Spiking Neural Networks on EEG data processing

André Furlan

Universidade Estadual Paulista Júlio de Mesquita Filho

2023

Introduction

Introduction I

The databse and sources used in this work are available in ⟨https://github.com/ensismoebius/deepLearnning⟩):

Spiking Neural Networks (SNNs) mimic in some degree the workings of the brain by utilizing action potentials, in contrast to continuous values transmitted between neurons in more traditional neural networks.

The term "Spiking" originates from the behavior of biological neurons, which sporadically emit action potentials, creating voltage spikes that convey information (KASABOV, 2019). Figure 1 illustrates these spikes.

An SNN **is not** a one-to-one simulation of neurons. Instead, it approximates certain computational capabilities of specific biological properties. Some studies like (JONES; KORDING, 2020) created models closer to natural neurons exploring the nonlinearity of dendrites and other neuron features yielding remarkable results in the classification.

Introduction II

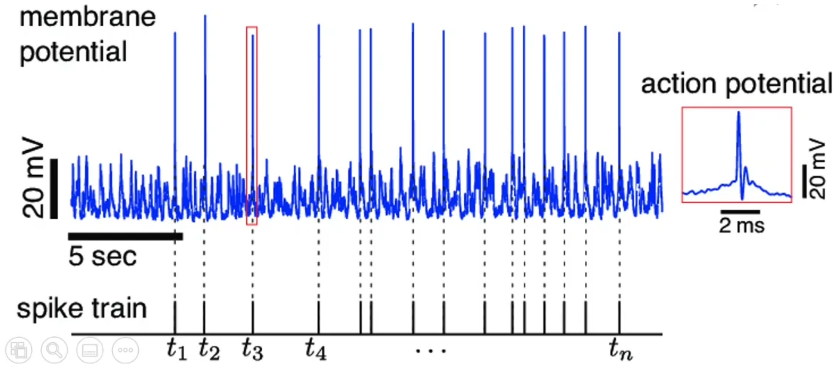


Figure: Spikes from a noisy signal. Source (GOODMAN et al., 2022)

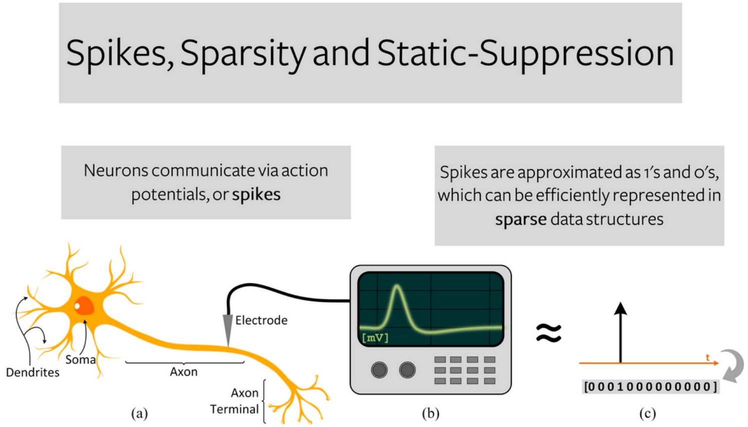
Characteristics

General characteristics I

SNNs possess several noteworthy characteristics that distinguish them from traditional machine learning techniques, including classical neural networks. These distinctions encompass (KASABOV, 2019):

- Proficiency in modeling temporal, spatial-temporal, or spectro-temporal data.
- Effectiveness in capturing processes involving various time scales.
- Seamless integration of multiple modalities, such as sound and vision, into a unified system.
- Aptitude for predictive modeling and event prediction.
- Swift and highly parallel information processing capabilities.
- Streamlined information processing.
- Scalability, accommodating structures ranging from a few tens to billions of spiking neurons.
- Minimal energy consumption when implemented on neuromorphic platforms.

General characteristics II


Spikes, Sparsity and Static-Suppression

Neurons communicate via action potentials, or **spikes**

Spikes are approximated as 1's and 0's, which can be efficiently represented in **sparse** data structures

Figure: Sparsity on Spike Neuron Networks. Source: (ESHRAGHIAN et al., 2023)

Energy characteristics I

Spiking Neural Networks (SNNs) are often considered power-efficient for several reasons:
Event-Driven Processing: SNNs are inherently event-driven. Instead of constantly updating neuron activations and synapse weights as in traditional artificial neural networks (ANNs), SNNs only transmit spikes (action potentials) when a neuron's membrane potential reaches a certain threshold. This event-driven approach reduces the amount of computation required and can lead to significant energy savings.
Sparse Activity: SNNs tend to exhibit sparse activity, meaning that only a small percentage of neurons are active at any given time. This sparsity reduces the number of computations that need to be performed, which is especially beneficial for hardware implementations where most of the energy consumption comes from active components. See Figure 3.
Low Precision: SNNs can often work with lower precision than ANNs. While ANNs typically use high-precision floating-point numbers for neuron activations and synaptic weights, SNNs can use lower precision fixed-point or binary representations. Lower precision computations require less energy to perform.

Energy characteristics II

Neuromorphic Hardware: SNNs can be efficiently implemented on specialized neuromorphic hardware, which is designed to mimic the energy-efficient behavior of biological neural systems. These hardware platforms are optimized for the event-driven nature of SNNs, further reducing power consumption.
Energy-Aware Learning Rules: SNNs can employ learning rules that take into account energy efficiency. For example, some learning rules prioritize strengthening or weakening synapses based on their contribution to network activity, which can lead to more energy-efficient learning.
Spike Encoding: SNNs can encode information in the timing and frequency of spikes, which can be a highly efficient way to represent and process data, particularly for event-based sensors like vision sensors or auditory sensors.

Energy characteristics III

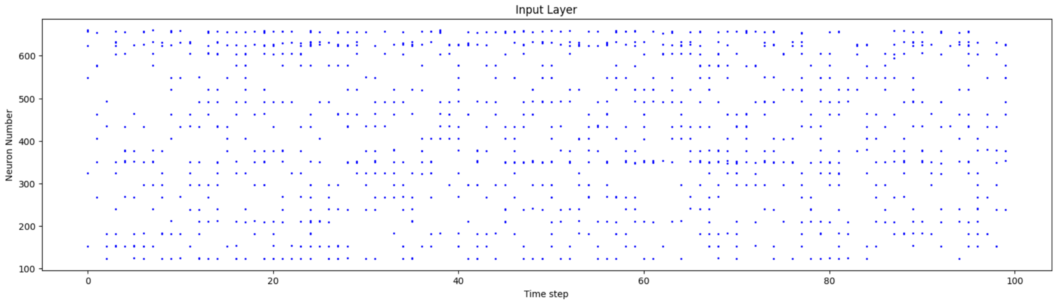


Figure: Sparse activity of a SNN

Theory

Theory: Spikes and potentials I

In order to emulate such behavior, let's begin with a simple model: The "Leaky Integrate and Fire neuron" (LIF). The LIF model describes the evolution of membrane potential which the potential decay over time.

$$\tau \cdot \frac{dV}{dt} = -V \tag{1}$$

When a neuron receives a spike, the membrane potential V increases according to a synaptic weight w.

$$V = V + w \tag{2}$$

Such behaviors are depicted in Figure 4.

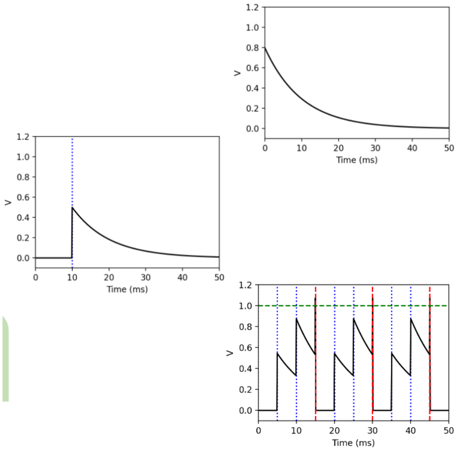


Figure: Evolution of a Spike. Source (GOODMAN et al., 2022)

Theory: Spikes and potentials II

A spike neuron acts like an **resistor-capacitor** circuit (RC).

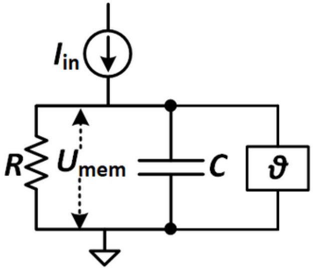


Figure: Resistor-capacitor circuit model. Source: (ESHRAGHIAN et al., 2023)

```
def convert_to_time(data, tau=5, threshold=0.01):
  spike_time = tau * torch.log(data / (data -
      threshold))
  return spike_time
```

Listing: "Code for RC model"

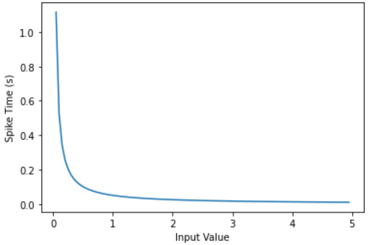


Figure: The smaller the value, the later the spike occurs with exponential dependence

Theory: Spikes and potentials III

```python

class Neuron:
    """
    Leak integrate and fire neuron — LIF

    """
    def __init__(self, tau=10, threshold=1):

        # Initial membrane voltage
        self.voltage = 0

        # The smaller tau is the faster the voltage decays
        # When tau is large the neuron acts as an intergrator summing its inputs
        # and firing when a certain threshold is reached.
        # When tau is small the neuron acts as a coincidence detector, firing a
        # spike only when two or more input arrive simultaneosly.
        self.tau = tau

        # The threshold above which the neuron fires
        self.threshold = threshold

        # Time step for decaying (i still don't known what this really is)
        # Bigger the number faster the decay
        self.timeStep = 0.5

        # The rate by which the membrane voltage decays each time step
        self.alpha = np.exp(-self.timeStep/self.tau)

```

Theory: Spikes and potentials IV

```
    def set_tau(self, tau):
        self.tau = tau
        self.alpha = np.exp(-self.timeStep/self.tau)

    def fire_spike(self):
        if self.voltage > self.threshold:
            self.voltage = 0
            return 1
        return 0

    def add_synaptic_weight(self, weigth):
        # Membrane voltage integration
        self.voltage += weigth

    def iterate(self):
        # Membrane voltage leak
        self.voltage = max(self.voltage * self.alpha, 0)
        return self.fire_spike()
```

Listing: "Leak integrate and fire neuron - LIF"

Demonstration: Simulating a spike neuron I

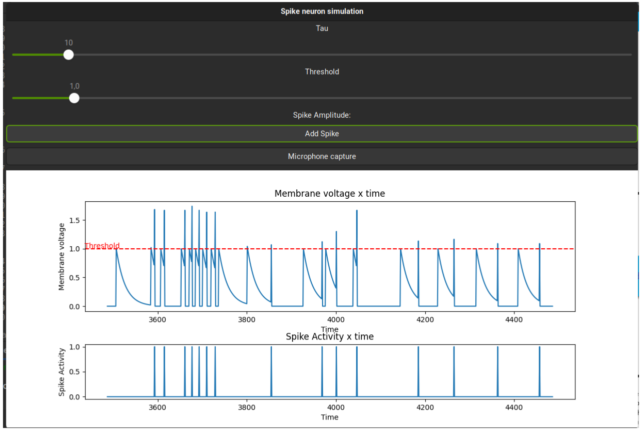


Figure: A simulation a of spiking neuron

Theory: Training overview

As shown in Figure 4, when a neuron reaches a certain threshold, it resets ($V = 0$). Note that $V = 0$ is just one of the possible values of reset, after this the neuron may or may not enter in a refractory period. **These, an another values cited ahead, can be parameterized or learned**.

How do SNNs get trained? Well, this is still an open question. An SNN neuron has an activation-function behavior that is more relatable to a **step function**. Therefore, in principle, we can't use gradient descent-based solutions because this kind of function **is not** differentiable (KASABOV, 2019).

But there are some insights out there that may shed some light on this subject: While some *in vivo/in vitro* observations show that brains, in general, learn by strengthening/weakening and adding/removing synapses or even by creating new neurons or other cumbersome methods like RNA packets, there are some more acceptable ones like the ones in the slide (KASABOV, 2019):

Theory: Training overview

- Spike Timing-Dependent Plasticity (STDP): The idea is that if a pre-synaptic neuron fires **before** the post-synaptic one, there is a strengthening in connection, but if the post-synaptic neuron fires before, then there is a weakening.
- Surrogate Gradient Descent: The technique **approximates** the step function by using another mathematical function, which is differentiable (like a sigmoid), in order to train the network. These approximations are used only **in the backward pass**, while keeping the step function in the forward pass (KASABOV, 2019).
- Evolving Algorithms: Use the selection of the fittest throughout many generations of networks.
- Reservoir/Dynamic Computing: **Echo state networks** or **Liquid state machines** respectively. These will be discussed further in this presentation.

Theory: Surrogate gradient descent

Gradient descent is a well known method to adjust the weights of a network, however it can't be used **directly** in SNNs because its behavior on activation obeys a **Heaviside function** which is not differentiable and its gradients are all zero.

Surrogate gradient descent is a method in which the derivative of the *sigmoid* or another similar function is used just in the **backward pass**.

```python
from snntorch import surrogate
spike_grad = surrogate.fast_sigmoid()
```

Listing: Surrogate in snnTorch

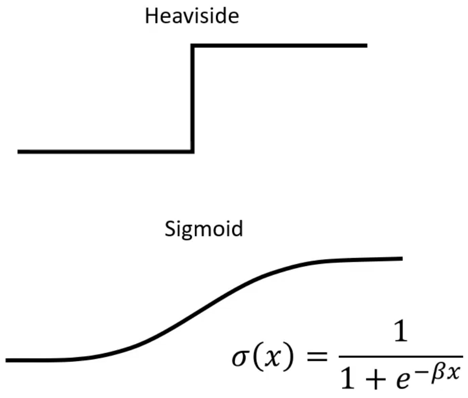


Figure: Heaviside sigmoid functions plots

Theory: Coding I

Despite SNNs can work in conjunction with another types of layers in deep networks they are made to exploit data that varies in time (audio, videos etc.) then, sometimes we need to make some static data be presented in time-varying fashion.

For now and according to (ESHRAGHIAN et al., 2023) there is three ways to do this:

- **rate coding**: Data with higher values spikes more than data with lesser values. Not so power efficient but very robust to noise.
- **latency coding**: Data with higher values spikes first, data with lesser values spikes last. Very power efficient but not so good against noise.
- **delta modulation**: Changes in data creates spikes. Good to detect movement but very prone to noises.

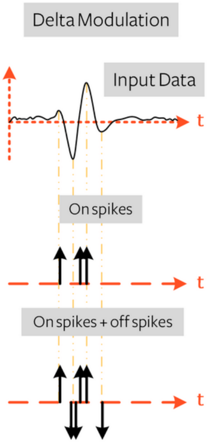


Figure: Delta modulation

Theory: Coding II

- **Rate coding**: mnist_rate.mp4
- **Latency coding** mnist_latency.mp4

Experiments

Experiment: Architecture

CNN architecture

The experiments objective is to compare a convolutional neural network (CNN) with an SNN.
The chosen architecture was the following:

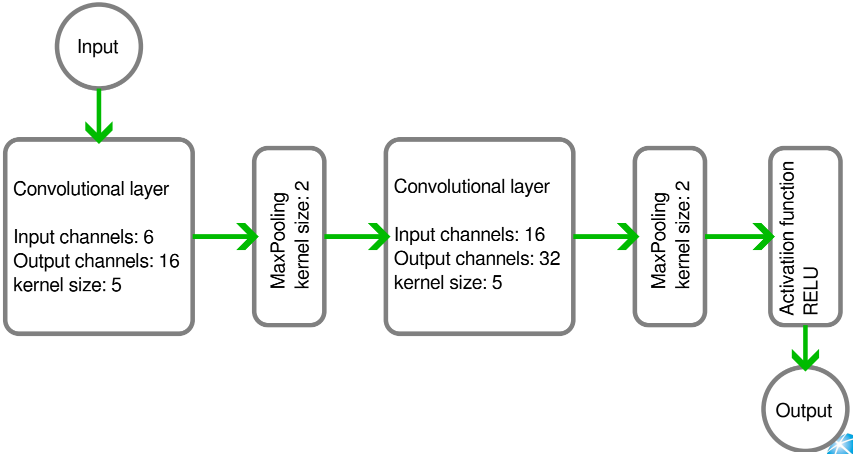


Figure: Convolutional neural network architecture

Experiment: Architecture

SNN architecture

The experiments objective is to compare a convolutional neural network (CNN) with an SNN.
The chosen architecture was the following:

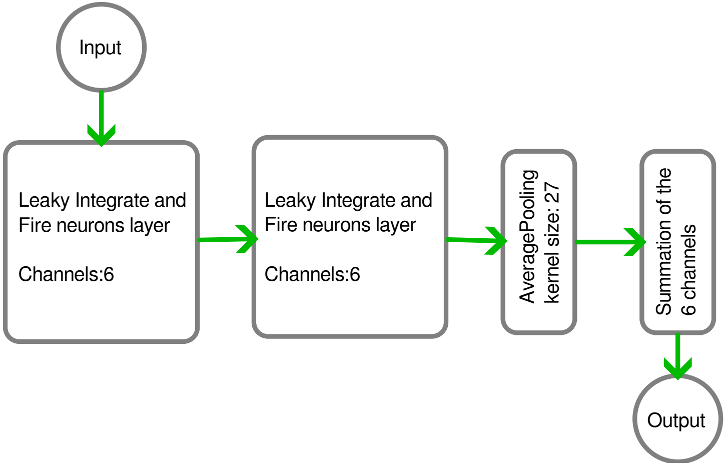

Input
Leaky Integrate and Fire neurons layer
Channels:6
Leaky Integrate and Fire neurons layer
Channels:6
AveragePooling kernel size: 27
Summation of the 6 channels
Output

Figure: Spiking neural network architecture

Experiment: Architecture

Code architecture

The code can be found at simulations/article folder in Google collab notebook format or in pure Python.

The SNN and CNN was trained according to the following parameters:

- **batch size**: 10
- **number of epochs**: 100

Some parameters are exclusive to SNNs like:

- **insist**: 4 (amount of times the same data is presented to the network)
- **neuron decay rate**: 0.9 (the rate LIF neuron loses its voltage)

```

self.model = nn.Sequential(
  nn.Conv1d(
    in_channels=input_channels,
    out_channels=16,
    kernel_size=5
  ),
  nn.MaxPool1d(kernel_size=2),

  nn.Conv1d(
    in_channels=16,
    out_channels=32,
    kernel_size=5
  ),
  nn.MaxPool1d(kernel_size=2),

  nn.Flatten(),
  nn.Linear(feature_size, 128),
  nn.ReLU(),
  nn.Linear(128, 27)
)
```

Listing: CNN architecture

Experiment: Architecture

Code architecture

The code can be found at simulations/article folder in Google collab notebook format or in pure Python.

The SNN and CNN was trained according to the following parameters:

- **batch size**: 10
- **number of epochs**: 100

Some parameters are exclusive to SNNs like:

- **insist**: 4 (amount of times the same data is presented to the network)
- **neuron decay rate**: 0.9 (the rate LIF neuron loses its voltage)

```
self.model = nn.Sequential(
  nn.Linear(inputLength, 4096),
  snn.Leaky(
    beta=neuronDecayRate,
    spike_grad=spike_grad
  ),
  nn.Linear(4096, 27),
  snn.Leaky(
    beta=neuronDecayRate,
    spike_grad=spike_grad
  ),
  nn.AvgPool1d(27, stride=1)
)
```

Listing: SNN architecture

Experiment: Database I

The used database (PRESSEL-CORETO; RUFINER; GAREIS, 2016) can be found at https://sinc.unl.edu.ar/grants/brain-computer-interfaces and consists in a set of EEG data recorded with 6 electrodes during a imagined speech of the words in spanish: "A", "E", "I", "O", "U", "Arriba", "Abajo", "Adelante", "Atrás", "Derecha", "Izquierda".

Each channel captured 4096 samples and has the following ids: 'F3','F4', 'C3', 'C4', 'P3', 'P4' according to 10-20 system (KLEM; AL., 1999).

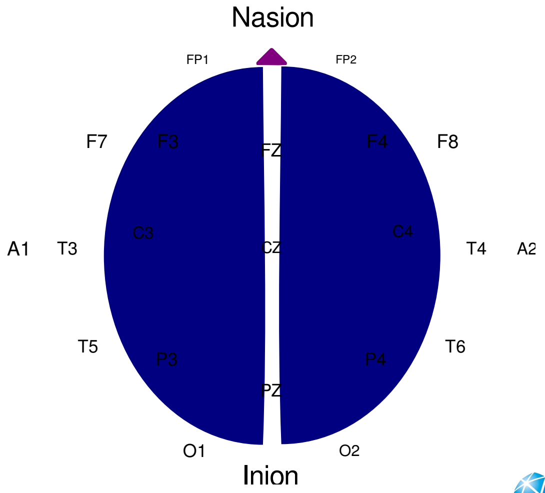


Figure: 10-20 system. Source: (VICENTE, 2023)

Experiment: Database II

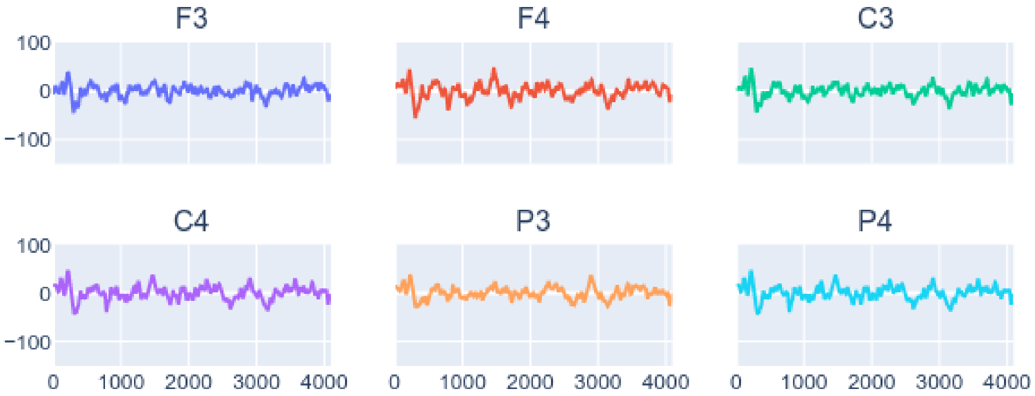


Figure: Sample of data from database

Experiment: Results

A snapshot of the models can be found at simulations/article/neuralNetworks folder.

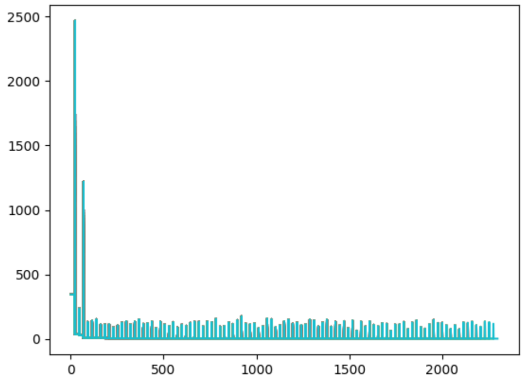


Figure: CNN Results. The horizontal axis represents iterations, vertical the loss

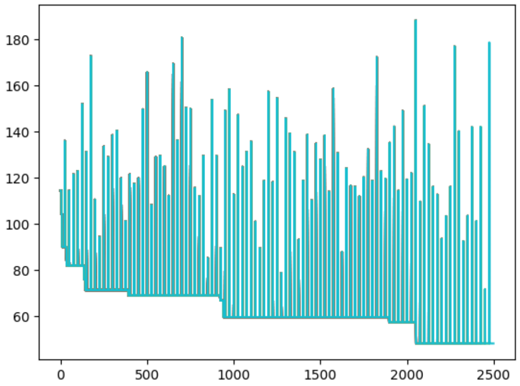


Figure: SNN Results. The horizontal axis represents iterations, vertical the loss

Referências I

ESHRAGHIAN, J. K. et al. Training spiking neural networks using lessons from deep learning. *Proceedings of the IEEE*, v. 111, n. 9, p. 1016–1054, 2023.

GOODMAN, D. et al. *Spiking Neural Network Models in Neuroscience - Cosyne Tutorial 2022*. Zenodo, 2022. Disponível em: ⟨https://doi.org/10.5281/zenodo.7044500⟩.

JONES, I. S.; KORDING, K. P. *Can Single Neurons Solve MNIST? The Computational Power of Biological Dendritic Trees*. 2020. Disponível em: ⟨https://arxiv.org/abs/2009.01269⟩.

KASABOV, N. K. *Time-space, spiking neural networks and brain-inspired artificial intelligence*. [S.l.]: Springer, 2019.

KLEM, H. O. L. G. H.; AL., H. H. J. et. The ten-twenty electrode system of the international federation. the international federation of clinical neurophysiology. *Electroencephalography and clinical neurophysiology. Supplement*, v. 52, 1999.

Referências II

PRESSEL-CORETO, G.; RUFINER, H. L.; GAREIS, I. E. *Diseño y elaboración de una base de datos pública de registros electroencefalográficos orientados a la clasificación de habla imaginada. (Undergraduate project)*. Dissertação (Mestrado) — Universidad Nacional de Entre Ríos - Facultad de Ingeniería, 2016. Disponível em: ⟨http://sinc.unl.edu.ar/sinc-publications/2016/PRG16⟩.

VICENTE, E. *Sistema 10-20 para localizar alvos terapêuticos em EMT*. 2023. Disponível em: ⟨https://www.kandel.com.br/post/como-localizar-os-pontos-de-estimulacao-para-estimulacao-magnetica-transcraniana⟩.