PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Escuela de Ingeniería Eléctrica
PONTIFICIA UNIVERSIDAD CATOLICA DE VALPARAISO

Marcel Fernández Fellay

Inteligencia Artificial para Electroencefalograma

Informe Proyecto de Titulación Ingeniero Electrónico

Escuela de Ingeniería Eléctrica
Facultad de Ingeniería

Valparaíso, 14 de julio de 2021

[redacted]

Inteligencia Artificial para Electroencefalograma

Marcel Rodrigo André Fernández Fellay

Informe Proyecto de Titulación Ingeniero Electrónico,

aprobada por la comisión de la

Escuela de Ingeniería Eléctrica de la

Facultad de Ingeniería de la

Pontificia Universidad Católica de Valparaíso

conformada por

Sr. Francisco Javier Alonso Villalobos

Profesor Guía

Sr. José Antonio Cartes Valenzuela

Segundo Revisor

Valparaíso, 14 de julio de 2021

Resumen

El presente proyecto se basa en la confección de un Electroencefalograma mediante el uso de una placa OpenBCI Cyton para la toma de muestras y un microcontrolador Nvidia Jetson Nano para el análisis e inteligencia artificial. Se abarcan los conceptos relacionados a los impulsos eléctricos del cerebro, como también sus unidades y significados. Se hace énfasis en el armado de un dispositivo que se distinga por su bajo costo y simplicidad respecto a los dispositivos existentes en el mercado.

Se presenta el método de muestreo y análisis, el cual fue realizado utilizando Python en su totalidad. Dentro del análisis se presentan distintos conceptos como lo son las transformadas de Fourier y el diseño e implementación de filtros. Por otro lado, se realiza el desarrollo de algoritmos de inteligencia artificial como el Autoencoder y K-Medios. Se establece su funcionamiento, como también el significado de los resultados obtenidos mediante el procesamiento de datos y sus posibles aplicaciones siendo este proyecto la base para ellas.

Palabras clave: Inteligencia Artificial (IA), Procesamiento de datos, Algoritmos, Electroencefalograma, Python.

Abstract

This project is based on the assembly of an Electroencephalogram with the use of an OpenBCI Cyton board for sample taking and a Nvidia Jetson Nano computer for analysis and usage of artificial intelligence. Concepts such as brain electrical impulses are covered, as well as its units and meanings. Importance is given in the assembly of a device that distinguishes itself from other devices currently offered by its low cost and simplicity.

The sampling and analysis method is presented, which was done in its entirety with the use of Python. Different concepts such as Fourier Transforms and digital filter implementation will be covered. On the other hand, different artificial intelligence algorithms such as Autoencoder and K-Means are presented and elaborated. Its operation is established, as well as the meaning of the results obtained through data processing and its possible applications, being this project the canvas for them.

Keywords: Artificial Intelligence (AI), Signal Processing, Algorithms, Electroencephalogram, Python.

Índice general

1 Introducción

Inteligencia Artificial es aquel conjunto de tecnologías computacionales y electrónicas diseñadas para realizar operaciones autónomas comparables a la inteligencia humana. Los principales tipos de Inteligencia Artificial que se utilizan en la actualidad son el aprendizaje automático, que consiste en el desarrollo de algoritmos para analizar y predecir; redes neuronales, que consiste en un aprendizaje automático modelado; aprendizaje profundo (Deep learning), el cual corresponde a una forma de aprendizaje que utiliza capas de computación para aprender datos complejos y no estructurados.

Los usos de la inteligencia artificial son variados, ya que ayudan a optimizar y mejorar el rendimiento de empresas y procesos mediante procesos que anteriormente se realizaban de forma humana (tendencias y modas, por ejemplo). También se utiliza esta tecnología para usos médicos, como lo es la búsqueda y filtrado de pacientes en alto riesgo, o la interpretación de imágenes en cirugías y radiografías para detección de anomalías. También existe el uso de la inteligencia artificial para el procesamiento de impulsos cerebrales, o Electroencefalografía. Esta consiste en el estudio de impulsos eléctricos mediante electrodos, los cuales llevan información que puede resultar crucial para diagnosticar o tratar trastornos relacionados.

Otro punto importante del uso de inteligencia artificial en la medicina y tecnología es el desarrollo de dispositivos y sistemas de ayuda para personas con movilidad reducida o faltante en alguno de sus miembros. Debido a que existe esta constante necesidad de mejorar los actuales sistemas y crear nuevos dispositivos, es que la inteligencia artificial juega un rol clave, aprendiendo de los actuales datos para generar nuevas soluciones.

Por otro lado, el electroencefalograma es una de las máquinas biológicas más complejas utilizadas actualmente en la medicina. Las señales obtenidas por esta máquina, como también su análisis y clasificación siempre ha sido una tarea de gran dificultad. El desarrollo de electroencefalogramas de fácil adquisición es una tarea importante para la masificación de estos dispositivos, lo cual afectaría de manera directa la popularidad o el aumento de uso de estos, masificando el estudio de las señales cerebrales.

La inteligencia artificial demuestra ser una herramienta útil para el análisis de señales cerebrales, ya que permitiría el análisis y clasificación de estas señales de manera automatizada y precisa.

Objetivo General

El objetivo general y principal de este proyecto es realizar un Electroencefalograma (EEG) para estudios científicos a un costo menor en comparación a las actuales ofertas, con capacidades de utilizar Inteligencia Artificial. Se realizará el análisis mediante métodos de filtrado y algoritmos de inteligencia artificial. El fin es aportar con datos y mejorar el entendimiento actual de las señales que nuestro cerebro emite con el movimiento de los dedos índices de ambas manos.

Objetivos Específicos

- Creación de un Electroencefalograma.
    - Utilizar Interfaz Cerebro-Computadora OpenBCI para la interpretación de las señales obtenidas por los electrodos conectados a la cabeza del usuario al realizar acciones (presionar botones), siendo registradas en un computador.
    - Utilización de Nvidia Jetson Nano para el procesamiento de las señales cerebrales.

- Procesamiento de señales.
    - Utilizar métodos de filtrado de señal, como también algoritmos de programación relacionados a Machine Learning y Deep Learning para la clasificación, visualización y entrenamiento de modelos artificiales para señales cerebrales.

- Ámbito Social.
    - Utilizar estos nuevos conocimientos para ampliar la información actual de las señales cerebrales, como también aportar con un código que permita a otros usuarios utilizar o adoptar su uso para otras aplicaciones como ayudas remotas.

2 Antecedentes y Estado del Arte

2.1 Descripción del Problema

El estudio de señales cerebrales para la creación y desarrollo de sistemas de ayuda para personas con menor capacidad de movimiento es un campo específico y con menos desarrollo en comparación a otras tecnologías, como lo es la industria del entretenimiento (Netflix, YouTube, Facebook, entre otras). Esto puede comprobarse al observar el actual mercado de prótesis y asistentes remotos.

Lo anterior se debe al gran problema que surge al estudiar las señales cerebrales de manera externa (a través de electrodos conectados en la piel), ya que existe una baja precisión debido a que estas señales son de voltaje muy pequeño, por lo que son susceptibles a ruido ocasionado por otros componentes presentes (cables, componentes electrónicos). Otro obstáculo existente es la diferencia que existe entre persona y persona en las señales cerebrales, por lo que las aplicaciones del tipo cerebral se deben realizar de manera personal, adecuando los sistemas al individuo.

Por otro lado, los actuales dispositivos que permiten el estudio de señales cerebrales son de difícil acceso y de alto costo. Esto se traduce en una difícil adquisición de datos e información relacionada al cuerpo humano de manera cerebral en la electrónica.

Se realizará entonces un dispositivo EEG (electroencefalograma) que contenga a su vez poder de procesamiento, y al mismo tiempo se realizarán pruebas para conocer y aislar las señales esperadas, así poder ser registradas con mayor precisión, para luego ser procesadas como datos útiles. Se aislará esta información con el fin de aportar un algoritmo o un código general que pueda ser utilizado como base de futuras aplicaciones.

2.2 Situación Actual y Estado del Arte

La inteligencia artificial es una de las tecnologías que podrían causar uno de los mayores impactos en la tecnología y la ciencia, sin embargo, existen variadas limitaciones y problemáticas que surgen con esta nueva tecnología. El primer y mayor problema tiene que ver con los datos. Existe mucha información de muchas procedencias distintas, por lo que inevitablemente se encontrarán datos erróneos. El uso de estos datos causará entonces que los modelos se entrenen con una parcialidad (bias), lo que al momento de tomar decisiones significará una nueva aparición de problemas, esta vez, relacionado quizás con género, raza o ética.

Luego se tiene el problema del poder de procesamiento. Debido a la estructura propia que tiene cada sistema de inteligencia artificial, el poder de computación necesario para procesar y construir grandes volúmenes de datos es inmenso.

Debido a estos problemas es que las aplicaciones de IA son a menor escala y en ambientes controlados y supervisados. Esto se puede aplicar en la medicina, donde la inteligencia artificial actúa como un soporte para los especialistas al momento de tomar decisiones. Un ejemplo de uso de esta tecnología en la actualidad es la clasificación de lesiones benignas y malignas en la piel. Otro uso es el cálculo de regiones adecuadas para radiografías de cuello y cráneo.

Así también el estudio de señales cerebrales se encuentra limitado a establecimientos médicos o altamente especializados, siendo un campo muy pequeño y poco accesible para establecimientos educativos, por lo que la aparición de sistemas electroencefalográficos de fácil uso y bajo costo sería fundamental para aumentar el estudio de señales cerebrales o musculares como un área popular de la ingeniería electrónica.

Estado del arte

A continuación, se presentan estudios y proyectos tanto nacionales como internacionales relacionados al uso de la Inteligencia Artificial para dispositivos médicos.

-Un acercamiento de inteligencia artificial para clasificar y analizar trazas de Electroencefalograma-

C. Castellaro, G. Favaro, A. Castellaro, A. Casgrande, S. Castellaro, D.V. Puthenparampil, C. Fattorello Salimbeni “An artificial intelligence approach to classify and analyse EEG traces”, Neurophysiologie Clinique/Clinical Neurophysiology, Volume 32, Issue 3, 2002, Páginas 193-214.

Se presenta un sistema completamente automático para clasificación y análisis de electroencefalogramas (EEG) en adultos. El sistema está basado en una red neuronal artificial que clasifica iteraciones individuales de las trazas.

Se creó el software Smart EEG para analizar todo tipo de grabación digital de electroencefalograma, luego de una subdivisión por modo de adquisición y edad del paciente. La interpretación final de la traza la realizó un doctor, quien lee los resultados obtenidos automáticamente por el software. Smart EEG funciona en tres etapas:

1. Extraer un número de trazas de la grabación digitalizada del EEG.
2. Clasificación de las trazas utilizando parámetros preestablecidos.
3. Se provee una visión general de la salida de la red neuronal.

Los resultados de 2000 muestras analizadas tanto por el software como por profesionales fueron:

1. 80% buenos resultados.
2. 18% nivel suficiente, pero con necesidad de cambios de parámetros.
3. 2% nivel insuficiente, los cuales se debieron a falsos positivos.

Los resultados al ingresar una red neuronal en el análisis de las muestras fueron:

1. 94% buenos resultados.
2. 5% nivel suficiente.
3. 1% nivel insuficiente.

-Inteligencia Artificial en cuidados neuro críticos-

Fawaz Al-Mufti, Vincent Dodson, James Lee, Ethan Wajswol, Chirag Gandhi, Corey Scurlock, Chad Cole, Kiwon Lee, Stephan A. Mayer "Artificial intelligence in neurocritical care", Journal of the Neurological Sciences, Estados Unidos, 2019, Páginas 1-4.

La toma de decisiones es muy importante, por lo que la inteligencia artificial es una gran herramienta para esta. En cuidados neuro críticos, existen modelos de IA que predicen la presión intracraneal, los cuales aún se encuentran en fases experimentales. La automatización de ciertos procesos (flujo automático de fluidos, por ejemplo) podrían mejorar la precisión y rapidez con la que se lidia con estos cambios de presión.

Otro uso de la IA es en el control y monitoreo de convulsiones y las drogas asociadas. Asistencia computacional para detectar convulsiones está demostrada, lo que muestra ser una utilidad para asistir a trabajadores médicos no especializados que se encargan de los cuidados de pacientes con convulsiones.

En cuidados intensos, la presión sanguínea es muy importante, ya que cambios de estas pueden dañar órganos y dejar daños permanentes. Aquí existen niveles de hipertensión (moderado vs severo) y el tiempo según esto se desarrolla (urgencia vs emergencia). El uso de IA puede realizarse mediante la aplicación de drogas para estabilizar la presión sanguínea de manera automatizada.

Otro punto importante es la ventilación. Muchos pacientes con daños craneales presentan problemas respiratorios. La sobre oxigenación puede ser útil para tratar elevaciones de presión intracraneal, pero al mismo tiempo, causar hipertensión intracraneal. La inteligencia artificial es una posible solución para monitorear los cambios de suministros respiratorios para mantener un equilibrio de presiones en el cráneo.

- Inteligencia Artificial en enfermedad celíaca-

Muhammad Khawar Sana, Zeshan M. H ussain, Muhammad Haisum Maqsood, Pir Ahmad Shah "Artificial intelligence in celiac disease", Computers in Biology and Medicine, 2020.

Debido a la cantidad de exámenes a los que deben someterse los pacientes con problemas gastrointestinales, el exceso de endoscopías puede ser dañino, y la alternativa existente es la endoscopia por video cápsula, la cual es lenta y poco precisa por los movimientos súbitos de ella en el intestino. Aquí es cuando la inteligencia artificial juega un rol clave. Al registrar los resultados de varios pacientes sometidos a endoscopía por video cápsula, se obtuvieron datos utilizados para el entrenamiento de esta inteligencia artificial. Así pudo crearse un modelo de análisis que no se basa en la percepción humana. Esto logró diagnosticar enfermedad celíaca de manera correcta, además de indicar el grado de severidad de esta en el paciente.

-Neuralink, un Start-up de Elon Musk obtiene simbiosis con inteligencia artificial-

A. Kulshreshth, A. Anand, A. Lakanpal "Neuralink – An Elon Musk Start-up Achieve symbiosis with artificial intelligence", ICCCIS, 2019.

Neuralink, registrado en 2016, busca crear una simbiosis general entre el hombre y las máquinas o la inteligencia artificial.

Se utiliza una interfaz cerebro-máquina capaz de conectar a las personas a cualquier máquina capaz de leer las entradas enviadas por nuestro cerebro. Esta interfaz tiene el poder de ayudar a las personas con un gran rango de desórdenes clínicos como funciones motoras y sentidos disfuncionales, siendo esta interfaz el paso a la creación de un canal de alto ancho de banda para transferir señales utilizando matrices de hilos y electrodos.

Neuralink tendrá electrodos que leerán las señales cerebrales emitidas por las neuronas, las cuales amplificará y enviará a las máquinas que a su vez actuarán de la manera deseada. Estos electrodos también soportan la escritura, lo cual ayudará en el tratamiento de desórdenes cerebrales.

-Prótesis de mano virtual movida por señales de encefalografía EEG-

K. Correa, A. Vivas "Prótesis de mano virtual movida por señales encefalograficas – EEG", Prospect, Vol 14, N2, 2016, páginas 99-110,

Se busca manipular una prótesis de mano en un ambiente virtual de simulación utilizando una interfaz natural basada en una BCI (Brain Computer Interface). Utilizando señales encefalograficas captadas a través de electrodos conectados sobre el cráneo del usuario, quien puede producir órdenes sencillas a través del software utilizado.

Los movimientos escogidos para la prótesis de mano virtual fueron agarres de pinza, cilíndrico y esférico, al igual que abrir y cerrar la mano. Utilizando la suite CognitiveTM se realiza una etapa de entrenamiento para encontrar y guardar los patrones cerebrales para distintas acciones de control.

-Análisis de transformada de Fourier de corto tiempo en señal EEG obtenida al escribir-

C.W.N.F Che Wan Fadzal, W. Mansor, L. Y. Khuan, A. Zabidi "Short-time Fourier Transform Analysis of EEG Signal from Writing", IEEE 8th International Colloquium on Signal Processing and its Applications, 2012.

La transformada de Fourier de corto tiempo provee una ventaja al revelar los contenidos de frecuencia de una señal en cada punto de ella. Esta información puede ser utilizada para proveer control y realizar distintas tareas en interfaces Cerebro-Computadora. En este estudio se describe el análisis de señales EEG utilizando STFT (Short-time Fourier Transform) al escribir y relajarse.

Las señales EEG fueron tomadas de varones diestros de 20-27 años, sentados cómodamente en una silla. Luego, se les pidió realizar tres actividades, de dos minutos cada una. Cuatro electrodos fueron utilizados para registrar la señal EEG, ubicados en las posiciones C3, C4, P3, P4.

El análisis de Fourier es importante para describir el contenido en frecuencia de la señal en cada punto temporal. Se implemento STFT luego de utilizar un filtro pasa bandas con frecuencia entre 8-30 Hz.

Para un estado de relajación, se obtuvo el rango de frecuencias entre 8-12.5 Hertz, mientras que, al realizar la escritura, las frecuencias correspondían al rango de 11-28.38 Hertz.

-Optimización no homogénea para filtros espaciales en interfaces cerebro-computadora para electroencefalograma-

T. Kam, H. Suk, S. Lee "Non-Homogeneous Spatial Filter Optimization for EEG-based Brain-Computer Interfaces", International Winter Workshop on Brain-Computer Interface (BCI), 2013

Atenuación o realce de poder en bandas de frecuencia específicas sobre la corteza psicomotora, llamada Event-Related Desynchronization (ERD, Desincronización Evento-Relacionada) o Event-Related Synchronization (ERS, Sincronización Evento-Relacionada) respectivamente, son fenómenos en las actividades cerebrales relacionadas con el movimiento imaginario de partes corporales. Sin embargo, es conocido que la naturaleza de las muestras obtenidas por señales EEG son no-estacionarias y altamente variables a través del tiempo y frecuencia. En este estudio se propone un método novedoso, llamado filtro no-homogéneo.

El método consiste en descomponer señales EEG en 9 bandas frecuenciales entre 4Hz y 40Hz con ventadas de 4Hz de tamaño. Para cada ventana se aplica un filtro espacial. Cada señal original se separa en 9 segmentos temporales utilizando ventanas de 1 segundo traslapadas en un 50%. Luego, se aplica un método estadístico llamado F-test (Prueba F) con el cual se encuentran diferencias en las muestras utilizando varianza dentro de las clases o entre las clases. Finalmente, se entrenan clasificadores para cada segmento de tiempo-frecuencia con las características seleccionadas, basado en algoritmos de análisis discriminante lineal (LDA) y máquinas de vector soporte (MVS).

-Generalización y Extracción de Componentes utilizando interfaz cerebro-computadora basado en FFT y Red Neural Multicapa-

K. Nakayama, Y. Kaneda, A. Hirano "A Brain Computer Interface Based on FFT and Multilayer Neural Network -Feature Extraction and Generalization- ", International Symposium on Intelligent Signal Processing and Communication Systems, 2007

En este estudio, una red neural multicapa es aplicada a una interfaz cerebro-computadora. La amplitud de la FFT (Fast Fourier Transform, Transformada rápida de Fourier) de las señales cerebrales so utilizada como información de entrada. Se introdujeron técnicas para el preprocesamiento de las señales. Se incluye la segmentación a través del eje temporal para respuestas rápidas, normalización no-lineal para enfatizar información importante, promedio de muestras de señales cerebrales para suprimir efectos de ruido, reducción en el número de muestras para realizar una pequeña red, entre otras.

Se utilizaron cinco tareas mentales en las muestras de señales cerebrales, como Gráfico de Línea Base, Multiplicación, Composición de letras, Rotación de objetos 3D, conteo de números.

Las ondas cerebrales se muestrearon en intervalos de 10 segundos y con una frecuencia de muestreo de 250Hz. Luego se segmentaron las muestras en el eje del tiempo en segmentos de 0,5 segundos, seguido por la obtención de amplitud utilizando FFT. Luego se redujeron las muestras utilizando un promedio para disminuir el ruido agregado en las señales cerebrales, finalizado con una normalización no-lineal.

-Identificación de ruidos al pestañear utilizando electroencefalograma inalámbrico para sistema de interfaz cerebro-computadora-

S. Sridhar, U. Ramachandraiah, E. Sathish, G. Muthukumaran, P. Rajendra "Identification of Eye Blink Artifacts using Wireless EEG headset for Brain Computer Interface System", IEEE Sensors, 2018

La utilización de cascos inalámbricos para Electroencefalograma es explorada mundialmente para innumerables aplicaciones bio-médicas. Usualmente, las señales EEG se ven adulteradas con ruido indeseado tal como el pestañeo, movimiento muscular, movimiento mandibular, fuentes de poder, etc. Este estudio presenta procedimientos experimentales para detectar y analizar ruido generado por el pestañeo, con condiciones al cerrar y abrir los ojos, utilizando un histograma y la transformada rápida de Fourier.

La amplitud del ruido provocado al pestañear es mayor en comparación a la de ojos abiertos o cerrados. La amplitud varía entre 0.5 y 3Hz.

El método de registro de señales se hizo en 4 personas en la edad de 25-35 y se muestrearon las señales utilizando una frecuencia de muestreo de 250Hz y analizadas utilizando FFT (Fast Fourier Transform). La señal también fue filtrada utilizando un filtro rechaza banda en 50Hz para eliminar el ruido producido por la fuente de poder.

-Análisis de Transformada Rápida de Fourier y Clasificación de EEG para Silla de Ruedas controlada con Señales Cerebrales -

S. Swee, L. You "Fast Fourier Analysis and EEG Classification Brainwave Controlled Wheelchair", 2nd International Conference on Control Science and Systems Engineering, 2016

En este estudio se realiza el análisis de la transformada rápida de Fourier para clasificación basada en silla de rueda controlada por electroencefalograma (EEG). Esta silla de rueda está directamente controlada por el cerebro. Así, no requiere retroalimentación física por parte del usuario. El proyecto apunta a mejorar el poder cerebral. Esto se conoce como fuerza de concentración. Al incrementar el uso y la concentración del cerebro, se reduce el riesgo de degeneración cerebral. Para lograr esto se utiliza el método de BCI (Brain-Computer interface), el cual permite al cerebro comunicarse directamente con los mecanismos eléctricos de la silla de ruedas. El método de análisis fue la transformada rápida de Fourier y clasificación EEG.

La construcción se basó en una silla de ruedas que utiliza dos motores de scooter eléctricos, un microcontrolador Arduino, un casco EEG Emotiv EPOC y un módulo bluetooth para la comunicación inalámbrica.

Se encontró que utilizando un dispositivo EEG de 16 canales fue adecuado para proveer datos con mayor precisión en tiempo real. El método utilizado pudo lograr un 90% de precisión al detectar comandos mentales del usuario.

-Silla de ruedas controlada por señales cerebrales-

Rodrigo Quevedo, experto en robótica, presentó la idea de crear un sistema que permite el movimiento de sillas de ruedas mediante las señales cerebrales obtenidas de un casco conectado a la cabeza. El

proyecto llamado Over Mind busca el movimiento sin contacto físico, es decir, usando solo las señales cerebrales.

Para lograr el funcionamiento de esta aplicación, las señales cerebrales son obtenidas mediante neuro-sensores. Estas señales se procesan mediante software que traduce impulsos eléctricos en órdenes de movimiento, permitiendo el desplazamiento de las ruedas.

-Prótesis de mano controlada por señales musculares-

Mario Olivares, Ingeniero en automatización y robótica de la Universidad Andrés Bello, ideó y creó una prótesis de mano 3D funcional a bajo costo. Esta prótesis contiene tres electrodos que se conectan a los músculos del brazo, las cuales detectan movimiento y contracción muscular, transformándolo en señales eléctricas. Estas señales eléctricas son procesadas y luego entregadas a un microcontrolador, el cual se encarga de realizar los movimientos.

Los movimientos logrados en la mano prostética pueden ser juntar dedos pulgar e índice, permitiendo escribir. Así también juntar todos los dedos, permitiendo agarrar objetos. Esto permite mayor independencia de movimientos.

Este proyecto nace del estudio de las señales eléctricas entregadas por los músculos al generar movimientos. Estas señales eran luego registradas y procesadas para poder ser aplicadas en algún sistema robótico.

-Sistemas EEG similares-

Existen numerosas opciones de electroencefalogramas para el estudio de ondas cerebrales en el ámbito científico, siendo el principal problema su alto costo y necesidad de utilizar o comprar software propio de compañía que ofrece su sistema EEG. Se puede observar un costo mayor a $700 dólares. Algunos productos similares a lo que se busca realizar en este proyecto son Emotiv Epoc X, Wearable Sensing DSI7, GTEC g.Nautilus, Neuroelectrics Enobio 8.

3 Marco Teórico

3.1 Sistemas neurales.

Los sistemas biológicos realizar tareas a velocidades extremadamente altas, como lo es el reconocimiento de objetos, colores y palabras. Todo esto se realiza a nivel cerebral, donde existe una red de neuronas interconectadas donde se lleva y procesa la información de los receptores del cuerpo humano (ej: ojos, oídos). Esta red se ha querido replicar de manera virtual, con la inteligencia artificial, llamadas también Redes Neuronales.

Una sola neurona no tiene la capacidad para procesar grandes cantidades de información, por lo que se necesita un flujo de información entre neuronas para lograrlo. Lo anterior se demuestra en Figura 3-1:

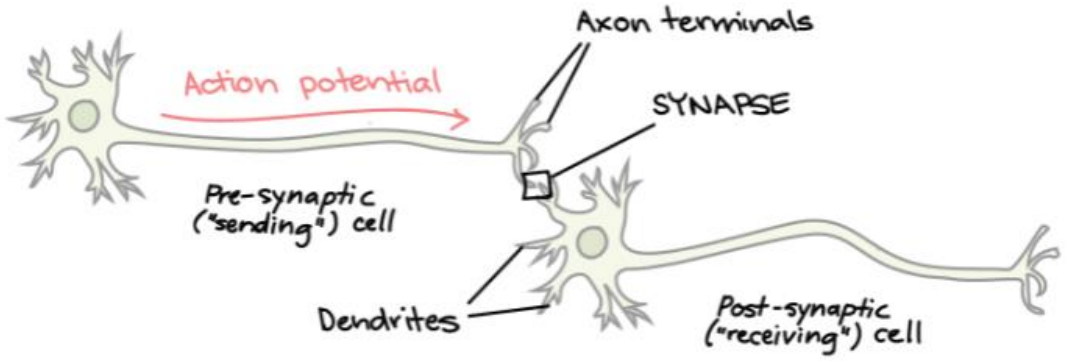

Figura 3-1: Red de 3 neuronas [19]

Se puede apreciar en Figura 3-1 que la interconexión de neuronas permite el flujo de información, realizada por sinapsis, que corresponde al espacio entre una neurona y otra, donde también se transmiten los impulsos nerviosos.

En los sistemas artificiales, la representación de las neuronas son nodos, en donde existe siempre una entrada y una salida, conectadas a través de una red. La Figura 3-2 demuestra esta red de nodos.

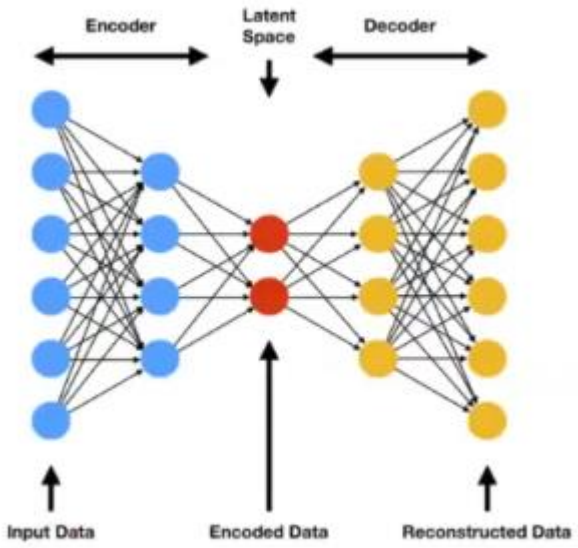

Figura 3-2: Representación de Sistema Neuronal [21]

Al igual que en el cuerpo humano, las neuronas deben ser entrenadas para saber cómo responder a cierto estímulo. En Figura 3-2 esto se puede explicar de la siguiente forma:

- Codificador: Aquí ingresan los datos, se realiza una excitación de los nodos que comienzan a agrupar y analizar esta información.
- Espacio Latente: Representación de la información comprimida. Basado en información anterior se realiza el análisis de los datos que provienen de la entrada.
- Decodificador: Aquí se convierten los datos ya analizados en respectivas respuestas deseadas por el sistema.

Naturalmente, la inteligencia artificial está muy lejos de parecerse o acercarse a la gran capacidad de procesamiento que el cerebro humano posee. Al analizar ambos sistemas, se realizó la Tabla 3-1 para comparar distintas características:

Tabla 3-1: Comparación entre Nodos y Neuronas

Característica	Sistema Artificial	Sistema Biológico
Cantidad de neuronas o nodos	Rango 10-1000	Estimación de $8.6 \cdot 10^{10}$
Tipo de red	Nodos independientes, procesamiento síncrono	Red interconectada, procesamiento asíncrono
Velocidad	Activación continua, dependiente del tipo de	Depende de características biológicas (edad, sexo,

	hardware (Procesador, memoria)	temperatura, etc.) Alrededor de 200 activaciones por segundo
Consumo de poder	Dependiente de tecnología actual. Con un procesador gráfico Nvidia Titan RTX, alrededor de 280 Watts.	Se estima un consumo del 20% de la energía corporal en el cerebro, equivalente a 20 Watts.
Tolerancia	Fallas de nodos pueden causar pérdidas de información	Debido al tipo de red, fallas entre neuronas no ocasionan pérdidas de información
Aprendizaje	Modelo predefinido, no se agregan ni se eliminan nodos	No conocido completamente, cerebro permite creación y modificación de neuronas y sus funciones

3.2 Interfaz Cerebro-Computadora.

La entrada a utilizar serán los impulsos recibidos en la interfaz cerebro-computadora a través de un electroencefalograma. En la Figura 3-3 se muestran las posiciones adecuadas de los electrodos en la cabeza y orejas, según el sistema 10-20:

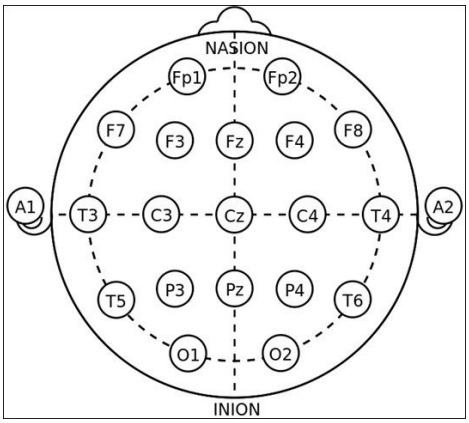

Figura 3-3: Posición de electrodos, sistema 10-20 [20]

Una de las opciones para el análisis de las señales cerebrales es mediante el uso del software libre de OpenBCI, mostrado en Figura 3-4:

Figura 3-4: Software OpenBCI [8]

Los estímulos cerebrales podrán interpretarse según sus cambios de amplitud (Voltaje) en el tiempo, también utilizando herramientas matemáticas como lo son las transformadas de Fourier en frecuencia. Un ejemplo de la señal emitida al pestañear se muestra en Figura 3-5:

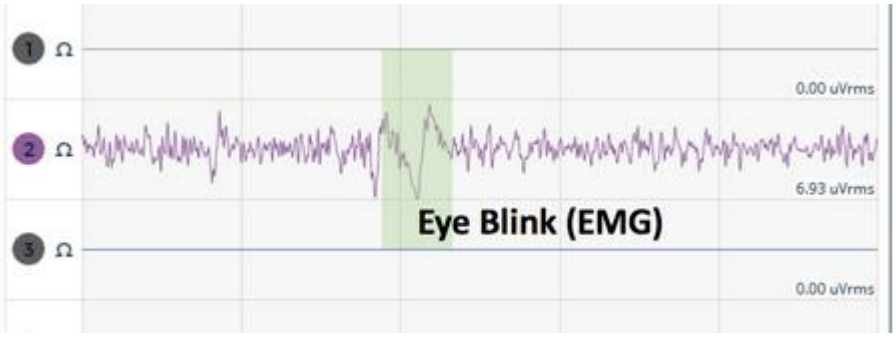

Figura 3-5: Señal al pestañear (Electromiograma) [8]

Un ejemplo de una subida en la transformada rápida de Fourier se aprecia en Figura 3-6:

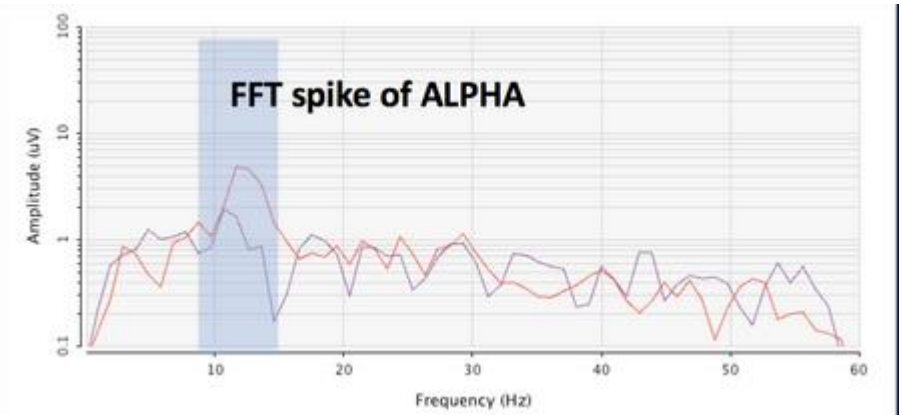

Figura 3-6: Transformada rápida de Fourier [8]

Es conocido que las señales cerebrales presentan comportamientos que pueden ser llamados rítmicos en ciertos momentos. Un ejemplo de esto es el pestañeo en gente saludable, donde se espera una vibración de 7 a 13 veces en un segundo, patrón al cual se le llaman ondas alfa. Figura 3-6 muestra las componentes en frecuencia de un ejemplo de estas ondas.

Por otro lado, utilizando Python se puede realizar también la transformación de señales, como se muestra en el ejemplo Figura 3-7:

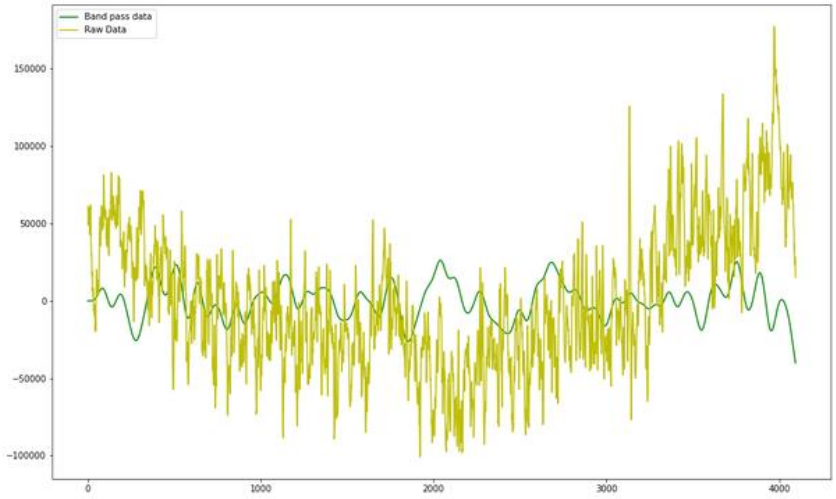

Figura 3-7: Señal Sin procesar y Señal Filtrada [21]

Se tiene en la Figura 3-7 la señal original del EEG como también la señal original al pasar por un filtro pasa banda. A continuación, se ve el resultado al ser sometidas estas señales a la transformada rápida de Fourier en Figura 3-8:

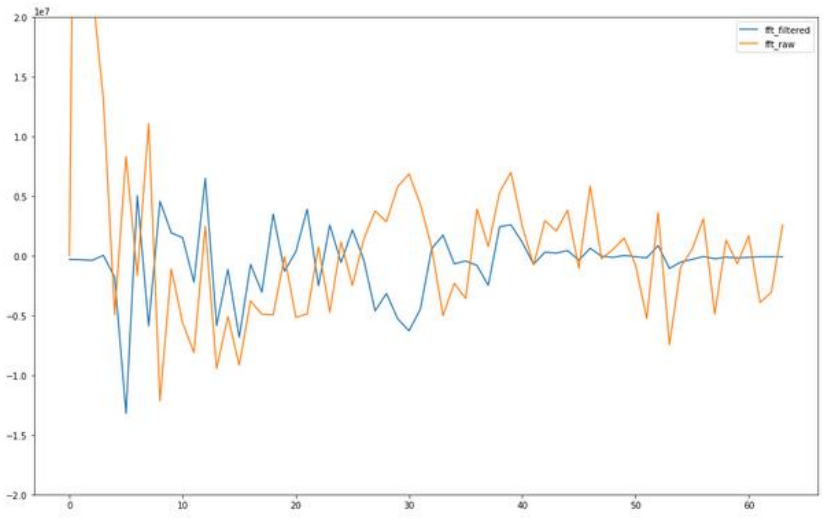

Figura 3-8: Transformada Rápida de Fourier para señal original y filtrada [21]

La utilidad de la transformada de Fourier se debe a que una señal oscilatoria puede ser representada o descompuesta en múltiples señales sinusoidales a distintas frecuencias. Esto permite un análisis óptimo de las señales, ya que así se pueden encontrar las frecuencias naturales de ciertos impulsos biológicos, resultando en un pico o peak en la representación gráfica de la transformada de Fourier como se mostró en Figura 3-6.

Pero ¿Qué es la transformada rápida de Fourier (FFT)?

Transformada rápida de Fourier o Fast Fourier Transform (FFT) es una herramienta que permite revelar las componentes útiles en frecuencia de una señal, aun cuando dicha señal presenta ruido. Esta transformada viene de la Transformada de Fourier, que corresponde a un algoritmo matemático que permite extraer información de la frecuencia al descomponer las señales periódicas como una combinación de senos y cosenos. Con la necesidad de aplicar esta transformación en señales digitales, se propuso una versión discreta (Transformada Discreta de Fourier), la cual permitió la creación de la FFT como una versión más rápida y eficiente.

Figura 3-9 demuestra de manera gráfica esta transformada. En el dominio del tiempo se descompone la señal en senos y cosenos, mostrándose como picos en el dominio de la frecuencia.

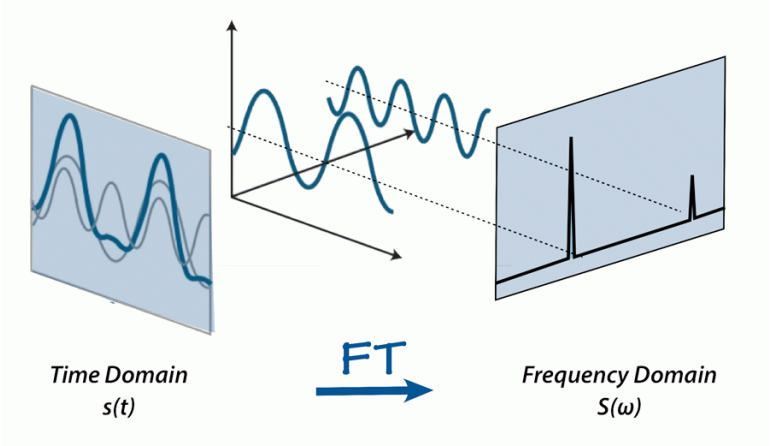

Figura 3-9: Demostración gráfica transformada de Fourier [22]

Cabe destacar que, debido a la incapacidad de los computadores de procesar una señal continua, las señales se separan en grupos de muestras con información discreta en tiempos de muestreo específicos, llamándose tasa o frecuencia de muestreo a la velocidad de este muestreo. A modo de ejemplo, que un dispositivo utilice una tasa de muestreo de 500Hz significa que 500 muestras serán requeridas en un segundo, por lo que el intervalo entre dos puntos adyacentes será de 1/500s lo que es igual a 2 ms.

3.3 Machine Learning.

Machine Learning o aprendizaje automático es un tipo de Inteligencia Artificial que busca crear sistemas que aprenden automáticamente. Utilizando distintos algoritmos, se busca que la máquina sea capaz de entregar datos y predecir resultados.

Machine Learning se divide en dos tipos de aprendizajes, Supervisado y No supervisado.

Aprendizaje supervisado generalmente se realiza para el contexto de clasificación, o cuando se requiere utilizar datos etiquetados. Ejemplos de aplicaciones de este tipo de aprendizaje es la regresión logística, redes neuronales artificiales y máquinas de vector soporte.

Un ejemplo de este tipo de aprendizaje se muestra en Figura 3-10:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267

Figura 3-10: Dataset para Aprendizaje Supervisado [18]

En Figura 3-10 se tienen distintas características para 9 vehículos (acortados de datos mayores) y se busca crear un modelo de regresión lineal simple que permita entregar una predicción en base a la suma residual de cuadrados entre la variable independiente (Enginesize o tamaño de motor) y la variable dependiente (CO2Emissions o emisiones de CO2). [18]

La Figura 3-11 muestra la relación entre emisiones y tamaño de motor:

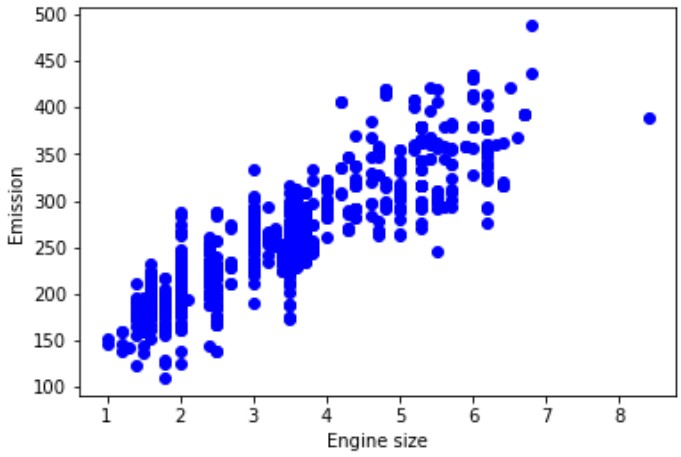

Figura 3-11: Emisiones vs Tamaño de motor

Luego, al realizarse el entrenamiento de datos, se obtiene la siguiente recta que se ajusta al modelo de regresión y que permite predecir emisiones según el tamaño del motor, siendo el modelo de regresión la línea roja en la Figura 3-12.

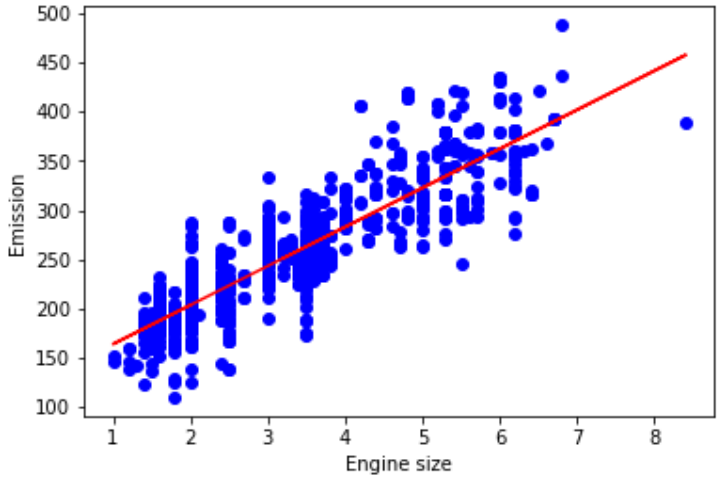

Figura 3-12: Modelo obtenido de regresión lineal simple

Por otro lado, el aprendizaje no supervisado se refiere al uso de datos para ser agrupados o representados, como también la estimación de densidad. Aquí se busca aprender la estructura inherente de los datos que no tienen etiqueta. Ejemplos de aplicaciones para este aprendizaje es el agrupamiento k-medias, análisis de componente principal y autoencoders.

Un autoencoder es un tipo de red neuronal en aprendizaje no supervisado, el cual aprende una representación de los datos ingresados, siendo entrenado para la eliminación de agentes externos tales como ruido. Un ejemplo de esto se ve en Figura 3-13:

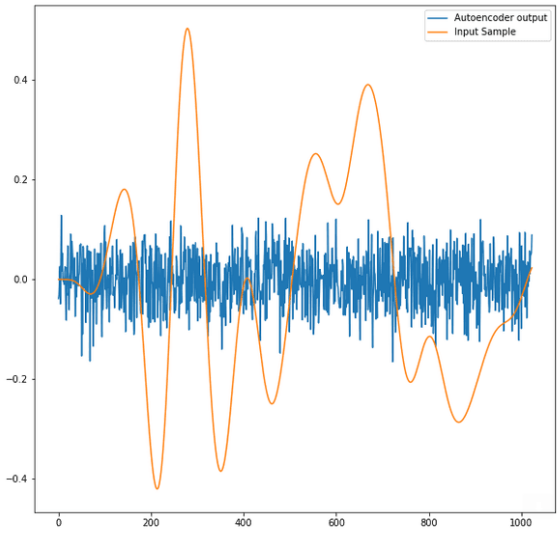

Figura 3-13: Señal y Salida iniciales Autoencoder [21]

Se puede apreciar de Figura 3-13 que la señal inicial dista completamente de la salida del autoencoder, por lo que se realiza un entrenamiento automático al realizarse el análisis por la totalidad de los datos múltiples veces, las que se llaman épocas.

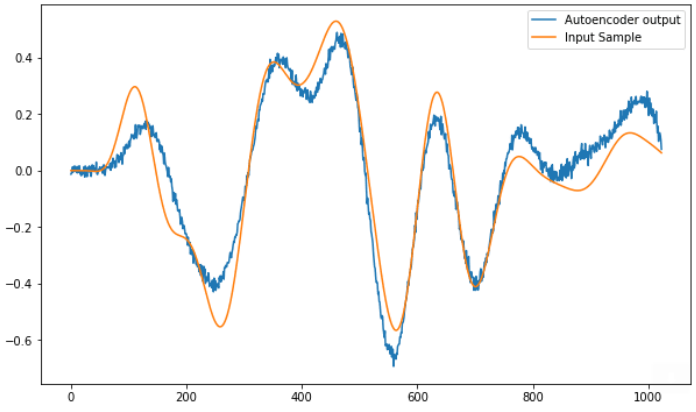

Figura 3-14: Salida Autoencoder en 5 épocas [21]

Se puede apreciar de Figura 3-14 que al pasar 5 épocas el autoencoder comienza a tomar forma, existiendo aún ruido y diferencias entre la entrada y salida.

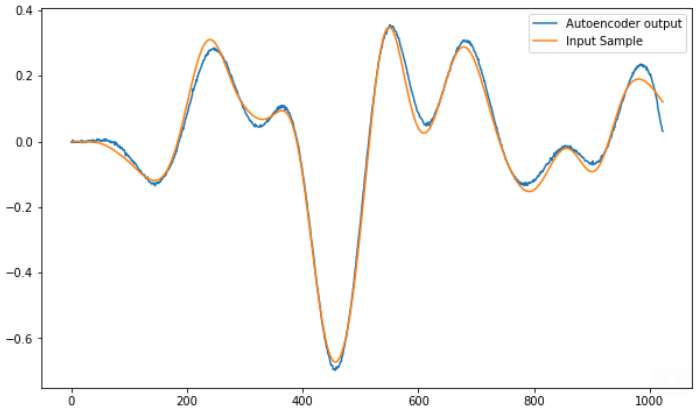

Figura 3-15: Salida Autoencoder en 20 épocas [21]

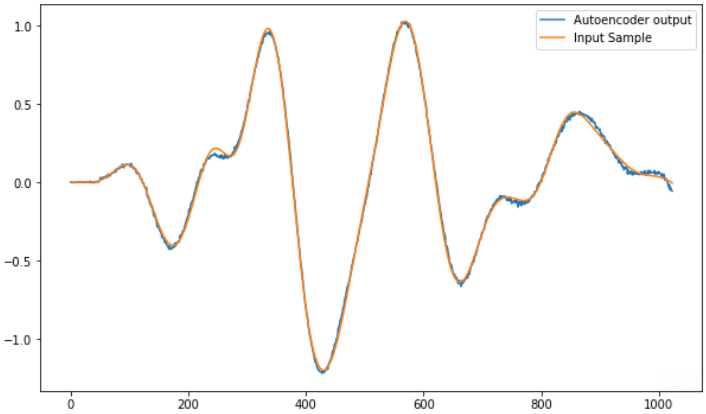

Figura 3-16: Salida Autoencoder en 50 épocas [21]

Luego, se puede apreciar en Figura 3-15 que al pasar 20 épocas el ruido ya es disminuido en manera considerable, como también el modelo aprendido por la red neuronal es cercano a la entrada, y finalmente en Figura 3-16 al pasar 50 épocas la diferencia que existe en la entrada y salida es mínima.

Otro método de aprendizaje no supervisado que se utilizará es el algoritmo de K-medios. Este se utiliza para datos que no están categorizados, y busca agrupar estos datos en distintas categorías, siendo el número de grupos representados por la variable K. Este algoritmo funciona de la siguiente manera:

1. Se realiza una suma de cuadrados al moverse de un grupo de datos a otro.
2. Para cada punto se encuentra el primer y segundo grupo más cercano para así obtener el centroide de este grupo.
3. Se realiza el cálculo de la distancia Euclidiana desde el centroide a cada dato.
4. Se encuentra el nuevo centroide y se repite el paso 3 y 4 hasta que no se encuentra un nuevo centroide.

El uso de este algoritmo en señales cerebrales permite la detección de anomalías o la diferenciación entre impulsos cerebrales. Para este proyecto se utilizará para detectar el impulso específico del movimiento realizado por el sujeto de prueba, de modo que se pueda definir si la actual señal es una acción o no.

4 Electroencefalograma: Implementación

A continuación, se presentan la información relacionada a la confección y configuración del dispositivo de electroencefalografía. Se inicia con una descripción técnica de los componentes y la posición utilizados de los electrodos. También se presenta lo relacionado a requisitos de software y programación para lograr un funcionamiento del sistema.

4.1 Materiales y software.

El dispositivo a utilizar para la toma de datos o señales cerebrales es una placa basada en la OpenBCI Cyton de 8 canales, la cual se ve en Figura 4-1:

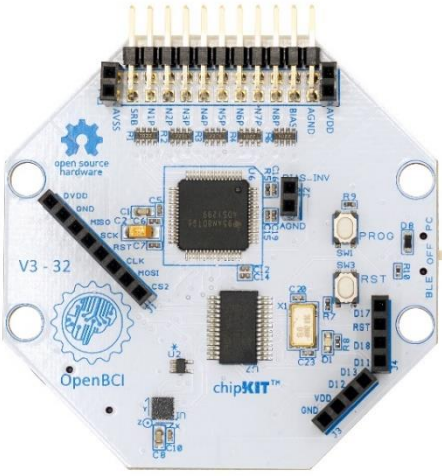

Figura 4-1: Dispositivo EEG Cyton 8Ch. [8]

El EEG Cyton contiene un microcontrolador PIC32MX250F128B [28] con un bootloader chipKIT UDB32-MX2-DIP. La conversión análoga-digital se realiza mediante un ADS1299 [29].

Los canales a utilizar corresponden a: SRB, N1P, N2P, N3P, N4P, N5P, N6P, N7P, N8P, BIAS. Siendo los canales N1P a N8P los que adquieren datos, mientras que SRB corresponde al pin de referencia, es decir, el que realiza la diferencia entre el valor obtenido por los canales, y el valor base de la cabeza utilizada para la toma de muestras. BIAS corresponde al canal de supresión de ruido, o tierra. [24]

Estos canales se conectan mediante el uso de electrodos de disco bañados en oro (Gold Cup Electrodes) similares a los que se aprecian en Figura 4-2:

Figura 4-2: Electrodos EEG [8]

Finalmente, estos electrodos se fijaron en un gorro de silicona para EEG, el cual se muestra en Figura 4-3:

Figura 4-3: Gorro Silicona EEG [Fuente: producto]

Utilizando el sistema 10-20 mostrado en Figura 3-3, los electrodos con sus respectivos canales corresponden a:

- SRB: A2
- N1P: Fp2
- N2P: Fp1
- N3P: C3
- N4P: C4
- N5P: P3
- N6P: P4
- N7P: O2
- N8P: O1
- BIAS: A1

Debido a que se quiere lograr obtener la información cerebral al apretar botones con los dedos índices, las regiones cerebrales de interés son:

Lóbulo Parietal: Funciones principales de integración sensomotora, es decir, de todo aquello relacionado a los sentidos y a la motricidad, como también de la somatosensación, que corresponde a aquellas acciones o reacciones causadas por estímulos como lo son el tacto, temperatura, posición y nocicepción o estímulos potencialmente dañinos contra la piel. [25]

Lóbulo Frontal: Funciones conductuales y habilidades cognitivas. Dentro de este lóbulo se encuentra el área motora del cerebro, la que se encarga en la planificación y ejecución de movimientos.[30]

De igual manera, se conectaron dos electrodos en lo que corresponde al Lóbulo Occipital, el cual está encargado del área visual del cuerpo. Se hizo esto con el fin de poder tener un rango completo del cerebro dentro de las mediciones.

Por otro lado, para poder concretar la toma de muestras, se utiliza la computadora Jetson Nano 2[GB] de Nvidia, mostrada en Figura 4-4:

Figura 4-4: Nvidia Jetson Nano 2GB [23]

Este dispositivo corresponde a una pequeña computadora que permite la ejecución de redes neuronales para clasificación, detección, segmentación y procesamiento de información. Contiene una unidad de procesamiento gráfico Nvidia Maxwell de 128 núcleos, y una unidad de procesamiento central ARM A57 de 4 núcleos. Además, contiene 2[GB] de memoria RAM de 64-bits. Por otro lado, utiliza una tarjeta microSD como almacenamiento, y su sistema operativo predeterminado es Ubuntu de Linux. [23]

La conexión entre la computadora Jetson Nano y el dispositivo EEG Cyton se realizó mediante USB, el diagrama de Figura 4-5 explica la conexión realizada:

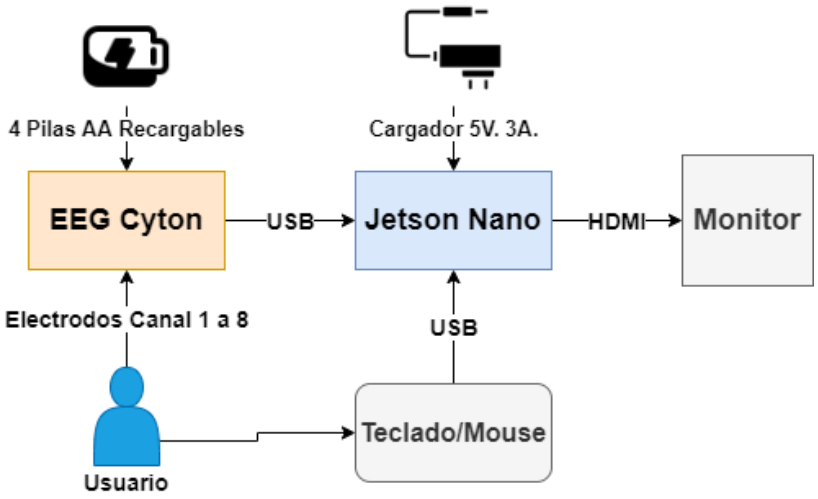

Figura 4-5: Diagrama simple de Conexión

Una vez confeccionado el sistema, se debió configurar la Jetson Nano, instalando el software necesario para el correcto funcionamiento y compatibilidad con el EEG. La programación se realiza en Python 3.7, utilizando la aplicación de código abierto Jupyter Notebook [31]. Las librerías específicas a utilizar son las siguientes:

- Numpy>=1.14.3
- Pandas>=0.22.0
- Scipy>=1.0.0
- Brainflow
- Tensorflow
- Pyserial>=2.7
- Requests>=2.7
- Jupyter
- Bitstring

5 Electroencefalograma: Desarrollo

A continuación, se continúa la confección del proyecto, en esta sección se desarrolla lo relacionado al código utilizado para registrar las señales cerebrales (toma de muestras) y su posterior procesamiento para realizar el análisis de datos.

5.1 Toma de muestras

Una vez configuradas las librerías en Python, se procede a ejecutar la aplicación Jupyter Notebook, y se comienza con la etapa de programación para el programa de muestreo. La librería Brainflow contiene las herramientas y configuraciones necesarias para el dispositivo de OpenBCI, el código se muestra en Listado 5-1:

Listado 5-1: Código para registro de señales

```
import argparse

import time

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scipy.signal

import brainflow

from brainflow.board_shim import BoardShim, BrainFlowInputParams, LogLevels, BoardIds

#Se importaron las librerías necesarias para las funciones a utilizar. La librería Brainflow contiene los parámetros específicos para la placa Cyton de OpenBCI tal como el muestreo de 250[Hz]. También se asigna el puerto USB de la Jetson Nano en la que está conectada esta placa, los canales (8) y se prepara el stream de datos. Se seleccionan las últimas 1000 muestras para ser registradas en un archivo csv. Se transponen los datos de manera que cada canal se registre de manera vertical, y una vez guardado el archivo csv el sistema se detiene y entrega un mensaje de término.

def main():
```

```
    BoardShim.enable_dev_board_logger()

    params = BrainFlowInputParams ()

    board_id=0

    params.serial_port = '/dev/ttyUSB0'

    eeg_channels = BoardShim.get_eeg_channels(BoardIds.CYTON_BOARD.value)

    board = BoardShim(board_id, params)

    board.prepare_session()

    board.start_stream()

    BoardShim.log_message(LogLevels.LEVEL_INFO.value, 'start sleeping in the main thread')

    print('Comienzo')

    time.sleep(5)

    data= board.get_current_board_data(1000)

    board.stop_stream()

    board.release_session()

    datf= pd.DataFrame(np.transpose(data))

    #print(' Data From the Board')

    #print(df.head(10))

    DataFilter.write_file(data, 'test2raw.csv','w')

    print('Muestras guardadas en archivo .csv')
if __name__ == "__main__":

    main()
```

Una vez los electrodos se encuentran conectados a la cabeza, utilizando un gel conductor eléctrico para lograr un buen traspaso de señal, se ejecuta el programa, almacenándose las muestras tomadas en un archivo con formato csv. Es importante mencionar que las muestras se tomaron al mover los dedos índices, por lo que al momento de procesar se quiere buscar el canal con mayor visualización de este movimiento. Se genera entonces el archivo .csv el cual se ejecuta en Excel, donde se aprecian los datos por canal, en micro Volts [μV], mostrado en Figura 5-1:

0	0	0	0	0	0	0	0
-5068.63803	23251.9386	-22693.0779	-24449.3216	1744.95599	-22086.4516	-28769.2209	8157.33619
-5560.93521	23249.8823	-21979.4985	-25167.2596	2437.59184	-21608.9736	-28617.8548	8551.75508
-5279.41499	23469.354	-21580.6316	-24704.355	2805.54626	-21226.2671	-28222.5866	8879.99044
-4544.53433	23590.5228	-22023.2632	-23623.9163	2301.8497	-21458.7252	-28154.2573	8657.52353
-4473.45578	23418.4815	-22760.7143	-23518.8631	1623.09428	-22024.5596	-28511.7511	8191.91434
-5055.11523	23231.6432	-22699.2247	-24434.4577	1736.59643	-22087.4798	-28760.1461	8145.06509
-5551.77099	23236.3818	-21988.0592	-25159.213	2434.55201	-21606.5373	-28608.1542	8540.80272
-5281.73957	23442.979	-21590.7793	-24719.6883	2808.22847	-21227.3847	-28236.9811	8862.42197
-4556.38076	23582.9232	-22011.6627	-23634.1087	2317.47357	-21448.354	-28160.8734	8658.01527
-4470.88533	23443.359	-22746.4986	-23499.8195	1633.75606	-22009.2263	-28506.1632	8203.33608

Figura 5-1: Muestras en 8 canales [uV]

Se observan las primeras 10 muestras (filas) en los 8 canales (columnas) del EEG. Estos datos registrados son los que contienen la información que posteriormente deberá ser procesada para adquirir información relevante. El proceso de muestreo debe realizarse repetidas veces para poder seleccionar las muestras con menor ruido posible, asegurándose que los electrodos tengan un buen contacto con la piel.

5.2 Procesamiento de muestras

Una vez se obtienen las muestras, éstas deben pasar por una serie de procesos para poder ser visualizadas y poder también obtener información útil. El primer paso corresponde a la importación o lectura del archivo .csv que se obtuvo anteriormente, para luego pasar por diversos filtros. Es importante definir estos filtros, para poder entender su uso en esta aplicación.

Filtro pasa banda: Similar al filtro pasabajos, también existen bajas frecuencias asociadas al ruido. El cerebro humano no funciona a frecuencias menores a aquellas llamadas frecuencias Delta (0.5 – 4 [Hz]), al mismo tiempo, las frecuencias Delta están asociadas al estado de sueño correspondiente a la quiescencia o NREM (Non-rapid eye movement [26]), el estado de sueño ligero. Hay baja actividad cerebral, por lo que frecuencias menores a éstas pueden ser catalogadas como ruido.

Filtro notch [27] o elimina banda: Se utiliza para aislar un rango de frecuencias en específico. Debido a que la Jetson Nano está conectada al suministro eléctrico, existe un ruido asociado a la frecuencia de corriente de 50[Hz]. En este caso se va a dar uso de este filtro para poder aislar la frecuencia relacionada a esta corriente. Se procede en Listado 5-2:

Listado 5-2: Código inicialización de variables y lectura de datos

```
#Se inician o importan las librerías a ser utilizadas. En este caso es importante el uso de tensorflow y sklearn ya que estos incluyen las herramientas para el desarrollo y procesamiento de algoritmos de inteligencia artificial, como también el procesamiento de datos.

import argparse

import time
```

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import scipy

from scipy import signal

import brainflow

import tensorflow as tf

from sklearn.metrics import accuracy_score, precision_score, recall_score

from sklearn.model_selection import train_test_split

from tensorflow.keras import layers, losses

from tensorflow.keras.datasets import fashion_mnist

from tensorflow.keras.models import Model

from brainflow.board_shim import BoardShim, BrainFlowInputParams, LogLevels, BoardIds

from brainflow.data_filter import DataFilter, FilterTypes, AggOperations, NoiseTypes, WindowFunctions

#A continuación se definen las funciones representativas de Filtro del tipo Butterworth. Se define un filtro pasa banda de orden 6 con frecuencia de corte baja y alta, la cual se puede llamar de la librería Scipy. La segunda función se define para utilizar este filtro en una variable que puede contener múltiples datos.

def butter_bandpass(lowcut, highcut, fs, order=6):

    nyq = 0.5 * fs

    low = lowcut / nyq

    high = highcut / nyq

    sos = scipy.signal.butter(order, [low, high], analog=False, btype='band', output='sos')

    return sos

def butter_bandpass_filter(data, lowcut, highcut, fs, order=6):

    sos = butter_bandpass(lowcut, highcut, fs, order=order)

    y = scipy.signal.sosfilt(sos, data)

    return y

#Se abre el archivo csv en Python y debido a que el formato en el que se registró utiliza tabulaciones como medio de separación, se reemplazan estas tabulaciones con comas (‘,’). Se guarda un archivo que contiene la
```

```
misma información que el original, pero con estas nuevas separaciones y luego se lee y se asigna a la variable "raw".

with open('raw/botones2.csv') as fin, open('coma/botones.csv', 'w') as fout:

   for line in fin:

      fout.write(line.replace('\t', ','))

raw = DataFilter.read_file('coma/botones.csv')
```

Una vez definidos los filtros a utilizar, se realiza la lectura de los datos tomados del dispositivo EEG. A continuación, se tiene el código correspondiente al uso de estos filtros, como también a la visualización y transformación de los datos en Listado 5-3.

Listado 5-3: Código filtrado y graficado filtro Notch

```
#Se inician diccionarios en Python, con el fin de registrar distintos arreglos de datos (uno por cada canal) dentro de ellos.

channel=dict()

var=dict()

outputSignal=dict()

#Aquí se inician las variables para la utilización del primer filtro del tipo Notch en 50[Hz], esto para remover la interferencia o ruido generado por el paso de corriente del suministro eléctrico. Las variables 'i' y 'x' se utilizan para contar, las frecuencias corresponden a la de muestreo y la que se desea quitar, Q es el factor de calidad relacionado a la función del filtro Notch.

i=0

fs = 250.0  # Frecuencia de muestreo (Hz)

f0 = 50.0  #Frecuencia a quitar de la señal (Hz)

Q = 100.0  # Factor de calidad

b, a = signal.iirnotch(f0, Q, fs)

#Aquí se realiza un ciclo de 1 a 8, es decir en los 8 canales, para filtrar estos 50[Hz]. 'std' corresponde a la desviación estándar de los datos. Para reducir el ruido se debe también calcular el promedio de cada canal a cada uno de sus datos y luego este arreglo se divide por la desviación estándar para finalmente ser pasada por el filtro de 50[Hz]. Hay que mencionar que se ajusta cada canal a partir de las 31 muestras para evitar posible ruido o errores de muestreo que pudieron existir al inicio de cada muestreo.

for i in range (1,9):

   channel[i]=raw[i,31:]
```

```
    std=np.std(channel[i])
    var[i]=channel[i]-np.mean(channel[i])
    var[i]=var[i]/std
    outputSignal[i] = signal.filtfilt(b, a, var[i])
#A continuación se comienzan a graficar las funciones. Aquí se tiene un subplot (es decir, múltiples gráficas en una sola figura) Con los resultados del filtro de tipo notch de 50[Hz] para cada canal, es decir 8 plots.
plt.figure(figsize=(16,10))
plt.subplot(8, 1, 1)
plt.plot(outputSignal[1],label='Canal 1')
plt.legend()
plt.title('Señal Filtrada 50[Hz] Notch ', fontsize=20)
plt.subplot(8, 1, 2)
plt.plot(outputSignal[2], label='Canal 2')
plt.ylabel('Magnitud', fontsize=18)
plt.subplots_adjust(hspace=0.5)
plt.legend()
plt.subplot(8, 1, 3)
plt.plot(outputSignal[3], label='Canal 3')
plt.subplots_adjust(hspace=0.5)
plt.legend()
plt.subplot(8, 1, 4)
plt.plot(outputSignal[4], label='Canal 4')
plt.subplots_adjust(hspace=0.5)
plt.legend()
plt.ylabel('Magnitud', fontsize=18)
plt.subplot(8, 1, 5)
plt.plot(outputSignal[5], label='Canal 5')
plt.subplots_adjust(hspace=0.5)
```

```
plt.legend()

plt.subplot(8, 1, 6)

plt.plot(outputSignal[6], label='Canal 6')

plt.subplots_adjust(hspace=0.5)

plt.legend()

plt.ylabel('Magnitud', fontsize=18)

plt.subplot(8, 1, 7)

plt.plot(outputSignal[7], label='Canal 7')

plt.subplots_adjust(hspace=0.5)

plt.legend()

plt.subplot(8, 1, 8)

plt.plot(outputSignal[8], label='Canal 8')

plt.xlabel('Muestras', fontsize=20)

plt.subplots_adjust(hspace=0.5)

plt.legend()
```

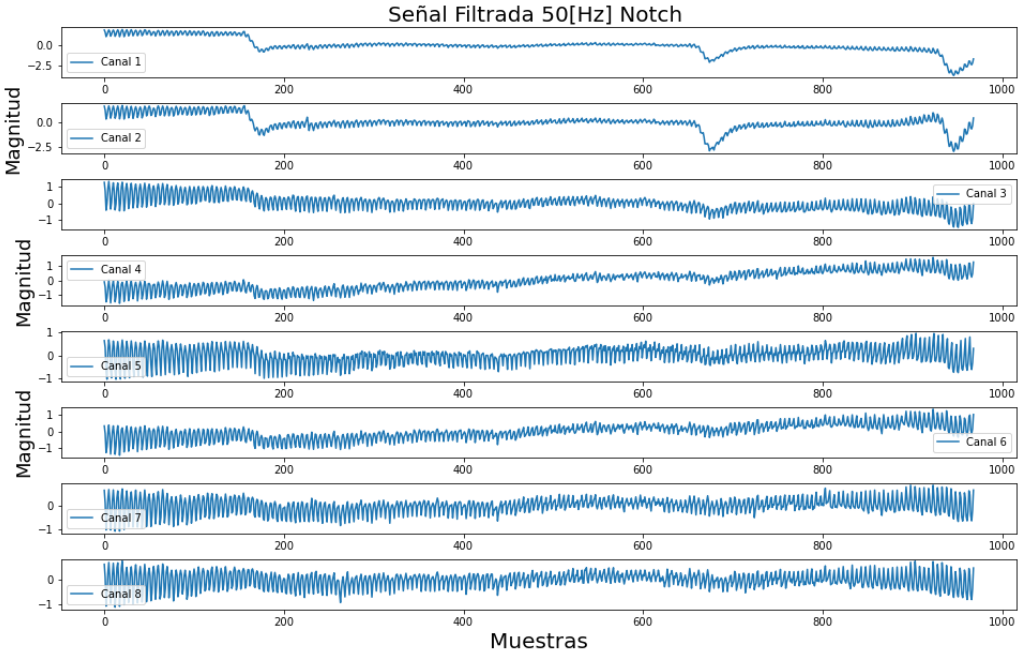

Figura 5-2: Señales (Ch. 1 a Ch. 8) filtradas con filtro Notch 50[Hz]

Se tienen en Figura 5-2 las 8 señales graficadas luego de ser pasadas por un filtro de 50[Hz]. Desde ya se puede tener una idea de cuáles serán los canales en donde se registró de mejor manera el movimiento de los dedos. Pero esto podrá visualizarse mejor una vez ejecutado el segmento de Listado 5-4.

Listado 5-4: Código filtrado y graficado de canales

```
#Ahora se realiza el gráfico de las señales pasadas por el filtro Butterworth de rango 2-35[Hz]. Se utiliza este rango ya que es el rango estimable en el que el cerebro funciona durante tareas normales. Se realiza nuevamente el procesamiento por rangos, siendo cada gráfica utilizada para 2 canales. Se tendrá entonces la gráfica de las señales filtradas al rango mencionado (utilizando la variable a la cual se le realizo el filtro notch), y las gráficas de la señal sin filtrar para contrastar.

plt.figure(figsize=(16,10))

plt.title('Datos EEG, Filtro 1.5-35[Hz]')

for x in range(1,3):

   filtered = butter_bandpass_filter(outputSignal[x], lowcut=2, highcut=35, fs=250, order=6)

   plt.plot(var[x], c='y', label='Raw Data channel' + str(x))

   plt.plot(filtered, label='Channel'+str(x))

plt.legend()

# Canales 3 y 4

plt.figure(figsize=(16,10))

plt.title('Datos EEG, Filtro 1.5-35[Hz]')

for x in range(3,5):

   filtered = butter_bandpass_filter(outputSignal[x], lowcut=2, highcut=35, fs=250, order=6)

   plt.plot(var[x], c='y', label='Raw Data channel' + str(x))

   plt.plot(filtered, label='Channel'+str(x))

plt.legend()

# Canales 5 y 6

plt.figure(figsize=(16,10))

plt.title('Datos EEG, Filtro 1.5-35[Hz]')

for x in range(5,7):

   filtered = butter_bandpass_filter(outputSignal[x], lowcut=2, highcut=35, fs=250, order=6)

   plt.plot(var[x], c='y', label='Raw Data channel' + str(x))
```

```
    plt.plot(filtered, label='Channel'+str(x))
plt.legend()
# Canales 7 y 8
plt.figure(figsize=(16,10))
plt.title('Datos EEG, Filtro 1.5-35[Hz]')
for x in range(7,9):
    filtered = butter_bandpass_filter(outputSignal[x], lowcut=2, highcut=35, fs=250, order=6)
    plt.plot(var[x], c='y', label='Raw Data channel' + str(x))
    plt.plot(filtered, label='Channel'+str(x))
plt.legend()
```

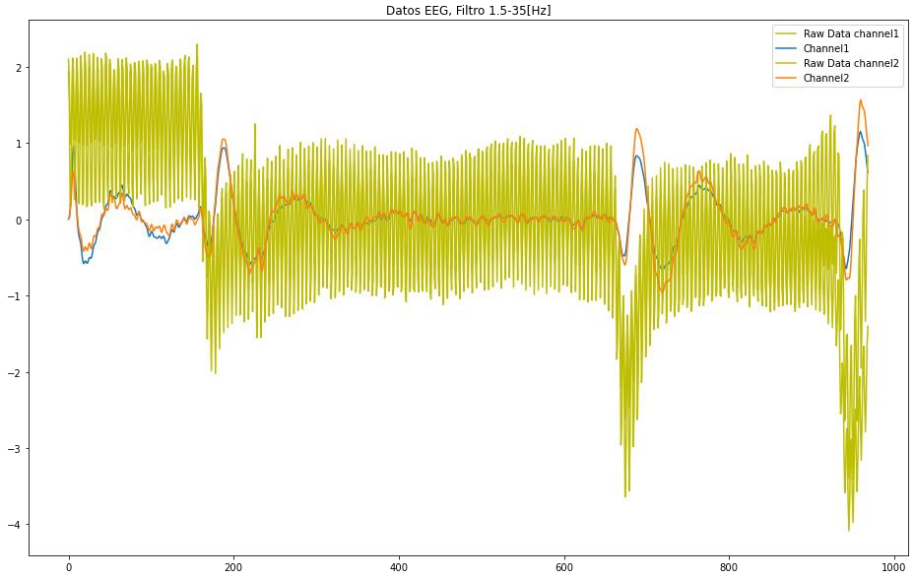

Figura 5-3: Señales de canales 1 y 2

Se tienen en Figura 5-3 los primeros dos canales, correspondientes a Fp2 y Fp1 en Figura 3-3, en los que se puede apreciar el impulso registrado que ocurre al existir movimiento de dedos índices tanto en las muestras 0-300 como alrededor de las 700 muestras y cercano a las 1000 muestras. Se observa una subida de amplitud mucho mayor al resto de la señal, lo cual indica que el cerebro está enviando un impulso eléctrico distinto al estado de reposo. Esto permite saber en qué posición del gráfico o de qué rango de muestras existen impulsos registrados con claridad.

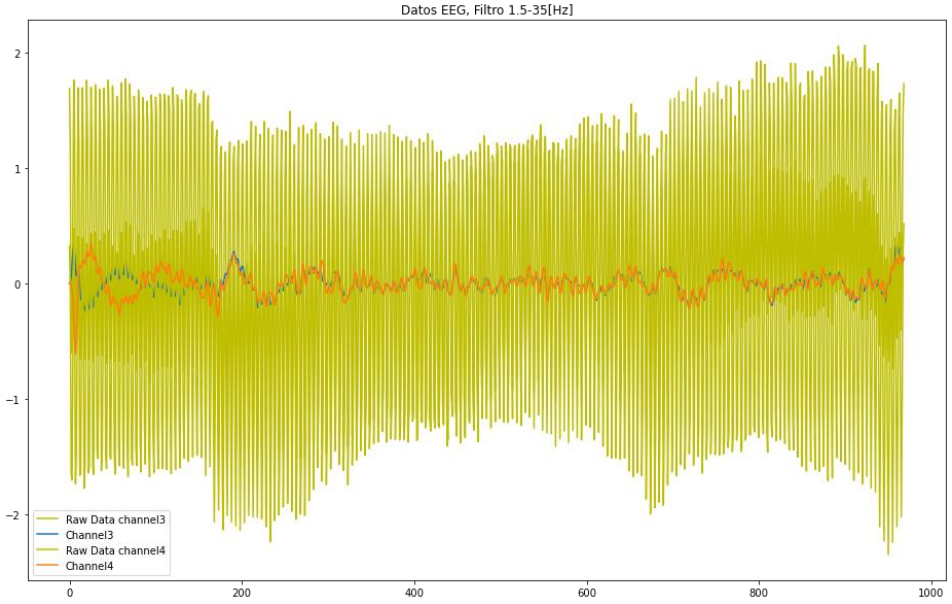

Figura 5-4: Señales de canales 3 y 4

En Figura 5-4, siendo los canales C3 y C4 en Figura 3-3, se observa que a medida que los electrodos se alejan del lóbulo frontal, existe una atenuación de las señales. De igual manera se aprecia una oscilación o impulso marcado distinto al resto de la señal, pero esta vez sólo puede ser observado a simple vista alrededor de las 200 y 700 muestras.

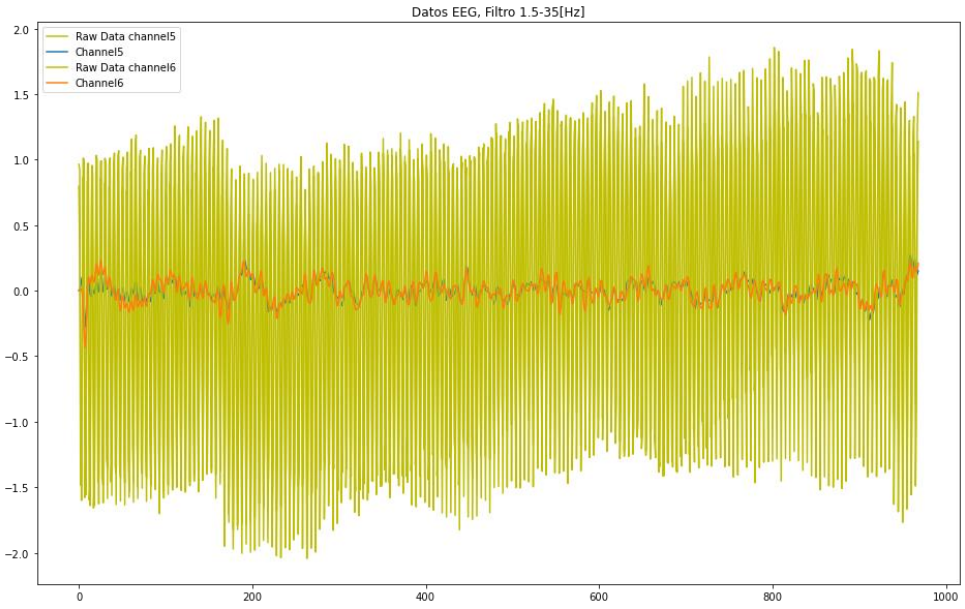

Figura 5-5: Señales de canales 5 y 6

Se observa en Figura 5-5, siendo los canales P3 y P4 en Figura 3-3, que a partir del canal 5 ya es difícil definir qué rango de muestras corresponde a un estado de reposo y a un estado de acción o movimiento relacionado a los dedos índices.

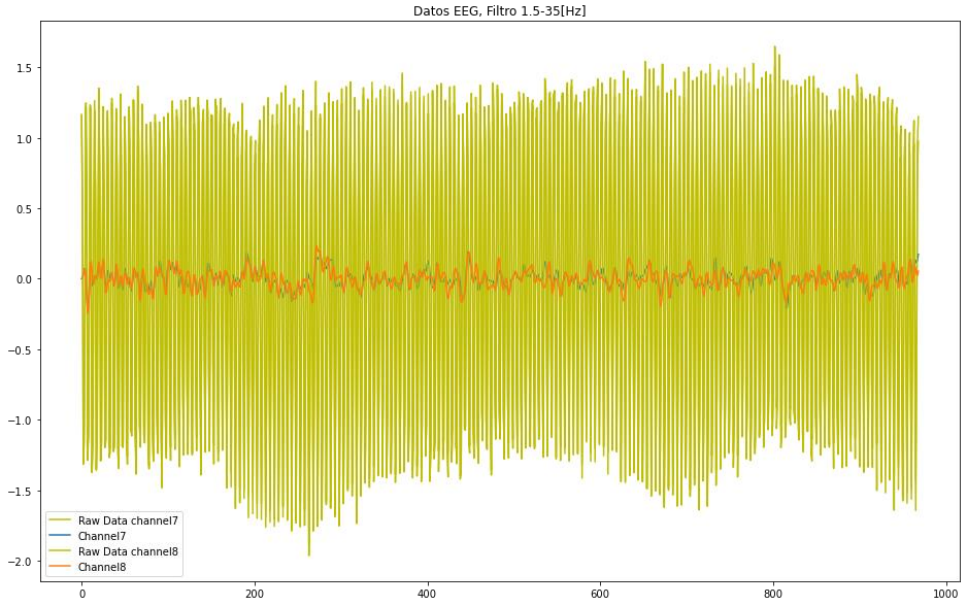

Figura 5-6: Señales de canales 7 y 8

Finalmente, en Figura 5-6, siendo los canales O2 y O1 en Figura 3-3, se observa que en los electrodos occipitales no se puede diferenciar ninguna parte de la señal para ser relacionada con el movimiento de los dedos, lo cual tiene sentido ya que esta parte del cerebro está relacionada principalmente con la visión.

En Listado 5-5 se utiliza la transformación de Fourier para definir qué rangos de frecuencia son los predominantes para cada canal.

Listado 5-5: Código clasificación Fourier

```
#El código acá se realizó para cada canal, pero se muestra el ejemplo en el primer canal para evitar la repetición y extensión excesiva, además que la señal seleccionada para el posterior procesamiento corresponde al canal 1 que fue el que mejor visualización entregó. Se define nuevamente la variable para el filtro de 2-35[Hz] de modo de evitar posibles datos erróneos. Se utilizan valores absolutos del rango de señal utilizado, y se calcula la transformada de Fourier discreta en una dimensión que contiene la librería Numpy. Luego se realiza la transformada de Fourier discreta esta vez para entregar frecuencias.

filtered = butter_bandpass_filter(outputSignal[1], lowcut=2, highcut=35, fs=250, order=6)

data = filtered[600:850]  # rango en el que se aprecia el impulso

fft_vals = np.absolute(np.fft.rfft(data))

fft_freq = np.fft.rfftfreq(len(data), 1.0/fs)
```

```
#Se definen las bandas cerebrales en rangos de frecuencia.

eeg_bands = {'Delta': (2, 4),

        'Theta': (4, 8),

        'Alpha': (8, 12),

        'Beta': (12, 30),

        'Gamma': (30, 45)}

#Se inicia un diccionario para registrar las muestras en cada banda. Se realiza un ciclo for para empezar a registrar los valores de frecuencias entregados por la FFT y ser asignado a cada banda. Luego se calcula un promedio de estas señales en frecuencia para posteriormente establecer cuál fue la banda predominante en cada canal.

eeg_band_fft = dict()

for band in eeg_bands:

   freq_ix = np.where((fft_freq >= eeg_bands[band][0]) &

                (fft_freq <= eeg_bands[band][1]))[0]

   eeg_band_fft[band] = np.mean(fft_vals[freq_ix])

# Aquí se realiza el gráfico de las bandas con su valor promedio.

df = pd.DataFrame(columns=['band', 'val'])

df['band'] = eeg_bands.keys()

df['val'] = [eeg_band_fft[band] for band in eeg_bands]

ax = df.plot.bar(x='band', y='val', legend=False)

ax.set_xlabel("Banda EEG")

ax.set_ylabel("Amplitud promedio de Banda")
```

Al ejecutar Listado 5-5 se realiza la transformación relacionada a la explicación de Figura 3-9, donde los rangos de frecuencia se catalogan según su amplitud media, lo que sirve para indicar qué rangos o bandas cerebrales se encuentran presentes y cuáles predominan durante la acción de movimiento [32]. Esto se aprecia en Figura 5-7 y Figura 5-8.

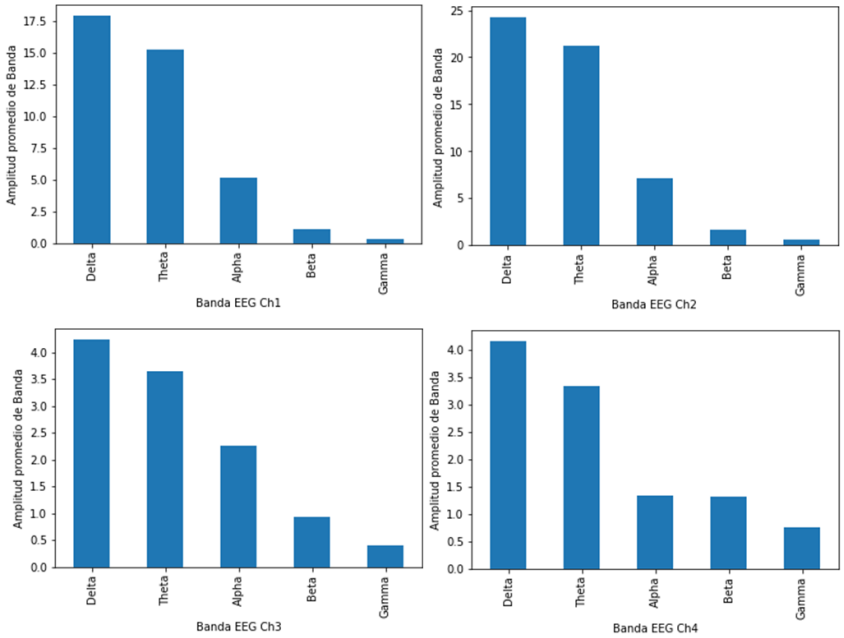

Figura 5-7: Bandas cerebrales para canales 1-4

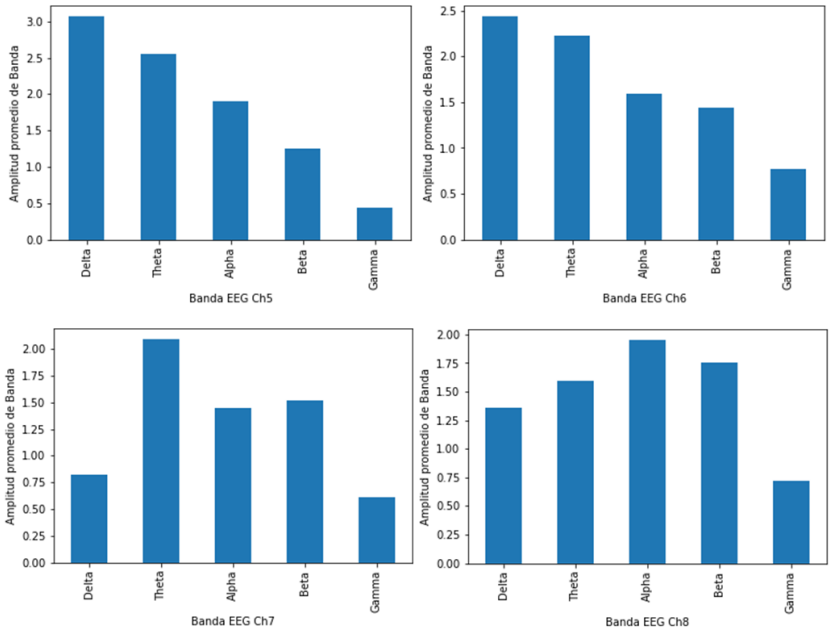

Figura 5-8: Bandas cerebrales para canales 5-8

Se puede apreciar de Figura 5-7 y Figura 5-8 que las bandas predominantes en los lóbulos frontal y central corresponden a la Delta y Theta, disminuyendo en amplitud a medida que se alejan los electrodos del lóbulo frontal. También se observa una amplitud promedio mayor en los canales 1 a 4. Con esta información se confecciona la Tabla 5-1, la cual ayudará a observar las amplitudes por banda con mayor detalle.

Tabla 5-1: Amplitud por Bandas de frecuencia por canal

Canal	Banda Delta	Banda Theta	Banda Alpha	Banda Beta	Banda Gamma
1	17.5	15.0	5.0	1.5	0.5
2	24.0	21.0	5.0	2.0	1.0
3	4.5	3.5	2.2	1.0	0.5
4	4.0	3.4	1.4	1.4	0.6
5	3.0	2.5	2.0	1.3	0.4
6	2.4	2.3	2.0	1.5	0.7
7	0.75	2.0	1.3	1.5	0.5
8	1.25	1.6	1.8	1.75	0.75

Se puede ver en Tabla 5-1 que en los dos primeros canales predominan los rangos Delta y Theta con amplitudes superiores a 15[-], mientras que a medida que va subiendo el número de canal, estos rangos se van perdiendo, obteniéndose amplitudes menores a 5[-], lo que podría indicar que la frecuencia emitida por el cerebro al mover los dedos índices se encuentra contenida dentro del rango de 2-8[Hz] (Delta y Theta [32]) en el lóbulo frontal. También se aprecia un cambio en la banda Alpha, que se encuentra en su máximo en las mediciones de los primeros 2 canales, con una amplitud de 5[-].

En Listado 5-6 se procederá a entrenar el modelo de Autoencoder. Corresponde a una red neural que se entrena mediante código para copiar la entrada a la salida. Un uso popular que tiene este algoritmo es el reconocimiento de letras o números escritos a mano. El algoritmo realiza una codificación de la entrada a una representación latente de menor dimensión, para luego decodificar la representación de vuelta a una señal. El Autoencoder con esto aprende a comprimir los datos al mismo tiempo que se minimiza el error de reconstrucción.

Listado 5-6: Código inicialización de datos Autoencoder

```
#Se inicializan las variables de entrenamiento y prueba. Se toman las muestras de 600 a 900 para entrenar y un rango de 100:350 para pruebas. Se utiliza el canal 1 para entrenar y el canal 2 para probar el modelo entrenado. Se calculan máximos y mínimos de tal forma de poder normalizar la señal completa a valores que vayan de 0 a 1. Luego se llevan ambas variables a un formato adecuado para su procesamiento (float32).

filtered = butter_bandpass_filter(outputSignal[1], lowcut=2, highcut=35, fs=250, order=6)

filtered2=butter_bandpass_filter(outputSignal[2], lowcut=2, highcut=35, fs=250, order=6)

train_data=filtered[600:900]

test_data=filtered[100:350]
```

```
atrain_data=filtered2[600:900]

atest_data=filtered2[100:350]

min_val = tf.reduce_min(train_data)

max_val = tf.reduce_max(train_data)

train_data =(train_data - min_val) / (max_val - min_val)

test_data = (test_data - min_val) / (max_val - min_val)

train_data = tf.cast(train_data, tf.float32)

test_data = tf.cast(test_data, tf.float32)

atrain_data =(atrain_data - min_val) / (max_val - min_val)

atest_data = (atest_data - min_val) / (max_val - min_val)

atrain_data = tf.cast(atrain_data, tf.float32)

atest_data = tf.cast(atest_data, tf.float32)

#Se realiza el gráfico de las muestras que se utilizarán para entrenar el modelo.

plt.figure(figsize=(16,10))

plt.grid()

plt.plot(np.arange(300), train_data)

plt.title("EEG")

plt.show()
```

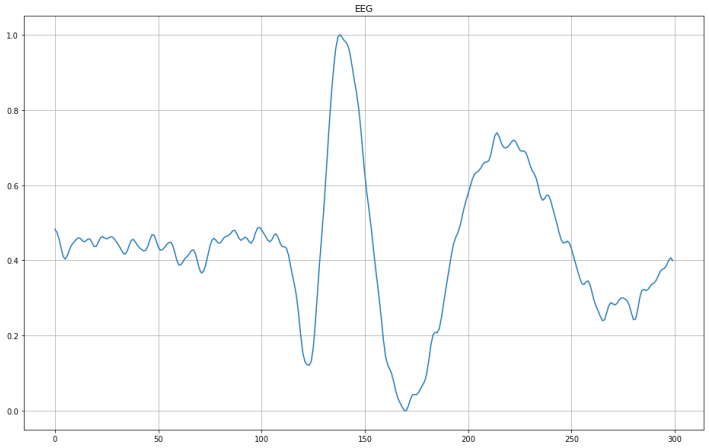

Figura 5-9: Muestras correspondientes a train_data del canal 1

Se utilizó el segmento de la señal mostrado en Figura 5-9 ya que corresponde a un impulso cerebral marcado, y lo que se quiere lograr es el entrenamiento de un modelo que reconozca estos impulsos.

Listado 5-7: Código algoritmo Autoencoder

```
# Se inicializa el Autoencoder bajo un modelo de detección de anomalías. Se utiliza el modelo secuencial de Tensorflow Keras que transforma capas a objetos con características de entrenamiento e inferencia. Las capas a utilizar son del tipo “RElu” o Rectified Linear Unit Activation (Activación de unidad lineal rectificada). El resultado corresponde a una codificación y decodificación de datos que corresponderán al modelo entrenado.

class AnomalyDetector(Model):

 def __init__(self):

  super(AnomalyDetector, self).__init__()

  self.encoder = tf.keras.Sequential([

   layers.Dense(32, activation="relu"),

   layers.Dense(16, activation="relu"),

   layers.Dense(8, activation="relu")])

  self.decoder = tf.keras.Sequential([

   layers.Dense(16, activation="relu"),

   layers.Dense(32, activation="relu"),

   layers.Dense(140, activation="sigmoid")])

 def call(self, x):

  encoded = self.encoder(x)

  decoded = self.decoder(encoded)

  return decoded

# Aquí se llama entonces a la clase definida anteriormente del Autoencoder, utilizando un optimizador llamado “Adam” que corresponde a un método de gradiente descendente estocástico, siendo este un método iterativo para optimizar una función objetiva basada en la suavidad de la señal. La pérdida a calcular se basa en “mae” Mean Absolute Error o Error Absoluto Promedio.

autoencoder = AnomalyDetector()

autoencoder.compile(optimizer='adam', loss='mae')

# Aquí se realiza el fit o ajuste del modelo entrenado según el número de épocas. Una época es una iteración por todos los datos de 0 a 100%. Se realiza entonces para 5 épocas.

history = autoencoder.fit(train_data, train_data,
```

```
        epochs=5,
        batch_size=64,
        validation_data=(test_data, test_data),
        shuffle=True)
encoded_data = autoencoder.encoder(test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()
# Se procede a graficar el input correspondiente a los datos de prueba, como también la información decodificada, correspondiente al modelo entrenado, llamado “Reconstrucción”.
plt.title('Entrada y Reconstrucción Autoencoder 5 épocas ', fontsize=12)
plt.plot(test_data, 'b')
plt.plot(decoded_data, 'r')
plt.legend(labels=["Input", "Reconstruccion"])
plt.show()
# Se realiza el ajuste para 15 épocas y se grafica de igual manera a la anterior.
history2 = autoencoder.fit(train_data, train_data,
        epochs=15,
        batch_size=64,
        validation_data=(test_data, test_data),
        shuffle=True)
plt.figure()
encoded_data = autoencoder.encoder(test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()
plt.title('Entrada y Reconstrucción Autoencoder 15 épocas ', fontsize=12)
plt.plot(test_data, 'b')
plt.plot(decoded_data, 'r')
plt.legend(labels=["Input", "Reconstruccion"])
plt.show()
# Se realiza el ajuste para 25 épocas y se grafica.
```

```
history3 = autoencoder.fit(train_data, train_data,
        epochs=25,
        batch_size=64,
        validation_data=(test_data, test_data),
        shuffle=True)
plt.figure()
encoded_data = autoencoder.encoder(test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()
plt.title('Entrada y Reconstrucción Autoencoder 25 épocas ', fontsize=12)
plt.plot(test_data, 'b')
plt.plot(decoded_data, 'r')
plt.legend(labels=["Input", "Reconstruccion"])
plt.show()
# Se realiza el ajuste para 50 épocas y se grafica.
history4 = autoencoder.fit(train_data, train_data,
        epochs=50,
        batch_size=64,
        validation_data=(test_data, test_data),
        shuffle=True)
plt.figure()
encoded_data = autoencoder.encoder(test_data).numpy()
decoded_data = autoencoder.decoder(encoded_data).numpy()
plt.title('Entrada y Reconstrucción Autoencoder 50 épocas', fontsize=12)
plt.plot(test_data, 'b')
plt.plot(decoded_data, 'r')
plt.legend(labels=["Input", "Reconstruccion"])
plt.show()
```

El resultado entregado por Listado 5-7 corresponde tanto a la reconstrucción realizada por el modelo entrenado, como también la pérdida entregada en cada época que se verá más adelante en Figura 5-11. En Figura 5-10 se observa la fiabilidad de la reconstrucción al ser comparada con la entrada de entrenamiento.

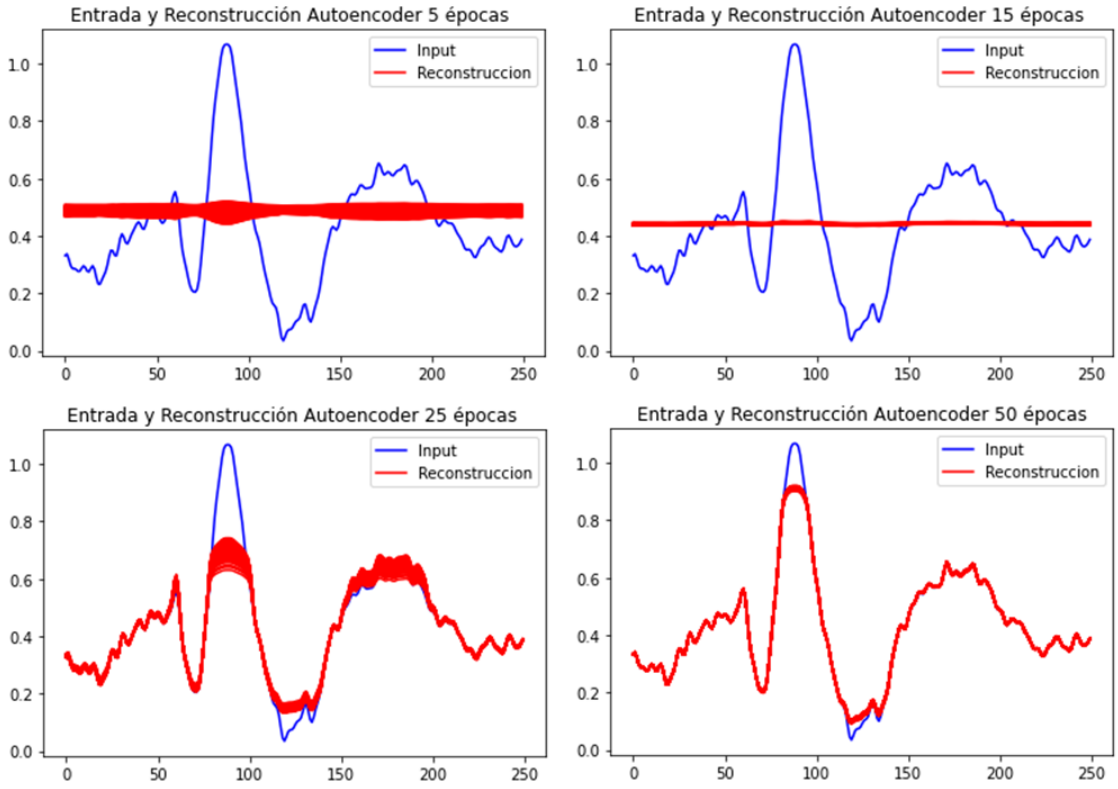

Figura 5-10: Entrada vs Reconstrucción por épocas

Se puede apreciar de Figura 5-10 que mientras mayor sea el número de épocas o iteraciones, mejor será el resultado del modelo entrenado, por lo que entregará mayor confianza. Se continúa el código para obtener la información relacionada a la pérdida en Listado 5-8.

Listado 5-8: Código gráfico de pérdidas y validación

```
# Aquí se graficarán las pérdidas de validación y entrenamiento ocurridas durante el proceso de creación o ajuste del modelo del Autoencoder. Primero se tiene para 5 épocas.

plt.plot(history.history["loss"], label="Pérdida entrenamiento")

plt.plot(history.history["val_loss"], label="Pérdida validación")

plt.title('Pérdidas en 5 épocas', fontsize=12)

plt.legend()

# Para 15 épocas.
```

```
plt.figure()

plt.plot(history2.history["loss"], label="Pérdida entrenamiento")

plt.plot(history2.history["val_loss"], label="Pérdida validación")

plt.title('Pérdidas en 15 épocas', fontsize=12)

plt.legend()

# Para 25 épocas.

plt.figure()

plt.plot(history3.history["loss"], label="Pérdida entrenamiento")

plt.plot(history3.history["val_loss"], label="Pérdida validación")

plt.title('Pérdidas en 25 épocas', fontsize=12)

plt.legend()

# Para 50 épocas.

plt.figure()

plt.plot(history4.history["loss"], label="Pérdida entrenamiento")

plt.plot(history4.history["val_loss"], label="Pérdida validación")

plt.title('Pérdidas en 50 épocas', fontsize=12)

plt.legend()

# Este segmento de código corresponde al ajuste del modelo entrenado a 50 épocas al ingresar una entrada distinta al canal 1. Se utilizó el canal 2 de la señal EEG para poder visualizar el modelo siendo aplicado.

plt.figure()

encoded_data = autoencoder.encoder(atest_data).numpy()

decoded_data = autoencoder.decoder(encoded_data).numpy()

encoded_data = autoencoder.encoder(atest_data).numpy()

decoded_data = autoencoder.decoder(encoded_data).numpy()

plt.plot(atest_data, 'b')

plt.plot(decoded_data, 'r')

plt.legend(labels=["Input prueba Ch2.", "Reconstrucción"])

plt.show()
```

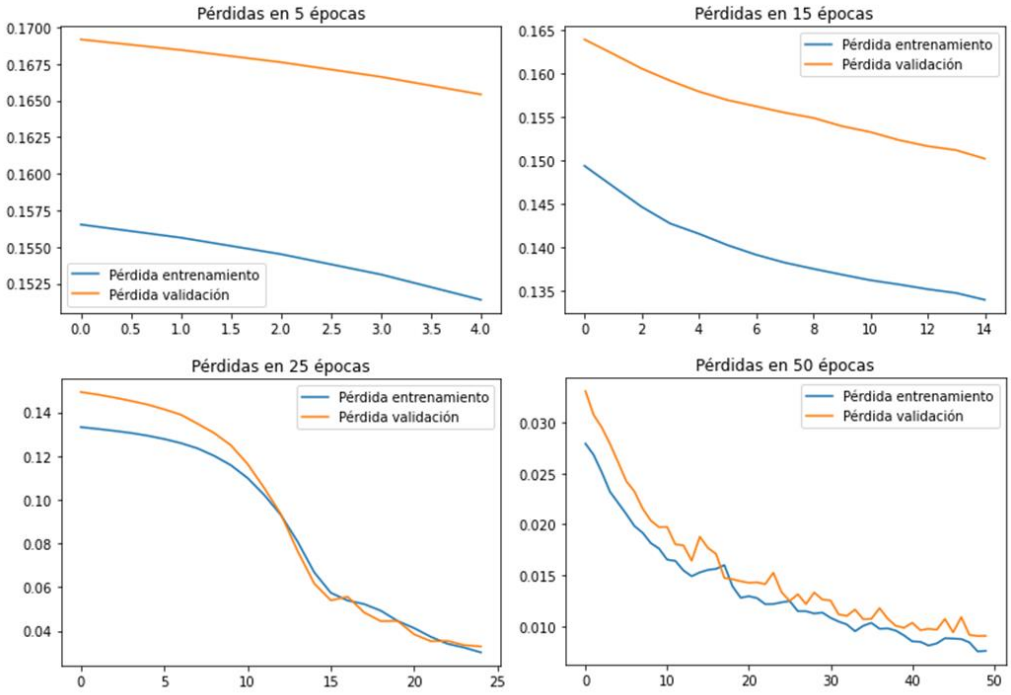

Figura 5-11: Gráficos de pérdida entrenamiento vs validación

Se tiene en Figura 5-11 los gráficos que muestran la pérdida de información una vez validado el modelo con los datos de prueba. También se puede concluir que mientras mayor sea el número de épocas, menor será la pérdida y más confiable el modelo entrenado. Se entregó también el gráfico al utilizarse este modelo, observado en Figura 5-12.

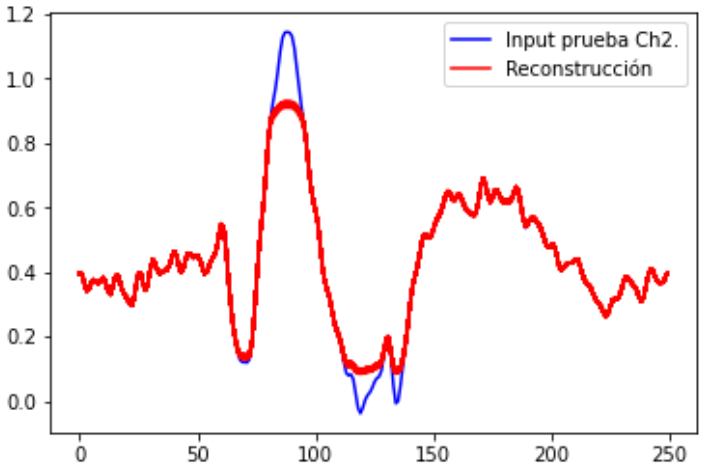

Figura 5-12: Prueba de reconstrucción utilizando segmento del canal 2

Se tiene Figura 5-12 que muestra como entrada una parte de la señal correspondiente al canal 2, durante el mismo período de muestras que se utilizó para el canal 1 de entrenamiento. Debido a que en el canal dos el peak o pico que existe alrededor de la muestra 80 es mayor que el contenido en el canal uno, el modelo no lo reconstruye por completo, lo mismo con la bajada existente alrededor de la

muestra 125. Es justamente esta diferencia la que se utiliza para detectar anomalías, ya que generalmente el modelo entrenado se realiza con información "normal" y se prueba posteriormente con información "anormal". La utilización del modelo en esta aplicación podría significar un mayor impulso emitido por un sector del cerebro respecto al movimiento de los dedos, como también un movimiento más fuerte respecto al anterior.

A continuación, en Listado 5-9 y Listado 5-10 se procede a realizar otro algoritmo de inteligencia artificial llamado K-medios. Al igual que con el Autoencoder, la función de este algoritmo es de búsqueda de anomalías, con la diferencia de que en el Autoencoder se entrena un modelo, mientras que K-medios realiza agrupaciones para realizar la comparación entre datos establecidos como parámetros normales y aquellos que pueden ser anomalías.

Listado 5-9: Código algoritmo K-medios parte 1

```
# Se importan las librerías a utilizar específicas para el modelo K-medios contenidos en sklearn

from sklearn.cluster import KMeans

from sklearn.preprocessing import scale

# Se selecciona el rango a utilizar de la señal, las muestras 600 a 850 y se calcula el promedio de estas para luego restar este valor a los datos.

kdatos = filtered[600:850]

kprom=np.average(x)

kdatos=(kdatos-kprom)

# Se establece el rango de estos datos para poder ser graficada la muestra a utilizar

kdatos_ax = range(250)

plt.plot(kdatos_ax, kdatos)

plt.show()
```

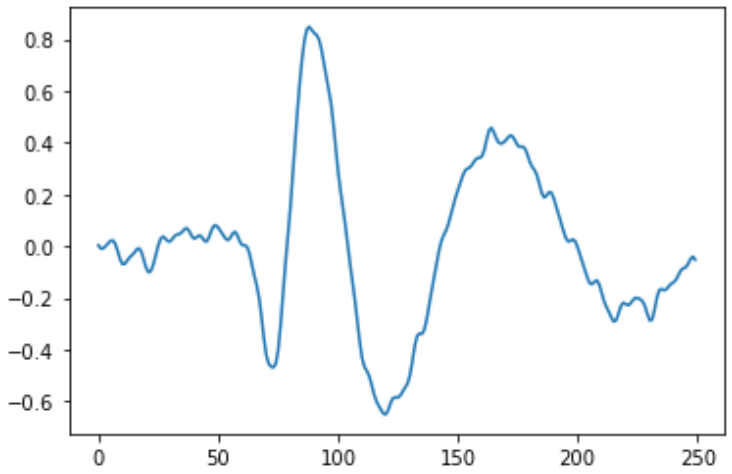

Figura 5-13: Segmento que muestra el impulso cerebral

Se observa en Figura 5-13 la muestra seleccionada para utilizar el algoritmo K-means o K-medios, siendo similar a la utilizada para el Autoencoder, pero con 50 datos menos, para centrarse de mejor forma en el impulso existente.

Listado 5-10: Código algoritmo K-medios parte 2

```
#Se remodelan los datos al adecuado y se inicializa el algoritmo de K-Medios estableciendo como 1 el número de agrupaciones, al cual se le ajustan los datos.

kdatos=kdatos.reshape(-1,1)

kmeans = KMeans(n_clusters = 1).fit(kdatos)

# Se obtiene el centro de la agrupación para poder realizar el cálculo de la distancia Euclidiana.

cent = kmeans.cluster_centers_

dist= sqrt((kdatos - cent)**2)

order_index = argsort(dist, axis = 0)

# Se selecciona la sensibilidad del sistema, a menor valor, mayor será la cantidad de datos que se tomarán como anomalías, lo cual se verá en el gráfico a continuación como puntos rojos.

indexes = order_index[-20:]

values = kdatos[indexes]

# Se realiza el gráfico de la señal original con los puntos establecidos como anomalías por el algoritmo K-medios.

plt.plot(kdatos_ax, kdatos)

plt.scatter(indexes, values, color='r')

plt.show()
```

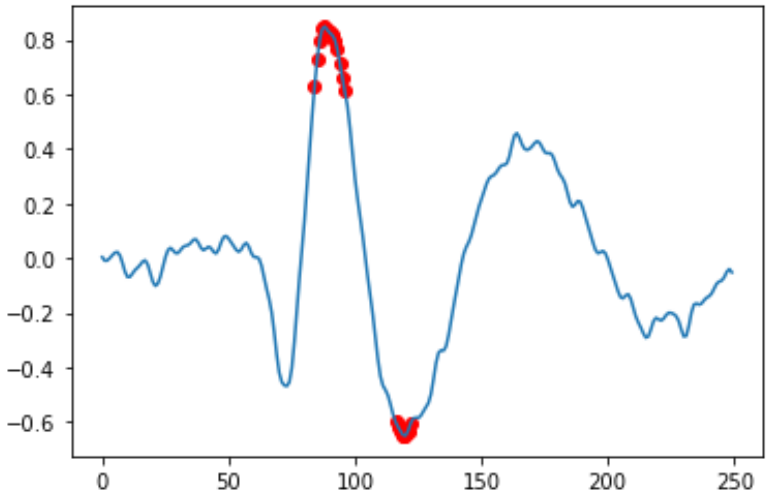

Figura 5-14: Segmento con anomalías marcadas (en rojo)

En Figura 5-14 se tiene finalmente el resultado del algoritmo K-medios. Se puede observar que el peak principal como su bajada se encuentran fuera del rango "normal" establecido por la sensibilidad del algoritmo.

La utilidad de este algoritmo para el caso de Electroencefalograma es el uso de estas anomalías como acciones, es decir, asignar acciones a ejecutar cada vez que el sistema detecte estas anomalías. Se puede utilizar este algoritmo para otros tipos de señales, como estado de relajación versus estado de concentración, o para detectar estados de ánimo. También se podría utilizar para entrenar un modelo en una persona con baja capacidad de movimiento para utilizar el cerebro como "controlador" de una silla de ruedas o de prótesis o ayudas electrónicas.

Discusión y conclusiones

Mediante el estudio realizado en este proyecto se puede afirmar lo positivo que resulta la utilización de inteligencia artificial en el procesamiento y visualización de señales cerebrales. El uso de estos algoritmos permite la apertura de muchas posibles aplicaciones tal como la que se mostró anteriormente que corresponde a la detección de anomalías. Al registrar señales cerebrales de una persona en distintos estados como relajación, concentración o acción, se puede lograr el entrenamiento de un modelo que utilice estas anomalías como gatillo para la toma de acciones, como lo sería por ejemplo el movimiento de una silla de ruedas al entrar en un estado de concentración, o algún estado de fácil adquisición y que sea controlable por el usuario. Por otro lado, este mismo tipo de algoritmo permitiría la comparación de señales cerebrales entre una y más persona, encontrándose de manera automática las diferencias entre ellas, lo que es un gran uso en la actualidad para la detección y diagnóstico de enfermedades como lo son la epilepsia, y la ayuda que significa al permitir la búsqueda de impulsos que gatillen estas condiciones.

En lo que respecta al análisis de señales, se pudo hacer uso de filtros de distintos tipos para permitir la eliminación del ruido producido por fuentes de baja frecuencia, como también aquella introducida por el suministro eléctrico del hogar. Esto afectó a la señal de manera positiva para lograr una mejor visualización de la señal cerebral, lo que permitió diferir entre canales al momento de seleccionar la señal más apta para el posterior procesamiento. También se pudo catalogar canal por canal según rangos de frecuencia o bandas cerebrales al estudiar el impulso específico del movimiento de los dedos índices.

Por otro lado, la realización de este proyecto permitió solucionar problemas existentes como también realizar aportes como se describe a continuación:

Creación de un Electroencefalograma de bajo costo: Se realizó este proyecto utilizando materiales con un costo total de 375 dólares, lo cual es un precio competitivo en comparación a la oferta actual en el mercado.

Utilización de Inteligencia Artificial en EEG: Debido al rumbo en open-source o código abierto que tomó este proyecto, se podrá acceder libremente al código escrito y utilizado para la finalización de este proyecto. Por otro lado, debido a la utilización de Jupyter Notebooks, esto permite una integración fácil en lo que significa la toma de muestras y el análisis de estas.

Toma de muestras útiles: Se tomaron múltiples muestras las cuales podrán ser utilizadas para el uso de algoritmos de inteligencia artificial, como también para la visualización y procesamiento de estas.

Plantilla para futuras aplicaciones: El uso de este código no está limitado al proyecto realizado, sino que también sirve como plantilla para su posterior desarrollo en múltiples otros proyectos relacionados a las señales cerebrales, como puede ser la integración de tecnologías computacionales mediante el uso de señales cerebrales.

Finalmente, el desarrollo de este proyecto permitió ampliar conocimientos en el campo biológico, permitiendo el estudio de ámbitos cerebrales mediante la utilización de dispositivos electrónicos y tecnologías de programación como Machine Learning.

Debido a que el estudio se realizó registrando sólo el movimiento de dedos índices, no todos los electrodos fueron de utilidad, por lo que ampliar el estudio de señales cerebrales no solo a movimientos de dedos, sino también movimiento de otras partes del cuerpo, o analizar las respuestas cerebrales frente a los sentidos (olfato, visión, audición, gusto, tacto). También se puede realizar un estudio de carácter mental como lo son los estados de ánimo, concentración o relajación, lo que permitiría hacer un mejor uso de los electrodos en lóbulos distintos al frontal. Otra forma de mejorar el proyecto es utilizando o agregando mayor cantidad de algoritmos de inteligencia artificial, como también utilizando distintos códigos de programación o distintas tecnologías como lo son OpenAI e IBM Watson.

Bibliografía

[1] Oracle, «Oracle,» [En línea]. Available: https://www.oracle.com/cl/artificial-intelligence/what-is-artificial-intelligence.html.

[2] M. Clinic, «MayoClinic,» [En línea]. Available: https://www.mayoclinic.org/es-es/tests-procedures/eeg/about/pac-20393875.

[3] U.S. Department of Health & Human Services, «Nibib.nih,» [En línea]. Available: https://www.nibib.nih.gov/espanol/temas-cientificos/inteligencia-artificial-ia.

[4] A. Kulshreshth, A. Anand y A. Lakanpal, «Neuralink- An Elon Musk Start-up Achieve symbiosis with Artificial Intelligence,» *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS),* Octubre 2019.

[5] C.Castellaro, G.Favaro, A.Castellaro, A.Casagrande, S.Castellaro, D.V.Puthenparampil y C. Salimbeni, «An artificial intelligence approach to classify and analyse EEG traces,» *Neurophysiologie Clinique/Clinical Neurophysiology,* vol. 32, nº 3, pp. 193-214, Junio 2002.

[6] M. K. Sana, Z. M.Hussain, P. AhmadShah y M. HaisumMaqsood, «Artificial intelligence in celiac disease,» *Computers in Biology and Medicine,* vol. 125, Octubre 2020.

[7] FawazAl-Mufti, VincentDodson, JamesLee, EthanWajswol, C. Gandhi, C. Scurlock, C. Cole, K. Lee y f. S. A.Mayer, «Artificial intelligence in neurocritical care,» *Journal of the Neurological Sciences,* vol. 404, pp. 1-4, 2019.

[8] OpenBCI, «Openbci,» [En línea]. Available: https://openbci.com/.

[9] K. Arana y A. Vivas, «Prótesis de mano virtual movida por señales encefalograficas - EEG,» *Prospect,* vol. 144, nº 2, pp. 99-110, 2016.

[10] C. Fadzal, W. Mansor, L. Khuan y A. Zabidi, «Short-time Fourier Transform Analysis of EEG Singal from Writing,» de *IEEE 8th International Colloquium on Signal Processing and its Applications*, 2012.

[11] T. Kam, H. Suk y S. Lee, «Non-Homogeneous Spatial Filter Optimization for EEG-based Brain-Computer interfaces,» de *International Winter Workshop on Brain-Computer Interface (BCI)*, 2013.

[12] K. Nakayama, Y. Kaneda y A. Hirano, «A bRain Computer Interface Based on FFT and Multilayer Neural Network,» de *International Sympsoium on Intelligent Signal Processing and Communication Systems*, 2007.

[13] Y. Kim, N. Kwak y S. Lee, «Classification of Motor Imagery for Ear-EEG based Brain-Computer Interface,» de *6th International Conference on Brain and Computer Interface (BCI)*, 2018.

[14] S. Sridhar, U. Ramachandraiah, E. Sathish, G. Muthukumaran y P. Rajendra, «Identification of Eye Blink Artifacts using Wireless EEG Headset for Brain-Computer Interface System,» de *IEEE Sensors*, 2018.

[15] S. Swee y L. You, «Fast Fourier Analysis and EEG Classification Brainwave Controlled Wheelchair,» de *2nd International Conference on Control Science and Systems Engineering*, 2016.

[16] A. Bablani, E. Damodar Reddy, K. Venkatanareshbabu y R. Dharavath, «A Multi Stage EEG Data Classification Using K-Means and Feed Forward Neural Network,» *Original Article,* vol. Volume 8, nº Issue 3, pp. P7128-724, 2020.

[17] MIT, «www.deeplearningbook.org,» [En línea]. Available: https://www.deeplearningbook.org. [Último acceso: 23 Junio 2021].

[18] EDX, «Machine Learning (aprendizaje automático) con Python: una introducción práctica,» [En línea]. Available: https://www.edx.org/es/course/machine-learning-aprendizaje-automatico-con-python. [Último acceso: 14 07 2021].

[19] K. Academy, «The Synapse,» [En línea]. Available: https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/the-synapse. [Último acceso: 14 07 2021].

[20] E. R. Society, «ERS-Education,» [En línea]. Available: https://www.ers-education.org/lrmedia/2016/pdf/298830.pdf. [Último acceso: 14 07 2021].

[21] «AIoT - Artificial Intelligence on Thoughts,» [En línea]. Available: https://www.hackster.io/dnhkng/aiot-artificial-intelligence-on-thoughts-f62249. [Último acceso: 14 07 2021].

[22] SapienLabs, «Brain Waves, Sine Waves and the Fourier Transform,» [En línea]. Available: https://sapienlabs.org/brain-waves-sine-waves/. [Último acceso: 14 07 2021].

[23] Nvidia, «Nvidia Developer,» [En línea]. Available: https://developer.nvidia.com/embedded/jetson-nano-2gb-developer-kit. [Último acceso: 14 07 2021].

[24] OpenBCI, «Docs OpenBCI,» [En línea]. Available: https://docs.openbci.com/docs/02Cyton/CytonDataFormat. [Último acceso: 14 07 2021].

[25] Physio-Pedia, «Somatosensation,» [En línea]. Available: https://www.physio-pedia.com/Somatosensation. [Último acceso: 07 14 2021].

[26] P. D, A. GJ y F. D, «National Center for Biotechnology Information,» [En línea]. Available: https://www.ncbi.nlm.nih.gov/books/NBK10996/. [Último acceso: 07 14 2021].

[27] G. Ellis, «Filters in Control Systems,» *Control System Design Guide (Fourth Edition),* pp. 166-169, 2012.

[28] Microchip, «PIC32MX250F128B,» [En línea]. Available: https://www.microchip.com/wwwproducts/en/PIC32MX250F128B. [Último acceso: 14 07 2021].

[29] T. Instruments, «ADS1299,» [En línea]. Available: https://www.ti.com/product/ADS1299. [Último acceso: 14 07 2021].

[30] R. M. El-Baba y M. P. Shury, «Neuroanatomy, Frontal Cortex,» [En línea]. Available: https://www.ncbi.nlm.nih.gov/books/NBK554483/. [Último acceso: 14 07 2021].

[31] Jupyter, «Jupter Notebook,» [En línea]. Available: https://jupyter.org/about. [Último acceso: 14 07 2021].

[32] P. A. Abhang, B. W. Gawali y S. C. Mehrotra, «Technological Basics of EEG Recording and Operation of Apparatus,» *Introduccion to EEG - and Speech-Based Emotion Recognition,* vol. Chapter 2, pp. 19-50, 2016.