.0	46.0	35.0	96.8	89.0
64 Channels	50 Bins	99.2	94.0	99.5	97.0	57.8	38.0	98.5	95.0	88.0	73.0	69.0	52.0	95.8	87.0	93.0	91.0	27.3	15.0	79.2	72.0
	125 Bins	99.2	90.0	100.0	98.0	56.8	42.0	90.5	93.0	86.8	70.0	69.8	49.0	98.0	85.0	79.2	75.0	11.2	8.0	64.8	59.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	98.5	92.0	98.2	94.0	83.8	77.0	99.0	91.0	81.5	75.0	78.0	74.0	76.8	64.0	96.2	93.0	69.2	48.0	93.2	84.0
	250 Bins	96.8	90.0	94.8	88.0	90.5	80.0	98.0	91.0	72.2	72.0	83.0	82.0	39.5	47.0	90.8	80.0	63.7	55.0	92.0	82.0
64 Channels	50 Bins	99.8	94.0	99.0	90.0	90.2	85.0	98.0	92.0	90.5	92.0	96.8	93.0	89.8	75.0	93.5	93.0	74.0	58.0	89.8	81.0
	125 Bins	98.2	95.0	93.5	82.0	84.8	79.0	94.0	85.0	99.0	95.0	91.2	87.0	84.5	63.0	95.0	93.0	19.5	18.0	89.8	84.0

Table A.3: Network structure c12c24f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.8	97.0	100.0	97.0	99.8	97.0	99.8	96.0	98.5	90.0	99.3	90.0	99.5	90.0	99.8	95.0	98.8	69.0	100.0	96.0
	250 Bins	99.8	97.0	99.8	96.0	100.0	96.0	100.0	96.0	98.5	84.0	98.0	89.0	98.0	89.0	99.5	96.0	97.5	67.0	99.5	92.0
64 Channels	50 Bins	100.0	96.0	100.0	98.0	100.0	97.0	99.8	97.0	99.0	79.0	98.0	77.0	99.8	84.0	99.8	97.0	99.0	63.0	100.0	97.0
	125 Bins	100.0	97.0	100.0	98.0	99.8	96.0	100.0	98.0	97.8	85.0	98.8	80.0	100.0	87.0	100.0	99.0	99.0	63.0	99.8	95.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	100.0	97.0	99.5	92.0	99.8	92.0	99.5	93.0	99.8	95.0	99.3	94.0	97.5	77.0	99.5	95.0	97.5	82.0	99.8	91.0
	250 Bins	100.0	95.0	99.5	90.0	99.8	94.0	100.0	93.0	100.0	97.0	98.8	90.0	97.3	82.0	100.0	95.0	99.0	80.0	99.5	92.0
64 Channels	50 Bins	99.8	95.0	100.0	93.0	99.5	90.0	99.8	94.0	99.0	97.0	99.8	97.0	98.0	74.0	100.0	97.0	97.0	74.0	100.0	93.0
	125 Bins	99.8	97.0	99.3	95.0	99.5	93.0	99.5	95.0	98.8	97.0	99.3	98.0	96.7	78.0	100.0	96.0	98.8	79.0	99.5	91.0

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.2	93.0	99.8	96.0	99.5	96.0	95.5	89.0	92.0	73.0	91.8	85.0	84.5	72.0	88.8	81.0	17.8	10.0	56.2	45.0
	250 Bins	99.5	97.0	99.8	93.0	99.0	94.0	99.8	96.0	97.0	80.0	85.8	74.0	63.0	53.0	97.0	89.0	44.5	29.0	91.0	80.0
64 Channels	50 Bins	99.2	94.0	98.8	95.0	99.2	94.0	95.2	95.0	77.5	58.0	71.2	56.0	97.5	77.0	87.2	84.0	9.0	9.0	74.5	68.0
	125 Bins	99.2	94.0	98.8	96.0	90.0	85.0	82.8	77.0	64.8	60.0	83.8	59.0	90.2	78.0	71.0	63.0	10.0	8.0	70.5	60.0

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	98.2	96.0	97.0	88.0	97.0	87.0	97.2	87.0	96.0	87.0	96.2	89.0	81.2	65.0	88.5	82.0	50.2	41.0	83.2	79.0
	250 Bins	98.0	91.0	98.8	92.0	97.5	87.0	98.0	88.0	82.5	78.0	96.2	87.0	78.8	64.0	97.8	91.0	50.2	40.0	91.8	89.0
64 Channels	50 Bins	97.2	93.0	98.2	91.0	96.0	91.0	93.5	84.0	99.0	95.0	97.2	90.0	89.5	67.0	92.2	85.0	60.2	49.0	77.0	70.0
	125 Bins	95.0	89.0	81.0	72.0	86.2	86.0	91.2	83.0	98.5	94.0	98.8	94.0	92.0	72.0	69.8	73.0	60.0	54.0	70.5	61.0

A.2 Model compression

This section will show the tables relating to the model compression associated to the synapses reduction and pruning step.

A.2.1 Synapse reduction model

Table A.4: Network c6c12f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Temporal Contrast																									
		Complete network								First quartile								Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.8	95.0	99.5	94.0	99.2	93.0	99.0	93.0	99.8	95.0	99.5	93.0	99.0	93.0	98.5	93.0	97.5	91.0	98.5	94.0	34.2	29.0	95.5	89.0	31.8	32.0	56.2	47.0	93.2	86.0	43.0	41.0
	250 Bins	95.2	91.0	96.5	82.0	97.5	91.0	98.5	88.0	96.8	92.0	95.8	80.0	98.2	89.0	98.0	89.0	82.8	79.0	94.0	78.0	10.0	10.0	81.0	70.0	54.5	50.0	85.5	72.0	10.5	11.0	20.0	16.0
64 Channels	50 Bins	99.2	94.0	99.8	97.0	59.0	40.0	98.2	96.0	99.2	91.0	99.5	97.0	88.5	82.0	97.8	95.0	95.5	86.0	98.8	95.0	10.0	10.0	94.8	94.0	20.8	20.0	60.5	54.0	10.0	10.0	86.0	74.0
	125 Bins	99.5	90.0	100.0	98.0	55.8	42.0	90.2	93.0	97.5	89.0	99.5	97.0	69.8	52.0	89.8	91.0	45.8	42.0	98.5	96.0	10.0	10.0	82.0	69.0	10.2	10.0	37.8	24.0	10.0	10.0	51.5	38.0

Spiking Neural Network

Gammatone filter bank		Temporal Contrast																									
		Complete network								First quartile								Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	98.8	93.0	98.2	94.0	83.8	77.0	98.8	91.0	98.5	92.0	96.5	92.0	88.8	78.0	98.8	90.0	94.5	90.0	84.5	77.0	52.0	45.0	97.8	87.0	48.8	48.0	91.5	89.0	10.0	10.0	14.2	15.0
	250 Bins	96.5	89.0	95.5	88.0	90.2	81.0	98.0	91.0	97.2	91.0	94.2	89.0	84.5	76.0	99.2	89.0	90.5	86.0	84.2	82.0	11.0	10.0	98.2	90.0	63.2	61.0	33.2	34.0	10.0	10.0	94.2	89.0
64 Channels	50 Bins	99.8	94.0	99.0	90.0	90.8	86.0	98.0	92.0	99.5	94.0	99.0	90.0	80.5	77.0	98.0	92.0	97.5	91.0	98.2	90.0	19.2	19.0	96.5	92.0	93.8	85.0	90.0	82.0	11.2	12.0	83.5	76.0
	125 Bins	98.0	95.0	93.2	82.0	84.8	79.0	93.5	85.0	98.2	96.0	93.2	84.0	71.8	65.0	92.8	85.0	85.0	80.0	83.2	81.0	22.0	17.0	92.0	82.0	33.0	30.0	46.8	49.0	9.8	10.0	81.8	68.0

Table A.5: Network c12c24f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Temporal Contrast																									
		Complete network								First quartile								Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.2	93.0	99.8	96.0	99.5	96.0	95.5	89.0	99.0	96.0	99.5	96.0	99.2	95.0	96.0	88.0	93.5	84.0	98.2	94.0	17.5	15.0	94.8	90.0	18.8	25.0	45.8	35.0	10.0	10.0	45.5	35.0
	250 Bins	99.5	97.0	99.8	92.0	99.2	94.0	99.8	95.0	99.2	97.0	99.8	91.0	95.0	84.0	99.8	96.0	97.2	93.0	96.2	91.0	10.2	10.0	99.5	96.0	49.2	49.0	46.2	36.0	10.0	10.0	90.2	85.0
64 Channels	50 Bins	99.2	94.0	98.8	96.0	99.0	93.0	95.0	94.0	99.8	93.0	98.8	96.0	98.5	90.0	94.5	93.0	99.0	93.0	96.5	91.0	97.0	94.0	93.0	90.0	37.0	40.0	47.2	40.0	10.0	10.0	68.5	65.0
	125 Bins	99.8	94.0	99.0	96.0	90.5	85.0	83.2	77.0	99.0	93.0	98.8	96.0	91.2	86.0	82.0	78.0	97.2	91.0	92.8	92.0	28.2	20.0	79.5	73.0	49.8	37.0	53.5	41.0	6.8	6.0	66.2	63.0

Spiking Neural Network

Gammatone filter bank		Temporal Contrast																									
		Complete network								First quartile								Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	98.5	96.0	96.8	88.0	97.0	88.0	97.2	87.0	97.8	95.0	96.0	87.0	96.5	86.0	97.2	86.0	65.5	66.0	92.0	83.0	94.2	85.0	88.8	83.0	18.0	16.0	57.5	46.0	49.5	38.0	70.2	69.0
	250 Bins	98.5	92.0	98.8	92.0	97.2	86.0	98.2	88.0	92.0	83.0	98.5	92.0	91.0	78.0	98.5	90.0	65.0	69.0	97.0	94.0	86.5	81.0	97.8	90.0	20.5	24.0	47.5	44.0	41.0	38.0	70.0	69.0
64 Channels	50 Bins	97.5	94.0	97.8	90.0	95.2	91.0	93.5	83.0	96.0	91.0	98.0	91.0	71.8	70.0	94.2	84.0	95.2	90.0	94.5	86.0	13.2	10.0	88.2	82.0	61.0	63.0	52.8	49.0	9.2	9.0	71.0	68.0
	125 Bins	95.8	89.0	81.2	71.0	86.0	85.0	91.2	84.0	95.0	89.0	78.8	67.0	37.5	35.0	90.0	83.0	84.8	78.0	70.2	59.0	7.8	8.0	86.5	82.0	16.0	14.0	24.0	23.0	6.5	6.0	68.5	65.0

A.2.2 Pruned model

Table A.6: Network c6c12f2 after pruning accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast															
		Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	100.0	96.0	100.0	95.0	100.0	97.0	100.0	96.0	100.0	96.0	100.0	95.0	99.8	95.0	99.8	96.0
	250 Bins	100.0	96.0	99.8	96.0	100.0	98.0	100.0	97.0	100.0	96.0	99.8	95.0	99.5	98.0	100.0	97.0
64 Channels	50 Bins	100.0	98.0	100.0	100.0	100.0	98.0	100.0	97.0	100.0	96.0	100.0	97.0	100.0	98.0	100.0	97.0
	125 Bins	99.8	98.0	100.0	98.0	100.0	97.0	100.0	98.0	99.8	97.0	100.0	99.0	99.8	98.0	100.0	99.0

Gammatone filter bank		Temporal Contrast															
		Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	100.0	95.0	100.0	96.0	100.0	89.0	100.0	95.0	100.0	95.0	99.8	96.0	99.5	88.0	100.0	94.0
	250 Bins	100.0	92.0	100.0	91.0	99.8	92.0	100.0	95.0	100.0	93.0	100.0	94.0	99.3	94.0	99.5	95.0
64 Channels	50 Bins	100.0	96.0	100.0	95.0	100.0	92.0	99.8	94.0	100.0	97.0	99.8	96.0	99.5	94.0	100.0	94.0
	125 Bins	99.8	95.0	100.0	94.0	100.0	94.0	100.0	93.0	100.0	93.0	99.8	94.0	99.3	95.0	100.0	94.0

Spiking Neural Network

Butterworth filter bank		Temporal Contrast															
		Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.8	95.0	100.0	95.0	99.0	94.0	100.0	96.0	99.2	94.0	100.0	94.0	99.2	94.0	99.0	95.0
	250 Bins	98.0	94.0	92.8	78.0	99.2	90.0	99.2	89.0	93.0	89.0	93.5	77.0	99.2	96.0	100.0	94.0
64 Channels	50 Bins	99.5	94.0	99.8	98.0	97.8	94.0	99.2	95.0	99.8	94.0	100.0	96.0	99.8	96.0	99.8	97.0
	125 Bins	99.2	94.0	100.0	98.0	86.8	78.0	94.0	93.0	99.2	94.0	99.5	99.0	96.0	90.0	96.2	97.0

Gammatone filter bank		Temporal Contrast															
		Median								Third quartile							
		TBR		SF		MW		PFM		TBR		SF		MW		PFM	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
32 Channels	50 Bins	99.2	94.0	99.2	94.0	92.5	77.0	99.2	93.0	98.0	93.0	98.8	94.0	86.2	77.0	97.2	85.0
	250 Bins	96.5	90.0	88.2	82.0	97.0	86.0	94.2	88.0	86.8	83.0	87.5	85.0	95.5	86.0	97.8	92.0
64 Channels	50 Bins	100.0	96.0	99.2	91.0	95.8	84.0	99.0	92.0	99.5	97.0	98.8	90.0	97.5	93.0	98.0	91.0
	125 Bins	98.0	94.0	96.2	88.0	95.0	90.0	96.5	86.0	98.0	95.0	91.0	81.0	87.5	89.0	93.2	84.0

Appendix B

Accuracy tables for Wireless Sensor Data Mining (WISDM) dataset

B.1 Subset1

B.1.1 Complete network

In this section is reported the train and test accuracies of the various network in the case of the WISDM in particular case for subset1, in which different encoding algorithms and feature extraction is applied.

Table B.1: Network c6c12f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	95.4	83.3	96.7	73.3	91.7	91.7	96.7	85.0	92.5	73.3	92.1	81.7	89.2	85.0	97.1	80.0	90.8	75.0	95.0	76.7
8 Channels	18 Bins	98.8	88.3	97.1	83.3	82.5	83.3	99.2	83.3	94.2	85.0	95.8	85.0	95.0	81.7	98.8	88.3	70.0	56.7	94.2	83.3
16 Channels	18 Bins	99.2	83.3	97.5	90.0	89.2	76.7	100.0	91.7	92.9	85.0	97.5	91.7	97.9	93.3	99.6	90.0	56.7	55.0	92.9	81.7

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	95.8	70.0	95.4	66.7	90.4	66.7	95.8	60.0	95.4	75.0	94.6	70.0	91.3	61.7	94.2	63.3	91.3	61.7	91.3	61.7
8 Channels	18 Bins	88.3	65.0	95.8	68.3	90.0	63.3	97.1	68.3	95.8	71.7	96.2	81.7	96.2	70.0	95.8	68.3	90.0	68.3	96.2	58.3
16 Channels	18 Bins	90.8	68.3	98.8	60.0	82.1	61.7	96.7	63.3	96.7	66.7	97.9	68.3	95.8	71.7	96.7	70.0	95.8	61.7	98.3	71.7

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	37.9	33.3	80.4	63.3	35.4	28.3	92.5	68.3	80.8	73.3	39.2	35.0	36.2	26.7	80.8	66.7	45.4	36.7	22.1	20.0
8 Channels	18 Bins	57.5	58.3	90.0	75.0	22.9	21.7	95.0	83.3	76.2	81.7	57.5	58.3	39.6	38.3	82.5	71.7	22.5	23.3	61.3	58.3
16 Channels	18 Bins	36.2	31.7	97.1	88.3	18.8	18.3	97.1	93.3	80.0	86.7	64.6	65.0	58.8	53.3	92.1	75.0	28.7	28.3	63.3	68.3

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	24.6	21.7	59.6	36.7	38.8	31.7	82.1	61.7	58.3	45.0	42.9	40.0	71.7	53.3	82.1	61.7	61.7	45.0	76.2	63.3
8 Channels	18 Bins	19.6	15.0	35.4	25.0	29.2	26.7	49.2	46.7	50.8	50.0	47.5	45.0	30.4	28.3	43.3	43.3	60.8	48.3	55.8	50.0
16 Channels	18 Bins	33.8	35.0	36.2	21.7	26.2	21.7	46.2	43.3	43.3	41.7	53.8	55.0	30.0	28.3	39.2	41.7	27.9	26.7	59.2	48.3

Table B.2: Network c12c24f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	97.1	85.0	96.7	73.3	98.8	81.7	98.3	85.0	97.1	80.0	97.9	86.7	96.7	85.0	95.8	68.3	92.9	81.7	97.5	83.3
8 Channels	18 Bins	97.1	88.3	98.8	83.3	93.3	83.3	98.3	86.7	96.2	90.0	95.8	86.7	98.3	88.3	97.9	85.0	79.2	63.3	93.8	80.0
16 Channels	18 Bins	98.8	88.3	99.2	90.0	86.3	78.3	98.3	95.0	94.2	85.0	97.5	95.0	98.3	93.3	97.9	88.3	16.7	16.7	95.0	81.7

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	95.4	61.7	97.1	73.3	95.0	68.3	96.7	65.0	96.7	66.7	95.8	76.7	92.9	75.0	96.2	66.7	95.8	63.3	96.2	70.0
8 Channels	18 Bins	92.1	68.3	94.6	75.0	93.3	63.3	98.8	66.7	97.5	75.0	90.8	75.0	95.0	71.7	96.7	61.7	90.8	65.0	97.5	68.3
16 Channels	18 Bins	94.2	60.0	97.5	70.0	89.6	65.0	96.2	63.3	97.1	75.0	96.2	76.7	95.0	63.3	95.8	73.3	96.2	60.0	95.8	73.3

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	80.4	68.3	87.1	66.7	57.9	48.3	92.5	80.0	75.4	63.3	56.2	53.3	70.8	76.7	82.5	60.0	44.6	40.0	18.3	16.7
8 Channels	18 Bins	60.4	60.0	84.6	71.7	19.6	16.7	96.7	86.7	90.8	85.0	69.6	68.3	91.7	80.0	92.5	78.3	25.4	28.3	60.8	58.3
16 Channels	18 Bins	83.3	71.7	95.8	86.7	19.6	16.7	97.1	90.0	94.6	85.0	71.7	71.7	57.5	60.0	96.7	86.7	16.7	16.7	87.5	88.3

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	45.4	41.7	67.1	53.3	50.0	45.0	87.1	61.7	72.5	53.3	54.2	48.3	37.5	31.7	86.7	61.7	75.0	55.0	74.6	66.7
8 Channels	18 Bins	40.4	38.3	53.8	36.7	28.7	20.0	62.9	56.7	48.3	40.0	52.9	48.3	30.8	28.3	70.8	56.7	62.1	48.3	67.1	51.7
16 Channels	18 Bins	47.5	43.3	55.4	46.7	35.0	21.7	64.2	50.0	31.2	26.7	65.4	58.3	31.7	31.7	50.8	46.7	46.7	40.0	68.3	50.0

B.2 Model compression

This section will show the tables relating to the model compression associated to the synapses reduction and pruning step.

B.2.1 Synapse reduction model

Table B.3: Network c6c12f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Mixed class																			
		Complete network								First quartile						Median				Third quartile							
		SF		PFM		HSA		PHASE		SF		PFM		HSA		PHASE		SF		PFM		HSA		PHASE			
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	97.1	88.3	97.1	93.3	80	86.7	92.1	75	96.7	86.7	97.9	93.3	78.3	83.3	90.8	76.7	93.3	85	67.9	68.3	66.7	71.7	85.4	78.3	68.8	65	25.4	20	17.9	16.7	68.3	61.7

Spiking Neural Network

Gammatone filter bank		Mixed class																			
		Complete network								First quartile						Median				Third quartile							
		PFM		BSA		PHASE		BURST		PFM		BSA		PHASE		BURST		PFM		BSA		PHASE		BURST			
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	82.1	61.7	71.7	53.3	82.1	61.7	76.3	63.3	82.1	63.3	65.8	50	83.3	63.3	77.9	63.3	72.1	58.3	56.3	43.3	77.1	63.3	57.5	38.3	36.7	26.7	35.8	21.7	38.3	35	29.6	23.3

Table B.4: Network c12c24f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Mixed class																			
		Complete network								First quartile						Median				Third quartile							
		SF		PFM		PHASE		BURST		SF		PFM		PHASE		BURST		SF		PFM		PHASE		BURST			
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	95.8	86.7	97.1	90	96.7	86.7	87.5	88.3	93.3	83.3	96.3	90	96.7	86.7	88.8	90	90.4	80	94.6	81.7	95.8	90	85.4	86.7	57.5	53.3	72.1	66.7	75.4	70	60	63.3

Spiking Neural Network

Gammatone filter bank		Mixed class																			
		Complete network								First quartile						Median				Third quartile							
		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST			
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	87.1	61.7	75	55	86.7	61.7	74.6	66.7	82.9	61.7	68.3	45	86.7	63.3	67.9	56.7	76.3	58.3	50	36.7	84.6	61.7	52.5	45	58.3	48.3	17.9	16.7	50.4	48.3	31.7	31.7

B.2.2 Pruned model

Table B.5: Network c12c24f2 after pruning accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Mixed class															
		Median								Third quartile							
		SF		PFM		PHASE		BURST		SF		PFM		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	100.0	93.3	100.0	96.7	100.0	88.3	97.1	80.0	99.2	93.3	99.6	94.9	99.6.8	88.3	96.7	86.7

Gammatone filter bank		Mixed class															
		Median								Third quartile							
		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	98.3	65.0	99.2	66.7	98.8	65.0	97.1	66.7	97.1	65.0	94.2	75.0	98.3	65.0	92.9	71.7

Spiking Neural Network

Butterworth filter bank		Mixed class															
		Median								Third quartile							
		SF		PFM		PHASE		BURST		SF		PFM		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	97.1	95.0	97.5	88.3	98.3	90.0	86.3	88.3	98.3	91.7	99.2	91.7	97.9	88.3	73.8	75.0

Gammatone filter bank		Mixed class															
		Median								Third quartile							
		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	86.3	61.7	75.4	56.7	85.8	60.0	72.9	61.7	76.7	56.7	68.3	56.7	84.2	56.7	58.8	50.0

B.3 Subset2

In this section is reported the train and test accuracies of the various network in the case of the WISDM in particular case for subset2, in which different encoding algorithms and feature extraction is applied.

Table B.6: Network c6c12f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	92.5	37.1	91.8	58.6	80.4	58.6	96.8	61.4	85.7	65.7	77.5	61.4	89.6	62.9	91.8	58.6	62.9	52.9	74.3	50.0
8 Channels	18 Bins	96.4	50.0	95.4	52.9	83.2	57.1	95.7	52.9	87.9	68.6	76.4	65.7	93.2	71.4	92.1	57.1	47.9	38.6	72.5	58.6
16 Channels	18 Bins	95.7	51.4	95.0	48.6	93.9	50.0	95.4	62.9	96.4	64.3	91.1	67.1	96.4	67.1	98.2	61.4	14.6	14.3	68.6	58.6

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	88.9	38.6	88.9	41.4	91.4	37.1	87.9	51.4	82.9	37.1	88.9	45.7	82.5	42.9	91.8	45.7	82.5	42.9	77.5	47.1
8 Channels	18 Bins	92.9	40.0	92.5	31.4	85.0	34.3	92.1	45.7	89.6	48.6	88.2	45.7	90.7	48.6	92.1	44.3	91.1	42.9	89.6	44.3
16 Channels	18 Bins	91.8	32.9	94.3	34.3	93.6	38.6	97.1	48.6	94.6	50.0	91.8	50.0	92.9	44.3	95.4	58.6	94.6	44.3	94.3	44.3

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	33.2	15.7	72.1	50.0	27.1	25.7	81.1	57.1	42.5	27.1	45.0	42.9	46.4	32.9	59.3	48.6	23.2	21.4	28.6	24.3
8 Channels	18 Bins	50.4	30.0	72.9	48.6	41.8	32.9	73.2	50.0	34.3	25.7	46.1	42.9	69.6	54.3	76.8	51.4	14.3	14.3	41.8	44.3
16 Channels	18 Bins	70.7	37.1	76.4	40.0	46.8	28.6	71.4	52.9	50.4	34.3	64.6	55.7	42.9	32.9	84.6	55.7	14.3	14.3	43.2	42.9

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	42.5	24.3	54.3	31.4	36.1	17.1	67.5	45.7	66.4	28.6	43.2	38.6	32.9	22.9	56.4	42.9	52.1	28.6	63.6	38.6
8 Channels	18 Bins	25.4	21.4	54.6	20.0	30.0	18.6	51.4	38.6	32.5	25.7	23.6	20.0	14.3	14.3	51.8	42.9	22.1	10.0	45.7	35.7
16 Channels	18 Bins	30.0	12.9	53.6	24.3	25.7	11.4	43.2	27.1	33.2	22.9	25.0	21.4	19.3	14.3	42.1	31.4	30.0	15.7	47.5	34.3

Table B.7: Network c12c24f2 accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	93.6	42.9	95.0	50.0	95.7	48.6	96.1	55.7	93.9	70.0	78.2	68.6	94.6	67.1	96.8	55.7	50.4	40.0	75.4	50.0
8 Channels	18 Bins	94.3	57.1	96.1	52.9	86.4	60.0	94.6	62.9	93.2	61.4	83.2	62.9	96.4	72.9	96.8	61.4	42.1	34.3	71.1	60.0
16 Channels	18 Bins	94.6	62.9	94.6	52.9	93.9	38.6	95.0	61.4	97.1	58.6	96.4	67.1	97.9	67.1	95.7	62.9	14.3	14.3	69.3	52.9

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	93.9	30.0	92.9	28.6	93.9	38.6	90.4	47.1	80.4	48.6	92.9	48.6	91.4	48.6	91.4	51.4	89.3	42.9	88.6	44.3
8 Channels	18 Bins	93.9	41.4	93.6	40.0	92.1	40.0	95.4	52.9	94.6	44.3	96.1	50.0	90.7	47.1	95.7	58.6	92.9	41.4	95.0	51.4
16 Channels	18 Bins	93.9	38.6	92.9	31.4	95.4	37.1	95.4	40.0	95.4	47.1	96.4	52.9	93.2	41.4	94.6	51.4	93.2	40.0	93.6	44.3

Spiking Neural Network

Butterworth filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	60.0	22.9	61.4	34.3	71.4	41.4	64.3	51.4	46.1	31.4	50.7	47.1	35.0	31.4	69.6	52.9	18.6	20.0	22.9	17.1
8 Channels	18 Bins	66.8	42.9	78.9	48.6	57.9	38.6	70.4	52.9	59.3	52.9	58.9	51.4	65.4	52.9	78.6	62.9	21.1	18.6	44.6	47.1
16 Channels	18 Bins	76.8	47.1	86.8	54.3	58.6	30.0	81.8	64.3	76.8	47.1	77.5	65.7	70.0	52.9	85.0	57.1	14.3	14.3	43.6	42.9

Gammatone filter bank		Temporal Contrast								Filter & Oprimizer						Global Referenced				Latency/ISI	
		TBR		SF		MW		PFM		HSA		MHSA		BSA		PHASE		TTFS		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	64.6	30.0	43.6	27.1	51.4	27.1	81.8	48.6	67.5	48.6	64.3	44.3	27.5	17.1	72.5	45.7	52.1	40.0	66.4	44.3
8 Channels	18 Bins	43.9	25.7	56.1	32.9	42.5	22.9	61.1	37.1	40.4	28.6	33.2	24.3	44.6	30.0	64.6	44.3	60.7	40.0	51.8	38.6
16 Channels	18 Bins	57.1	18.6	65.4	27.1	50.4	15.7	70.4	35.7	47.1	27.1	48.6	28.6	24.3	20.0	61.4	40.0	17.5	14.3	47.1	31.4

B.4 Model compression

This section will show the tables relating to the model compression associated to the synapses reduction and pruning step.

B.4.1 Synapse reduction model

Table B.8: Network c6c12f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Mixed class																									
		Complete network								First quartile								Median								Third quartile			
		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	71.4	52.9	64.6	55.7	84.6	55.7	43.2	42.9	65.7	48.6	62.9	52.9	80.7	50	39.6	41.4	54.6	30	58.6	52.9	70	40	35.7	37.1	49.3	27.1	45	42.9	54.6	37	27.9	24.3

Spiking Neural Network

Gammatone filter bank		Mixed class																									
		Complete network								First quartile								Median								Third quartile			
		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST		PFM		MHSA		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	67.5	45.7	43.2	38.6	56.4	42.9	63.6	38.6	66.1	47.1	62.5	40	58.2	44.3	57.9	34.3	53.6	42.9	44.3	38.6	58.2	48.6	39.6	30	26.8	22.9	32.9	30	42.5	30	26.1	18.6

Table B.9: Network c12c24f2 after synapses reduction accuracy (%)

Spiking Neural Network

Butterworth filter bank		Mixed class																									
		Complete network								First quartile								Median								Third quartile			
		SF		PFM		MHSA		PHASE		SF		PFM		MHSA		PHASE		SF		PFM		MHSA		PHASE		SF		PFM		MHSA		PHASE	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	86.8	54.3	81.8	64.3	77.5	65.7	85	57	86.1	52.9	81.9	62.9	74.6	65.7	85.7	60	85	51.4	76.4	62.9	67.9	64.3	85.7	58.6	52.5	35.7	35	22.9	61.4	57.1	63.2	50

Spiking Neural Network

Gammatone filter bank		Mixed class																									
		Complete network								First quartile								Median								Third quartile			
		PFM		HSA		PHASE		BURST		PFM		HSA		PHASE		BURST		PFM		HSA		PHASE		BURST		PFM		HSA		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	81.8	48.6	67.5	48.6	72.5	45.7	66.4	44.3	80.7	45.7	63.9	47.1	72.9	47.1	64.3	44.3	75.7	41.4	45.7	37.1	69.3	41.4	66.1	41.4	43.6	24.3	15	20	39.6	30	45	27.1

B.4.2 Pruned model

Table B.10: Network c12c24f2 after pruning accuracy (%)

Convolutional Neural Network

Butterworth filter bank		Mixed class															
		Median								Third quartile							
		SF		PFM		MHSA		PHASE		SF		PFM		MHSA		PHASE	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	99.3	57.1	98.9	57.1	97.9	71.4	97.5	62.9	95.4	55.7	97.1	60.0	95.7	68.6	96.8	61.4

Gammatone filter bank		Mixed class															
		Median								Third quartile							
		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	93.9	52.9	92.1	41.4	96.1	51.4	95.0	41.4	90.4	50.0	87.1	42.8	92.1	57.1	88.6	42.9

Spiking Neural Network

Butterworth filter bank		Mixed class															
		Median								Third quartile							
		SF		PFM		MHSA		PHASE		SF		PFM		MHSA		PHASE	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
16 Channels	18 Bins	86.3	50.0	82.1	53.3	77.1	60.0	87.1	58.3	75.0	43.3	66.7	48.3	70.8	60.0	72.1	53.3

Gammatone filter bank		Mixed class															
		Median								Third quartile							
		PFM		TTFS		PHASE		BURST		PFM		TTFS		PHASE		BURST	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
4 Channels	24 Bins	82.9	43.3	57.1	30.0	76.7	46.7	63.3	43.3	63.3	36.7	50.0	38.3	70.4	53.3	57.1	41.7

Acknowledgements

First of all I would like to thank my mother, who with great sacrifices has allowed the completion of this long journey, giving me the privilege of following my passions, something for which I am immensely grateful, supporting me morally and economically, encouraging me to do better and supporting myself in the most difficult moments, always believing in me. I would also like to thank my brother and my sister who encouraged me with great sarcasm, and my uncles and in particular my grandfather who are always available to listen and provide precious advice.

A special thanks goes to my friends met during this journey, without whom it would not have been the same, Andrea Ser., Andrea Ste., Cecilia, Ettore, Francesca, Francesco, Georgi, Nicola, Raffaele, who each of them has contributed in a significant way during my personal growth, transforming very difficult moments into moments of great joy.

Last but certainly not least, I would like to thank my supervisor Gianvito Urgese for giving me the opportunity to develop a thesis on a topic that fascinates me a lot, allowing me to better understand my passions, and together with my co-advisors Evelina Forno and Vittorio Fra, who with great patience and dedication have constantly supported and encouraged me throughout my path, always fueling my enthusiasm.

Ringraziamenti

Per prima cosa vorrei ringraziare mia madre, che con grandi sacrifici ha permesso il compimento di questo lungo viaggio, dandomi il privilegio di seguire le mie passioni, cosa di cui ne sono immensamente grato, supportandomi moralmente ed economicamente, spronandomi a fare sempre meglio e sorreggendomi nei momenti più difficili, credendo sempre in me. Vorrei inoltre ringraziare mio fratello e mia sorella che con grande sarcasmo mi hanno incoraggiato, e hai miei zii e in particolare a mio nonno sempre disponibili ad ascoltare e fornire preziosi consigli.

Un ringraziamento speciale va poi ai miei amici incontrati durante questo viaggio, senza i quali non sarebbe stato lo stesso, Andrea Ser., Andrea Ste., Cecilia, Ettore, Francesca, Francesco, Georgi, Nicola, Raffaele, i quali ognuno di loro ha contribuito in un modo significativo durante la mia crescita personale, trasformando momenti molto difficili, in momenti di grande gioia.

Per ultimi ma di certo non per importanza, vorrei ringraziare il mio relatore Gianvito Urgese per avermi dato l'opportunità di svolgere una tesi in un argomento che mi affascina molto, permettendomi di comprendere meglio le mie passioni, e assieme ai miei co-advisor Evelina Forno e Vittorio Fra, che con grande pazienza e dedizione mi hanno costantemente supportato e incoraggiato durante tutto il mio percorso, alimentando sempre il mio entusiasmo.

Bibliography

[1] Dan Claudiu Ciresan et al. «Convolutional Neural Network Committees for Handwritten Character Classification». In: *2011 International Conference on Document Analysis and Recognition*. IEEE, Sept. 2011, pp. 1135–1139. ISBN: 978-1-4577-1350-7. DOI: 10.1109/ICDAR.2011.229. URL: http://ieeexplore.ieee.org/document/6065487/.

[2] Iuri Frosio, Karen Egiazarian, and Kari Pulli. «Machine learning for adaptive bilateral filtering». In: ed. by Karen O. Egiazarian, Sos S. Agaian, and Atanas P. Gotchev. Mar. 2015, p. 939908. DOI: 10.1117/12.2077733. URL: http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2077733.

[3] Theodoros Theodorou, Iosif Mporas, and Nikos Fakotakis. «An Overview of Automatic Audio Segmentation». In: *International Journal of Information Technology and Computer Science* 6.11 (Oct. 2014), pp. 1–9. ISSN: 20749007. DOI: 10.5815/ijitcs.2014.11.01. URL: http://www.mecs-press.org/ijitcs/ijitcs-v6-n11/v6n11-1.html.

[4] Stuart Jonathan Russell and Peter Norvig. *Artificial Intelligence*. Feb. 2022. ISBN: 9780137903955.

[5] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. «CIFAR-100 (Canadian Institute for Advanced Research)». In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[6] Gary M. Weiss. «WISDM Smartphone and Smartwatch Activity and Biometrics Dataset». In: *UCI Machine Learning Repository: WISDM Smartphone and Smartwatch Activity and Biometrics Dataset Data Set* 7 (2019), pp. 133190–133202.

[7] J.C. Bezdek et al. «Will the real iris data please stand up?» In: *IEEE Transactions on Fuzzy Systems* 7.3 (June 1999), pp. 368–369. ISSN: 10636706. DOI: 10.1109/91.771092. URL: http://ieeexplore.ieee.org/document/771092/.

[8] David Silver et al. «Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm». In: (Dec. 2017). arXiv: 1712.01815. URL: http://arxiv.org/abs/1712.01815.

[9] David Silver et al. «Mastering the game of Go without human knowledge». In: *Nature* 550.7676 (Oct. 2017), pp. 354–359. ISSN: 0028-0836. DOI: 10.1038/nature24270. URL: http://www.nature.com/articles/nature24270.

[10] Volodymyr Mnih et al. «Playing Atari with Deep Reinforcement Learning». In: (Dec. 2013). DOI: 1312.5602. arXiv: 1312.5602. URL: http://arxiv.org/abs/1312.5602.

[11] D F Karya, P Katias, and T Herlambang. «Stock price estimation using ensemble Kalman Filter square root method». In: *Journal of Physics: Conference Series* 1008 (Apr. 2018), p. 012017. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1008/1/012017. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1008/1/012017.

[12] Stanley Baek et al. «Accurate vehicle position estimation using a Kalman filter and neural network-based approach». In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, Nov. 2017, pp. 1–8. ISBN: 978-1-5386-2726-6. DOI: 10.1109/SSCI.2017.8285360. URL: http://ieeexplore.ieee.org/document/8285360/.

[13] Hardie Cate, Fahim Dalvi, and Zeshan Hussain. «DeepFace: Face Generation using Deep Learning». In: (Jan. 2017). arXiv: 1701.01876. URL: http://arxiv.org/abs/1701.01876.

[14] *Carver Mead - Caltech*. http://www.carvermead.caltech.edu/index.html.

[15] Shih Chii Liu et al. «Asynchronous binaural spatial audition sensor with 2×64×4 Channel output». In: *IEEE Transactions on Biomedical Circuits and Systems* 8.4 (2014), pp. 453–464. ISSN: 19324545. DOI: 10.1109/TBCAS.2013.2281834.

[16] Misha A. Mahowald and Carver Mead. «The silicon retina.» In: *Scientific American* 264 5 (1991), pp. 76–82.

[17] «Loihi: A Neuromorphic Manycore Processor with On-Chip Learning». In: *IEEE Micro* 38.1 (Jan. 2018), pp. 82–99. ISSN: 0272-1732. DOI: 10.1109/MM.2018.112130359. URL: http://ieeexplore.ieee.org/document/8259423/.

[18] Wolfgang Maass. «Networks of spiking neurons: The third generation of neural network models». In: *Neural Networks* 10.9 (1997), pp. 1659–1671. ISSN: 08936080. DOI: 10.1016/S0893-6080(97)00011-7.

[19] «Medical gallery of Blausen Medical 2014». In: *WikiJournal of Medicine* 1.2 (2014). ISSN: 20024436. DOI: 10.15347/wjm/2014.010. URL: https://en.wikiversity.org/wiki/WikiJournal_of_Medicine/Medical_gallery_of_Blausen_Medical_2014.

[20] A. L. Hodgkin and A. F. Huxley. «A quantitative description of membrane current and its application to conduction and excitation in nerve». In: *The Journal of Physiology* 117.4 (Aug. 1952), pp. 500–544. ISSN: 0022-3751. DOI: 10.1113/jphysiol.1952.sp004764. URL: https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1952.sp004764.

[21] *Nernst equation - Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/Nernst_equation.

[22] L.F Abbott. «Lapicque's introduction of the integrate-and-fire model neuron (1907)». In: *Brain Research Bulletin* 50.5-6 (Nov. 1999), pp. 303–304. ISSN: 03619230. DOI: 10.1016/S0361-9230(99)00161-6. URL: https://linkinghub.elsevier.com/retrieve/pii/S0361923099001616.

[23] Roger M. Enoka and Jacques Duchateau. «Rate Coding and the Control of Muscle Force». In: *Cold Spring Harbor Perspectives in Medicine* 7.10 (Oct. 2017), a029702. ISSN: 2157-1422. DOI: 10.1101/cshperspect.a029702. URL: http://perspectivesinmedicine.cshlp.org/lookup/doi/10.1101/cshperspect.a029702.

[24] Wulfram Gerstner et al. *Neuronal dynamics: From single neurons to networks and models of cognition*. 2014, pp. 1–577. ISBN: 9781107447615. DOI: 10.1017/CBO9781107447615.

[25] «A Survey of Encoding Techniques for Signal Processing in Spiking Neural Networks». In: *Neural Processing Letters* 53.6 (Dec. 2021), pp. 4693–4710. ISSN: 1370-4621. DOI: 10.1007/s11063-021-10562-2. URL: https://link.springer.com/10.1007/s11063-021-10562-2.

[26] Dzung L. Pham, Chenyang Xu, and Jerry L. Prince. «Current Methods in Medical Image Segmentation». In: *Annual Review of Biomedical Engineering* 2.1 (Aug. 2000), pp. 315–337. ISSN: 1523-9829. DOI: 10.1146/annurev.bioeng.2.1.315. URL: https://www.annualreviews.org/doi/10.1146/annurev.bioeng.2.1.315.

[27] *Medel - Blog*. https://blog.medel.com/how-the-cochlear-understands-so-many-different-sounds/.

[28] Daniel L Schacter, Daniel T Gilbert, and Daniel M Wegner. *Psychology*. 2nd ed. New York, NY: Worth, May 2011.

[29] Alice F Healy. *Handbook of psychology: Experimental psychology v. 4*. en. Ed. by Alice F Healy and Robert W Proctor. Handbook of Psychology. 12 Volume Set. Nashville, TN: John Wiley & Sons, Jan. 2003.

[30] By James T George and Elizabeth Elias. «A 16-Band Reconfigurable Hearing Aid using Variable Bandwidth Filters». In: *Global Journal of Researches in Engineering: f Electrical and Electronics Engineering* 14.1 (2014).

[31] E. Ambikairajah, J. Epps, and L. Lin. «Wideband speech and audio coding using gammatone filter banks». In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2.February (2001), pp. 773–776. ISSN: 15206149. DOI: 10.1109/icassp.2001.941029.

[32] A. G. Katsiamis, E. M. Drakakis, and R. F. Lyon. «Practical gammatone-like filters for auditory processing». In: *Eurasip Journal on Audio, Speech, and Music Processing* 2007 (2007). ISSN: 16874714. DOI: 10.1155/2007/63685.

[33] E. D. Adrian and Yngve Zotterman. «The impulses produced by sensory nerve endings». In: *The Journal of Physiology* 61.4 (Aug. 1926), pp. 465–483. ISSN: 00223751. DOI: 10.1113/jphysiol.1926.sp002308. URL: https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.1926.sp002308.

[34] T. J. Gawne, T. W. Kjaer, and B. J. Richmond. «Latency: another potential code for feature binding in striate cortex». In: *Journal of Neurophysiology* 76.2 (Aug. 1996), pp. 1356–1360. ISSN: 0022-3077. DOI: 10.1152/jn.1996.76.2.1356. URL: https://www.physiology.org/doi/10.1152/jn.1996.76.2.1356.

[35] Troy W. Margrie and Andreas T. Schaefer. «Theta oscillation coupled spike latencies yield computational vigour in a mammalian sensory system». In: *The Journal of Physiology* 546.2 (Jan. 2003), pp. 363–374. ISSN: 0022-3751. DOI: 10.1113/jphysiol.2002.031245. URL: https://onlinelibrary.wiley.com/doi/10.1113/jphysiol.2002.031245.

[36] Roland S Johansson and Ingvars Birznieks. «First spikes in ensembles of human tactile afferents code complex spatial fingertip events». In: *Nature Neuroscience* 7.2 (Feb. 2004), pp. 170–177. ISSN: 1097-6256. DOI: 10.1038/nn1177. URL: http://www.nature.com/articles/nn1177.

[37] Robert Galambos and Hallowell Davis. «THE RESPONSE OF SINGLE AUDITORY-NERVE FIBERS TO ACOUSTIC STIMULATION». In: *Journal of Neurophysiology* 6.1 (Jan. 1943), pp. 39–57. ISSN: 0022-3077. DOI: 10.1152/jn.1943.6.1.39. URL: https://www.physiology.org/doi/10.1152/jn.1943.6.1.39.

[38] Simon Thorpe, Denis Fize, and Catherine Marlot. «Speed of processing in the human visual system». In: *Nature* 381.6582 (June 1996), pp. 520–522. ISSN: 0028-0836. DOI: 10.1038/381520a0. URL: http://www.nature.com/articles/381520a0.

[39] Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience*. Aug. 2005. ISBN: 9780262541855.

[40] David Heeger. «Poisson Model of Spike Generation». In: *Handout* (2000), pp. 1–13. URL: http://www.cns.nyu.edu/$%5Csim$david/handouts/poisson.pdf.

[41] Elisa Donati et al. «Discrimination of EMG Signals Using a Neuromorphic Implementation of a Spiking Neural Network». In: *IEEE Transactions on Biomedical Circuits and Systems* 13.5 (2019), pp. 793–801. ISSN: 19409990. DOI: 10.1109/TBCAS.2019.2925454.

[42] Chang Gao et al. «Real-time speech recognition for iot purpose using a delta recurrent neural network accelerator». In: *Proceedings - IEEE International Symposium on Circuits and Systems* 2019-May (2019), pp. 6–10. ISSN: 02714310. DOI: 10.1109/ISCAS.2019.8702290.

[43] Nik Dennler et al. «Online Detection of Vibration Anomalies Using Balanced Spiking Neural Networks». In: *2021 IEEE 3rd International Conference on Artificial Intelligence Circuits and Systems, AICAS 2021* 1 (2021), pp. 1–4. DOI: 10.1109/AICAS51828.2021.9458403. arXiv: 2106.00687.

[44] Enea Ceolini et al. «Event-driven Pipeline for Low-latency Low-compute Keyword Spotting and Speaker Verification System». In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* 2019-May (2019), pp. 7953–7957. ISSN: 15206149. DOI: 10.1109/ICASSP.2019.8683669.

[45] Federico Corradi and Giacomo Indiveri. «A Neuromorphic Event-Based Neural Recording System for Smart Brain-Machine-Interfaces». In: *IEEE Transactions on Biomedical Circuits and Systems* 9.5 (2015), pp. 699–709. ISSN: 19324545. DOI: 10.1109/TBCAS.2015.2479256.

[46] Minhao Yang et al. «A 0.5 V 55 μW 64 $\times$ 2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing». In: *IEEE Journal of Solid-State Circuits* 51.11 (2016), pp. 2554–2569. ISSN: 00189200. DOI: 10.1109/JSSC.2016.2604285.

[47] «SPIKER : Analog Waveform to Digital Spiketrain Conversion in ATR ' s Arti cial Brain (CAM-Brain) Project Abstract 1 Introduction 3 Hough Spiker Algorithm (HSA) 2 Spike Interval Information Coding (SIIC)». In: 1 (), pp. 3–6.

[48] Julien Dupeyroux. «A toolbox for neuromorphic sensing in robotics». In: (Mar. 2021).

[49] Christoph Kayser et al. «Spike-Phase Coding Boosts and Stabilizes Information Carried by Spatial and Temporal Spike Patterns». In: *Neuron* 61.4 (2009), pp. 597–608. ISSN: 08966273. DOI: 10.1016/j.neuron.2009.01.008. URL: http://dx.doi.org/10.1016/j.neuron.2009.01.008.

[50] Jaehyun Kim et al. «Deep neural networks with weighted spikes». In: *Neurocomputing* 311 (2018), pp. 373–386. ISSN: 18728286. DOI: 10.1016/j.neucom.2018.05.087.

[51] Wenzhe Guo et al. «Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems». In: *Frontiers in Neuroscience* 15.March (2021), pp. 1–21. ISSN: 1662453X. DOI: 10.3389/fnins.2021.638474.

[52] Zihan Pan et al. «An Efficient and Perceptually Motivated Auditory Neural Encoding and Decoding Algorithm for Spiking Neural Networks». In: *Frontiers in Neuroscience* 13.01 (2020), pp. 1–17. ISSN: 1662453X. DOI: 10.3389/fnins.2019.01420. arXiv: 1909.01302.

[53] Jackson Zohar. «Free Spoken Digit Dataset». 2020. DOI: 10.5281/zenodo.1342401. URL: https://github.com/Jakobovski/free-spoken-digit-dataset.

[54] Gary M. Weiss, Kenichi Yoneda, and Thaier Hayajneh. «Smartphone and Smartwatch-Based Biometrics Using Activities of Daily Living». In: *IEEE Access* 7 (2019), pp. 133190–133202. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2940729. URL: https://ieeexplore.ieee.org/document/8835065/.

[55] Jyotibdha Acharya et al. «A comparison of low-complexity real-time feature extraction for neuromorphic speech recognition». In: *Frontiers in Neuroscience* 12.MAR (2018), pp. 1–15. ISSN: 1662453X. DOI: 10.3389/fnins.2018.00160.

[56] Jithendar Anumula et al. «Feature representations for neuromorphic audio spike streams». In: *Frontiers in Neuroscience* 12.FEB (2018), pp. 1–12. ISSN: 1662453X. DOI: 10.3389/fnins.2018.00023.

[57] Juan P. Dominguez-Morales et al. «Deep Spiking Neural Network model for time-variant signals classification: A real-time speech recognition approach». In: *Proceedings of the International Joint Conference on Neural Networks* 2018-July (2018). DOI: 10.1109/IJCNN.2018.8489381.

[58] Qian Liu, Yunhua Chen, and Steve Furber. «Noisy Softplus: an activation function that enables SNNs to be trained as ANNs». In: (2017). ISSN: 2331-8422. arXiv: 1706.03609. URL: http://arxiv.org/abs/1706.03609.

[59] Vittorio Fra et al. «Human activity recognition: suitability of a neuromorphic approach for on-edge AIoT applications». In: *Neuromorphic Computing and Engineering* 2.1 (Mar. 2022), p. 014006. ISSN: 2634-4386. DOI: 10.1088/2634-4386/ac4c38. URL: https://iopscience.iop.org/article/10.1088/2634-4386/ac4c38.

[60] Aaron R. Voelker, Ivana Kajic, and Chris Eliasmith. «Legendre memory units: Continuous-time representation in recurrent neural networks». In: *Advances in Neural Information Processing Systems* 32.NeurIPS (2019). ISSN: 10495258.