UNIVERSIDAD DE VALLADOLID

FACULTAD DE MEDICINA

ESCUELA DE INGENIERÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA BIOMÉDICA

Desarrollo y evaluación de una herramienta de eliminación online de artefactos oculares de la señal de electroencefalograma

Autor:

D. Juan Para Maeso

Tutores:

D. Diego Marcos Martínez

Dr. Roberto Hornero Sánchez

Valladolid, 21 de septiembre de 2023

TÍTULO:	**Desarrollo y evaluación de una herramienta de eliminación online de artefactos oculares de la señal de electroencefalograma**
AUTOR:	**D. Juan Para Maeso**
TUTORES:	**Dr. Roberto Hornero Sánchez** **D. Diego Marcos Martínez**
DEPARTAMENTO:	**Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE:	**Dr. Roberto Hornero Sánchez**
VOCAL:	**Dr. Jesús Poza Crespo**
SECRETARIO:	**Dr. Carlos Gómez Peña**
SUPLENTE 1:	**Dra. María García Gadañón**
SUPLENTE 2:	**Dr. Daniel Álvarez González**

FECHA:	
CALIFICACIÓN:	

Agradecimientos

En primer lugar, quiero transmitir mi agradecimiento a mi tutor Roberto Hornero por darme la oportunidad de llevar a cabo este trabajo. Mención especial merece mi tutor Diego Marcos, debido a la ayuda y el apoyo constante que me ha brindado a lo largo de todos estos meses de trabajo. No me quiero olvidar de Sergio Pérez, Eduardo Santamaría, Rubén Ruiz y Álvaro Fernández, que también han aportado su granito de arena a este trabajo, aportando ideas y resolviendo dudas siempre con una sonrisa. También quiero agradecer al resto de estudiantes de doctorado e investigadores del Grupo de Ingeniería Biomédica el trato recibido, han hecho de esta experiencia algo inolvidable.

Finalmente, quiero destacar el apoyo y compresión que me han transmitido desde que comencé con el trabajo mi familia y amigos.

Muchas gracias.

Resumen

El electroencefalograma (EEG) es una técnica neurofisiológica que permite registrar la actividad eléctrica del cerebro. Para ello se emplean electrodos colocados en la superficie del cuero cabelludo. Debido a la baja amplitud que presenta la señal de EEG, es una señal muy susceptible de ser enmascarada por ruido y señales independientes de la actividad cerebral. Los movimientos oculares y los parpadeos generan grandes perturbaciones, que son medibles en los electrodos, denominadas artefactos oculares. En consecuencia, se requiere de un método de detección y filtrado que permita recuperar la señal de EEG lo más limpia posible y conservando la actividad cerebral. En la actualidad existen métodos capaces de detectar estos artefactos oculares, sin embargo, o presentan un bajo rendimiento en la clasificación o se limitan a la detección de parpadeos.

El objetivo de este TFG era desarrollar un novedoso algoritmo de detección de artefactos oculares a través de inteligencia artificial. Además, el algoritmo debía ser capaz de llevarlo a cabo en tiempo en tiempo real y no requerir de señales adicionales al EEG. Para ello se adquirió una base de datos con artefactos oculares etiquetados y se entrenó con ella la red neuronal EEG-Inception. Se comprobó la validez del modelo y del algoritmo mediante una validación cruzada intersujeto. En ella el algoritmo demostró una elevada sensibilidad en la detección de los parpadeos, 97.6 % y de los movimientos oculares, 91 %. Además, el algoritmo fue evaluado sobre una base de datos propia, que no incluía movimientos oculares. En ella se obtuvo una precisión en la clasificación de parpadeos del 96 %. Los resultados están al nivel de los que presentan otros algoritmos de la literatura para clasificación de parpadeos. Sin embargo, ninguno de estos permite también la detección de movimientos oculares. El resultado final de este trabajo es un algoritmo confiable en la detección de artefactos oculares, aplicable en tiempo real y que no requiere de calibración, ni de una señal adicional que registre la actividad ocular.

Palabras clave

Electroencefalografía, artefactos oculares, detección automática, aprendizaje profundo.

Abstract

The electroencephalogram (EEG) is a neurophysiological technique that performs the recording of the brain's electrical activity. Electrodes are placed on the surface of the scalp. Due to the low amplitude of the EEG signal, it is highly susceptible to being masked by noise and signals unrelated to brain activity. Eye movements and blinks generate significant disturbances that can be measured in the electrodes. These disturbances are known as ocular artifacts. A detection and filtering method is required to recover the EEG signal, as clean as possible while preserving brain activity. Currently, there are methods capable of detecting these ocular artifacts. However, they either exhibit poor classification performance or are limited to blink detection.

This thesis aims to develop a novel algorithm for detecting ocular artifacts using artificial intelligence. Furthermore, the algorithm must perform in real time and not require additional signals to the EEG. To achieve this, a database with labeled ocular artifacts was acquired and used to train the EEG-Inception neural network. The validity of the model and algorithm was tested through inter-subject cross-validation. In it, the algorithm demonstrated a high sensitivity in detecting blinks, 97.6%, and eye movements, 91%. Additionally, the algorithm was evaluated on a private database, achieving a blink classification accuracy of 96%. The results are comparable to those presented by other algorithms in the literature for blink classification. However, none of these algorithms also enable the detection of eye movements. The outcome of this work is a reliable algorithm for detecting ocular artifacts that does not require calibration or an additional signal to record the ocular activity and can be applied in real time.

Keywords

Electroencephalography, ocular artifacts, automated detection, deep learning.

Índice

Índice de tablas

Índice de figuras

Capítulo 1: Introducción

1.1. Introducción a la ingeniería biomédica

La aplicación de la ingeniería al ámbito medico tiene un largo recorrido. Ya en el antiguo Egipto, hace más de 3000 años, se empleaban prótesis funcionales, como la hallada de un dedo gordo en Tebas-Oeste [1]. El desarrollo de la instrumentación eléctrica y electrónica durante el siglo XIX generó multitud de dispositivos aplicables a la medicina. Entre los avances más destacables, se encuentran la invención de la máquina de rayos X por Roentgen y los registros de actividad eléctrica del corazón mediante electrodos adheridos a la piel por A.D. Waller. Lo que sentó las bases para la invención del electrocardiograma (ECG) por Einthoven y del electroencefalograma (EEG) de la mano de Hans Berger.

El aumento del conocimiento y la mejora de la tecnología han generado una disciplina enorme y diversa, que hoy en día se conoce como ingeniería biomédica y que se puede definir como:

"Disciplina en la que se aplican principios de ingeniería eléctrica, mecánica, química, óptica y otros, con el objetivo de comprender, modificar o controlar sistemas biológicos. Además, abarca el diseño y la fabricación de productos que puedan monitorear las funciones fisiológicas de los individuos, así como asistir en el diagnóstico y tratamiento de los pacientes" [2].

La ingeniería biomédica abarca grandes áreas del conocimiento. Algunos ejemplos son:

- Aplicación de análisis de sistemas de ingeniería a problemas biológicos.
- Detección, medición y monitoreo de señales fisiológicas mediante biosensores.
- Procedimientos y dispositivos terapéuticos y de rehabilitación.
- Desarrollo de dispositivos para reemplazar o mejorar funciones corporales.
- Análisis informático de datos relacionados con el paciente y toma de decisiones clínicas, por ejemplo, mediante informática médica e inteligencia artificial.
- Imágenes médicas para la representación gráfica de detalles anatómicos o funciones fisiológicas.
- Creación de nuevos productos biológicos mediante biotecnología e ingeniería de tejidos.
- Desarrollo de nuevos biomateriales para su uso en el cuerpo.
- Interpretación diagnóstica más objetiva y cuantificable, a través de técnicas de procesamiento de señales de biomédicas.

1.2. Las señales biomédicas

Las señales biomédicas o bioseñales hacen referencia a las señales de origen eléctrico, mecánico, químico o térmico que se originan en un sistema biológico [3]. Existen multitud de señales biomédicas debido al número casi ilimitado de mecanismos biológicos capaces de generarlas. Las bioseñales se miden y analizan para extraer datos acerca del estado fisiológico de un individuo [3]. La técnica empleada para la medición

de la señal depende de su naturaleza. En la Tabla 1 se encuentran algunas de las señales biomédicas y el biosensor empleado en su medición.

Señal Biomédica	Biosensor
Electroencefalografía (EEG)	Electrodos en el cuero cabelludo para medir la actividad eléctrica del cerebro [4].
Electroencefalografía (ECG)	Electrodos de contacto en la piel para medir la actividad eléctrica del corazón [4].
Electromiografía (EMG)	Electrodos de superficie o agujas en los músculos para medir la actividad eléctrica muscular [4].
Magnetoencefalografía (MEG)	Sensores magnéticos hechos con superconductores, dispuestos alrededor de la cabeza, que permiten registrar la actividad magnética del cerebro [4].
Presión arterial	Sensores de presión colocados en el brazo o muñeca para medir la presión arterial [5].
Flujo arterial	Sensor piezoeléctrico mide la velocidad de la sangre mediante el efecto Doppler [6].
Presión intracraneal	Catéteres intracraneales o sensores de fibra óptica para medir la presión dentro del cráneo [7].
Flujo pulmonar	Sensor de presión diferencial entre dos puntos que permite calcular el flujo de aire exhalado o inhalado a través de la boca [8].
Gasometría	Sensores químicos que miden los niveles de gases como oxígeno, dióxido de carbono y pH en la sangre [9].
Pulsioximetría	Sensores ópticos en la punta de los dedos o lóbulos de las orejas para medir la saturación de oxígeno en la sangre [10].
Glucosa en sangre	Biosensores químicos que utilizan enzimas para medir los niveles de glucosa en la sangre [11].
Temperatura corporal	Termómetros de contacto o infrarrojos para medir la temperatura del cuerpo [12].

Tabla 1 Señales biomédicas y el tipo de sensor que se emplea.

Para señales de variación temporal lenta, como la pulsioximetría o la temperatura corporal de forma general, se emplean valores únicos para la extracción de conclusiones. Sin embargo, también existen excepciones como la monitorización de la pulsioximetría en apnea. Por otro lado, señales como el EEG o el ECG, son señales temporales complejas,

es decir, que si bien se puede realizar un análisis visual preliminar, este es subjetivo y no permite extraer toda la información. En la mayoría de las aplicaciones esto hace que, además de la adquisición de la señal, sea necesario realizar un procesado para realzar la información relevante. Este procesado consiste principalmente en la eliminación de ruido y artefactos, que enmascaran la señal y en la aplicación de transformaciones (*e.g.*, al dominio de la frecuencia) que permiten extraer características difíciles de detectar con la inspección visual [2].

1.3. Actividad cerebral eléctrica

La actividad cerebral eléctrica registrada mediante dispositivos como el EEG es una suma de la actividad eléctrica de grandes poblaciones de neuronas y células gliales que operan en sincronía [4]. Las neuronas que más contribuyen a esta actividad son las piramidales, muy presentes y ordenadas en la corteza cerebral [4]. En sus dendritas, se producen cambios de voltaje medibles, denominados potenciales postsinápticos, los cuales se generan por movimientos de iones, como resultado de una sinapsis con otra neurona [4, 5]. La señal medida en el cuero cabelludo se comporta como una señal oscilatoria. Debido a esto, la señal se suele dividir en distintas bandas de frecuencia, denominadas ritmos [14]. La razón de este comportamiento oscilatorio se debe a que distintos grupos de neuronas actuando en sincronía generan interferencias constructivas que aumentan la amplitud del EEG. Sin embargo, señales asíncronas generan interferencias destructivas que derivan en un EEG de baja amplitud y mayor frecuencia [14].

La actividad eléctrica cerebral se puede registrar a tres niveles diferentes, en función de profundidad craneal a la que se realiza la medición. Los *local field potential* (LFP) permiten registrar corrientes iónicas que se propagan a través del medio extracelular. De esta manera, se pueden modelar grupos pequeños de neuronas y comprender mejor el funcionamiento de las mismas [13]. En esta señal los electrodos se ubican en el interior de la corteza, lo que permite captar la información muy cerca de las fuentes generadoras [13]. En la superficie de la corteza cerebral, se registra el electrocorticograma (ECoG). Esta técnica destaca por su notable resolución espacial y alta relación señal ruido (SNR). Para su adquisición se requiere de cirugía invasiva. Esto hace que el tipo de registro más empleado sea el EEG convencional, que emplea electrodos colocados sobre el cuero cabelludo [15]. Esta técnica implica menor riesgo, pero también reduce la resolución espacial.

La actividad cerebral que alcanza el cuero cabelludo se distribuye por el espacio como una señal continua. Para discretizarla de forma efectiva, se requiere una correcta cantidad y distribución de los electrodos. Un mayor número de electrodos mejora la resolución espacial y proporciona información más detallada. El EEG se aplica en varias áreas de la medicina y la investigación, y una de las aplicaciones más revolucionarias es la creación de interfaces cerebro-computadora (BCI, por sus siglas en inglés *Brain Computer Interface*).

1.4. Brain Computer Interface (BCI)

Los sistemas BCI son tecnologías diseñadas para establecer una comunicación directa entre el cerebro y dispositivos externos, sin necesidad de utilizar los músculos o el habla [4]. La actividad cerebral se adquiere y procesa en tiempo real. Esto permite extraer información y traducir esta actividad en comandos o estados mentales del sujeto para controlar distintas aplicaciones. Existen diferentes técnicas que permiten implementar sistemas BCI, principalmente se utiliza el EEG, sin embargo, también se emplea el MEG, la resonancia magnética funcional, la espectroscopía de infrarrojo cercano y el electrocorticograma entre otros [16]. Predomina el uso del EEG debido a su elevada resolución temporal, portabilidad y bajo coste.

El objetivo principal de estos sistemas es dotar de mayor independencia a personas con graves discapacidades motoras o enfermedades neurodegenerativas [4]. Para ello, los sistemas BCI se emplean como sistemas de comunicación, basados en la selección secuencial de letras en una matriz para componer palabras y frases, el control de prótesis y sillas de ruedas, o el tratamiento de enfermedades basado en *neurofeedback* [4]. Este último método consiste en proporcionar realimentación al paciente en tiempo real sobre determinados patrones de su actividad cerebral (*e.g.*, la potencia de su actividad cerebral eléctrica en una determinada banda de frecuencia en una región concreta de la corteza cerebral). De esta manera, el usuario puede aprender a modificar estos patrones, con el objetivo de inducir plasticidad cerebral, que provoque un impacto positivo en los síntomas de la patología del paciente.

A pesar de los beneficios del EEG con respecto a otros métodos de medida de la señal cerebral, el voltaje que se registra es muy bajo (en el orden de los microvoltios), lo cual hace que sea muy susceptible de ser enmascarado con otras fuentes de señal, que se denominan artefactos. [17]. Su influencia es aún mayor sobre los sistemas BCI, dado que requieren de técnicas de filtrado automáticas en tiempo real [18]. Sin embargo, la mayoría de los métodos empleados hasta el momento requieren de una señal adicional para registrar el ruido, o no son viables debido a la complejidad temporal del algoritmo. Existen multitud de señales generadoras de artefactos, sin embargo, los artefactos oculares son los más disruptivos y difíciles de evitar en los sistemas BCI [19]. Estos se generan por los parpadeos o movimientos oculares durante el registro. Y generan potenciales que afectan principalmente a los electrodos de la región frontal [4].

1.5. Hipótesis y objetivos

En la señal de EEG están presentes la actividad cerebral y otras señales de diversos orígenes que enmascaran la señal de interés. En este TFG se parte de la hipótesis de que exclusivamente con la señal de EEG se puede detectar temporalmente cuándo la señal generada por parpadeos o movimientos oculares está enmascarando la actividad cerebral.

En este trabajo se propone un algoritmo que permita detectar y clasificar los artefactos oculares de forma automática, en tiempo real, sin la necesidad de la señal adicional de electrooculografía (EOG) y sin calibración previa. Para lograrlo se ha propuesto un algoritmo basado en una arquitectura de *deep learning*. Este algoritmo se podría aplicar

tanto en técnicas de filtrado de artefactos oculares, como para simplificar la tarea manual que supone la detección de artefactos.

A continuación, se definen los objetivos específicos que permitirán alcanzar el objetivo principal:

1. Buscar información acerca de los artefactos oculares en la señal de EEG, técnicas de detección y de filtrado.

2. Buscar información acerca del *deep learning* y su aplicación en estudios similares.

3. Adquirir y preprocesar una base de datos con artefactos oculares etiquetados.

4. Entrenar un modelo a partir de una red de *deep learning* capaz de clasificar los artefactos.

5. Crear un algoritmo a partir del modelo clasificador capaz de detectar los artefactos en tiempo real.

6. Interpretar y comparar los resultados extraídos con los del estado del arte.

7. Determinar limitaciones del algoritmo, extraer conclusiones y proponer posibles líneas futuras.

1.6. Estructura del TFG

La memoria de este TFG está dividida en ocho capítulos: (i) introducción; (ii) electroencefalograma; (iii) redes neuronales; (iv) revisión del estado del arte; (v) materiales y métodos; (vi) resultados; (vii) discusión de resultados y (viii) conclusiones.

En el Capítulo 1: Introducción, se hace una breve introducción de la Ingeniería Biomédica y se describen algunas señales junto con los sensores empleados para su registro. También se abordan el origen y las técnicas de registro de la actividad cerebral eléctrica. Posteriormente se explican los sistemas BCI y se describe brevemente la importancia que tienen los artefactos oculares. Por último, se detallan la hipótesis y objetivos del trabajo.

En el Capítulo 2: Electroencefalograma, se aborda de manera exhaustiva la señal de EEG, ya introducida en el anterior capítulo. En primer lugar, se estudian sus bases neurofisiológicas y usos principales. Después, se comentan los distintos ritmos fisiológicos de la actividad eléctrica cerebral y se describe el comportamiento de la señal ante un estímulo, generando el denominado potencial relacionado con evento. Posteriormente, se detallan los montajes de EEG que siguen el estándar internacional. A continuación, se describen los artefactos oculares atendiendo a sus bases fisiológicas y propagación sobre el cuero cabelludo. También se describen las bases del electrooculograma y se comentan las principales técnicas de filtrado. Por último, se abordan los sistemas BCI, explicando las distintas señales de control y las etapas que se incluyen en estos sistemas.

En el Capítulo 3: Redes neuronales, se describen las redes neuronales, primero introduciendo el *deep learning* para después adentrarse en las redes neuronales convolucionales y finalmente, en las redes *inception*.

En el Capítulo 4: Revisión del estado del arte, se presenta una revisión del estado del arte. Esta se centra en algoritmos de detección de artefactos oculares y en algoritmos de clasificación de componentes independientes asociadas a estos artefactos.

En el Capítulo 5. Materiales y métodos, se describen los materiales y métodos empleados para alcanzar los resultados finales. Primero, se describe el preprocesado de las bases de datos. Después, se describe el entrenamiento de la red EEG-Inception para crear el modelo. Seguidamente, se explica cómo se integra el modelo en el algoritmo. Finalmente, se describe la validación cruzada llevada a cabo sobre el modelo y el algoritmo y se evalúa la complejidad temporal del algoritmo.

En el Capítulo 6. Resultados, se muestran los resultados obtenidos, tanto para el modelo como para el algoritmo.

En el Capítulo 7: Discusión de resultados, se lleva a cabo la discusión de resultados, se comprueba el cumplimiento de los objetivos y se comentan las limitaciones del algoritmo.

En el Capítulo 8: Conclusiones, se extraen conclusiones del trabajo realizado, se describen las contribuciones del algoritmo y se comentan algunas líneas futuras de investigación.

Capítulo 2: Electroencefalograma

2.1. Introducción

El EEG es un método ampliamente utilizado para el monitoreo de la actividad cerebral. A pesar de no ser la única señal empleada con este fin, destaca frente a otras técnicas debido a varias ventajas significativas [11, 12]. En primer lugar, el EEG permite adquirir la señal cerebral eléctrica de forma no invasiva. El método de adquisición se basa en la colocación de electrodos en la superficie del cuero cabelludo. Estos electrodos son capaces de detectar la actividad eléctrica generada por los potenciales neuronales, que se ha difundido por el volumen cerebral, han atravesado las distintas capas de las meninges y el cráneo, y finalmente llegan al cuero cabelludo muy atenuados. La no invasividad convierte al EEG en un método seguro y ampliamente aceptado en la práctica clínica y en la investigación [21]. En segundo lugar, el EEG es un método portátil y versátil. Existen equipos de EEG que son compactos y fácilmente transportables [4]. De esta manera, se facilita su uso en diversos entornos clínicos y de investigación. Además, se han desarrollado sistemas inalámbricos que permiten la adquisición de señales EEG sin la necesidad de cables, lo cual mejora la comodidad del paciente [22]. Otra ventaja importante del EEG es su alta resolución temporal. Esto implica que esta técnica puede registrar cambios en la actividad cerebral con una precisión temporal muy alta, del orden de milisegundos [4]. En consecuencia, el EEG permite analizar eventos cortos y capturar rápidas fluctuaciones en la actividad cerebral. La alta resolución temporal es especialmente útil en estudios que examinan respuestas cerebrales en tiempo real.

Por otro lado, la actividad generada en las neuronas se propaga a través de los diferentes tejidos, estos actúan como conductores imperfectos, que atenúan y dispersan la señal. Por consiguiente, la actividad cerebral captada por un electrodo no se debe únicamente a las neuronas ubicadas debajo del mismo [4]. La consecuencia de esta situación es una disminución en la resolución espacial. Esto representa un desafío en tareas específicas que implican el estudio de actividades altamente localizadas o en aquellas que se centran en las relaciones entre diferentes regiones (*e.g.*, conectómica). En tales casos, es común emplear técnicas de localización de fuentes o aplicar filtros espaciales para abordar este problema.

2.2. Usos principales de EEG

El electroencefalograma tiene varias aplicaciones bien establecidas en el diagnóstico y monitorización de pacientes, así como en áreas de investigación. Las principales son:

1. La evaluación del estado de conciencia en pacientes con afecciones como somnolencia, estupor o coma. Permite determinar la actividad cerebral mediante el análisis por bandas de frecuencia y la reactividad a estímulos [23].

2. El diagnóstico y monitorización de la epilepsia. Puede detectar anomalías en la actividad eléctrica cerebral que son características de las crisis epilépticas y de los

periodos intericticales. Además, el EEG es utilizado en la monitorización de los pacientes para ajustar la medicación y evaluar la eficacia de los tratamientos [24].

3. El estudio de los trastornos del sueño. Permite identificar patrones de actividad cerebral relacionados con el sueño y evaluar la calidad y la arquitectura de este [25].

4. Protocolo de confirmación la muerte cerebral en casos de lesiones cerebrales graves. La ausencia de actividad eléctrica cerebral característica del EEG puede indicar la pérdida irreversible de la función cerebral [26].

5. En el ámbito de la investigación, el EEG se utiliza en diversas áreas de la neurociencia, como la neurociencia computacional o cognitiva. También se utiliza en los sistemas BCI donde se aprovecha la actividad cerebral registrada por el EEG para controlar dispositivos externos o comunicarse directamente con ordenadores [4].

2.3. Ritmos fisiológicos en la actividad eléctrica cerebral

El EEG es una señal compleja que de forma general no se puede interpretar a simple vista [21]. Mediante el procesamiento de la señal se extraen datos cuantitativos o características que permiten analizar la señal de forma objetiva y extraer la información de interés [4]. Es común realizar una caracterización de la señal dividiéndola en las diferentes bandas de frecuencia o ritmos [27]:

- Delta: frecuencias menores a 4 Hz
- Theta: frecuencias entre 4 y 8 Hz
- Alpha: frecuencias entre 8 y 13 Hz
- Beta: frecuencias entre 13 y 30 Hz
- Gamma: frecuencias superiores a 30 Hz

En la Figura 1 se representa la señal de EEG descompuesta en cada una de estas bandas de frecuencia. En función de cómo se reparte la potencia de la señal en estas bandas, se puede inferir el estado en el que se encuentra el sujeto. Si el sujeto esta relajado, predominarán bajas frecuencias con grandes amplitudes [27]. Esto es una consecuencia directa de las interferencias constructivas generadas por la sincronía entre neuronas. Sin embargo, momentos de gran esfuerzo cognitivo hacen que disminuya de forma considerable esta sincronía, generando un EEG más irregular y de mayor frecuencia [27]. El EEG de una persona en estado de alerta y con los ojos abiertos presentará una gran cantidad de actividad α y β, sin embargo, una persona que está profundamente dormida presentará ondas δ de gran amplitud [27]. Los parámetros espectrales son utilizados de forma rutinaria en la monitorización intraoperatoria, estudios del sueño, diagnóstico de epilepsia, *etc*. [16, 17]. Además de realizarse análisis espectral, también se presta especial atención a características específicas que pueden ser indicativas de estados fisiológicos. Durante el sueño, por ejemplo, se observan formas de ondas características que aparecen

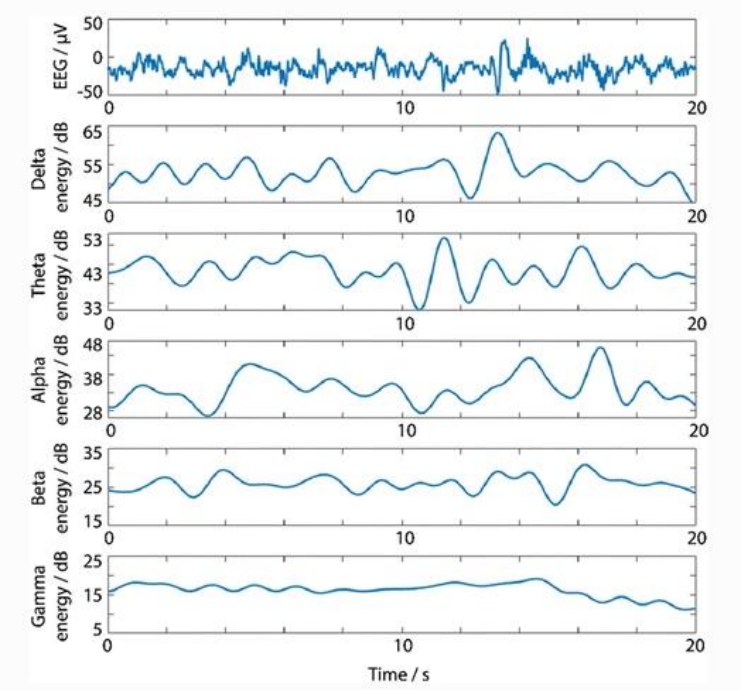

Figura 1 Señal recogida por el canal C4 y divida en las bandas de frecuencias [21].

en la fase N2 del sueño sin movimientos oculares rápidos, los denominados "complejos-k" [28]. Estos patrones presentan diversas morfologías y se definen como la combinación de una onda rápida (8-16 Hz) y una onda δ [28]. En el caso de la epilepsia, se busca identificar los picos interictales, que son actividad eléctrica anormal que ocurre entre los episodios de convulsiones. Estos picos interictales pueden manifestarse como breves descargas o cambios en el patrón de ondas cerebrales en el EEG [24].

2.4. Potencial relacionado con evento

Los registros de EEG no se realizan únicamente en condiciones de reposo, sino que también es común que el sujeto realice tareas cognitivas, (*e.g.*, en una sucesión de tonos regulares detectar la presencia de un tono inusual). En el cerebro se producen respuestas eléctricas a causa de los estímulos sensoriales. Estas respuestas se denominan potenciales relacionados con evento (ERP, por sus siglas en ingles *Event-related potential*) y permiten estudiar como el cerebro procesa la información sensorial de manera precisa. Los ERPs son muy utilizados debido a sus aplicaciones en el diagnóstico de TDAH [29], de esquizofrenia [21] y de la dependencia a algunas drogas [22]. Además, también se pueden emplear como señal de control en sistemas BCI basados en la detección de eventos [4]. Estos sistemas permiten controlar dispositivos a través de la selección de comandos.

En el análisis de los ERPs se estudian formas de onda o deflexiones características de la señal. Sin embargo, los ERPs son comúnmente de menor amplitud que el ruido y difíciles de distinguir en un solo evento [30]. Por lo tanto, es necesario aumentar la SNR para poder analizar la señal. Para ello, la técnica más empleada es el promediado, que se realiza empleando segmentos alineados temporalmente respecto al estímulo [30]. El promediado

es capaz de reducir el ruido estacionario, de tal forma que el ruido resultante es proporcional a la raíz de la media cuadrática del ruido original e inversamente proporcional a la raíz cuadrada del número de épocas [30]. La Figura 2 ilustra el proceso de promediado del ERP y en la esquina superior derecha se pueden observar las siguientes formas de onda N1, P2, N2 y P3. En la Tabla 2 se encuentran características principales de estas deflexiones, junto con la interpretación de las mismas.

	N1	P2	N2	P3
Tiempo tras estímulo	150 y 200 ms	150 y 250 ms	200 y 300 ms	300 y 500 ms
Tipo de estímulo	Visual o auditivo	Visual, auditivo o táctil	Visual, auditivo o táctil	Visual, auditivo o táctil
Región	Occipital o temporal	Frontal	Frontal	Parietal
Interpretación	Percepción temprana de características físicas básicas del estímulo	Procesos cognitivos como la percepción, la atención y la memoria	Atención selectiva y el procesamiento semántico	Memoria de trabajo, toma de decisiones, detección de estímulos inesperados y la discriminación entre relevantes e irrelevantes

Tabla 2 Características principales de las principales deflexiones que pueden aparecer durante ERPs [31].

Estas deflexiones no siempre están presentes, esto se debe a varios factores como el tipo de estímulo, el estado de atención, la edad o la condición clínica, que hacen que algunos componentes puedan ser más prominentes en ciertas personas y menos evidentes en otras.

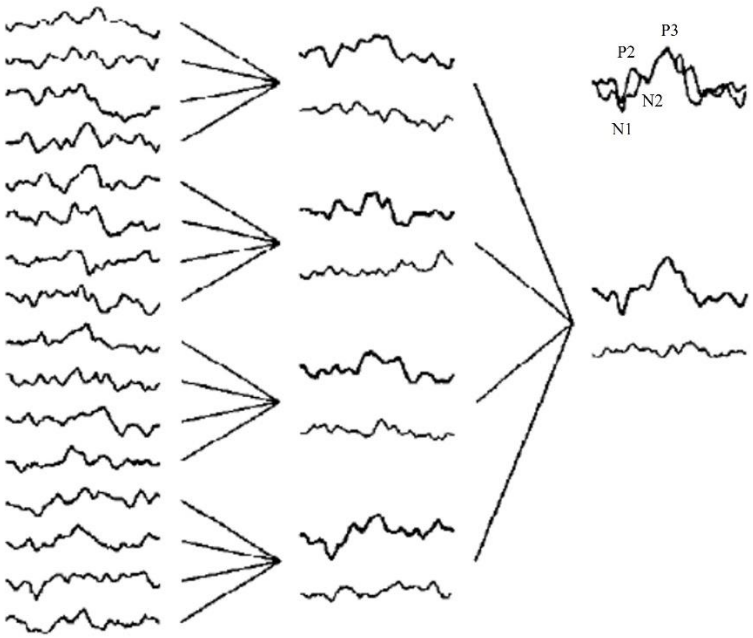

Figura 2 Promediado de potenciales evocados auditivos registrados como respuesta a la detección de un estímulo diana. La primera columna representa los *single-trials*. De forma ilustrativa en la segunda columna se muestra el promedio de grupos de 4 épocas junto con la estimación del ruido residual. En la tercera columna se muestra el promedio de los 16 *single-trials.* En la esquina superior derecha se muestran dos señales que corresponden al promedio de 8 *single-trials* cada una, de esta manera se ilustra la replicabilidad de la señal. En ambas señales se pueden apreciar las deflexiones N1, P2, N2 y P3 [30].

2.5. Montaje

El montaje del EEG hace referencia a las posiciones que ocupan los electrodos como conjunto [4]. En la práctica habitual se emplean los sistemas de posicionamiento de electrodos estandarizados, que permiten la comparación entre estudios y facilitan así la investigación científica. En la Figura 3 se muestran los sistemas internacionales 10–20, 10–10 y 10–5. El montaje más utilizado es el denominado 10-20 [32]. En este sistema los electrodos se encuentran separados una distancia igual al 20% de la distancia total entre nasión (el punto en el puente de la nariz) y el inion (el punto más prominente en la parte posterior del cráneo) para distribuir los electrodos en a lo largo de las líneas que conectan estos dos puntos en función de esta distancia relativa [5]. En el caso de los sistemas 10-10 y 10-5 se emplean el 10% y el 5% de la distancia, respectivamente.

De forma general en estos sistemas, se emplea una combinación de letra y número para identificar cada región. La letra indica el lóbulo cerebral en el que se encuentra ubicado (*e.g.*, F para frontal, T para temporal, *etc.*), mientras que el número determina si el electrodo está en el hemisferio derecho si es par, o en el hemisferio izquierdo si es impar y la distancia al centro (mayor valor, más alejado). Si se encuentra en el centro, el número se sustituye por la letra z.

El número de electrodos dependerá de la aplicación concreta del registro, el sistema 10-20 fue diseñado para ubicar números relativamente pequeños de electrodos, típicamente 21. Los otros sistemas, con mayor densidad de electrodos como el sistema 10-10 permite hasta 74 y el 10-5 hasta 128.

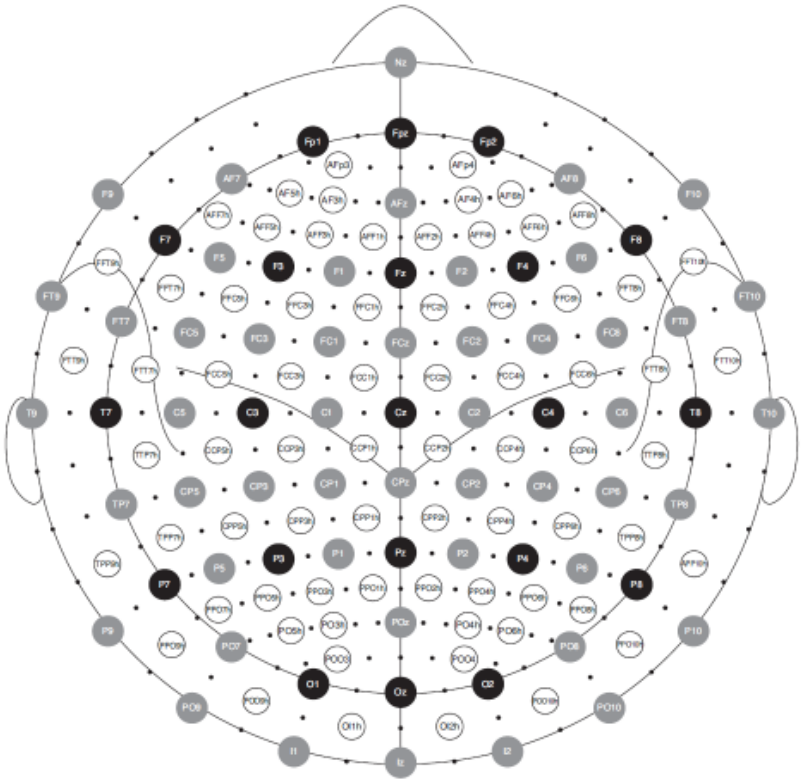

Figura 3 El sistema 10-20 está representado por los círculos negros. El 10-10 presenta los 21 electrodos del sistema 10-20 y 53 electrodos adicionales representados por círculos grises. Los electrodos adicionales del sistema 10-5 se están indicados por los círculos blancos [4].

2.6. Artefactos oculares en EEG

2.6.1. Introducción

El EEG es una señal de muy baja amplitud, lo que hace que sea muy susceptible a ser contaminada por diferentes fuentes de ruido y artefactos. Los artefactos son señales no deseadas que introducen cambios sobre la señal de interés [33]. Realizar un preprocesado adecuado es fundamental para obtener una señal limpia y confiable sobre la cual realizar un análisis más preciso. Los artefactos pueden ser generados por diversas causas, como factores fisiológicos, movimientos del sujeto o fuentes externas [4]. Entre los artefactos fisiológicos se incluyen los generados por el parpadeo, movimiento ocular, actividad cardíaca, actividad muscular y respiración. En la Figura 4 se representan los tres artefactos fisiológicos principales. Por otro lado, los artefactos externos pueden surgir de interferencias ambientales, como luces, ruido eléctrico y otros equipos electrónicos cercanos, así como del propio movimiento de los electrodos o los cables utilizados para realizar el registro del EEG [34]. Los artefactos oculares, que engloban a parpadeos y movimientos de ojos son de especial importancia en el análisis del EEG [34]. La actividad asociada a estos genera grandes interferencias, que afectan principalmente a la región frontal del EEG [4]. Además, los artefactos oculares se solapan frecuencialmente con la actividad cerebral y ocurren con asiduidad durante la adquisición, a pesar de que se le suele indicar al sujeto que no parpadee, ni mueva los ojos [34]. Esto supone que en los sistemas BCI los artefactos oculares sean los que más afectan a la señal y los más difíciles de evitar [19].

2.6.2. Dipolo corneo-retinal

Existen dos fuentes fisiológicas responsables de artefactos oculares, la principal es el dipolo corneo-retinal (CRD) y en menor medida los músculos extraoculares, principalmente el recto lateral [35]. Entre la retina y la córnea existe una diferencia de potencial [36]. Como consecuencia, el globo ocular es considerado un dipolo equivalente, denominado el dipolo corneo-retinal. Considerando este modelo, el polo positivo se encuentra en la córnea y presenta una diferencia de potencial respecto a la retina de entre

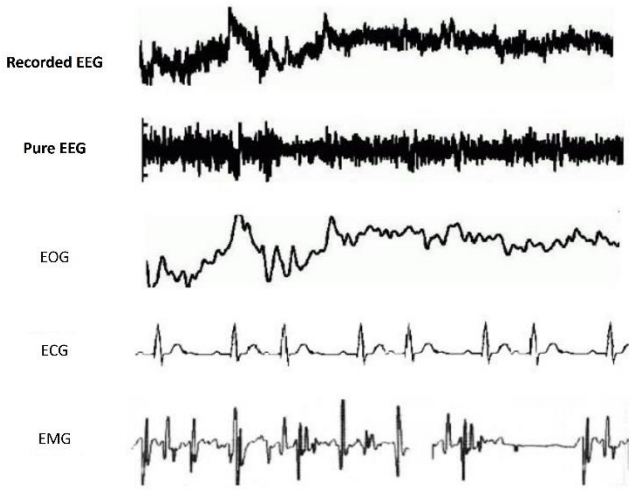

Figura 4 Artefactos fisiológicos presentes en EEG [34].

0.4 y 1 mV [36]. Durante los movimientos oculares el ojo rota y cambia la orientación del CRD, lo que genera una variación del campo eléctrico que puede ser captada por los electrodos [36]. La amplitud de esta variación se ve influida por la proximidad del electrodo al ojo, y el ángulo y la dirección en la cual se orienta el globo ocular. Todos los movimientos del ojo se pueden representar como combinaciones de movimientos en los ejes horizontal y vertical. Se ha observado que no hay diferencia significativa en la amplitud generada entre los movimientos ascendentes y descendentes en los movimientos verticales, ni entre los movimientos horizontales. Por otro lado, los movimientos horizontales generan amplitudes significativamente mayores en comparación con los verticales [35]. Estas diferencias en las amplitudes se deben a las características anatómicas de la cabeza y de los músculos extraoculares, principalmente del recto lateral [35]. Además, la contracción de los músculos extraoculares también contribuye a la generación de otro tipo de artefacto, los picos sacádicos. Este tipo de potenciales son generados por movimientos oculares rápidos o sacadas y se caracterizan por un aumento breve y transitorio en la potencia de la señal de EEG en el rango de frecuencia de 20 a 90 Hz [37]. Este rango está lejos del de movimientos más lentos, en los que su comportamiento en el dominio espectral se puede confundir con actividad lenta del EEG, como δ o θ [38].

El parpadeo es una función esencial de los ojos que ayuda a hidratar la superficie de la córnea y la conjuntiva, así como eliminar sustancias irritantes. Se pueden producir de forma voluntaria o como un acto reflejo, debido a un estímulo luminoso fuerte o para mantener la córnea húmeda. En un estado de relajación la frecuencia de parpadeo es de 15 a 20 veces por minuto, sin embargo, hay que tener en cuenta factores como fatiga o el nivel de atención [39]. Debido a que la conductividad del cráneo es baja, la piel es el medio por el que se propagan los potenciales oculares. Al deslizarse el parpado sobre la córnea, se genera una variación en el campo eléctrico generado por el dipolo corneo-retinal y se propaga la actividad ocular a lo largo del cuero cabelludo. Esta actividad puede llegar a ser detectada hasta en electrodos de regiones occipitales y resulta en un potencial de elevada amplitud (hasta 800 µV) y alta frecuencia [39].

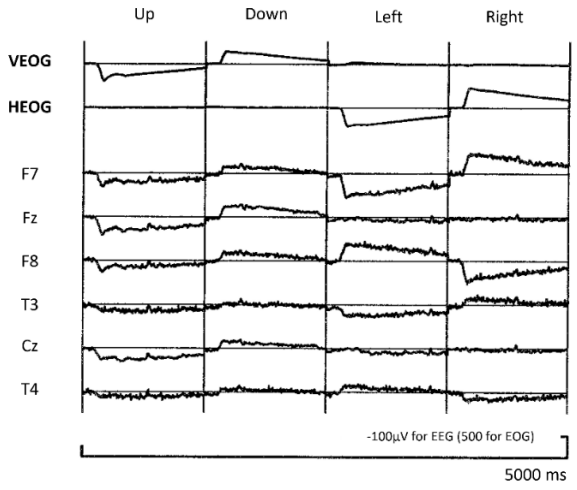

Figura 5 Electrodos de EOG y EEG ante movimientos de ojos verticales y horizontales [35].

Atendiendo a la propagación de los artefactos sobre el cuero cabelludo, en los movimientos oculares verticales y los parpadeos, el potencial ocular viaja a lo largo del eje sagital. Mientras que la actividad de los movimientos horizontales se propaga siguiendo el eje coronal, afectando en mayor medida a las regiones laterales [38]. De forma general, el gradiente del potencial es mayor en zonas próximas a los ojos, como la región frontal y menor en regiones posteriores, parietal, temporal y occipital. En la Figura 5 se aprecia de una forma clara como los movimientos laterales afectan más a los electrodos F7 y F8 colocados lateralmente a los ojos, mientras que el electrodo Fz ubicado en el centro de la región frontal, capta de forma más evidente el artefacto vertical.

En la actualidad existen métodos capaces de detectar parpadeos únicamente con la señal de EEG. De forma general, se emplean umbrales de voltaje que son capaces de detectar estos eventos debido a su gran amplitud. Este tipo de técnicas, sin embargo, no se aplican sobre los movimientos oculares. La detección de estos se ve dificultada por diversos factores de variabilidad intra e intersujeto que hace que no sean siempre tan evidentes en la señal de EEG [17]. Entre estos factores destacan la geometría de la cabeza, la conductancia de los tejidos y las interfases entre el electrodo y el cuero cabelludo. En consecuencia, el método más fiable para la detección de los artefactos oculares consiste en el corregistro de la actividad eléctrica ocular mediante el EOG.

2.6.3. Electrooculograma

El EOG registra la señal generada por dipolo corneo-retinal y las variaciones que se generan con los movimientos y parpadeos. Se colocan pequeños electrodos, similares a los de EEG, en regiones adyacentes a los ojos, como se puede ver en la Figura 6. El EOG considera que los movimientos de ambos ojos están conjugados y que sus correspondientes dipolos se mueven de forma sincronizada.

Los electrodos ubicados en arriba y debajo de los ojos permiten registrar los movimientos verticales y parpadeos, mientras que los ubicados en el a la izquierda y derecha registran los horizontales. Cada pareja genera señales independientes que se denominan vertical EOG (VEOG) y horizontal EOG (HEOG), respectivamente. Estas se calculan a partir de la diferencia de amplitud de los canales y aportan información adicional con respecto a los EOG por sí mismos [36].

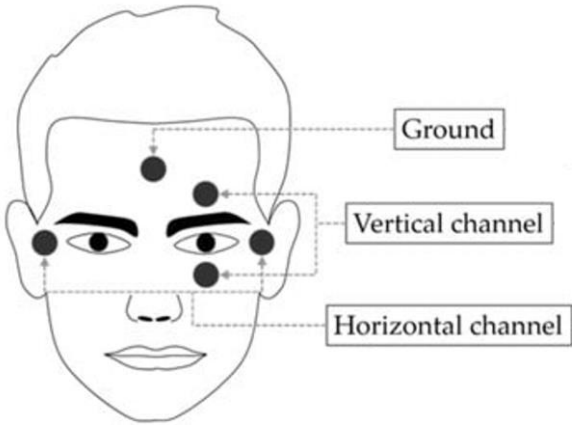

Figura 6 Posicionamiento de electrodos para un EOG convencional [40].

2.6.4. Filtrado de artefactos oculares

Hay múltiples formas de abordar los artefactos oculares. Si el procesado no requiere ser realizado en tiempo real, el método más comúnmente empleado es el análisis de componentes independientes (ICA, por sus siglas en ingles *Independent Component Analysis*). El uso de esta técnica se fundamenta en la hipótesis de que la señal EEG es la combinación lineal de información de origen neuronal y artefactos, por lo que se puede descomponer en componentes estadísticamente independientes y eliminar las componentes no cerebrales [36]. Sin embargo, si se dispone de un número limitado de electrodos, es probable que varias fuentes de actividad cerebral y no cerebral se mezclen y dificulten la separación. Además, esta técnica implica un alto costo computacional, y el proceso de selección de componentes se lleva a cabo de manera manual. Debido a estos motivos, ICA no se puede operar en tiempo real, lo que impide su aplicabilidad en BCI.

Otra metodología existente es la regresión, capaz de estimar la influencia de los artefactos oculares sobre el EEG y restarla, incluso en tiempo real. En este método se aplican filtros espaciales a través de señales captadas por canales de externos (EOG) para sustraer los artefactos estimados [34, 17]. La actividad de EEG corregida se puede obtener de la siguiente manera:

$$EEG_{cor} = EEG_{raw} - \gamma F(HEOG) - \delta F(VEOG) \qquad (1)$$

Los parámetros γ y δ hacen referencia a los coeficientes de transmisión de información entre las señales de EOG y EEG y buscan minimizar la influencia de los artefactos sobre la señal [34]. Para su estimación es necesario realizar un registro en el que el sujeto realice movimientos oculares y parpadeos. Existen dos limitaciones principales asociadas con la técnica de regresión. Por un lado, esta técnica ha sido criticada en algunos artículos debido a la eliminación de la actividad cerebral incluida erróneamente como artefacto [36]. Sin embargo, con otros métodos como ICA, tampoco se puede garantizar que no se vea reducida la actividad cerebral tras la eliminación de componentes, sobre todo en casos donde se emplee un número reducido de electrodos. Por otro lado, en la regresión existe la necesidad de registrar la señal de EOG, esto implica la colocación de electrodos adicionales y puede resultar incómodo para el sujeto durante la adquisición de datos, incluso generar movimientos de músculos faciales.

Otra técnica comúnmente aplicada es el filtro adaptativo, que posee la capacidad de estimar y suprimir en tiempo real los artefactos oculares [4]. Este tipo de filtros son capaces de ajustar sus parámetros de manera dinámica en función de las características particulares del EEG. En la Figura 7, se presenta la estructura típica de esta categoría de filtros.

Al igual que en el contexto de análisis de regresión, además de la señal principal de EEG, se monitorea una señal secundaria o de referencia, el EOG. Esta señal de referencia se somete a un proceso de filtrado digital con el propósito de obtener una estimación de la componente que corresponde al artefacto ocular [34]. Posteriormente, esta estimación se sustrae de la señal de EEG original, lo que resulta en la obtención de una señal resultante y un error asociado.

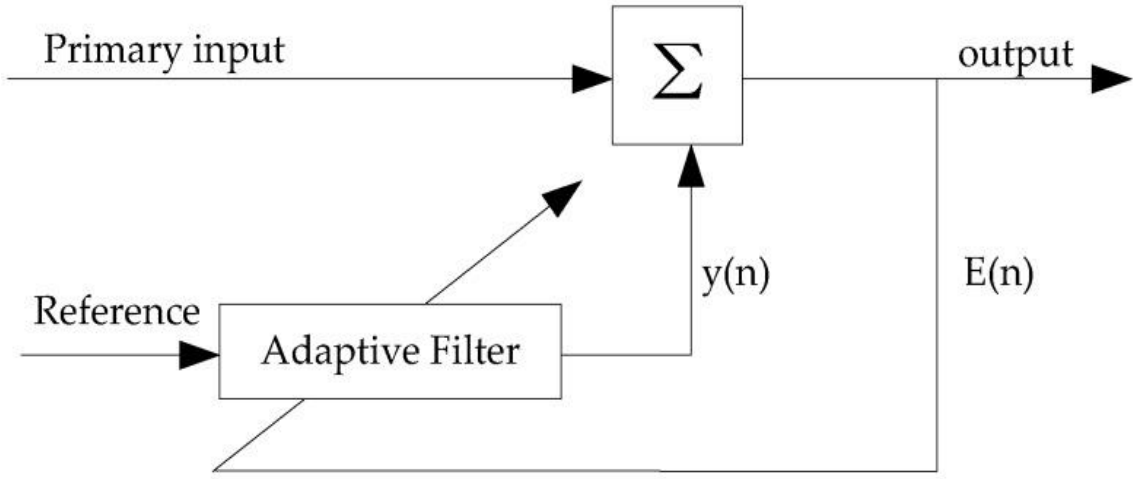

Figura 7 Esquema de filtro adaptativo [34].

El proceso adaptativo se fundamenta en la minimización del error cuadrático medio, a través de algoritmos de optimización, el más frecuentemente empleado es el *least mean square* (LMS) [41]. Este método asume que la señal de EOG únicamente está correlacionada con el artefacto ocular y no guarda relación con la actividad cerebral de interés. De esta manera, el filtro adaptativo itera para generar una salida que se asemeje considerablemente al ruido presente en la señal principal de EEG, que se sustrae dando como resultado una señal limpia. Este método de filtrado también ha demostrado ser efectivo sobre otros artefactos como la interferencia de red o el ECG [41].

Este enfoque, a pesar de reducir la eliminación de actividad cerebral en comparación con el método de regresión en algunos escenarios, aún demanda la utilización simultánea del EOG durante el proceso de registro.

Con la intención de superar estas limitaciones en 1994, Berg y Scherg [42] crearon un algoritmo de substracción de subespacios. Su método y los basados en el presentan una fase de calibración previa al registro. En ella se estiman de forma precisa las fuentes de artefactos a través de los canales de EEG y EOG. Posteriormente, durante el registro, se elimina esta contribución estimada de los canales de EEG sin la necesidad de utilizar señales de EOG. Estos algoritmos de substracción de subespacios son aptos para aplicaciones en tiempo real, gracias a la asunción de estacionariedad en los artefactos oculares y al bajo costo computacional una vez entrenado el algoritmo. En la fase calibración previa al registro, se provocan los artefactos de forma controlada. Con los datos recopilados durante la calibración, se calculan unos parámetros que conforman una matriz denominada de *unmixing* que permite transformar el EEG a los subespacios de fuentes. Una vez completada la calibración, la substracción de los subespacios asociados a los artefactos se puede reducir a una multiplicación de matrices.

El tiempo de calibración es limitado, debido a que durante los experimentos se trata de maximizar la duración de la tarea que realiza el sujeto. En el artículo de Kobler *et al*. [36] se desarrolló un paradigma de 5 min que permite registrar movimientos de ojos y parpadeos de una forma controlada y demostró que su algoritmo de substracción de subespacios podía atenuar el ruido hasta una hora después de la calibración. Este enfoque

es efectivo y elimina la necesidad de registros de EOG durante el registro principal. Sin embargo, aún se requiere el uso de EOG en la etapa de calibración para detectar temporalmente los artefactos.

En el presente TFG se propone como solución un algoritmo de detección automática de artefactos oculares. Esto permitiría que el algoritmo de substracción de subespacios fuera mucho más accesible y agilizaría el proceso de calibración, reduciendo el tiempo requerido. Además, la aplicabilidad en tiempo real del algoritmo de detección permitiría llevar a cabo una substracción de subespacios limitada a aquellos fragmentos de señal que presenten artefactos, evitando la perdida de información asociada a todas las técnicas de filtrado.

2.7. BCI

Los sistemas BCI permiten que el cerebro humano se comunique directamente con dispositivos electrónicos. Para llevar a cabo esta interacción se suele emplear la señal de EEG. Debido a que es una señal compleja y no estacionaria, los sistemas BCI empelan patrones concretos de la actividad cerebral, que sirven para identificar las intenciones de los usuarios y manejar así el dispositivo [4]. Estos patrones se denominan señales de control. Para detectarlos los sistemas BCI llevan a cabo una serie de etapas que permiten adquirir la señal de EEG, procesarla en búsqueda de las señales de control y por último llevar a cabo la aplicación final.

2.7.1. Señales de control

Existen dos tipos de señales de control: señales exógenas y señales endógenas. Las señales exógenas resultan de la respuesta a estímulos externos específicos que se le presentan al usuario [16]. Este tipo de señales requieren poco entrenamiento y presentan una alta tasa de transferencia (información transmitida por unidad de tiempo). Por otro lado, las señales endógenas demandan de atención permanente, lo cual puede ocasionar fatiga en algunos sujetos. Las señales endógenas se generan por la actividad cerebral controlada del usuario y no están vinculadas a estímulos. Esto implica que es el sujeto el que genera la señal de control a voluntad. Sin embargo, requieren un largo tiempo de entrenamiento, de hecho, algunos sujetos no son capaces de generar las señales [16]. Además, requieren de un número mayor de electrodos para obtener buenos resultados y presentan tasas de transferencia menores.

Las principales señales de control exógenas son [16]:

- ERP P300: El potencial P300 es una respuesta cerebral generada tras detectar un estímulo inesperado o relevante. Los sistemas BCI aprovechan este potencial a través del paradigma *odd-ball*, en el que se presentan multitud de estímulos regulares y predecibles, que se intercalan con los estímulos inusuales.
- Potenciales evocados visuales de estado estable (SSVEP): Son respuestas cerebrales generadas por estímulos visuales que se repiten a una frecuencia específica. Estos estímulos se generan mediante varias fuentes luminosas parpadeando a distintas frecuencias. Existe una sincronización entre la actividad

cerebral y la frecuencia de la fuente en la que se fija el sujeto. Lo que hace posible conocer en función de la respuesta cerebral hacia donde mira.

Las tasas de transferencia son elevadas, de entre 60 y 100 en los SSVEP y de entre 20 y 25 en los P300 [16].

Las principales señales de control endógenas son [16]:

- Ritmos sensoriomotores: En BCI se utilizan las bandas de frecuencia μ (8 – 13 Hz) y β (13 – 30 Hz), relacionadas con la actividad de las regiones sensoriales y motoras. μ ocupa el mismo rango de frecuencias que α, sin embargo, los ritmos α se asocian a la región occipital mientras que los ritmos μ se producen en la región somatosensorial o motora de la corteza. Para generar estos ritmos de una forma controlada se emplea *Motor imagery* (MI).
  En esta estrategia, la intención motora del sujeto se emplea como señal de control. De forma habitual, al imaginar el movimiento de la mano izquierda se pueden lograr disminuciones en las frecuencias μ y β en áreas específicas de la corteza motora.

- Potenciales corticales lentos: Son cambios de voltaje que se generan durante tiempos más prolongados y están asociados al movimiento y a procesos cognitivos y de atención. Es un tipo de señal endógena, por lo que deberá ser el sujeto el que aprenda a controlarla.

En este caso las tasas de transferencia son menores, de entre 3 y 35 bits/min en los ritmos sensoriomotores y de entre 5 y 15 para los potenciales corticales lentos [16].

2.7.2. Etapas de los sistemas BCI

Los sistemas BCI constan de cinco etapas: la adquisición de la señal cerebral, el preprocesado, el procesado a su vez dividido en: extracción de características y traducción de características en comandos y, por último, aplicación final. Estas etapas se llevan a cabo de forma secuencial, tal y como muestra la Figura 8.

1. Adquisición: se registra la señal de EEG y se realiza su acondicionamiento para el procesado posterior. La señal de EEG tiene amplitudes muy pequeñas, que varían entre 10 y 100 μV, por lo que es necesario amplificarla para mejorar su calidad y facilitar el procesado [21]. Después de la amplificación, la señal analógica de EEG se somete a una conversión analógica-digital (ADC). Esta etapa implica dos procesos principales: el muestreo y la cuantización. El muestreo consiste en tomar muestras de la señal de forma periódica en el tiempo. Estas muestras se toman a una frecuencia determinada, conocida como frecuencia de muestreo, que debe ser lo suficientemente alta para capturar adecuadamente la información contenida en la señal de EEG [2]. Para prevenir el *aliasing*, las señales analógicas suelen ser filtradas con un filtro paso bajo para asegurar que no existan componentes de frecuencias superiores al límite de Nyquist [2]. Una vez que se han tomado las muestras de la señal, se realiza la cuantización, que

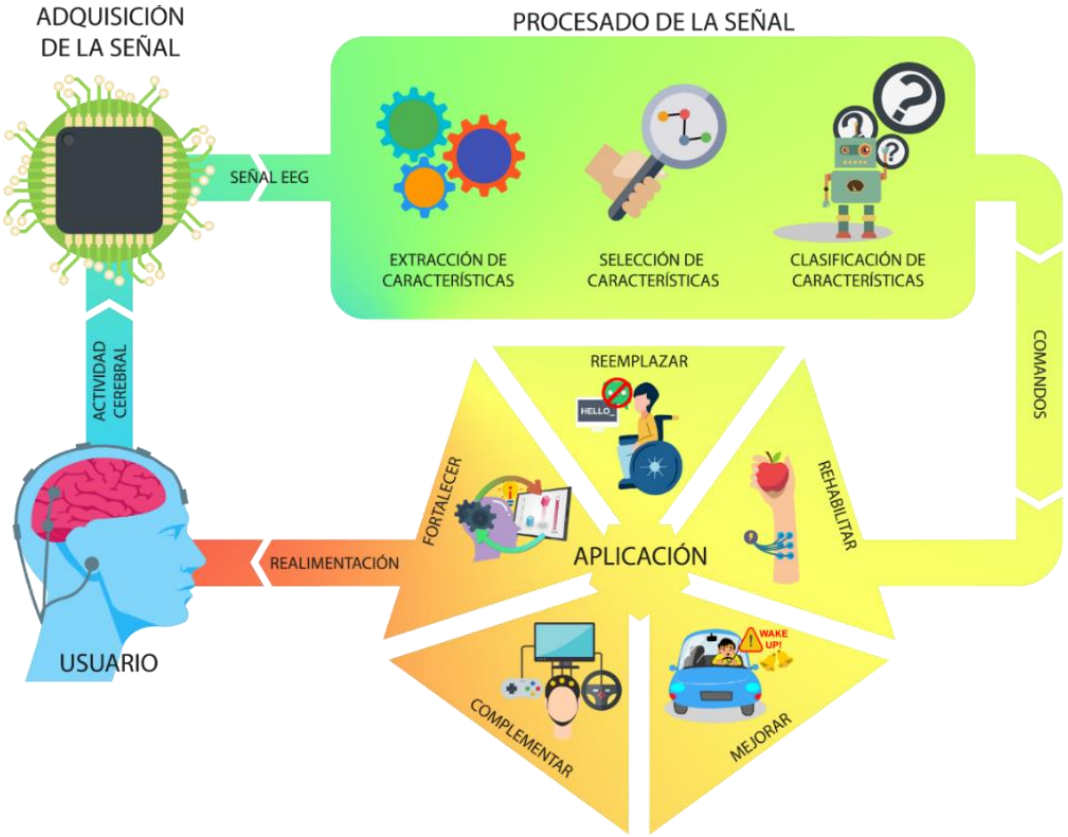

Figura 8 Esquema de las diferentes etapas de un sistema BCI**.**

implica asignar un valor digital a cada muestra analógica. La cuantización se basa en discretizar la amplitud de la señal en un conjunto finito de niveles discretos [2]. En la actualidad, muchos ADC utilizados en sistemas BCI son capaces de realizar tanto la amplificación como la conversión analógica-digital en una sola etapa [4].

2. El preprocesado trata de mejorar la señal a partir de eliminar el ruido y la información irrelevante que contenga la señal o realzando las características espaciales, temporales o espectrales de la señal [4]. Las técnicas en las que se basa el preprocesado son:
   - Filtrado en frecuencia: El filtrado que requiere la señal varía en función de la aplicación final. Es importante eliminar frecuencias que se encuentren fuera del rango de interés. Para una extracción de características general, es común es aplicar un filtro paso banda entre 0.5 y 40 Hz, y si la aplicación lo permite, seleccionar un rango aún menor (*e.g.*, 8-30 Hz aislando la actividad μ y β) [4].
   - *Decimation* o *downsampling*: Reducción de la señal al mínimo número de muestras efectivo, con el cual se consigue un mejor procesado y almacenamiento [4]. Al igual que ocurría durante la conversión analógico-digital, para prevenir *aliasing*, es importante utilizar previamente un filtro paso bajo, con una frecuencia de corte de la mitad de la frecuencia de muestreo final.
   - Normalización: Operación que transforma la señal para que tenga un valor medio centrado en 0 y desviación estándar de 1. Esto permite eliminar las

diferencias de amplitudes entre distintas señales y dar mayor importancia a la morfología de la onda.

- Filtrado espacial: Conjunto de operaciones que combinan y ajustan las contribuciones de distintos electrodos para aumentar la sensibilidad a fuentes concretas del cerebro, mejora su localización y reduce el ruido. Los filtros más comunes de este tipo son el *common-average reference* (CAR) y los filtros laplacianos [4].
- Detección y eliminación de interferencias ambientales y de artefactos biológicos: En la Figura 4 se muestra como algunos de estos artefactos afectan a la señal de EEG. En BCI, una limitación importante es que se trabaja en tiempo real, lo que impide hacer una selección estricta de épocas libres de artefactos, como se hace en el análisis off-line. En la actualidad, para eliminar los artefactos oculares se suelen emplear los filtros adaptativos, filtros espaciales y filtros paso alto [4]. Esto últimos se emplean dado que la mayor parte de la potencia de los movimientos y los parpadeos se ubica en las frecuencias bajas [4]. Sin embargo, la realización de un filtrado sistemático de estas bandas de baja frecuencia implica la pérdida permanente de información neural.

  En el caso de la actividad muscular, existe la dificultad añadida de que su rango de frecuencias varía entre 15 y 30 Hz, lo que coincide con la banda de frecuencia β [4]. Además, la señal muscular no se puede adquirir de forma aislada mediante EMG, como en el caso de los artefactos oculares mediante EOG. Este artefacto presenta una localización variable, que depende de la fuente muscular que lo genere. A menudo estos artefactos presentan patrones rítmicos, como los representados en la Figura 4.

  La interferencia de la red eléctrica es una fuente común de artefactos en muchos dispositivos de señal eléctrica [2]. Generalmente, funciona a una frecuencia de 50 o 60 Hz dependiendo de la región, en el caso de Europa es a 50 Hz. Esto ocurre debido a la radiación electromagnética que se propaga desde los cables y equipos eléctricos cercanos.

3. Una vez la señal ha sido acondicionada y se han eliminado los artefactos, la siguiente etapa es la extracción de características. Se entiende por características a la información que permita diferenciar las intenciones del usuario [4]. Las técnicas utilizadas para la extracción de características en un sistema BCI pueden clasificarse en dos grupos principales, dependiendo de la forma en que se procesa la señal. En el primer grupo, se encuentran los métodos que trabajan en el dominio temporal de la señal. Estas técnicas se caracterizan por tener un bajo coste computacional [4]. Un ejemplo de este tipo de método es el promediado sincronizado, que permite reducir el ruido estacionario y mejora la capacidad de detección de potenciales evocados. En el segundo grupo, se encuentran los métodos que trabajan en el dominio de la frecuencia. Ejemplos de técnicas en este grupo incluyen el cálculo de bandas de frecuencia específicas y el análisis espectral a través de la transformada rápida de Fourier.

4. La etapa de traducción consiste en el procesamiento del vector de características mediante algoritmos de clasificación. Este proceso tiene como objetivo la selección de comandos que el dispositivo es capaz de interpretar. Existen multitud de modelos que se pueden utilizar como algoritmo de traducción, la elección de mismo dependerá de la aplicación concreta del sistema BCI [4]. Los modelos se pueden separar en dos grupos principales, discriminantes y de regresión. Los modelos discriminantes se permiten "traducir" las características a categorías o clases concretas, como pueden ser si/no o arriba/abajo. Para esto se debe encontrar una frontera de decisión que separe las clases. Por otro lado, los modelos de regresión se utilizan para "traducir" las características a una variable continua, como podría ser el movimiento de un cursor o de una prótesis. Buscan establecer relaciones funcionales entre las características y la respuesta. Algunos modelos comúnmente utilizados son *Linear least-squares discriminant function,* clasificadores bayesianos, máquinas de vector soporte (SVP) y modelos no lineales (*e.g.*, Redes neuronales artificiales o análisis discriminante cuadrático) [4].

5. La última etapa es la de aplicación, recibe como entrada la señal de traducción de características y las convierte en acciones. La aplicación principal es la tecnología de asistencia (AT), que incluye los dispositivos diseñados para ayudar a personas con discapacidades funcionales. Las funcionalidades a las que más contribuyen son la comunicación, la movilidad y las actividades del día a día [43]. A continuación, se comentan algunas de ellas.

   Los sistemas BCI basados en señales de control exógenas (eg. P300 o SSVEP) permiten implementar aplicaciones de selección de comandos. Estos pueden usarse para el desarrollar matrices de selección de letras para comunicación (letterboard-based) o en el control de un teléfono inteligente. Por otro lado, sistemas BCI basados en señales de control endógenas (MI) pueden ser empleados en la selección de direcciones, aplicable a sillas de ruedas eléctricas [4]. Con entrenamiento los ritmos sesoriomotores permiten el movimiento en dos dimensiones.

   Fuera de la tecnología de asistencia se encuentran los sistemas BCI empleados para la rehabilitación neuronal y para el entrenamiento cognitivo. La rehabilitación neuronal está enfocada en pacientes con daño cerebral o lesiones en el sistema nervioso. MI ha demostrado que puede facilitar la recuperación de funciones motoras afectadas mediante la inducción de la plasticidad cerebral [44]. Para lograr que la terapia funcione es necesario dar al paciente realimentación o *neurofeedback* de su desempeño en tiempo real. De igual manera, el entrenamiento cognitivo también emplea *neurofeedback* y sus aplicaciones principales se centran en la normalización de los patrones de actividad cerebral afectados por un desorden de origen neuronal [45]. De esta manera, se espera influir positivamente en los síntomas de dichos desórdenes, como por ejemplo la atención en el TDAH [46].

Capítulo 3: Redes neuronales

3.1. Deep learning

El *deep learning* (DL) es una rama de *machine learning* (ML) basada en las redes neuronales artificiales (ANN) profundas (Figura 9) [47]. El ML es un campo de la inteligencia artificial que se enfoca en el desarrollo de modelos capaces de "aprender" y mejorar su rendimiento a partir de experiencias previas. Al aplicar un modelo de ML tradicional se debe realizar una extracción manual de las características relevantes, para que el modelo, a partir de ellas, identifique patrones y tome decisiones por sí mismo [48]. Las ANN son modelos matemáticos inspirados en el funcionamiento del cerebro humano [49]. Están compuestas por un conjunto de unidades fundamentales denominadas neuronas, que están unidas entre sí y organizadas en capas [49]. Las capas pueden ser de entrada, ocultas o de salida en función de la posición que ocupen en la red. Si todas las todas las neuronas de una capa están conectadas a todas las neuronas de la capa siguiente, se la conoce como capa *fully conected* o *dense*.

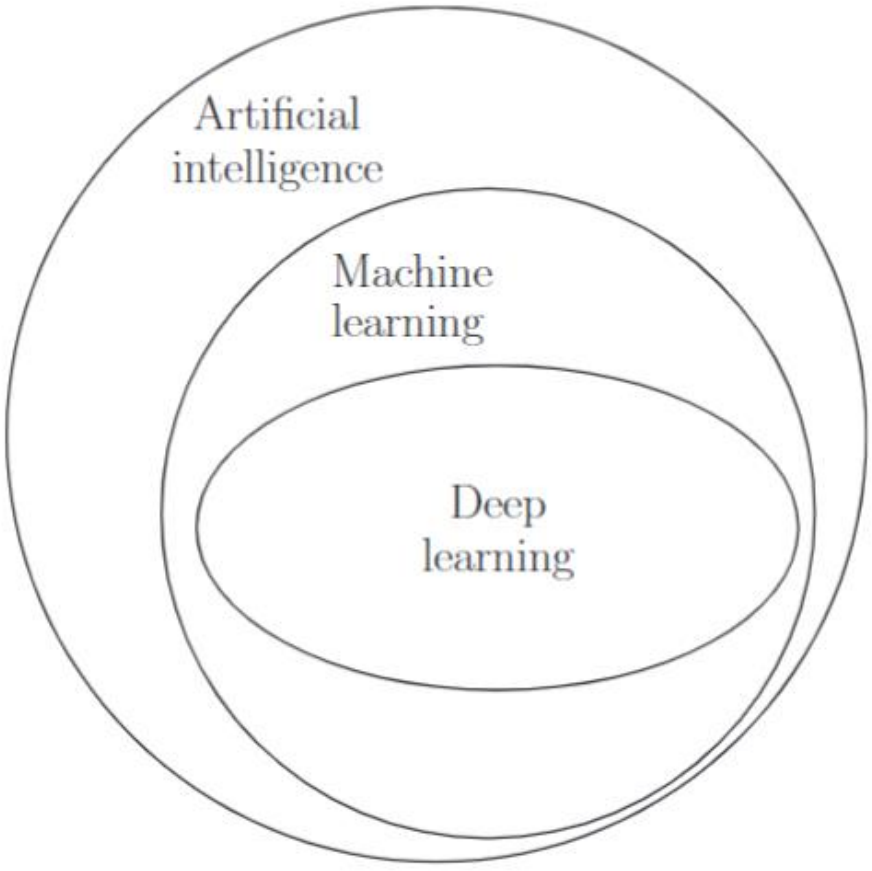

Figura 9 La relación entre la inteligencia artificial, el *machine learning* y el *Deep learning* [47].

Las entradas de las neuronas son multiplicadas por pesos y sumadas, para posteriormente aplicar una función de activación que da lugar a la salida de la neurona [49]. Durante el proceso de entrenamiento se ajustan los pesos de las conexiones. Esto se logra mediante la minimización de una función de pérdida, que se obtiene de comparar las salidas con los valores deseados [49]. Para calcular las contribuciones de los pesos al error es necesario emplear el algoritmo de *backpropagation* o retropropagación [48].

Una propiedad muy importante de estas redes es la capacidad de aprender relaciones no lineales entre los datos de entrada y de salida. Esto no ocurre en los algoritmos de ML convencionales como pueden ser la regresión logística o las máquinas de vector soporte. Para lograr la no linealidad se emplean funciones de activación no lineales que permiten alcanzar representaciones complejas de las características de entrada.

Las redes de DL son un tipo de ANN que a diferencia del resto de modelos de ML son capaces de extraer automáticamente las características relevantes a partir de los datos crudos. Esto les permite capturar patrones y características más complejas y abstractas, que suponen una representación más sofisticada de los datos de entrada de la que se consigue con los métodos tradicionales de extracción de características. Todo esto conduce a un mejor rendimiento en las tareas de clasificación [47]. En consecuencia, en los últimos años la popularidad de los modelos de DL ha crecido significativamente en diversos campos, entre ellos el análisis de EEG [43]. Han demostrado un gran desempeño en las tareas de detección de eventos cognitivos [50], detección de anomalías en el EEG [51] y control de los sistemas BCI [43].

Sin embargo, es importante señalar que el éxito del DL en EEG depende en gran medida de la cantidad y calidad de los datos disponibles, en algunos casos, los enfoques de ML más simples y basados en características específicas del dominio aún pueden ser efectivos.

Dentro de DL existen distintos tipos de redes neuronales, entre ellas destacan las redes neuronales convolucionales (CNN), las redes neuronales recurrentes (RNN) y las redes neuronales generativas adversiales (GAN) [47]. Todas ellas se han aplicado con éxito en aplicaciones concretas de EEG, dando lugar a multitud de arquitecturas diferentes. En el siguiente apartado se comentarán en profundidad las primeras.

3.2. Redes neuronales convolucionales

Las CNN fueron propuestas e introducidas por Yann LeCun en 1989 [47]. LeCun desarrolló las CNN como una arquitectura especializada para el reconocimiento de imágenes. Este tipo de redes aplican convoluciones, operaciones matemáticas capaces de extraer características relevantes de las imágenes. Estas convoluciones consisten en la multiplicación secuencial de la imagen con un *kernel* o matriz de pesos. Estos pesos se acturalizan de igual manera que los de las neuronas. Cada uno de estos *kernels* genera un *feature map* para una característica concreta [47]. En las capas iniciales, las características extraídas por las convoluciones serán bordes y texturas, que servirán a las capas más profundas para extraer características más complejas, como objetos enteros. La operación de convolución va seguida de una función de activación no lineal, comúnmente la *rectified linear unit* (ReLU), que convierte los valores negativos en ceros. Para un mejor

desempeño de la red es muy común aplicar seguidamente una capa de *pooling* que reduce la dimensión espacial para obtener características más robustas [47]. Dado que la operación de convolución, función de activación y capa de *pooling* se aplican de forma secuencial dentro de la red, se suele denominar al conjunto, capa de convolución (Figura 10).

Las arquitecturas típicas de CNN se basan en bloques de convolución para extraer características progresivamente más abstractas y reducir la dimensionalidad de los datos [47]. En las capas finales de estas redes, se suelen utilizar ANN *fully conected* que aprovechan las características previamente extraídas.

Dentro del campo de las redes convolucionales, existen diversas arquitecturas que han demostrado ser eficaces en diferentes aplicaciones. Las CNN fueron inicialmente diseñadas para el análisis de imágenes, lo que hace que las arquitecturas más importantes se centren en *visión por computadora*. LeCun en 1998 desarrollo LeNet-5, la primera arquitectura en implementar las convoluciones [52] (Figura 11). Su éxito incentivó el interés por el aprendizaje profundo, lo que desembocó en la aparición de nuevas arquitecturas como AlexNet [53] y VGG [20]. Todas estas redes han tenido una gran importancia en el desarrollo de nuevas arquitecturas más complejas y poderosas, como las redes *Inception* y las redes residuales (ResNet). Estas últimas arquitecturas han continuado empujando los límites de la visión por computadora y han demostrado ser altamente efectivas en diversas tareas de reconocimiento visual [55]. Sin embargo, su uso se ha extendido a diversos ámbitos debido a su alto rendimiento, permitiendo el análisis de otros tipos de datos, incluyendo la señal de EEG.

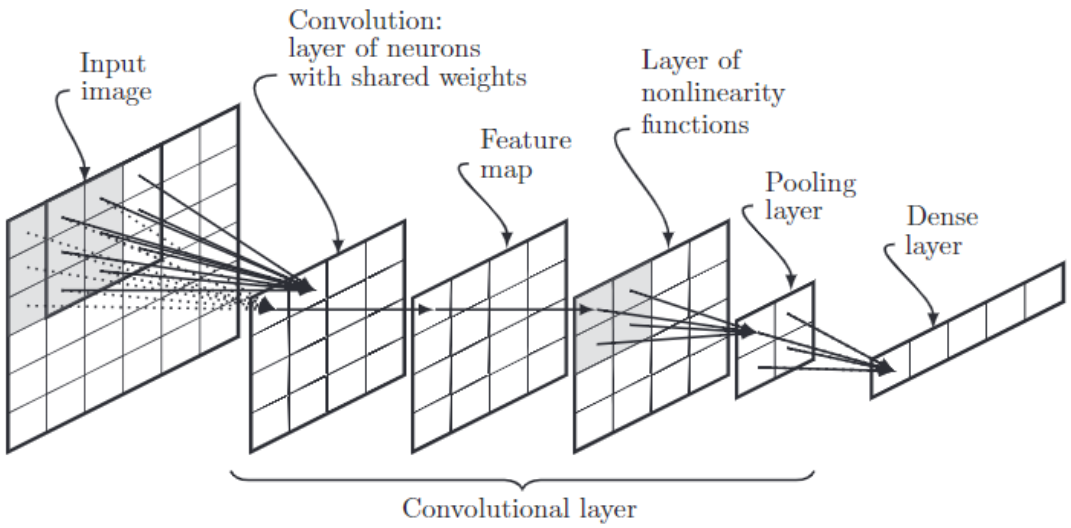

Figura 10 Simboliza las diferentes etapas de un bloque o capa convolucional [47].

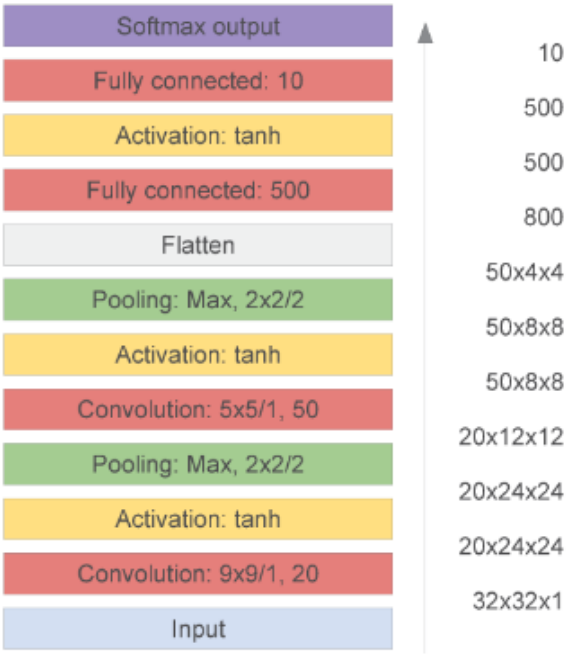

Figura 11 Arquitectura de la red LeNet-5 [55].

3.3. Arquitectura Inception

Esta arquitectura fue diseñada para abordar el desafío de elegir entre diferentes tamaños de *kernels* en las capas convolucionales de una red. La idea principal detrás de las redes *inception* es realizar convoluciones de forma paralela con diferentes tamaños de filtros, y luego combinar las salidas para obtener características tanto locales como de más alto nivel. Esto permite que la red elija automáticamente qué características y detalles capturar en cada capa [17].

La Figura 12 presentada a continuación muestra un módulo *inception* con 4 tipos de convoluciones diferentes, 1 x 1, 3 x 3, 5 x 5 y una de max pooling 3 x 3.

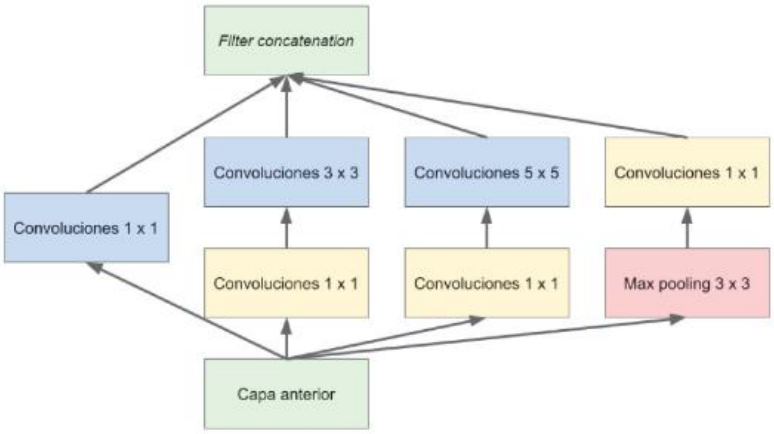

Figura 12 Ejemplo de módulo *inception* [55].

Los módulos *inception* son muy efectivos en casos donde la base de datos de entrenamiento es pequeña o la capacidad computacional es limitada [56]. Superan en estos casos a redes mucho más profundas y complejas. Esto se debe a que estas redes tienden más a sobreentrenar limitando la capacidad de generalizar y también acaparan los recursos computacionales. Dado que la señal de EEG es altamente dinámica y presenta patrones temporales transitorios, resulta útil para la realización de un análisis multiescala como el que permiten las redes *inception* [43]. A pesar de las ventajas potenciales de las arquitecturas *inception* para el procesamiento de EEG, hasta ahora han sido pocos los estudios que hayan explorado esta aplicación, habiéndose centrado la mayoría en arquitecturas más tradicionales [43].

La red EEG-Inception fue diseñada con el fin de mejorar la detección de ERPs. Los resultados obtenidos por la red superan de forma significativa a los de otras redes diseñadas para la detección de ERPs en señal de EEG [43]. La red es capaz de capturar características que suelen ser ignoradas debido a que se encuentran fuera de las bandas de frecuencia de mayor potencia en los ERPs. Para conseguir esto la red emplea los módulos *inception*, que le permiten procesar patrones de diferentes escalas temporales. EEG-Inception combina conceptos de visión por computadora, además de los bloques *inception*, también emplea convoluciones *depthwise*. Gracias a las convoluciones *depthwise* se pueden realizar convoluciones separadas para cada canal, es decir, cada filtro opera únicamente en un canal en lugar de en todos los canales a la vez. De esta manera, la extracción de características se separa dos etapas: una convolución temporal (convolución 2D) y una convolución espacial (convolución *depthwise* 2D). La finalidad principal de este tipo de convoluciones es la reducción del número de parámetros, sin embargo, en el contexto de EEG, también actúan como filtros espaciales para cada patrón temporal extraído de la capa anterior.

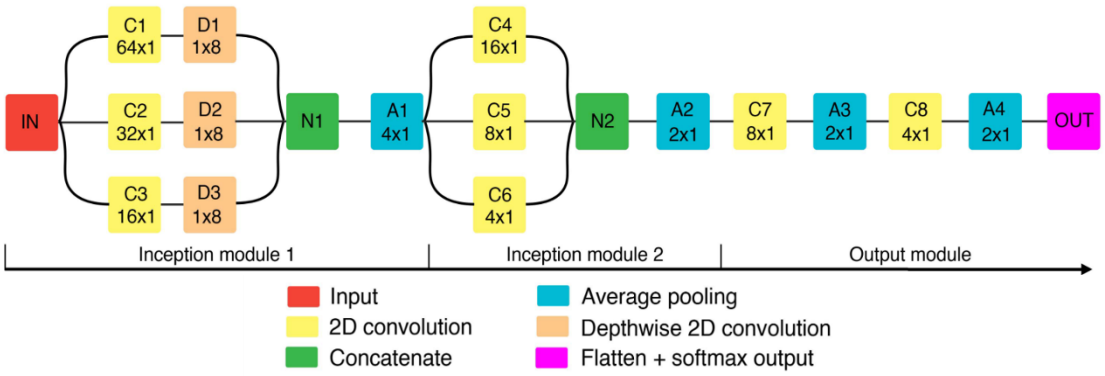

Figura 13 Descripción general de la arquitectura de EEG-Inception. Imagen adaptada de [43].

La red presenta la arquitectura mostrada en la Figura 13. En ella todas las capas de convolución y de convolución *depthwise* incluyen *batch normalization*, función de activación y regularización *dropout*. Se puede realizar un análisis de la arquitectura dividiendo la señal en bloques [43]:

1. Primer módulo *inception*: Procesa la señal de entrada en tres escalas temporales distintas. Para esta aplicación concreta, se emplearon ventanas temporales de 500 ms, 250 ms y 125 ms. Como se aprecia en la Figura 13, los tamaños de *kernel* son 64 × 1, 32 × 1 y 16 × 1, que resultan de la multiplicación de la frecuencia de muestreo (128 Hz) y la ventana temporal correspondiente. Posteriormente se procesa la señal en el dominio espacial mediante las convoluciones *depthwise*. Finalmente concatena la información y aplica un *average pooling* para reducir la dimensionalidad.

2. Segundo módulo *inception*: Para mantener los mismos tamaños de ventana temporal en este módulo los tamaños de *kernel* empleados son 16 × 1, 8 × 1 y 4 × 1. En este módulo se extraen características temporales de alto nivel, que tienen en cuenta todos los canales de EEG. De nuevo, se concatena la información y se aplica un *average pooling*.

3. Módulo de salida: Presenta dos capas de convolución seguidas de capas de *average pooling*, para extraer los patrones más relevantes y comprimir la información en unas pocas características. La capa de salida es una *fully conected* con la función *softmax*.

En el presente TFG, se ha presentado el EEG como una herramienta esencial en el ámbito del diagnóstico clínico y la investigación neurocientífica, con un énfasis particular en su aplicación en sistemas BCI. También, se ha destacado la susceptibilidad de la señal de EEG a la contaminación por artefactos oculares, lo cual puede comprometer la precisión de los resultados obtenidos. En el siguiente capítulo, se llevará a cabo una exhaustiva revisión del estado arte en relación con la detección de artefactos en las señales de EEG. Dada la versatilidad y eficacia demostrada por las redes neuronales, no sorprende que algunos de los algoritmos abordados en esta revisión incluyan arquitecturas como las CNN o las GAN.

Capítulo 4: Revisión del estado del arte

4.1. Introducción

A lo largo de décadas, el EEG ha desempeñado un papel esencial en el diagnóstico de trastornos neurológicos, la investigación de la cognición y la neurociencia en general. Sin embargo, la adquisición y análisis precisos de las señales de EEG se ven obstaculizados por la presencia de artefactos. La detección y eliminación de estos artefactos han sido un desafío constante en la comunidad científica, ya que la preservación de la integridad de la señal de EEG es crucial para la obtención de resultados fiables y la interpretación precisa. En este contexto, la automatización de los procesos de detección y eliminación de artefactos ha adquirido una importancia significativa, dado que los métodos manuales son inherentemente laboriosos, subjetivos y susceptibles a errores humanos.

Este estado del arte tiene como objetivo proporcionar una visión integral de los últimos métodos de detección automática de artefactos oculares en EEG. Se explorarán las estrategias y técnicas más innovadoras que abordan este desafío, dejando de lado aquellas que incluyan las señales adicionales y destacando las que operan en tiempo real. La efectividad de los algoritmos se mide en términos de precisión (*accuracy*) y precisión balanceada. Estas métricas se describen en el Apéndice A. Debido a la ausencia de métodos que aborden la detección temporal de movimientos oculares, el estado del arte está dividido en dos secciones. La primera sección (Apartado 4.2), se centra en algoritmos de detección de parpadeos. Por otro lado, en la segunda sección (Apartado 4.3) se describen algoritmos capaces de automatizar la clasificación de componentes independientes de ICA, incluyendo aquellas que presentan artefactos oculares.

4.2. Algoritmos de detección de parpadeos

Debido a las formas de onda características que se generan en los canales frontales durante los parpadeos, se pueden crear métodos muy simples basados en la observación de si la señal de EEG supera un determinado umbral de voltaje. A pesar de la sencillez inherente a estos algoritmos, se obtienen resultados aceptables y permiten detectar los parpadeos en tiempo real. En el artículo de Verleger [57], empleando un umbral de 50 μV sobre el canal Fz se logró una precisión del 73 % de parpadeos detectados.

Métodos más sofisticados emplean, además de umbrales, funciones simples como detección de mínimos y máximos o cálculo de distancias entre picos. Un ejemplo de estos métodos es el algoritmo desarrollado por Shachar *et al.* [58], que obtuvo un 100% de precisión y es capaz de operar en tiempo real. Sin embargo, fue evaluado sobre una pequeña base de datos que hace que este resultado no sea del todo concluyente.

En el artículo de Egambaram *et al.* [59], se diseñó un algoritmo que enventana la señal y analiza únicamente aquellas ventanas que presentan una correlación entre los canales Fp1 y Fp2 superior al 90 %. Las ventanas que cumplen esa condición son clasificadas como parpadeo si superan un umbral variable. Este se calcula como la media más dos desviaciones típicas calculadas con respecto a un segmento anterior a la ventana. Con el método propuesto se alcanzó una precisión en la detección del 96.6 %.

En el artículo de Nolan *et al.* [60], se describe un método automático basado en umbrales estadísticos para diversas características de la señal de EEG. Algunos ejemplos de estas características son: varianza y correlación de canales, rango de amplitud de las épocas, *kurtosis* de las componentes independientes calculadas mediante ICA, *etc*. Se determina que los datos están contaminados si la *z-score* de alguna de las características es superior a 3. Con este método se alcanzó una precisión del 97.64%. Sin embargo, el cálculo de algunas características hace que el método sea computacionalmente poco eficiente y que no sea capaz de trabajar en tiempo real.

En el artículo de Chang e Im [61], se desarrolla un método basado en la comparación plantillas o *fingerprints* de parpadeos. Este método requiere de la obtención de una plantilla de forma supervisada, seguida por el cálculo de una métrica de similitud entre la plantilla y los datos continuos del EEG, empleando una ventana móvil. La métrica de similitud empleada es *Dynamic Positional Wraping* una variante de la deformación dinámica del tiempo (DTW), que permite calcular la similitud entre dos señales teniendo en cuenta desfases temporales. Con este método se logró una precisión del 96.10%.

En algunos artículos se emplean los parpadeos generados de forma voluntaria como señales de control adicionales para sistemas BCI. Con este propósito, en el artículo de Giudice *et al.* [62], se presenta una red convolucional de una dimensión (1D-CNN), capaz de detectar y clasificar en tiempo real parpadeos voluntarios e involuntarios. Se obtuvo una precisión del 97.92 %.

4.3. Algoritmos de detección de componentes independientes

Dentro de los algoritmos de detección de artefactos oculares se incluyen a los algoritmos que tratan de automatizar la selección de componentes asociadas a artefactos encontradas en métodos de *blind source separation* (BSS). ICA a pesar de ser el método más empleado en el filtrado de artefactos, presenta un problema asociado, en ella las componentes se ordenan o bien de forma aleatoria o siguiendo un orden decreciente de varianza. Esto hace que se necesite de una clasificación manual de las mismas como señal o artefacto. Con la intención de automatizar este proceso, en varios artículos se describen métodos de clasificación de componentes independientes. Con esta metodología, a pesar de no hacer una detección temporal precisa de los artefactos, además de parpadeos se detectan otros artefactos asociados a la señal entre los que se incluyen los movimientos de los ojos.

El algoritmo desarrollado por Frølich *et al.* [63], emplea un clasificador de regresión logística multinomial. Este clasificador emplea 14 características espaciales, temporales y espectrales, para identificar componentes asociadas a actividad neural, cardiaca y muscular, así como movimientos laterales de los ojos, parpadeos y actividad mezclada. Se evaluó el rendimiento del algoritmo sobre dos bases de datos obteniendo un 74 % y un 62 % de precisión balanceada.

Por otro lado, Pion-Tonachini *et al.* [64] además de un clasificador, realizaron una amplia base de datos de componentes independientes etiquetadas. Las etiquetas empleadas distinguen entre componentes de origen cerebral, muscular, ocular, cardiaco, de red, de

canales y de otros orígenes. Su clasificador está basado en las redes GAN. En este caso, dado que es un problema de múltiples clases, existen tantas redes generadoras como etiquetas se quieren clasificar. Las redes generadoras intentan generar componentes independientes similares a las reales, mientras que el discriminador trata de determinar la probabilidad de que pertenezcan a una clase o a otra. Además de evaluar su algoritmo en su base de datos también evaluaron otros como el de Frølich *et al*. [63], estos algoritmos lograron una precisión balanceada de 62.3 % y 57.8 %, respectivamente. Sin embargo, demostraron una precisión balaceada de 85.5 % y 87.0 %, respectivamente, al evaluarse únicamente con dos clases, actividad cerebral y otros.

También es común emplear ICA en metodologías hibridas, como es el caso del algoritmo propuesto por Jafarifarmand *et al*, [18]. En él se combinan la técnica de ICA con un filtro adaptativo. Se obtienen primero las componentes independientes a partir de la señal de cuatro electrodos frontales. Se selecciona la señal de artefacto de las componentes independientes mediante la métrica de correlación absoluta, que evalúa la similitud entre los canales de EEG, Fp1 y Fp2 y las componentes independientes. Posteriormente esta señal se emplea como referencia en un filtro adaptativo para corregir la señal. El artículo no proporciona datos de precisión en la detección de la señal de artefacto y se centra en métricas que cuantifican la eliminación del artefacto y la preservación de la información neural.

Para aportar una visión general de este estado del arte, la Tabla 3 describe las características principales de cada uno de los algoritmos. En la tabla se indican los artefactos detectados, el método empleado, la precisión obtenida y la capacidad de operar en tiempo real, de cada uno de los algoritmos.

Artículo	Método	Artefactos	Precisión	Precisión balanceada	Tiempo real
Verleger [57]	Umbral	Parpadeos	73.00%	-	Si
Shachar *et al*. [58]	Umbrales y funciones simples	Parpadeos	100%	-	Si
Egambaram *et al*. [59]	Correlación y umbral	Parpadeos	96.60%	-	No
Nolan *et al.* [60]	Umbrales estadísticos de características	Parpadeos	97.64%	-	No
Chang e Im [61]	*Dynamic Positional Wraping*	Parpadeos	96.10%	-	No
Giudice *et al.* [62]	(1D) CNN	Parpadeos voluntarios y no voluntarios	97.92%	-	Si
Frølich *et al.* [63]	ICA-RLM	muscular, ocular, cardiaco	-	0.578	No
Pion-Tonachini *et al*. [64]	ICA-GAN	muscular, ocular, cardiaco, de red, de canales	-	0.623	No

Tabla 3 Métodos de detección de artefactos oculares sin el empleo de señal de EOG. Siglas: CNN: *Convolutional Neural Network,* ICA: Independent Component Analysis, RLM: Regresión Logística Multinomial, CNN: *Generative Adversarial Networks.*

Capítulo 5. Materiales y métodos

5.1. Introducción

El algoritmo se desarrolló enteramente en el lenguaje de programación Python. Para la transferencia de registros desde archivos de MATLAB a Python, se utilizó la biblioteca MNE (*Magnetoencephalography and Electroencephalography)*. Para llevar a cabo el preprocesamiento de estos registros, se emplearon las bibliotecas MEDUSA [65] y SciPy Signal. Finalmente, la implementación del modelo se llevó a cabo utilizando TensorFlow 2.11. El equipo utilizado para el entrenamiento y evaluación presenta un procesador *Intel Core i7*, @ 3.40 GHz y 16 GB RAM

En el siguiente subapartado de este capítulo, se describe detalladamente la base de datos empleada en el entrenamiento, junto con el preprocesado y etiquetado de artefactos. A continuación, se explica la metodología llevada a cabo para preprocesar los datos que se van a emplear en el entrenamiento del modelo. Después, se describe una base de datos creada en el laboratorio siguiendo un paradigma similar al que presenta la primera base de datos. Posteriormente, se explica el modelo clasificador y cómo se integra dentro del algoritmo. Finalmente, se aborda la validación cruzada realizada, así como la complejidad temporal del algoritmo.

5.2. Base de datos de entrenamiento

5.2.1. Contenido y estructura

La principal base de datos utilizada se describe en el artículo de Kobler *et al*. [36], que es de acceso público y consta de 59 registros de EEG realizados durante 5 estudios diferentes. Los estudios se llevaron a cabo en la Universidad Tecnológica de Graz y en ellos participaron 50 sujetos sanos. La finalidad los registros era la evaluación de un algoritmo de substracción de subespacios. La base de datos está compuesta de épocas con artefactos oculares etiquetados. Los registros se realizaron de acuerdo con el paradigma presentado en la Figura 14. En él se distinguen 4 condiciones diferentes. En función de la condición, el participante debe mantener los ojos abiertos y dirigir la mirada a un punto localizado en el centro de la pantalla (REST), seguir el estímulo a lo largo del eje horizontal de la pantalla (HORZ), seguirlo a lo largo del eje vertical (VERT) o parpadear como lo haría de forma involuntaria conforme el punto reduce su tamaño (BLINK). En el paradigma experimental, cada trial comienza con un segundo de preparación, seguido de 10 segundos de tarea y entre 2 y 3 segundos de descanso, véase Figura 14.

Cada estudio se subdividió en dos bloques experimentales. En términos generales, cada bloque consistió en 27 *trials*, de los cuales 9 correspondieron a la condición de REST y 6 a HORZ, 6 a VERT y 6 a BLINK. La duración aproximada de cada bloque fue de 5 minutos, y entre bloques existía un intervalo de separación de hasta 100 minutos. Esto permitió a Kobler *et al*. demostrar que un modelo invariante en el tiempo tiene la capacidad de describir los artefactos oculares.

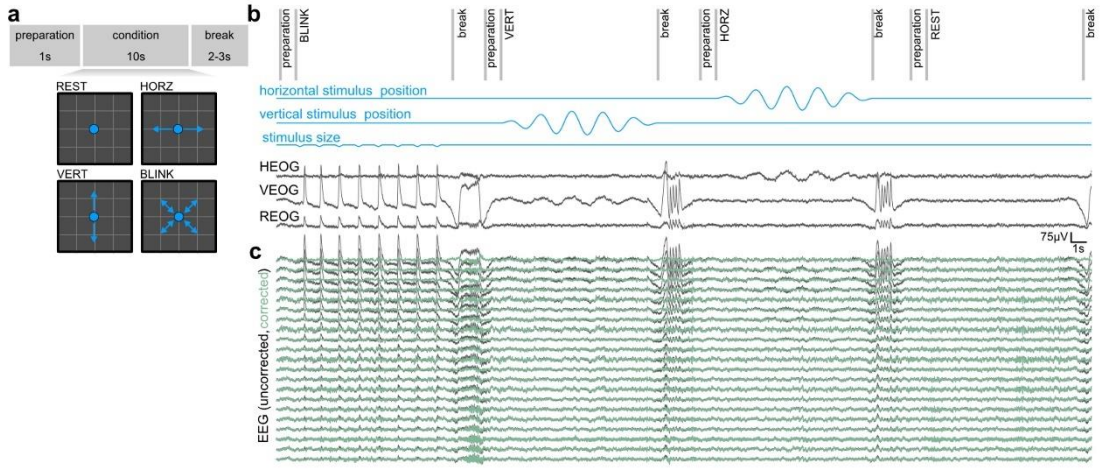

Figura 14 a Detalla la distribución temporal del paradigma, así como el movimiento del estimulo durante el registro. **b** Las lineas azules indican la posición y el tamaño del estímulo durante cuatro ensayos consecutivos. Las tres señales inferiores pertenecen a señales de electrooculograma horizontal (HEOG), vertical (VEOG) y radial (REOG), que es un promediado de las señales de EOG registradas. **c** Comparativa del EEG original y el corregido con el algoritmo desarrollado por Kobler et al. Imagen adaptada de [36].

Los tensores que contienen la señal en la base de datos están organizados de la siguiente manera *n* canales x *n* muestras x *n* épocas y las etiquetas de los canales EEG y EOG vienen dadas en un vector independiente.

Por otro lado, se proporciona un vector con las etiquetas del tipo de evento que ocurre en cada trial. Los valores asociados a cada evento son los siguientes:

1 - Reposo
2 - Movimiento de ojos horizontal
3 - Movimiento de ojos vertical
4 - Parpadeos

Por último, la base de datos incluye otro canal adicional denominado "artifactclasses" con la misma longitud que la señal EEG, en el cual a cada muestra se le asigna un valor en función del tipo de artefacto presente en ese instante temporal. De esta manera, se identifica el inicio y final de los distintos artefactos dentro de los *trials*. Los valores asociados a los distintos artefactos son los siguientes:

0 - Ninguno
1 - Movimiento de ojos hacia la derecha
2 - Movimiento de ojos hacia la izquierda
3 - Movimiento de ojos hacia arriba
4 - Movimiento de ojos hacia abajo
5 - Parpadeo
6 - Reposo

Cada estudio emplea un montaje EEG diferente, siguiendo todos el sistema internacional 10-10. En la Tabla 4 se detallan las características de cada uno de los estudios.

Estudio	Nº sujetos	Frecuencia muestreo	Nº canales EEG	Nº canales EOG	Posición referencia	Gorro	Amplificador
1	5	200	58	6	mastoide derecho	antiCAP	BrainAmp
2	15	200	64	6	mastoide derecho	antiCAP	BrainAmp
3	10	200	64	6	mastoide derecho	antiCAP	BrainAmp
4	15	100	61	3	lóbulo de la oreja derecha	g.GAMMAsys	g.USBamp
5	14	256	58	6	mastoide derecho	antiCAP	BrainAmp

Tabla 4 Características de los 5 estudios de la base de datos de Kobler *et al*. [36].

5.2.2. Preprocesado

En la base de datos todos los registros presentan un preprocesado común. A continuación, se describe la metodología llevada a cabo por Kobler *et al.* [36].

A las señales se les aplicaron dos filtros:

- Filtro *notch*, Butterworth de Segundo orden, frecuencias de corte 49 y 51 Hz.
- Filtro paso-alto, Butterworth de Segundo orden, frecuencia de corte 0.4 Hz.

Ambos se aplicaron de forma bidireccional, el propósito del filtro *notch* fue evitar el ruido de la red eléctrica y el de el paso-alto, las derivas. En la Figura 15 se puede apreciar la distribución espectral de potencia de una señal con la condición REST y comprobar la eficacia del filtrado.

Posteriormente se inspeccionaron las señales de forma visual para localizar canales con señal ruidosa que no registrara actividad cerebral. Sobre estos se aplicó una interpolación de *splines* esféricos. Esta técnica matemática permite proyectar los electrodos como un conjunto de puntos en una esfera, lo que facilita el cálculo de valores intermedios [66].

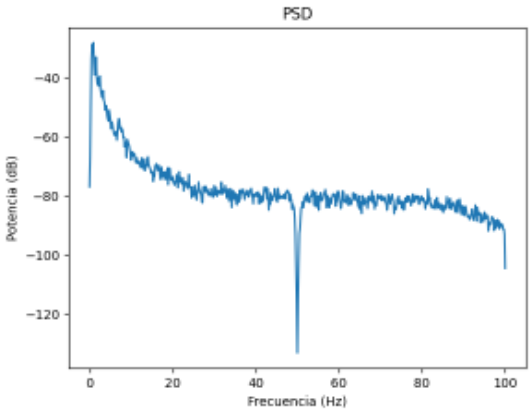

Figura 15 PSD de la señal en escala logaritmica (dB vs Hz).

Finalmente se extrajeron épocas de 8s de duración. De esta manera, se excluyó el primer y último segundo de los 10 segundos totales de condición que corresponden a cada trial (véase Figura 14). Tras un análisis visual de las épocas, se eliminaron aquellas contaminadas por artefactos musculares, saltos de voltaje y derivas de la línea base. Las épocas eliminadas representaron un 14% del total [36].

Durante el preprocesado también se calcularon las señales de HEOG y VEOG. Estas señales fueron filtradas con un filtro paso bajo, Butterworth de segundo orden y con una frecuencia de corte de 5 Hz. De igual manera que con la señal de EEG el filtro se aplicó bidireccionalmente.

5.2.3. Etiquetado de artefactos oculares

Se emplearon las nuevas señales calculadas de EOG para extraer los momentos de las épocas en los que había movimientos de ojos y parpadeos y así crear el canal adicional *artifactclasses*. En las épocas extraídas de los *trials* con la condición BLINK, se detectó como parpadeo los tramos de señal que excedieran un voltaje de 75 μV durante al menos 25 ms. En la Figura 16 a se aprecia como cuatro parpadeos afectan a la señal del canal F3, junto con el etiquetado del canal *artifactclasses*. En las épocas con la condición VERT, se realizó una clasificación entre movimientos oculares hacia arriba o hacia abajo. Se consideraron aquellos segmentos en los cuales la señal de VEOG se mantenía por encima de 10 μV o por debajo de -10 μV, respectivamente, durante al menos 200 ms. De igual manera, en las épocas con la condición HORT, los movimientos hacia la derecha o hacia la izquierda se detectaron cuando la *señal de* HEOG se mantenía por encima de 10 μV o por debajo de -10 μV, respectivamente, durante al menos 200 ms. Ejemplos del etiquetado de movimientos oculares verticales y horizontales se pueden apreciar en la Figura 16 c y d.

En los casos en los que se detectaron parpadeos fuera de la condición de BLINK, se consideraban como *outliers*. Para su detección, se emplearon dos umbrales, si la condición era VERT, se consideraban outliers a las muestras de los segmentos que superaban los 150 μV por al menos 200 ms, mientras que en las épocas de REST y HORZ el umbral era de 100 μV. Estos outliers, al igual que aquellos segmentos que no lleguen a los umbrales de parpadeos o movimientos de ojos, se clasifican como "nada" y se les da el valor 0 en el canal *artifactclasses*. Esto implica que en el canal *artifactclasses* asociado a cada época únicamente existen como máximo 3 valores posibles. Por ejemplo 0, 1, 2 en el caso de una época de movimientos de ojos horizontales o 0, 5 en el caso de una época de parpadeos. En la Figura 16 b se aprecia la identificación de un parpadeo en una época de REST.

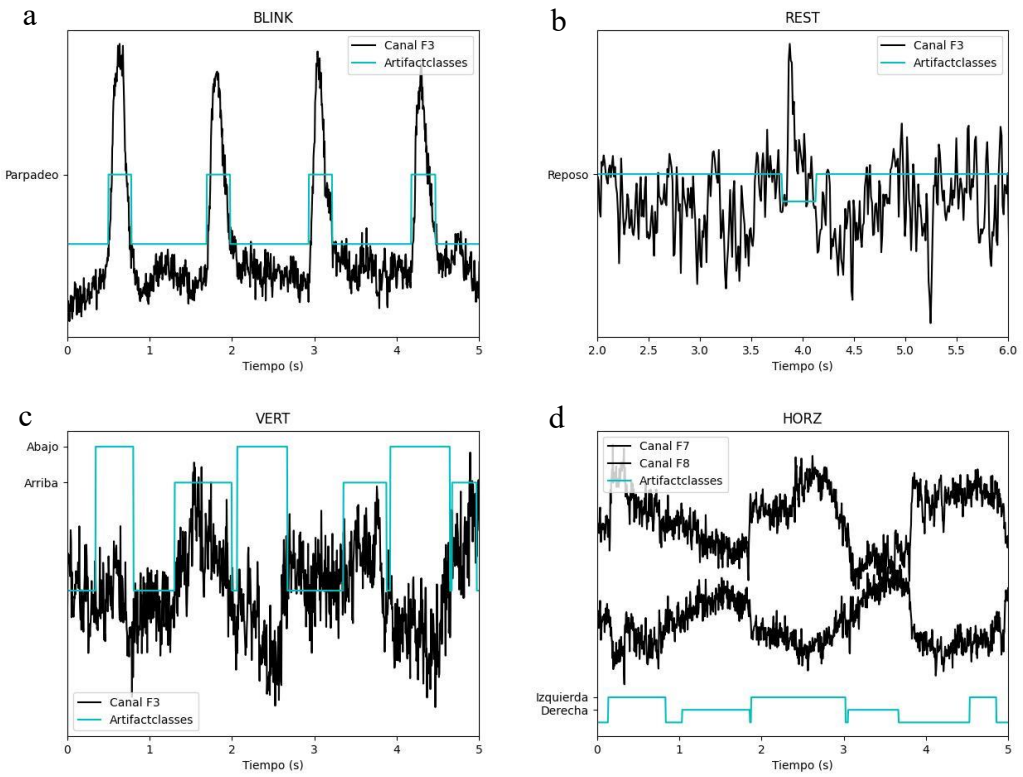

Figura 16 a Segmento de época de parpadeos registrada con el canal F3 y la clasificación de artifactclasses. **b** Segmento de época de *resting* con un parpadeo. **c** Segmento de época de movimiento vertical. **d** Segmento de época de movimientos horizontales registrada mediante F7 y F8.

5.3. Preprocesado para entrenamiento

Debido a que los registros presentan diferentes frecuencias de muestreo, es necesario realizar un remuestreado. Los tres primeros estudios presentan una frecuencia de muestreo de 200 Hz, el cuarto de 100 Hz y el quinto de 256 Hz. Se realizó un *downsampling* a 100 Hz sobre los tres primeros y el quinto. Para evitar el *aliasing*, previamente las señales fueron filtradas a 49 Hz. El filtro utilizado fue del tipo Butterworth, de orden 3.

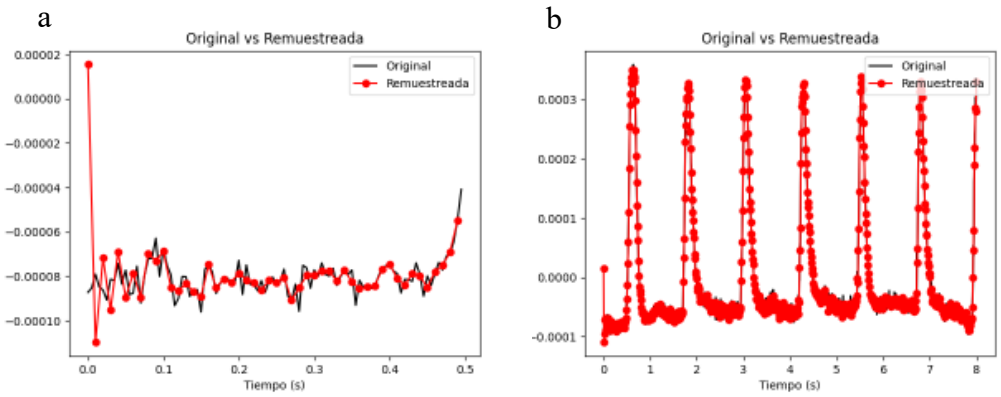

Figura 17 a Comparación de señal original frente a remuestreada, del tramo inicial de una época con la condición BLINK. **b** Época completa remuestreada.

Después del proceso de remuestreo, se detectaron irregularidades en el inicio de algunas épocas. En la Figura 17 se muestra el ejemplo de una época en la que ocurren estas irregularidades. Con la intención de omitir el segmento irregular, no se tuvo en cuenta como mínimo las 5 primeras muestras de las épocas, tanto en el entrenamiento del modelo, como en la evaluación del algoritmo de clasificación.

Tras el remuestreado se llevó a cabo una normalización de las épocas. Para ello de forma general se calculó el *z-score* de cada una de las muestras siguiendo la siguiente ecuación:

$$X[t] = \frac{X[t]-M}{Std} \quad (6)$$

donde *M* y *std* son el valor promedio y la desviación estándar de la señal en una época concreta. Este tipo de normalización se aplicó sobre todas las épocas excepto las de parpadeos. Si se realizara este tipo de normalización sobre las épocas con condición BLINK, la señal basal quedaría muy desplazada hacia valores negativos en comparación con el resto de las condiciones. Esto se debe a que la amplitud durante los parpadeos alcanza valores 10 veces superiores al rango normal de variación de la señal basal. En esas condiciones es muy fácil que el modelo aprendiera fijándose únicamente en los valores que tomaban las muestras y sacara muy buenos resultados. Es por esto que se aplicó un ajuste especial, calculando únicamente la media y desviación típica de los fragmentos de la época que no se clasifican como artefactos y normalizar la época respecto a ellos.

La literatura no indica ningún tamaño de ventana estándar. Por ejemplo, en [67] se emplea un tamaño de 450 ms y en [68] de 1000 ms. Para entrenar el modelo se emplearon ventas de diferentes tamaños y se determinó mediante heurística un tamaño óptimo de 600 ms.

El proceso de enventanado se realizó de forma diferente en función de la condición de la época. Para la condición REST, se realizó un enventanado con un solapamiento del 70%. Esto implica que cada ventana de 600 ms compartía 420 ms con la ventana adyacente. Con la intención de excluir las muestras iniciales, se descartaba la primera ventana. Y en el caso que existieran outliers, se excluían aquellas ventanas que los incluyeran. Para las condiciones HORZ y VERT, únicamente se tuvo en cuenta la presencia de movimientos horizontales o verticales, sin atender a si estos eran hacia la izquierda, derecha, arriba o abajo. Para aumentar el número de ventanas, se unieron aquellos segmentos separados por menos de 150 ms (Figura 18). El proceso de enventanado se realizó de manera similar que para la condición REST. El solapamiento empleado fue del 70 %. Para aquellos casos en los que la señal estuviese entera clasificada como movimiento de ojos, se descartó la primera ventana. Por otro lado, en las épocas con outliers o segmentos no clasificados, se realizó el enventanado sobre segmentos con una duración superior a 420 ms y sin tener en cuenta los 50 primeros ms.

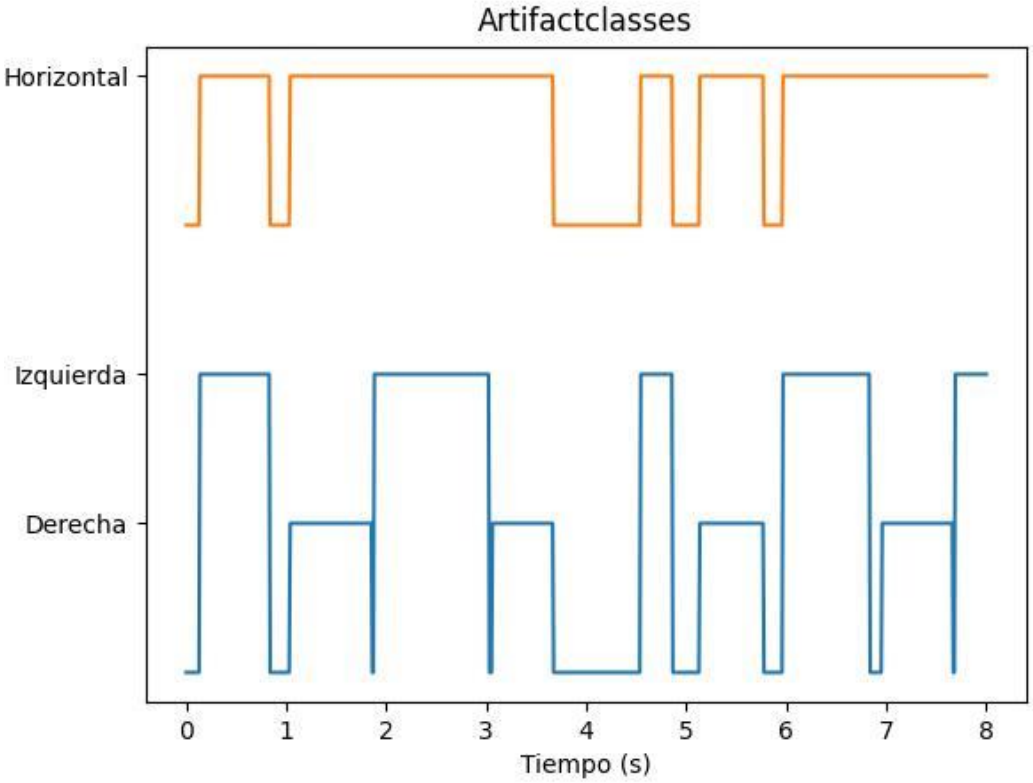

Figura 18 La señal cuadrada inferior representa el canal *artifactclasses* de una época de movimiento horizontal y distingue entre movimiento hacia derecha e izquierda. En la parte superior, se encuentra la señal cuadrada empleada para realizar el enventanado.

Por último, para la condición de BLINK el enventanado se usó una función de detección de picos sobre el canal *artifactclasses* para encontrar la muestra que se encontraba en el punto medio del parpadeo. A partir de este punto, se ubicaba una ventana central y si es posible una ventana adicional a cada lado con solapamiento del 50%, abarcando 1200 ms en total. Si el punto medio del parpadeo estaba a menos de 650 ms del inicio o 600 del final, no se creaban la primera o última ventana, respectivamente. De esta forma se evitaban también las irregularidades iniciales. En el enventanado realizado para entrenar al algoritmo se incluyen gran cantidad de muestras no clasificadas como parpadeo. Esto se puede apreciar en la Figura 19 Representa la metodología llevada a cabo en el enventanado de las épocas de parpadeos., en la que se representa el canal *artifactclasses,* junto con el canal F3 y las tres ventanas que se extraen de cada parpadeo. La inclusión de estas muestras que no fueron consideradas como parte del parpadeo por el etiquetado original por parte de Kobler *et al.* [36] se debe a que únicamente se consideró como parpadeo aquellas muestras de la señal que abarcaban el pico de voltaje (Figura 16 a), sin embargo, dado que estas muestras también se encuentran afectadas por el parpadeo, deben ser tomadas en consideración.

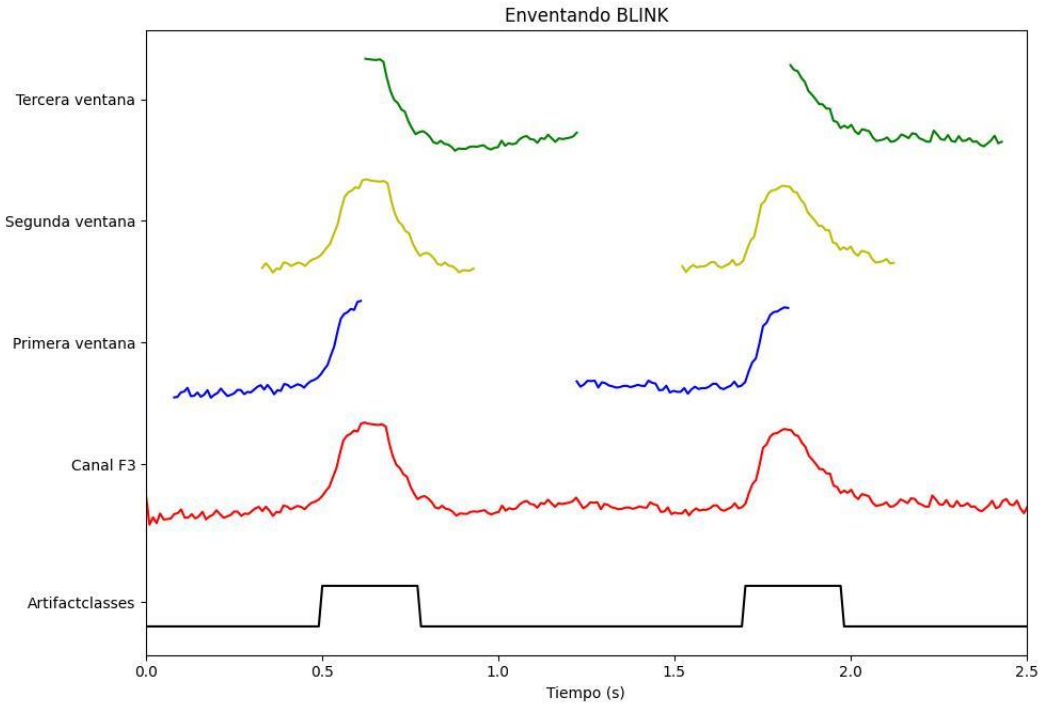

Figura 19 Representa la metodología llevada a cabo en el enventanado de las épocas de parpadeos.

Para entrenar el modelo, todas las ventanas debían tener la misma configuración de canales. Debido a que no todos los registros presentaban el mismo montaje, se seleccionaron 8 electrodos con la condición de que fueran comunes a todos los registros, permitieran caracterizar la propagación del potencial del artefacto en el espacio y fueran utilizados en la práctica común de BCI. Los canales elegidos fueron los siguientes: F3, F4, C3, C4, Cz, P3, P4 y Pz. Este montaje se puede emplear en el paradigma de MI [69], ya que permite registrar la señal de la región frontal y central, importantes en la preparación e imaginación del movimiento.

Tras las etapas descritas para la adaptación de la base de datos, la base resultante constaba de 28.233 ejemplos de REST, 13.771 de BLINK, 16.094 de HORZ y 10.793 de VERT. Todas estas ventanas contienen fragmentos de 600 ms.

5.4. Base de datos registrada en el laboratorio

5.4.1. Contenido y estructura

Esta base de datos se adquirió en el laboratorio, siguiendo un paradigma experimental similar al descrito en el artículo de Kobler *et al.* [36]. Sin embargo, esta base de datos se limita exclusivamente a las condiciones de REST y BLINK. Esta elección se fundamenta en que el registro de movimientos oculares carece de utilidad sin la información adicional del EOG, dado que la clasificación manual de estos artefactos no es tan sencilla como en el caso de los parpadeos.

En total, se llevaron a cabo 19 registros utilizando un grupo de 6 sujetos. Cada registro comprendió entre 4 y 8 épocas de reposo, seguidas de entre 4 y 10 épocas de parpadeos. De las 134 épocas de parpadeos registradas, se consideraron válidas 118, mientras que, de las épocas de reposo, se utilizaron 90 de las 106 registradas.

En este estudio, se empleó un amplificador g.USBamp (Guger Technologies OG, Graz, Austria) con 16 electrodos activos (FZ, F3, F4, F7, F8, FCZ, C3, C4, CZ, CPZ, P3, P4, PZ, PO7, PO8 y POZ) colocados en un gorro elástico siguiendo el estándar 10-10. El canal AFz se utilizó como tierra y se colocó una referencia común en el lóbulo de la oreja derecha. La señal se adquirió a una frecuencia de muestreo de 256 Hz.

La duración de las épocas registradas varió entre 5 y 10 segundos para *resting* y entre 12 y 30 para parpadeos. Durante los eventos de parpadeos, se generaron *time stamps* para tratar de ubicar temporalmente los parpadeos. Sin embargo, carecían de la precisión necesaria para ser utilizados como referencia en procesos de clasificación automática.

5.4.2. Preprocesado

A partir de la señal *raw* de EEG se obtuvieron las épocas de *resting* y parpadeos, para ello se emplearon los *time stamps* que marcaban el inicio y el final de la época. Tras la conversión a épocas se le aplicó el mismo preprocesado que tiene la otra base de datos:

Filtrados:

1. Filtro *notch*, Butterworth de Segundo orden, frecuencias de corte 49 y 51 Hz.
2. Filtro paso-alto, Butterworth de Segundo orden, frecuencia de corte 0.4 Hz.
3. Filtro paso-bajo, Butterworth de Segundo orden, frecuencia de corte 49 Hz.

Al tener una frecuencia de muestreo de 256 Hz fue necesario hacer el *downsampling* a 100 Hz. Posteriormente, se realizó una normalización de las épocas. En el caso de las épocas con la condición de *resting*, se realizó una normalización mediante *z-score*. Sin embargo, en el caso de las épocas con parpadeos se normalizaron respecto a la media y desviación típica medias de las épocas de *resting* con las que se registraron. De esta manera se logró evitar una normalización diferente a la que presentan las épocas de entrenamiento y lo que hubiese supuesto un peor rendimiento de la clasificación.

5.5. Modelo

La arquitectura de *Deep Learning* EEG-Inception diseñada para la detección de ERPs se consideró como una opción adecuada para la tarea de detectar los artefactos oculares. La red fue específicamente diseñada para EEG, con los distintos módulos *inception*, que extraen características en diferentes escalas temporales y las convoluciones *depthwise,* que permiten la extracción de características espaciales. La arquitectura presenta conceptos muy innovadores y que se esperan sean escalables a otros problemas.

En el presente TFG se ha tomado la implementación pública de EEG-Inception y se ha modificado acorde a las necesidades de este estudio. La Tabla 5 Comparativa de los parámetros de EEG-Inception [43] frente a la red modificada. presenta los valores predeterminados de la red original y los utilizados en este TFG.

Parámetro	Explicación	EEG-Inception	TFG
input_time	Duración de la época en ms.	1000	600
fs	Frecuencia de muestreo.	128	100
ncha	Número de canales de entrada.	8	8
filters_per_branch	Número de filtros en cada camino del módulo *inception*.	8	8
scales_time	Escala temporal en ms de cada convolución en los módulos *inception*.	500, 250, 125	240, 120, 60
dropout_rate	Proporción de neuronas que se desactivan durante el entrenamiento.	0.25	0.25
activation	Función de activación.	'elu'	'elu'
n_classes	Número de clases de salida	2	3
learning_rate	Tamaño del paso que el modelo toma en la dirección de la minimización del error durante el proceso de entrenamiento.	0.001	0.001

Tabla 5 Comparativa de los parámetros de EEG-Inception [43] frente a la red modificada.

Es bastante común emplear tamaños de ventana de 1s en la detección de ERP, sin embargo, no hay un tiempo de ventana establecido para la detección de artefactos. La señal se ve afectada de media entre 100 y 400 ms durante los parpadeos [39]. De forma heurística se obtuvo como tamaño óptimo de ventana 600 ms, capaz de abarcar la forma de onda completa del parpadeo. Otros tamaños de ventana evaluados fueron 400 ms, 500 ms, 700 ms y 1000 ms.

El número de canales empleado en este estudio coincide con el número de los propuestos por lo que no se modificó. Tampoco se consideró necesario modificar el número de filtros por camino de módulo *inception*, las funciones de activación, la tasa de *dropout*, ni el *learning rate*. Las escalas temporales óptimas se determinaron mediante heurística. En la Figura 20 se puede apreciar el esquema del modelo final. Se diferencia del esquema de EEG-Inception original en los tamaños de *kernel* de los módulos *inception*. La determinación del tamaño de *kernel* empleado en cada uno de los caminos se logra multiplicando la frecuencia de muestreo por la escala temporal correspondiente.

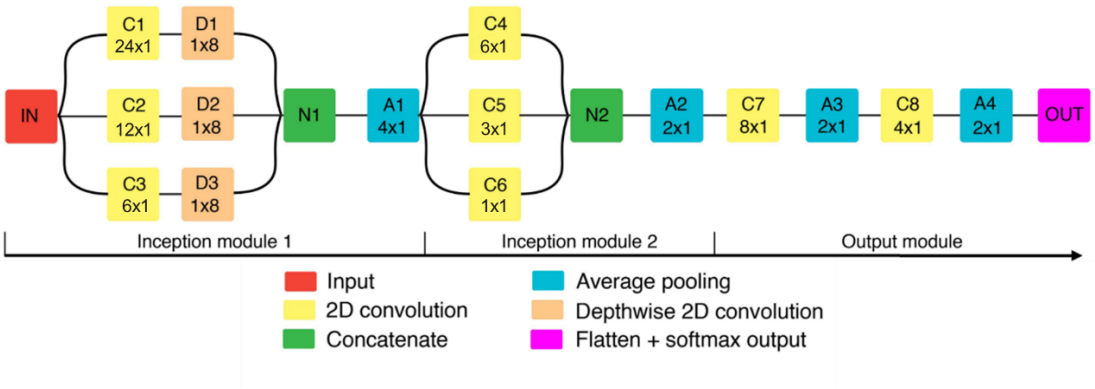

Figura 20 Esquema de la red EEG-Inception [43] modificada para este TFG.

5.6. Algoritmo

A partir del modelo de *Deep Learning*, se ha diseñado un algoritmo capaz de detectar temporalmente los artefactos en un registro de EEG. La señal puede constar de una o múltiples épocas, y no es necesario que tenga una duración específica. Sin embargo, sobre la señal es necesario haber realizado un preprocesado similar al que se aplicó sobre las épocas de las bases de datos.

El algoritmo diseñado en este TFG se subdivide en 3 etapas:

1. Enventanado de la señal recibida. El tamaño de ventana es de 600 ms y el solapamiento del enventanado es un parámetro que se puede modificar. A medida que se incrementa el solapamiento, se logra una mayor resolución en la clasificación del algoritmo, no obstante, esto conlleva un aumento en el costo computacional.
2. Cálculo de la probabilidad de pertenencia a una clase para cada ventana. Se aplica el modelo sobre todas las ventanas. Para cada muestra se acumulan las probabilidades de las ventanas en las que se haya incluida. En la Figura 21 se aprecia como se acumulan las probabilidades en tres vectores, uno para cada clase.
3. Clasificación de la muestra. Finalmente, se asigna la etiqueta de clase en función de la categoría que presente la probabilidad acumulada más alta. En la Figura 22 se aprecia como varía el vector de clasificación final en función de los vectores de probabilidad. En las figuras 21 y 22 el algoritmo presentaba un solapamiento del 90 %.

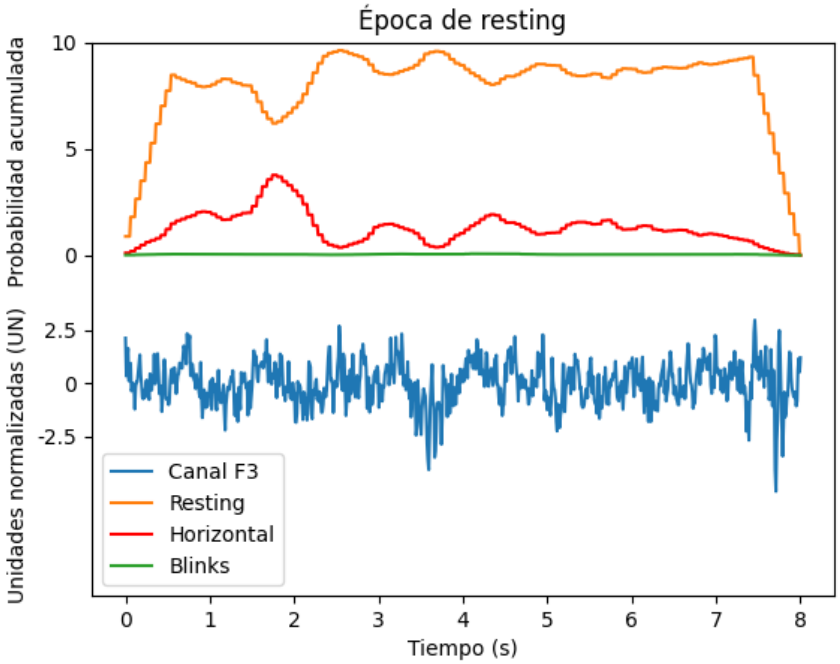

Figura 21 Ejemplo de época de *resting* correctamente clasificada. Se puede apreciar en la parte superior de la figura como se acumulan las probabilidades en los distintos vectores.

En la Figura 21, alrededor del segundo 2, se genera un pico en el vector "Horizontal", este pico se debe principalmente a la clasificación de dos ventanas como movimiento horizontal, con una probabilidad de entorno al 0.94. Sin embargo, gracias al solapamiento de estas con ventanas correctamente clasificadas, el error no perjudica el resultado final.

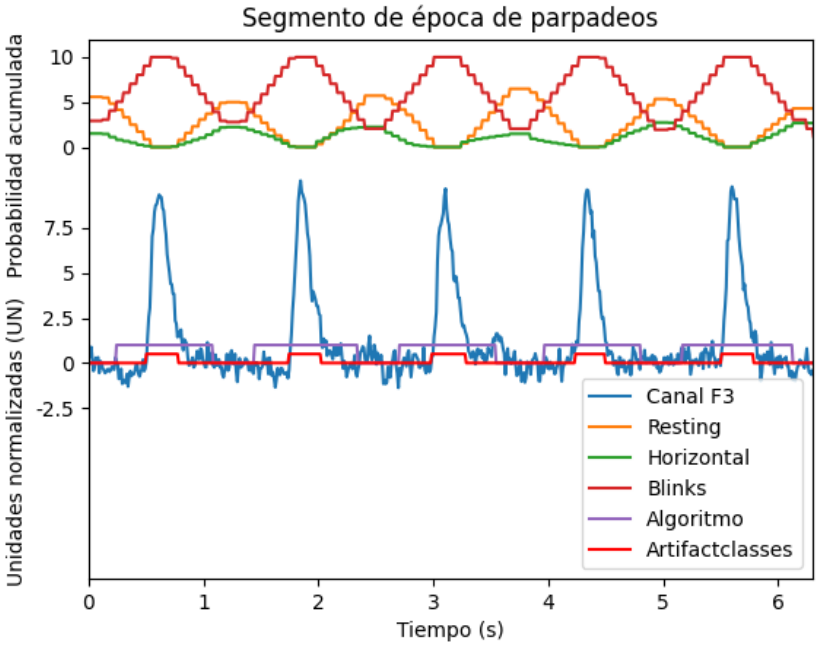

Figura 22 Representa un segmento de una época de parpadeos En la parte superior de nuevo se encuentran los vectores de probabilidad acumulada para cada clase y en la parte inferior se encuentra la señal del canal F3, junto con el canal ***artifactclasses*** y el vector de clasificación final del algoritmo.

5.7. Validación cruzada

5.7.1. Modelo

Para llevar a cabo una evaluación precisa del rendimiento del modelo, se optó por realizar una validación cruzada *k-fold*. En esta técnica, se repite el proceso de entrenamiento y evaluación *k* veces. El propósito principal de esta metodología radica en mitigar el sesgo inherente a una única partición de los datos en conjuntos de entrenamiento y validación[70]. Además, se busca aprovechar al máximo el conjunto de datos, ya que cada punto de datos se emplea tanto para entrenar como para validar el modelo en algún punto del proceso. Para este TFG se empleó un valor de $k = 6$. Además, se aplicó un proceso de reordenamiento aleatorio de las ventanas dentro de los registros con el fin de evitar cualquier sesgo en los datos que pudiera comprometer la validez de la validación cruzada.

De esta manera, los 59 registros se dividieron en 6 *folds* o subconjuntos de tamaño aproximadamente igual, donde 5 subconjuntos contenían 10 registros cada uno y el último subconjunto presentaba 9 registros. La aplicación de esta estrategia permite llevar a cabo una validación cruzada intersujeto. En consecuencia, las ventanas correspondientes a los registros de un sujeto no están presentes simultáneamente en los conjuntos de prueba y entrenamiento. Este enfoque garantiza la obtención de resultados más rigurosos y realistas en cuanto al rendimiento del modelo, evitando así la sobreestimación de su desempeño. Además, la validación intersujeto es especialmente útil en este caso, dado que es importante garantizar la aplicabilidad del modelo final sobre nuevos sujetos o participantes.

Una vez realizada la partición de datos, se llevó a cabo el proceso de entrenamiento y evaluación *k* veces. En cada iteración, se empleó uno de los *folds* como conjunto de test y los *k*-1 *folds* restantes como conjunto de entrenamiento. Las métricas empleadas en la evaluación de los modelos fueron las siguientes: *Recall, Precision, F1-score, Accuracy* y *Balance Accuracy.* Todas ellas se describen en el apéndice A.

5.7.2. Algoritmo

Para la evaluación del algoritmo se requieren segmentos de mayor tamaño. En consecuencia, se emplearon las épocas completas. Manteniendo los subconjuntos de test, se evaluó el algoritmo sobre cada *fold*, empleando el respectivo modelo. La validación cruzada mediante *k-fold* se utilizó para determinar la validez del modelo dentro del algoritmo. Se compararon muestra a muestra los resultados de la clasificación del algoritmo y la clasificación de la base de datos. Seguidamente, se calcularon las siguientes métricas: *Recall, Precision, F1-score* y *Accuracy*.

Una vez se determinó su validez, se entrenó un modelo definitivo empleando todos los registros en el conjunto de entrenamiento. Para evaluar la capacidad de este modelo, se decide emplear una base de datos diferente a la usada en el entrenamiento, la registrada en el laboratorio. Para esta base de datos se emplearon un equipo y condiciones diferentes, lo que aporta más robustez a los resultados. Las épocas de reposo se evaluaron muestra a muestra y se obtuvieron datos del *accuracy* del algoritmo en su clasificación. En las

épocas de parpadeos registradas en el laboratorio los parpadeos no estaban etiquetados. En consecuencia, fue necesaria una evaluación manual de la clasificación del algoritmo. Se calculó el *accuracy* de cada una de las épocas en función del número de parpadeos correctamente clasificados entre el número total de parpadeos.

5.8. Complejidad temporal

Para evaluar la posibilidad de aplicar este algoritmo en tiempo real, se midió la complejidad temporal de este algoritmo. La complejidad temporal de un algoritmo hace referencia al tiempo de cómputo que supone ejecutarlo. Para calcularla se realizó una simulación con un segmento de señal de 2 s. Este es un tamaño de ventana empleado en tareas de *neurofeedback*. Los cálculos se realizaron para distintos valores de solapamiento, 70 %, 80 % y 90 %. Se llevaron a cabo 70 iteraciones para cada uno de los solapamientos.

Capítulo 6. Resultados

6.1. Modelo

El modelo, desde el inicio de las pruebas, demostró un buen rendimiento en la tarea de clasificación de parpadeos y movimientos oculares horizontales. Sin embargo, con respecto a los movimientos oculares verticales, se evidenció una capacidad limitada del modelo para discernir entre estos y la señal de EEG en estado de reposo. Cabe señalar que los movimientos verticales exhiben menor influencia disruptiva en la señal de EEG que los horizontales o los parpadeos [35]. Debido a esto, y a la dificultad que le suponía al modelo su clasificación, se determinó que la opción más apropiada consistía en excluir los movimientos verticales del conjunto de datos y entrenar únicamente con el resto de las condiciones. Los resultados que se muestran a continuación únicamente tienen en cuenta estas tres clases. En la Figura 23 a, b y c se representan mediante *violinplot* las métricas *recall*, *precision* y *F1-score*, respectivamente y se compararan para cada una de las clases. En la Figura 23 d se comparan los valores de *accuracy* y *balance accuracy* obtenidos, también mediante un *violinplot*. La métrica de *balance accuracy* se calculó con la intención de aportar más información del desempeño del modelo, siendo especialmente útil en este caso debido a que la base de datos presenta diferencias importantes en el número de ventanas de cada clase. En la Figura 24 se representa en tanto por ciento la suma de las 6 matrices de confusión. En las filas refleja los valores reales y en las columnas los valores predichos. Finalmente, en la Tabla 6, se representan los valores promedio calculados de todas las métricas y su desviación típica asociada.

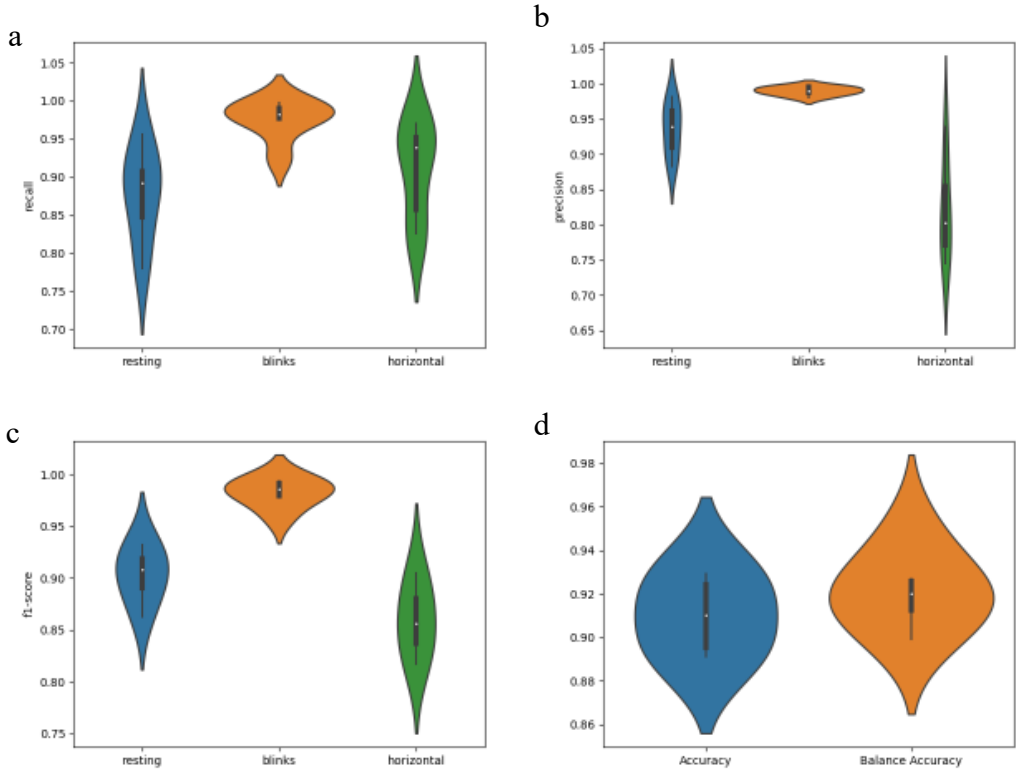

Figura 23 Metricas de evaluación de los 6 modelos representadas mediante *violinplot*. **a**. *Recall* calculados sobre las épocas de *resting*, *blinks* y *horizontal*. **b** y **c**. *Precision* y *F1-score*, respectivamente, representadas de la misma manera que *Recall*. **d**. *Violinplot* de las métricas: *accuracy* y *balance accuracy*.

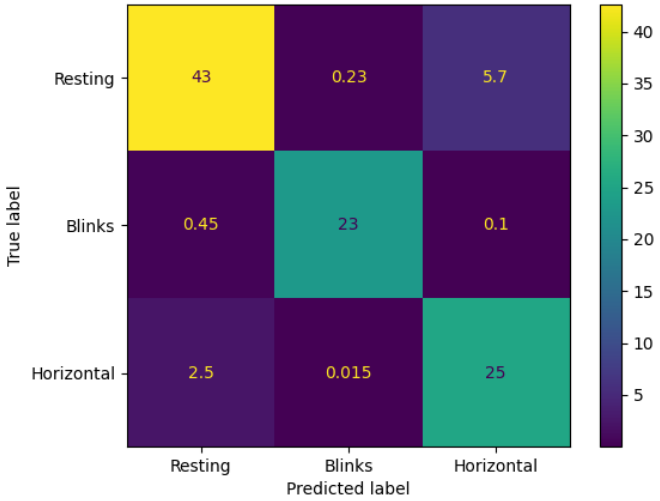

Figura 24 Representa en tanto por ciento la suma de los valores de las matrices de confusión calculadas para los distintos modelos.

Condición	Recall	Precision	F1-score	Accuracy	Balance Accuracy
Resting	87.7 ± 5.7	93.5 ± 3.5	90.3 ± 2.3	91.0 ± 1.6	92.1 ± 1.6
Blink	97.6 ± 2.4	99.0 ± 0.6	98.3 ± 1.2		
Horizontal	91.0 ± 5.8	82.0 ± 6.6	85.9 ± 3.0		

Tabla 6 Valores medios de las distintas métricas calculadas, junto con su desviación típica.

6.2. Algoritmo

6.2.1. Base de datos de entrenamiento

En la evaluación del algoritmo sobre esta base de datos se mantienen las mismas métricas que en la evaluación anterior a excepción de *balance accuracy*. En la Figura 25 a, b, c y d se representan mediante un *violinplot* las métricas *recall*, *precision*, *F1-score* y *accuracy*, respectivamente y se compararan los valores obtenidos en cada una de las épocas. En la Tabla 7 se muestran los valores medios de todas las métricas y su desviación típica asociada.

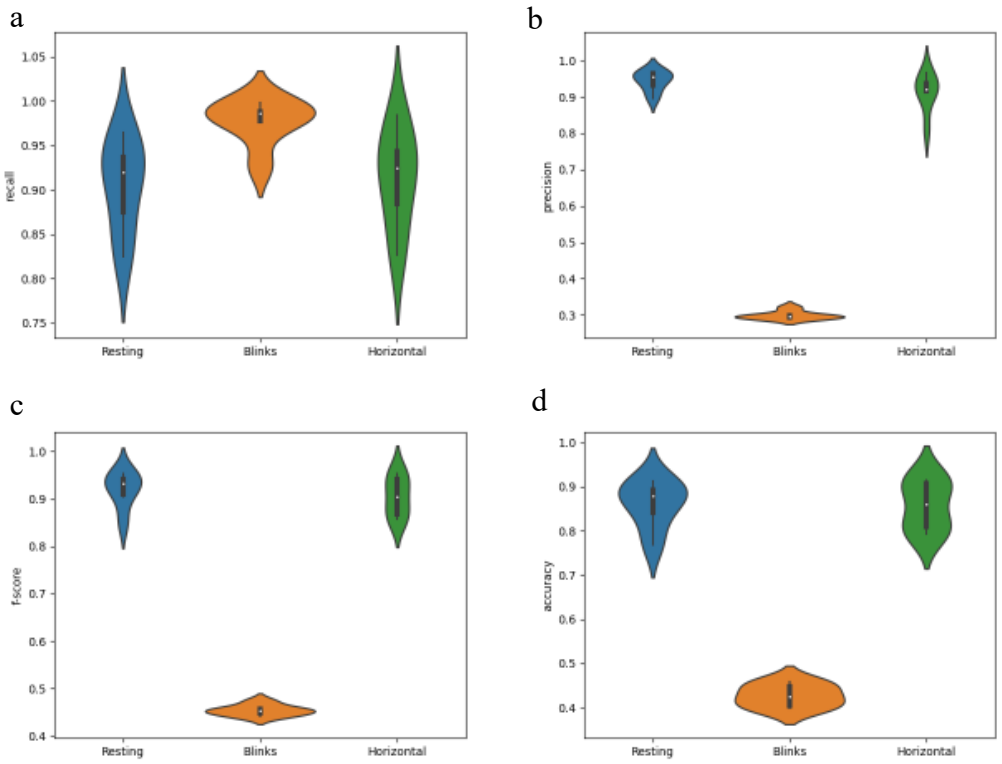

Figura 25 Metricas de evaluación del algoritmo representadas mediante *violinplot*. **a**. *Recall* calculados sobre las épocas de *resting*, *blinks* y *horizontal*. **b**, **c** y **d**. *Precision* y *F1-score*, *accuracy* respectivamente, representadas de la misma manera que *Recall*.

Condición	Recall	Precision	F1-score	Accuracy
Resting	90.5 ± 4.8	94.4 ± 2.5	92.0 ± 3.5	86.5 ± 4.8
Blink	97.7 ± 2.3	30.0 ± 1.0	45.4 ± 1.0	43.3 ± 2.4
Horizontal	91.3 ± 5.1	91.2 ± 4.9	90.5 ± 3.9	85.8 ± 5.1

Tabla 7 Valores medios de las distintas métricas calculadas, junto con su desviación típica.

6.2.2. Registros de laboratorio

En los registros de laboratorio se evaluaron las épocas de *resting* y de parpadeos de distinta forma. En las épocas que cumplían la primera condición, se consideró que todas las muestras cumplían la condición y que no presentaban artefactos. Por ende, debían ser clasificadas por el algoritmo enteramente como *resting* y se realizó una evaluación muestra a muestra. En segundo lugar, la clasificación del algoritmo para las épocas de parpadeos fue evaluada mediante inspección visual. Los resultados de las clasificaciones han sido representados mediante un diagrama de barras en la Figura 26. En la Tabla 8. se representan los valores de *accuracy* calculados para las épocas de reposo y de parpadeos.

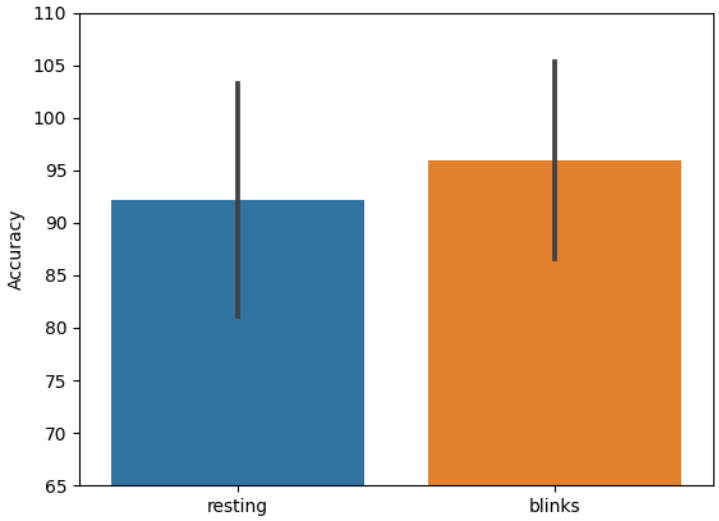

Figura 26 Diagrama de barras que representa la media del valor de *accuracy*, junto con una línea que muestra la desviación típica.

Condición	Accuracy
Resting	92.2 ± 11.0
Blink	96.0 ± 8.9

Tabla 8 Representa la *accuracy* calculada para las épocas de *resting* y parpadeos.

6.2.3. Complejidad temporal

Se obtuvieron los tiempos de cómputo representados en la Tabla 9.

Solapamiento	Tiempo
70 %	47.4 ms ± 1.25 ms
80 %	49 ms ± 1.64 ms
90 %	50.7 ms ± 0.96 ms

Tabla 9 Tiempos de cómputo calculados para solapamientos de 70, 80 y 90%

Capítulo 7: Discusión de resultados

7.1. Introducción

En la literatura científica se han documentado numerosos estudios relacionados con el procesamiento de señales de EEG y de estrategias para mitigar los artefactos que pueden manifestarse en estas señales. A pesar de esto, persiste la necesidad de desarrollar métodos con la capacidad de identificar automáticamente los artefactos con el fin de filtrarlos, tanto en aplicaciones *offline* como *online* [71]. En este contexto, se ha explorado la aplicación de la inteligencia artificial, concretamente, el uso de redes neuronales convolucionales, con el propósito de detectar segmentos de señales EEG que exhiben artefactos oculares. A diferencia de enfoques previos, la clasificación de estos artefactos se lleva a cabo prescindiendo de dispositivos adicionales o umbrales de voltaje. A pesar de que los umbrales pueden llegar a caracterizar los parpadeos, los movimientos oculares representan un desafío en términos de clasificación debido a su variabilidad, razón por la cual comúnmente se recurre al uso de EOG para su detección. El presente estudio se basa en el entrenamiento a partir de datos ya clasificados, proporcionados por Kobler *et al.* [36]. Se considera que el algoritmo propuesto logra una clasificación efectiva de estos eventos, marcando un avance significativo en la detección de artefactos oculares en señales EEG.

7.2. Discusión de resultados

En el presente TFG se introduce una metodología novedosa que permite distinguir entre tres condiciones: estado de reposo, parpadeo y movimiento horizontal de los ojos. Para el análisis de resultados se sigue un esquema similar al del capítulo anterior, primero se valorarán los resultados del modelo, posteriormente los del algoritmo sobre la base de datos de entrenamiento y finalmente el rendimiento del algoritmo sobre la base de datos registrada en el laboratorio.

7.2.1. Discusión de resultados del modelo

En el proceso de evaluación del modelo, se implementó una validación cruzada *k-fold* intersujeto. En ella destacaron los resultados obtenidos específicamente para la categoría de parpadeos, donde se lograron las métricas más altas para *recall*, *precisión* y *F1-score*. Este enfoque parece fundamentarse en la característica forma de onda que presenta esta clase de artefacto en la señal. Un análisis más detallado, centrado en la métrica de *recall*, revela que el modelo tiende a identificar mejor la mayoría de las ventanas que verdaderamente exhiben la condición de movimiento horizontal en comparación con las ventanas de *resting*. Esto indica que el modelo muestra mayor sensibilidad por la condición de movimiento de ojos que por la de reposo. Sin embargo, es importante destacar que, en cuanto a la precisión del modelo, es superior en la clasificación de ventanas clasificadas como *resting*. Esto podría sugerir que el modelo es más fiable en la detección de estados de reposo en comparación con su desempeño en la detección de

movimientos horizontales. Es importante resaltar que los resultados son adecuados, independientemente de las comparaciones.

7.2.2. Discusión de los resultados del algoritmo (validación)

El algoritmo diseñado fue evaluado comparando muestra a muestra las etiquetas predichas con las asignadas por el autor de la base de datos. Teniendo esto en cuenta, se observó respecto a las métricas del modelo una clara disminución del rendimiento en la clasificación de parpadeos, a pesar de que el *recall* se mantuviese elevado. Esto se debe a la forma del etiquetado que presenta el canal *artifactclasses*. Únicamente se consideraron parpadeos aquellos segmentos que superaron el umbral de 75 µV en la señal de VEOG por al menos 25 ms. Realizando el etiquetado de esta manera, se clasifican como reposo muestras a ambos extremos de los parpadeos que, a pesar de no superar este umbral, se encuentran influidas por este evento. Un ejemplo claro es la deflexión previa a los dos primeros parpadeos que se muestra en la Figura 27. En este sentido, el algoritmo propuesto realiza una clasificación más conservadora de los artefactos oculares, teniendo en cuenta toda la señal afectada por los mismos. Esto se puede apreciar en la Figura 27, en la que se ve como los segmentos inicial y final del parpadeo no se etiquetan como tal, sin embargo, el algoritmo si es capaz de capturar toda la señal afectada. En consecuencia, el algoritmo y la clasificación del canal *artifactclasses* difieren por un amplio margen, lo que justifica el bajo rendimiento reflejado en las métricas. El *recall*, sin embargo, se mantiene elevado, lo que indica que apenas existen falsos negativos, y por tanto, el algoritmo es capaz de capturar la inmensa mayoría de parpadeos. Esto es una condición indispensable para cualquier método de detección, que vaya a ser empleado en la eliminación de artefactos oculares.

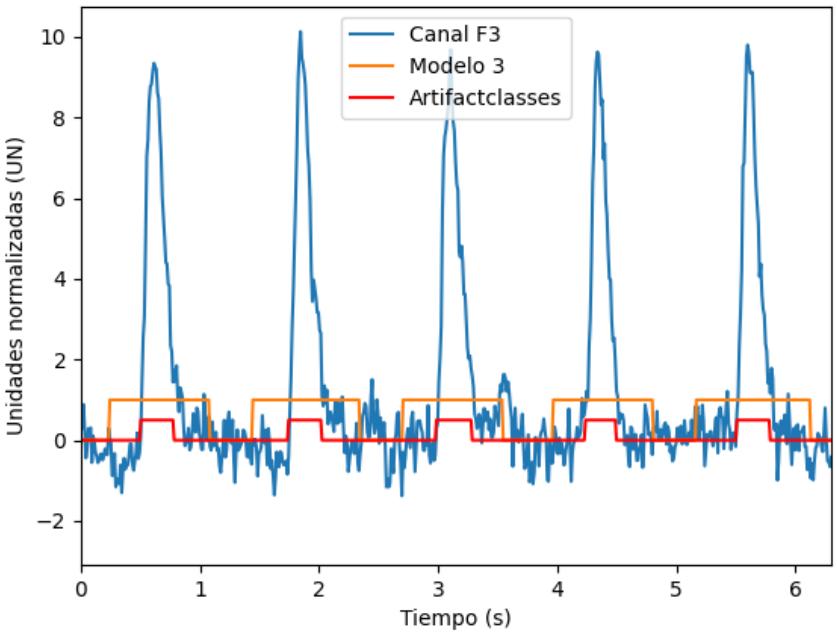

Figura 27 Ejemplo de un segmento de época del primer sujeto, con una duración de unos 6 s y con múltiples parpadeos. La señal representada en color azul es clasificada por las señales cuadradas naranja y roja, que representan el resultado del algoritmo (con el modelo 3) y el canal *artifactclasses,* respectivamente.

En el resto de las condiciones los resultados fueron muy buenos y se refuerzan las tendencias marcadas por el modelo. Para las épocas de *resting* se obtuvo una *precision* del 94.4%, que indica que el algoritmo realiza de forma excelente la tarea de identificación de épocas de *resting*.

En el caso de épocas de movimientos oculares, aparece un problema que hace que los resultados no sean del todo confiables. Las épocas contienen sucesivos movimientos oculares y están clasificadas en función de umbrales, lo que hace que, en ocasiones en los intervalos entre movimientos, la señal no sea clasificada como artefacto. Sin embargo, la señal basal o no clasificada de estas épocas no es del todo similar a la de *resting* y presenta variaciones que puede que la hagan más semejante a la actividad presente durante los movimientos oculares que a *resting*. Esto se puede apreciar en la Figura 28. En ella se muestra una época clasificada a tramos como movimiento de ojos. Sin embargo, nuestro algoritmo la ha considerado a todas las muestras como movimiento de ojos. Esto afecta claramente al accuracy, a la *precision* y en consecuencia al *F1-score*.

7.2.3. Discusión de los resultados del algoritmo (laboratorio)

A pesar de haberse realizado una validación cruzada bastante exigente, intersujeto y aplicando *k-fold* los resultados fueron satisfactorios. Esto permitió crear un modelo sin reservar registros para el grupo de test. Con la intención de evaluar este modelo en otros registros diferentes, se llevó a cabo la clasificación de registros adquiridos en el laboratorio. En esta base de datos las épocas no presentaban un etiquetado muestra a muestra. Tampoco presentaban épocas de movimientos de ojos. Todo esto se debe a la ausencia de un EOG para monitorizar la actividad ocular durante el registro de la prueba. Debido a la forma de onda tan característica que presenta el artefacto generado por los parpadeos, la señal pudo ser etiquetada de forma manual. Obteniendo un 96 ± 8.9% de accuracy en la predicción de estos eventos. Para las épocas de *resting* se obtuvo un 92.2 ± 11.0% de *accuracy*. Sin embargo, para ambas métricas los valores de desviación típica fueron elevados.

A pesar de no haberse realizado una evaluación del modelo final sobre una base de datos con épocas que presentaran movimiento de ojos, el algoritmo ha demostrado su fiabilidad a la hora de detectar correctamente épocas de *resting* y parpadeos. Los resultados obtenidos están al nivel de los algoritmos existentes en el estado del arte en cuanto a detección de parpadeos. Sin embargo, este algoritmo es capaz de detectar los movimientos oculares horizontales. Lo cual supone un gran salto respecto al resto de métodos y sienta las bases de una línea de investigación.

Este algoritmo puede ser empleado sin la necesidad de calibración previa sobre diferentes registros de EEG, independientemente del equipo con el que hayan sido registrados. A pesar de que este algoritmo no es capaz de reemplazar la señal de EOG en los métodos de regresión y de filtrado adaptativo, sí que reemplaza su función en los algoritmos de substracción de subespacios. Haciendo a esta técnica de filtrado tan interesante, mucho más accesible.

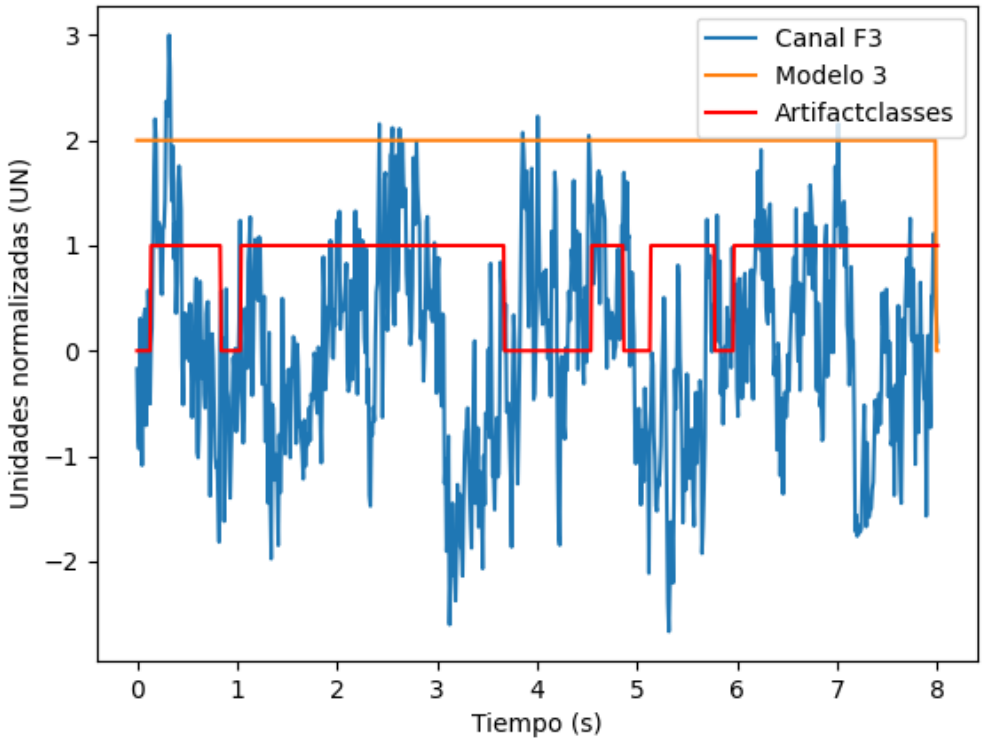

Figura 28 Se representa una época de movimientos oculares horizontales del primer sujeto. La señal representada en color azul es clasificada por las señales cuadradas naranja y roja, que representan el resultado del algoritmo y el canal *artifactclasses*, respectivamente.

7.2.4. Discusión del rendimiento en tiempo real

Las tareas de BCI requieren que los algoritmos implementados sean computacionalmente eficientes. Por ejemplo, las aplicaciones de *neurofeedback* requieren que el tiempo de cómputo no sea superior a los 350 ms [72]. El algoritmo desarrollado en el presente TFG ha demostrado tiempos de en torno a los 50 ms. El tiempo computacional del cálculo de métricas de análisis complejas para *neurofeedback* oscila alrededor de 140 ms [72], pero las más comunes tardan incluso menos. Esto permite determinar que el algoritmo podría aplicarse en la detección de artefactos, incluso ir acompañado de un método de substracción de subespacios, que son computacionalmente eficientes y de forma simultánea analizar la señal.

7.3. Limitaciones

A pesar de los adecuados resultados obtenidos en el presente estudio, es necesario señalar una serie de limitaciones asociadas:

1. La principal limitación que presenta el algoritmo es la incapacidad para detectar los movimientos oculares verticales. A pesar de que estos son los artefactos oculares que afectan en menor medida a la señal, su inclusión es esencial si se pretende crear un algoritmo que aspire a detectar todos los artefactos oculares.

2. El montaje empleado en el entrenamiento del modelo se escogió siguiendo los siguientes criterios: que los electrodos estuvieran presentes en todos los estudios, que fueran comúnmente empleados en tareas de BCI y que aportaran información relevante del artefacto. En consecuencia, para poder emplear el algoritmo el montaje empleado debe contener los electrodos seleccionados, lo cual limita su aplicabilidad.

3. La base de datos de Kobler *et al.* [36] contenía un número suficiente de registros, lo que permitió entrenar el modelo. Sin embargo, para la evaluación del algoritmo, la base de datos no presentaba un etiquetado óptimo. Lo cual afectó también a los resultados finales, tal y como se indica en la discusión.

4. Para normalizar la señal se requiere o bien una señal no demasiado saturada de parpadeos, o de épocas exentas de parpadeos y con actividad basal respecto a las cuales normalizar. En el caso de emplear este algoritmo sobre registros realizados siguiendo paradigmas como el de Kobler *et al.* [36], esto no supone un problema dado que se pueden normalizar respecto a las épocas de *resting*. Sin embargo, puede limitar la aplicabilidad del algoritmo en la detección de parpadeos *online*, si los valores de media y desviación típica se ven muy afectados.

Capítulo 8: Conclusiones

8.1. Introducción

A lo largo de este TFG, se han llevado a cabo una serie de tareas que han permitido el desarrollo del algoritmo. Primero, se realizó una investigación exhaustiva, analizando distintos algoritmos de detección y de filtrado, así como sus resultados. Esto permitió marcar unos objetivos claros, que incluían la detección de artefactos oculares de forma automática, en tiempo real y sin la necesidad de la señal adicional de electrooculografía (EOG), ni calibración previa. En este contexto se obtuvo una base de datos con artefactos oculares etiquetados. Esta base de datos fue preprocesada y enventanada, para posteriormente entrenar la red EEG-Inception. A partir del modelo creado se desarrolló el algoritmo final que demostró eficiencia y resultados adecuados. A continuación, se describen las aportaciones realizadas por este TFG a la comunidad científica y como el algoritmo contribuye a la innovación en esta área. Después, se expondrán las conclusiones extraídas de este trabajo y finalmente, se detallarán algunas líneas futuras.

8.2. Contribuciones

Es necesario resaltar las contribuciones que este TFG puede aportar a la comunidad científica en comparación con lo previamente hecho en el estado del arte:

1. Hay que destacar que el algoritmo desarrollado se distingue de los presentados en el estado del arte principalmente por su capacidad de detectar además de parpadeos, movimientos oculares horizontales.

2. El algoritmo permite simplificar la tarea de filtrado de los artefactos oculares, los más disruptivos y difíciles de evitar en aplicaciones de tiempo real [19]. Ya sea aplicándolo en la fase de calibración de métodos de substracción de subespacios o como clasificador en tiempo real de artefactos para ser filtrados.

3. Por otro lado, este algoritmo también permite simplificar las tareas de detección de artefactos manuales. La clasificación de parpadeos y, sobre todo, movimientos oculares, no siempre tan evidentes supone un gran avance y permite aligerar esta ardua tarea.

8.3. Conclusiones

A continuación, se presentan las conclusiones obtenidas a partir de este trabajo.

1. El algoritmo ha demostrado ser confiable en la clasificación de los parpadeos y los movimientos oculares horizontales, cumpliendo los objetivos marcados en un inicio.

2. A partir de la base de datos de Kobler et al. [36] se extrajeron un número adecuado de ventanas clasificadas. En total suman alrededor de 115 horas de registro. En la creación del modelo final no se incluyeron las ventanas asociadas a movimientos verticales, por lo que el total se redujo a 97 horas.

3. La realización de una validación cruzada exigente ha permitió determinar que el modelo era adecuado y que se podía emplear en la clasificación. La validación cruzada era intersujeto, lo cual permite aumentar la capacidad de generalización del modelo. Además, se realizó un *k-fold* que aporta una mayor robustez a los resultados y permite evitar el sesgo inherente a una única evaluación.

4. Los resultados obtenidos en la evaluación del algoritmo sobre otra base de datos han resultado satisfactorios. Además de lograr una alta precisión en la clasificación de parpadeos, este algoritmo también permite caracterizar los movimientos oculares horizontales. Algo que lo hace destacar frente a los algoritmos del estado del arte actual.

8.4. Líneas futuras

A continuación, se detallan algunas líneas futuras que se podrían implementar con la intención de perfeccionar el algoritmo y facilitar su uso y aplicación.

1. En un trabajo futuro, se podría plantear el uso de montajes alternativos con la intención de permitir detectar movimientos verticales. En consecuencia, se incluiría principalmente algún canal frontopolar. Por otro lado, introducir los electrodos F7 y F8 en el montaje, probablemente mejoraría la clasificación de los artefactos laterales.

2. Sería beneficioso incrementar el volumen de datos. Con la intención de entrenar el modelo con nuevos sujetos y equipos. Los modelos de DL tienden a mejorar su rendimiento a medida que aumenta la cantidad de datos y su diversidad. Por lo tanto, la adición de registros diversos contribuiría a una mejor generalización y rendimiento del algoritmo. Por otro lado, el diseño de una base de datos, enfocada a la evaluación de algoritmos similares aportaría unos resultados más fiables acerca del rendimiento del algoritmo.

3. La aplicación del algoritmo en tiempo real abre un gran abanico de posibilidades. Un ejemplo podría ser la aplicación de filtrado únicamente sobre segmentos que presenten artefactos. Esto es importante en aplicaciones de *neurofeedback* donde no interesa hacer un filtrado continuo, dado que esto puede suponer la perdida información neuronal.

4. En algunos métodos, concretamente los de substracción de subespacios se emplean unos minutos de registro previos al registro principal, para la calibración del algoritmo. Gracias a este algoritmo de detección, esta tarea de calibración dejaría de depender del uso de la señal de EOG. Haciendo este método más accesible y facilitando el desarrollo de nuevas técnicas de filtrado *online*.

5. Como tarea futura sería interesante desarrollar una función dentro de MEDUSA kernel con la que aplicar el algoritmo desarrollado. Incluso llegar a desarrollar una aplicación que aplique conjuntamente el algoritmo de detección y el de substracción de subespacios de Kobler *et al.* [36] con el fin de filtrar artefactos *online*.

6. También sería interesante saber en qué características de la señal se ha fijado EEG-Inception a la hora de clasificar las ventanas. Para ello se podría emplear alguna metodología ya establecida, como SHAP.

Apéndice A: Métricas de evaluación

Algunas de las métricas más comunes utilizadas en la evaluación del desempeño de algoritmos de detección de artefactos son: *F1-score, accuracy, recall, precision y Area Under the Receiver Operator Curve* (AUC) [73]. A continuación, se describen las empleadas en el presente TFG. Siendo TP las siglas para *true positive*; TN, para *true negative*; FP, para *false positive* y FN, para *false negative*.

La *accuracy* es la relación entre las muestras clasificadas correctamente respecto al número total de muestras en el conjunto de datos.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

El *recall,* mide la proporción de ejemplos TP que el modelo ha identificado correctamente en comparación con el total de ejemplos positivos reales.

$$Recall = \frac{TP}{TP + FN}$$

Por otro lado, *precision* denota la proporción de ejemplos clasificados como positivos por el modelo que son TP.

$$Precision = \frac{TP}{TP + FP}$$

A partir de estas dos métricas se obtiene el *F1-score,* que se calcula como la media armónica de *precision* y *recall* y sigue la siguiente ecuación:

$$F1 = 2\ x\ \frac{precision\ x\ recall}{precision + recall}$$

La *balance accuracy* sirve para abordar el problema de falta de balance entre las clases al calcular un promedio ponderado del *recall* de cada clase, la fórmula para calcularla es la siguiente:

$$Balance\ Accuracy = \sum_{i=0}^{n} \frac{TP_i}{TP_i + FP_i}$$

Apéndice B: Glosario de siglas y acrónimos

ADC: *Analog-to-Digital Converter.*

ANN: *Artificial Neural Network.*

BCI: *Brain Computer Interface.*

CAR: *Common Average Reference.*

CNN: *Convolutional Neural Network.*

CRD: *Corneo-Retinal Dipole.*

DL: *Deep Learning.*

EEG: Electroencefalograma.

ECG: Electrocardiograma.

EOG: Electrooculograma.

EMG: Electromiograma.

ERP: *Event-Related Potential.*

FN: Falso Negativo.

FP: Falso Positivo.

GAN: *Generative Adversarial Network.*

HEOG: *Horizontal Electrooculogram.*

ICA: *Independent Component Analysis.*

MEG: Magnetoencefalograma.

MI: *Motor Imagery.*

ML: *Machine Learning.*

RLM: Regresión Logística Multinomial.

RNN: *Recurrent Neural Network.*

SNR: *Signal-Noise Ratio.*

TDAH: Trastorno por Déficit de Atención con Hiperactividad.

TP: *True Positive.*

TN: *True Negative.*

VEOG: *Vertical Electrooculogram.*

Apéndice C: Principales funciones de código

Preprocesado

Funciones

```
#%%
def filtrado_remuestreado(registro, fs, filtro):
    """ Filtrado y remuestreado de la señal

        Parametros
        ----------
         registro : numpy.ndarray
            Épocas de señal de EEG. [n_épocas x n_canales x
n_muestras]
        fs : float
            Frecuencia de muestreo de la señal.
        filtro : medusa.frequency_filtering.IIRFilter
             Filtro IIR Butterworth.

        Devuelve
        -------
        registro : numpy.ndarray
            Registro remuestreado filtrado y remuestreado a 100 Hz.
[n_épocas x n_canales x n_muestras]
        """
    if fs == 200:
        artefactos = registro[:, -1, 0::2]
        filtro.fit(fs)
        # Le aplicamos el filtro y la trasponemos para poder aplicarla
el resample de medusa
        registro = filtro.transform(registro).transpose(0, 2, 1)
        # resampleamos a 100 Hz y trasponemos para recuperar [n_epocs
x n_canales x n_muestras]
        registro = medusa.epoching.resample_epochs(registro, [0,
8000], 100).transpose(0, 2, 1)
        # Recuperamos el canal de artefactos
        registro[:, -1, :] = artefactos
    elif fs == 256:
        filtro.fit(fs)
        # Le aplicamos el filtro y la trasponemos para poder aplicarla
el resample de medusa
        registro = filtro.transform(registro).transpose(0, 2, 1)
        # resampleamos a 100 Hz y trasponemos para recuperar [n_epocs
x n_canales x n_muestras]
        registro = medusa.epoching.resample_epochs(registro, [0,
8000],100).transpose(0, 2, 1)
        artefactos = np.around(registro[:, -1, :], 6)
        registro[:, -1, :] = artefactos
    elif fs == 100:
        artefactos = registro[:, -1, :]
        filtro.fit(fs)
        # Le aplicamos el filtro y la trasponemos para poder aplicarla
el resample de medusa
        registro = filtro.transform(registro)
        registro[:, -1, :] = artefactos
    return registro
```

```
def extraer(EEG: object, registro, artefacto: object = 'blink') ->
object:
    """ Encuentra las épocas dentro del registro que cumplan la
condición seleccionada

        Parametros
        ----------
        EEG : mne.io.eeglab.eeglab.EpochsEEGLAB
            Las épocas junto con información adicional de la señal.
        registro : numpy.ndarray
            Épocas de señal de EEG. [n_épocas x n_canales x
n_muestras]
        artefacto : str
            Artefacto seleccionado. ['blink', 'fixation',
'horizontal', 'vertical']

        Devuelve
        -------
        registro : numpy.ndarray
            registro con únicamente épocas de la condición
seleccionada. [n_épocas x n_canales x n_muestras]
        """
    event_id = EEG.event_id
    if artefacto == 'blink':
        id_parpadeos = []
        id_blink = []
        for i, j in enumerate(event_id):
            separacion = j.split(sep='/')
            if int(separacion[0]) == 4:
                id_parpadeos.append(event_id[j])

        for i, j in enumerate(EEG.events[:, 2]):
            if j in id_parpadeos:
                id_blink.append(i)
        return registro[id_blink, :, :]
    elif artefacto == 'fixation':
        id_parpadeos = []
        id_blink = []
        for i, j in enumerate(event_id):
            separacion = j.split(sep='/')
            if int(separacion[0]) == 1:
                id_parpadeos.append(event_id[j])

        for i, j in enumerate(EEG.events[:, 2]):
            if j in id_parpadeos:
                id_blink.append(i)

    elif artefacto == 'horizontal':
        id_horizontal = []
        id_blink = []
        for i, j in enumerate(event_id):
            separacion = j.split(sep='/')
            if int(separacion[0]) == 2:
                id_horizontal.append(event_id[j])
        for i, j in enumerate(EEG.events[:, 2]):
            if j in id_horizontal:
                id_blink.append(i)

    elif artefacto == 'vertical':
        id_vertical = []
```

```
        id_blink = []
        for i, j in enumerate(event_id):
            separacion = j.split(sep='/')
            if int(separacion[0]) == 3:
                id_vertical.append(event_id[j])
        for i, j in enumerate(EEG.events[:, 2]):
            if j in id_vertical:
                id_blink.append(i)

    return registro[id_blink, :, :]
```

```
def normalize_z_score(registro, artefacto = 'blink'):
    """ Normaliza la señal de forma acorde a la condición de la época

        Parametros
        ----------
        registro : numpy.ndarray
            Épocas de señal de EEG. [n_épocas x n_canales x
n_muestras]
        artefacto : str
            Artefacto seleccionado. ['blink', 'fixation',
'horizontal', 'vertical']
        Devuelve
        -------
        norm : numpy.ndarray
            Registro normalizado. [n_épocas x n_canales x n_muestras]
        """
    if artefacto == 'fixation':
        mean = np.mean(registro, 2)
        std = np.std(registro, 2)
        normalized_registro = (registro[:] - mean[:, :, np.newaxis]) /
std[:, :, np.newaxis]
        return normalized_registro
    elif artefacto == 'blink':
        norm = np.zeros(registro.shape)
        for i in np.arange(registro.shape[0]):
            indices_zeros = np.where(registro[i, -1, :] == 0)[0]
            rest = registro[i, :, indices_zeros]
            mean = np.mean(rest, 0)
            std = np.std(rest, 0)
            std[-1] = 1
            norm[i, :, :] = (registro[i, :, :] - mean[:, np.newaxis])
/ std[:, np.newaxis]
        return norm
    elif artefacto == 'horizontal':
        norm = np.zeros(registro.shape)
        for i in np.arange(registro.shape[0]):
            indices_zeros = np.where(registro[i, -1, :] != 0)[0]
            rest = registro[i, :, indices_zeros]
            mean = np.mean(rest, 0)
            std = np.std(rest, 0)
            std[-1] = 1
            norm[i, :, :] = (registro[i, :, :] - mean[:, np.newaxis])
/ std[:, np.newaxis]
        return norm
    elif artefacto == 'vertical':
        norm = np.zeros(registro.shape)
        for i in np.arange(registro.shape[0]):
            indices_zeros = np.where(registro[i, -1, :] != 0)[0]
            rest = registro[i, :, indices_zeros]
```

```
            mean = np.mean(rest, 0)
            std = np.std(rest, 0)
            std[-1] = 1
            norm[i, :, :] = (registro[i, :, :] - mean[:, np.newaxis])
/ std[:, np.newaxis]
        return norm
```

```
def extraer_normalizar(EEG, registro, artefacto = 'blink'):
    """ Normaliza la señal de forma acorde a la condición de la época

            Parametros
            ----------
            EEG : mne.io.eeglab.eeglab.EpochsEEGLAB
                Las épocas junto con información adicional de la
señal.
            registro : numpy.ndarray
                Épocas de señal de EEG filtradas y remuestreadas.
[n_épocas x n_canales x n_muestras]
            artefacto : str
                Artefacto seleccionado ['blink', 'fixation',
'horizontal', 'vertical']
            Devuelve
            -------
            registro : numpy.ndarray
                Registros normalizados que contienen únicamente épocas
con la condición deseada.
                [n_épocas x n_canales x n_muestras]
            """
    if artefacto == 'blink':
        registro = extraer(EEG, registro)
        artefactos = registro[:, -1, :]
        registro = normalize_z_score(registro, artefacto = artefacto)
        registro[:, -1, :] = np.where(artefactos < 3*10**-6, 0, 0.5)
    elif artefacto == 'fixation':
        registro = extraer(EEG, registro, artefacto)
        artefactos = registro[:, -1, :]
        registro = normalize_z_score(registro, artefacto)
        registro[:, -1, :] = np.where(artefactos < 3*10**-6, 0, 0.6)
    elif artefacto == 'horizontal':
        registro = extraer(EEG, registro, artefacto)
        artefactos = registro[:, -1, :]
        registro = normalize_z_score(registro, artefacto)
        registro[:, -1, :] = np.where(artefactos < 0.5*10**-6, 0, 0.1)
    elif artefacto == 'vertical':
        registro = extraer(EEG, registro, artefacto)
        artefactos = registro[:, -1, :]
        registro = normalize_z_score(registro, artefacto)
        registro[:, -1, :] = np.where(artefactos < 1.5*10**-6, 0, 0.1)
    return registro
```

```
def unir(r, distancia = 15):
    diff = np.diff(r[-1, :])
    inicios = np.where(diff > 0)[0]
    finales = np.where(diff < 0)[0]
    if r[-1, -1] > 0:
        finales = np.append(finales, 799)
    if r[-1, 0] == 0:  # unimos segementos separados menos de 15
muestras
        diferencias = np.where((inicios[1:] - finales[:-1]) <
```

```
distancia)[0]
        indices_finales = finales[diferencias]
        indices_inicios = inicios[diferencias + 1] + 1  # sumamos 1
para que quede con la muestra concreta
        for i, j in enumerate(indices_inicios):
            r[-1, indices_finales[i]:indices_inicios[i]] = 0.1
        diff = np.diff(r[-1, :])
        inicios = np.where(diff > 0)[0]
        finales = np.where(diff < 0)[0]

    elif r[-1, 0] > 0:  # unimos segementos separados menos de 15
muestras
        diferencias = np.where((inicios - finales[:-1]) <
distancia)[0]
        indices_finales = finales[diferencias]
        indices_inicios = inicios[diferencias] + 1  # sumamos 1 para
que quede con la muestra concreta
        for i, j in enumerate(indices_inicios):
            r[-1, indices_finales[i]:indices_inicios[i]] = 0.1
        diff = np.diff(r[-1, :])
        inicios = np.where(diff > 0)[0]
        finales = np.where(diff < 0)[0]
    if r[-1, -1] > 0:
        finales = np.append(finales, 799)
    return r, inicios, finales
```

```
def enventanado(registro, artefacto = 'blink', vent_size=int(0.5 *
100), sol= 50 / 100):
    """ Enventana la señal

        Parametros
        ----------
        registro : numpy.ndarray
            Épocas de señal de EEG. [n_épocas x n_canales x
n_muestras]
        artefacto : str
            Artefacto seleccionado ['blink', 'fixation', 'horizontal',
'vertical']
        vent_size : int
            Tamaño de ventana en muestras.
        sol : float
            Solapamiento entre las ventanas.
        Devuelve
        -------
        enventanado : numpy.ndarray
            Señal dividida en ventanas. [n_ventanas x n_canales x
n_muestras]
        """
    if artefacto == 'blink':
        cont = 0
        n_chan = registro.shape[1]
        solapamiento = int(vent_size * 0.5)
        enventanado = np.full((1000, n_chan, vent_size), np.nan)
        for y in np.arange(registro.shape[0]):
            r = registro[y, :, :]
            peaks = scipy.signal.find_peaks(r[-1, :])
            for i, j in enumerate(peaks[0]):
                if registro.shape[2] - j >= vent_size and j >
vent_size:
                    for x in np.arange(3):
```

```
                        enventanado[cont, :, :] = r[np.newaxis, :,
                                                    j + x * solapamiento
- vent_size:j + x * solapamiento]
                        cont += 1
                elif vent_size > registro.shape[2] - j >= solapamiento
and j >= vent_size:
                    for x in np.arange(2):
                        enventanado[cont, :, :] = r[np.newaxis, :,
                                                    j + x * solapamiento
- vent_size:j + x * solapamiento]
                        cont += 1
                elif registro.shape[2] - j < solapamiento and j >=
vent_size:
                    enventanado[cont, :, :] = r[np.newaxis, :, j -
vent_size:j]
                    cont += 1
                elif solapamiento < j <= vent_size:
                    for x in range(1,3):
                        enventanado[cont, :, :] = r[np.newaxis, :,
                                                    j + x * solapamiento
- vent_size:j + x * solapamiento]
                        cont += 1
                elif j <= solapamiento:
                    for x in range(2,3):
                        enventanado[cont, :, :] = r[np.newaxis, :,
                                                    j + x * solapamiento
- vent_size:j + x * solapamiento]
                        cont += 1

        filas_con_nan = np.isnan(enventanado[:, 0, 0])
        enventanado = enventanado[~filas_con_nan, :, :]

    elif artefacto == 'fixation':
        n_chan = registro.shape[1]
        enventanado = np.full((3000, n_chan, vent_size), np.nan)
        last_coc = 0
        paso = int(round(vent_size * (1 - sol), 6))
        for y in np.arange(registro.shape[0]):
            r = registro[y, :, :]
            diff = np.diff(r[-1, :])
            a = np.where(abs(diff) > 0)[0]
            cocientes = [0]
            if len(a) == 0:
                n_ventanas = int((registro.shape[2] - paso -
vent_size) / paso) + 1  # Eliminamos las primeras 25 muestras
                enventanado[last_coc:last_coc + n_ventanas, :, :] =
medusa.epoching.get_epochs(
                    r[:, paso:].transpose(1, 0), vent_size,
stride=paso).transpose(0, 2, 1)
                last_coc += n_ventanas
            else:
                inicios = np.where(diff > 0)[0]
                finales = np.where(diff < 0)[0]
                if r[-1, -1] > 0:
                    finales = np.append(finales, 799)
                if r[-1, 0] > 0:
                    cociente, residuo = np.divmod(finales[0],
vent_size)
                    enventanado[last_coc:last_coc + cociente, :, :] =
r[:,
```

```
residuo:cociente * vent_size + residuo].reshape(
                        n_chan, cociente, vent_size).transpose(1,0,2)
                    cocientes[0] = cociente
                    for i, j in enumerate(inicios):
                        distancia = finales[i + 1] - j
                        cociente, residuo = np.divmod(distancia,
vent_size)
                        enventanado[last_coc + cocientes[-1]:last_coc
+ cocientes[-1] + cociente, :, :] = r[:,

j:j + cociente * vent_size].reshape(
                        n_chan, cociente, vent_size).transpose(1,0,2)
                        cocientes.append(cociente + cocientes[-1])
                last_coc += cocientes[-1]
            if r[-1, 0] == 0:
                for i, j in enumerate(inicios):
                    distancia = finales[i] - j
                    cociente, residuo = np.divmod(distancia,
vent_size)
                    enventanado[last_coc + cocientes[-1]:last_coc +
cocientes[-1] + cociente, :, :] = r[:,

j:j + cociente * vent_size].reshape(
                        n_chan, cociente, vent_size).transpose(1,0,2)
                    cocientes.append(cociente + cocientes[-1])
                last_coc += cocientes[-1]
        filas_con_nan = np.isnan(enventanado[:, 0, 0])
        enventanado = enventanado[~filas_con_nan, :, :]

    elif artefacto == 'vertical' or artefacto == 'horizontal':
        n_chan = registro.shape[1]
        paso = int(round(vent_size * (1 - sol), 6))
        enventanado = np.full((1000, n_chan, vent_size), np.nan)
        last_coc = 0
        for y in np.arange(registro.shape[0]):
            r = registro[y, :, :]
            r, inicios, finales = unir(r)
            if len(inicios) == 0:  # si hay artefacto toda la época
                n_ventanas = int(
                    (registro.shape[2] - 25 - vent_size) / paso) + 1
# Eliminamos las primeras 25 muestras
                enventanado[last_coc:last_coc + n_ventanas, :, :] =
medusa.epoching.get_epochs(r[:, 25:].transpose(1, 0), vent_size,

stride=paso).transpose(0, 2, 1)
                last_coc += n_ventanas
            elif r[-1, 0] > 0 and finales[0] > vent_size + 5:
                n_ventanas = int((finales[0] - 5 - vent_size) / paso)
+ 1  # Eliminamos las primeras 5 muestras
                if n_ventanas > 1:
                    enventanado[last_coc:last_coc + n_ventanas, :, :]
= medusa.epoching.get_epochs(
                        r[:, 5:finales[0]].transpose(1, 0),
vent_size,
                        stride=paso).transpose(0, 2, 1)
                else:
                    enventanado[last_coc] = r[np.newaxis, :,
5:vent_size + 5]
                last_coc += n_ventanas

                for i, j in enumerate(inicios):
```

```
                    distancia = finales[i + 1] - j
                    if distancia >= vent_size + paso:
                        n_ventanas = int((distancia - vent_size) /
paso) + 1
                        enventanado[last_coc:last_coc + n_ventanas, :,
:] = medusa.epoching.get_epochs(
                            r[:, j:finales[i + 1]].transpose(1, 0),
vent_size,
                            stride=paso).transpose(0, 2, 1)
                        last_coc += n_ventanas
                    elif distancia > vent_size * 0.7 and finales[i +
1] < 790:
                        punto_medio = int(distancia / 2)
                        punto_inicio = j + punto_medio - int(vent_size
/ 2)
                        enventanado[last_coc] = r[np.newaxis, :,
punto_inicio:punto_inicio + vent_size]
                        last_coc += 1

            elif r[-1, 0] == 0:
                for i, j in enumerate(inicios):
                    distancia = finales[i] - j
                    if enventanado.flags.writeable:

                        if distancia > vent_size + paso:
                            n_ventanas = int((distancia - vent_size) /
paso) + 1
                            enventanado[last_coc:last_coc +
n_ventanas, :, :] = medusa.epoching.get_epochs(r[:,
j:finales[i]].transpose(1, 0), vent_size, stride=paso).transpose(0, 2,
1)
                            last_coc += n_ventanas
                        elif distancia > vent_size * 0.7 and
finales[i] < 790 and inicios[i] > 10:
                            punto_medio = int(distancia / 2)
                            punto_inicio = j + punto_medio -
int(vent_size / 2)
                            enventanado[last_coc, :, :] =
r[np.newaxis, :, punto_inicio:punto_inicio + vent_size]
                            last_coc += 1
                    else:
                        print(y)

        filas_con_nan = np.isnan(enventanado[:, 0, 0])
        enventanado = enventanado[~filas_con_nan, :, :]

    return enventanado
```

```
def preprocesado(camino, vent_size, sol, estudios, filtro, canales,
canales8):
    """ Preprocesado completo de la base de datos de Kobler et al.

        Parametros
        ----------
        camino : str
            Camino hasta la carpeta en la que se han almacenado las
carpetas de los estudios 1, 2, 3, 4 y 5.
        vent_size : int
            Tamaño de ventana en muestras.
        sol : float
```

```
            Solapamiento entre las ventanas.
        estudios : list
            Lista que contiene el número de cada sujeto que participó
en el registro.
        filtro : medusa.frequency_filtering.IIRFilter
            Filtro IIR Butterworth.
        canales : list
            Lista que contiene los vectores que indican cuales son los
canales de EEG y el de artifactclasses.
        canales8 : list
            Lista que contiene los vectores que indican cuales son los
canales de EEG seleccionados.
        Devuelve
        -------
        EEG : dict
            Diccionario con los objetos
mne.io.eeglab.eeglab.EpochsEEGLAB de cada registro.
        EEG_class : dict
            Diccionario con los registros filtrados y remuestreados.
        Blinks : dict
            Diccionario con las épocas de parpadeos.
        Fixation : dict
            Diccionario con las épocas de resting.
        Horizontal : dict
            Diccionario con las épocas de movimientos horizontales.
        Vertical : dict
            Diccionario con las épocas de movimientos verticales.
        Enventanado : dict
            Diccionario con los registros enventanados y divididos en
función de su condición.
        """
    EEG = {}
    EEG_class = {}
    Blinks, Fixation, Horizontal, Vertical = {}, {}, {}, {}
    Enventanado = {'Fixation': {}, 'Blinks': {}, 'Horizontal': {},
'Vertical': {}}

    for j, z in enumerate(estudios):
        for i in z:
            try:
                path = camino + '/study0'+ str(j+1) +'/study0'+
str(j+1) +'_p0' + str(i) + '_prep.set'
                EEG['study0'+ str(j+1) +'_p0' + str(i)] =
mne.io.read_epochs_eeglab(path)
                EEG_class['study0'+ str(j+1) +'_p0' + str(i)] =
EEG['study0'+ str(j+1) +'_p0' + str(i)].get_data()[:, canales[j], :]
                EEG_class['study0'+ str(j+1) +'_p0' + str(i)] =
filtrado_remuestreado(EEG_class['study0'+ str(j+1) +'_p0' + str(i)],

EEG['study0'+ str(j+1) +'_p0' + str(i)].info['sfreq'], filtro)
                Blinks['study0'+ str(j+1) +'_p0' + str(i)] =
extraer_normalizar(EEG['study0'+ str(j+1) +'_p0' + str(i)],

EEG_class['study0'+ str(j+1) +'_p0' + str(i)], artefacto='blink')
                Fixation['study0'+ str(j+1) +'_p0' + str(i)] =
extraer_normalizar(EEG['study0'+ str(j+1) +'_p0' + str(i)],

EEG_class['study0'+ str(j+1) +'_p0' + str(i)],

artefacto='fixation')
                Horizontal['study0'+ str(j+1) +'_p0' + str(i)] =
```

```
extraer_normalizar(EEG['study0'+ str(j+1) +'_p0' + str(i)],

EEG_class['study0'+ str(j+1) +'_p0' + str(i)],

artefacto='horizontal')
                Vertical['study0'+ str(j+1) +'_p0' + str(i)] =
extraer_normalizar(EEG['study0'+ str(j+1) +'_p0' + str(i)],

EEG_class['study0'+ str(j+1) +'_p0' + str(i)],

artefacto='vertical')

                Enventanado['Blinks']['study0'+ str(j+1) +'_p0' +
str(i)] = enventanado(Blinks['study0'+ str(j+1) +'_p0' + str(i)],
'blink',

vent_size, sol)[:, canales8[j], :]
                Enventanado['Fixation']['study0'+ str(j+1) +'_p0' +
str(i)] = enventanado(Fixation['study0'+ str(j+1) +'_p0' + str(i)],

'fixation', vent_size, sol)[:, canales8[j], :]
                Enventanado['Horizontal']['study0'+ str(j+1) +'_p0' +
str(i)] = enventanado(Horizontal['study0'+ str(j+1) +'_p0' + str(i)],

'horizontal', vent_size, sol)[
                                                                :,

canales8[j], :]
                Enventanado['Vertical']['study0'+ str(j+1) +'_p0' +
str(i)] = enventanado(Vertical['study0'+ str(j+1) +'_p0' + str(i)],

'vertical', vent_size, sol)[:,

canales8[j], :]


            except FileNotFoundError:
                path = camino + '/study0' + str(j + 1) + '/study0' +
str(j + 1) + '_p' + str(i) + '_prep.set'
                EEG['study0' + str(j + 1) + '_p0' + str(i)] =
mne.io.read_epochs_eeglab(path)
                EEG_class['study0' + str(j + 1) + '_p0' + str(i)] =
EEG['study0' + str(j + 1) + '_p0' + str(
                    i)].get_data()[:, canales[j], :]
                EEG_class['study0' + str(j + 1) + '_p0' + str(i)] =
filtrado_remuestreado(
                    EEG_class['study0' + str(j + 1) + '_p0' + str(i)],
                    EEG['study0' + str(j + 1) + '_p0' +
str(i)].info['sfreq'], filtro)
                Blinks['study0' + str(j + 1) + '_p0' + str(i)] =
extraer_normalizar(
                    EEG['study0' + str(j + 1) + '_p0' + str(i)],
                    EEG_class['study0' + str(j + 1) + '_p0' + str(i)],
artefacto='blink')
                Fixation['study0' + str(j + 1) + '_p0' + str(i)] =
extraer_normalizar(
                    EEG['study0' + str(j + 1) + '_p0' + str(i)],
                    EEG_class['study0' + str(j + 1) + '_p0' + str(i)],
                    artefacto='fixation')
                Horizontal['study0' + str(j + 1) + '_p0' + str(i)] =
extraer_normalizar(
```

```
                    EEG['study0' + str(j + 1) + '_p0' + str(i)],
                    EEG_class['study0' + str(j + 1) + '_p0' + str(i)],
                    artefacto='horizontal')
                Vertical['study0' + str(j + 1) + '_p0' + str(i)] =
extraer_normalizar(
                    EEG['study0' + str(j + 1) + '_p0' + str(i)],
                    EEG_class['study0' + str(j + 1) + '_p0' + str(i)],
                    artefacto='vertical')

                Enventanado['Blinks']['study0' + str(j + 1) + '_p0' +
str(i)] = enventanado(
                    Blinks['study0' + str(j + 1) + '_p0' + str(i)],
'blink',
                    vent_size, sol)[:, canales8[j], :]
                Enventanado['Fixation']['study0' + str(j + 1) + '_p0'
+ str(i)] = enventanado(
                    Fixation['study0' + str(j + 1) + '_p0' + str(i)],
                    'fixation', vent_size, sol)[:, canales8[j], :]
                Enventanado['Horizontal']['study0' + str(j + 1) +
'_p0' + str(i)] = enventanado(
                    Horizontal['study0' + str(j + 1) + '_p0' +
str(i)],
                    'horizontal', vent_size, sol)[

:,

canales8[j], :]
                Enventanado['Vertical']['study0' + str(j + 1) + '_p0'
+ str(i)] = enventanado(
                    Vertical['study0' + str(j + 1) + '_p0' + str(i)],
                    'vertical', vent_size, sol)[:,
                                                canales8[j],
:]
    return EEG, EEG_class, Blinks, Fixation, Horizontal, Vertical,
Enventanado
```

Código de ejecución

```
# A continuación se crea una lista denominada estudios, cada uno de
sus elementos contiene un array con los números
# asignados a los sujetos que participaron en el.
num_1 = np.arange(1,6)
num_2 = np.concatenate(([2, 3], np.arange(6, 19)))
num_3 = np.concatenate(([3], np.arange(19, 28)))
num_4 = np.concatenate(([3, 8, 24], np.arange(28, 40)))
num_5 = np.concatenate(([6, 10, 24], np.arange(40, 51)))
estudios = [num_1, num_2, num_3, num_4, num_5]

# Filtro que vamos a usar para todos los estudios.
filtro = medusa.frequency_filtering.IIRFilter(3, 49, 'lowpass',
axis=2)

#Tamaño de ventana y solapamiento que se va a emplear
vent_size = 60
sol = 0.7

# Camino hasta la carpeta de los estudios.
camino = 'D:/kobler-blink-rejection'

# Descarte de canales innecesarios en cada uno de los estudios
chan1 = np.concatenate((np.arange(16), np.arange(17, 21),
```

```
np.arange(22, 27), np.arange(28, 31), np.arange(32, 40),
                    np.arange(41, 45), np.arange(46, 64), [82]))
chan2 = np.concatenate((np.arange(6, 70), [88]))
chan3 = np.concatenate((np.arange(6, 70), [85]))
chan4 = np.concatenate((np.arange(32), np.arange(33, 35),
np.arange(36, 40), np.arange(41, 64), [79]))
chan5 = np.concatenate((np.arange(58), [79]))
canales = [chan1, chan2, chan3, chan4, chan5]

# Canales que se van a utilizar son ['F3','C3', 'P3', 'Cz', 'Pz',
'F4', 'C4', 'P4'] en este orden
# Estos son los canales seleccionados una vez se han elimando los
canales de EOG.
n_canal1 = np.array([3, 12, 21, 13, 22, 5, 14, 23])
n_canal2 = np.array([7, 27, 47, 29, 49, 11, 31, 51])
n_canal3 = np.array([7, 27, 47, 29, 49, 11, 31, 51])
n_canal4 = np.array([3, 12, 23, 13, 24, 5, 14, 25])
n_canal5 = np.array([0, 23, 49, 25, 51, 4, 27, 53])
canales8 = [n_canal1, n_canal2, n_canal3, n_canal4, n_canal5]

# Preprocesado
EEG, EEG_class, Blinks, Fixation, Horizontal, Vertical, Enventanado =
preprocesado(camino, vent_size, sol, estudios, filtro, canales,
canales8)
```

Entrenamiento del modelo final

Funciones

```
def crear_matrices(diccionarios, keys_val, keys_test,
vent_size):
    """ Genera la matriz con la que se entrena la red EEG-
Inception

        Parametros
        ----------
        diccionarios : dict
            Diccionario que contiene los registros
enventanados y dividiso en función de su condición.
        keys_val : list
            Lista que contiene los registros que se
emplearán en validación
        keys_test : list
            Lista que contiene los registros que se
emplearán en test.
        vent_size : int
            Tamaño de ventana en muestras.

        Devuelve
        -------
        train_set : numpy.ndarray
            Matriz que contiene las ventanas seleccionadas
para entrenamiento. [n_ventanas x n_muestras x 8_canales x
1]
        dev_set : numpy.ndarray
```

```
            Matriz que contiene las ventanas seleccionadas
para validación. [n_ventanas x n_muestras x 8_canales x 1]
        test_set : numpy.ndarray
            Matriz que contiene las ventanas seleccionadas
para test. [n_ventanas x n_muestras x 8_canales x 1]
        lista_train : list
            Lista con los números de ventanas asociadas a
Fixation, BLinks y Horizontal en la matriz de
entrenamiento.
        lista_dev : list
            Lista con los números de ventanas asociadas a
Fixation, BLinks y Horizontal en la matriz de validación.
        lista_test : list
            Lista con los números de ventanas asociadas a
Fixation, BLinks y Horizontal en la matriz de test.

        """
    lista_train = []
    lista_dev = []
    lista_test = []
    cont_train = 0
    cont_dev = 0
    cont_test = 0
    train_set = np.full((60000, vent_size, 8, 1), np.nan)
    dev_set = np.full((60000, vent_size, 8, 1), np.nan)
    test_set = np.full((60000, vent_size, 8, 1), np.nan)
    for w in diccionarios:
        for i, j in enumerate(diccionarios[w]):
            if j not in keys_test and j not in keys_val:
                train_set[cont_train:cont_train +
diccionarios[w][j].shape[0], :, :, :] =
diccionarios[w][j].transpose(
                    0, 2, 1)[:, :, :, np.newaxis]
                cont_train += diccionarios[w][j].shape[0]
            elif j in keys_val:
                dev_set[cont_dev:cont_dev +
diccionarios[w][j].shape[0], :, :, :] =
diccionarios[w][j].transpose(
                    0, 2, 1)[:, :, :, np.newaxis]
                cont_dev += diccionarios[w][j].shape[0]
            elif j in keys_test:
                test_set[cont_test:cont_test +
diccionarios[w][j].shape[0], :, :, :] =
diccionarios[w][j].transpose(
                    0, 2, 1)[:, :, :, np.newaxis]
                cont_test += diccionarios[w][j].shape[0]
        #%% lista con 3 valores, n_venanas de rest, blinks
y horz
        lista_train.append(cont_train)
        lista_dev.append(cont_dev)
        lista_test.append(cont_test)
```

```
    filas_con_nan = np.isnan(train_set[:, 0, 0, 0])
    train_set = train_set[~filas_con_nan, :, :, :]
    filas_con_nan = np.isnan(dev_set[:, 0, 0, 0])
    dev_set = dev_set[~filas_con_nan, :, :, :]
    filas_con_nan = np.isnan(test_set[:, 0, 0, 0])
    test_set = test_set[~filas_con_nan, :, :, :]
    return train_set, dev_set, test_set, lista_train,
lista_dev, lista_test
```

Código de ejecución

```
# El 90% de las ventanas se emplean en entrenamiento y el
otro 10% en validación, pero se separan en sujetos enteros.
keys_val = ['study01_p01', 'study02_p02', 'study03_p03',
'study03_p019', 'study04_p03', 'study05_p06']
keys_test = []
train_set, dev_set, test_set, lista_train, lista_dev,
lista_test = crear_matrices(Env, keys_val, keys_test,
vent_size= vent_size)
# Valor 0 para rest, 1 para parpadeos, 2 para movimiento
horizontal
label_train = np.concatenate((np.zeros((lista_train[0])),
np.ones((lista_train[1]-lista_train[0])),
np.ones((lista_train[2]-lista_train[1]))+1))
label_dev = np.concatenate((np.zeros((lista_dev[0])),
np.ones((lista_dev[1]-lista_dev[0])),
np.ones((lista_dev[2]-lista_dev[1]))+1))
# Se barajea la base de datos.
indexes_train = np.arange(len(label_train))
indexes_train_permuted =
np.random.permutation(indexes_train)
perm_label_train = label_train[indexes_train_permuted]
perm_train = train_set[indexes_train_permuted]
indexes_dev = np.arange(len(label_dev))
indexes_dev_permuted = np.random.permutation(indexes_dev)
perm_label_dev = label_dev[indexes_dev_permuted]
perm_dev = dev_set[indexes_dev_permuted]
#%%  TRAINING
os.environ['TF_FORCE_GPU_ALLOW_GROWTH'] = 'true'

# Create model
model = EEGInception(
    input_time=vent_size*10, fs=100, ncha=8,
filters_per_branch=8,
    scales_time=(240, 120, 60), dropout_rate=0.25,
    activation='elu', n_classes=3, learning_rate=0.001)



# Callbac   ks
early_stopping = EarlyStopping(
```

```
    monitor='val_loss', min_delta=0.0001,
    mode='min', patience=10, verbose=1,
    restore_best_weights=True)

# Fit model
fit_hist = model.fit(perm_train,
                     perm_label_train,
                     validation_data=(perm_dev,
perm_label_dev),
                     epochs=500,
                     batch_size=1024,
                     callbacks=[early_stopping])
#%% Save
model.save('modelo_final')
```

Algoritmo

Funciones

```
def algoritmo(epochs, muestras_ventana, solapamiento,
path_model):
    """ Detecta temporalmente los artefactos oculares

            Parametros
            ----------
            epochs : numpy.ndarray
                Épocas de la señal de EEG. [n_épocas x
n_muestras x 8_canales]
            muestras_ventana : int
                Número de muestras por ventana que emplea
el clasificador.
            solapamiento : float
                Solapamiento entre las ventanas
            path_model : int
                Camino hasta el modelo.

            Devuelve
            -------
            labels : numpy.ndarray
                Vector que asocia los valores 0 a REST, 1 a
BLINK y 2 a HORZ [n_épocas x n_muestras]

            """
    n_epocas = epochs.shape[0]
    muestras_epoca = epochs.shape[2]
    paso = int(round(muestras_ventana * (1 -
solapamiento),6))
    n_ventanas = int((muestras_epoca - muestras_ventana) /
paso) + 1
    r = np.zeros((n_epocas, n_ventanas, 8,
muestras_ventana))
```

```
    model = models.load_model(path_model)
    labels_blinks = np.zeros((n_epocas,muestras_epoca))
    labels_horiz = np.zeros((n_epocas,muestras_epoca))
    labels_rest = np.zeros((n_epocas,muestras_epoca))
    for y in range(n_epocas):
        r[y, :, :, :] = epoching.get_epochs(epochs[y, :,
:].transpose(1, 0), muestras_ventana,

stride=paso).transpose(0, 2, 1)
        ventana = r[y, :, :, :, np.newaxis].transpose(0, 2,
1, 3)
        probs = model.predict(ventana)
        for i in range(n_ventanas):
            labels_blinks[y,i * paso:i * paso +
muestras_ventana] += np.ones((muestras_ventana))*probs[i,1]
            labels_horiz[y,i * paso:i * paso +
muestras_ventana] += np.ones((muestras_ventana))*probs[i,2]
            labels_rest[y, i * paso:i * paso +
muestras_ventana] += np.ones((muestras_ventana)) * probs[i,
0]


    labels = np.argmax([labels_rest, labels_blinks,
labels_horiz], axis=0)
    return labels
```

Bibliografía

[1] A. G. Nerlich, A. Zink, U. Szeimies, y H. G. Hagedorn, «Ancient Egyptian prosthesis of the big toe», *Lancet Lond. Engl.*, vol. 356, n.º 9248, pp. 2176-2179, dic. 2000, doi: 10.1016/S0140-6736(00)03507-8.

[2] J. D. Bronzino, Ed., *The biomedical engineering handbook*, 3rd ed. en The electrical engineering handbook series. Boca Raton: CRC/Taylor & Francis, 2006.

[3] E. Kaniusas, «Fundamentals of Biosignals», en *Biomedical Signals and Sensors I: Linking Physiological Phenomena and Biosignals*, E. Kaniusas, Ed., en Biological and Medical Physics, Biomedical Engineering. Berlin, Heidelberg: Springer, 2012, pp. 1-26. doi: 10.1007/978-3-642-24843-6_1.

[4] J. R. Wolpaw y E. W. Wolpaw, Eds., *Brain-computer interfaces: principles and practice*. Oxford ; New York: Oxford University Press, 2012.

[5] J. K. Perloff, *Physical examination of the heart and circulation*, 4th ed. Shelton, CT: People's Medical Pub. House, 2009.

[6] P. R. Hoskins, «Measurement of arterial blood flow by Doppler ultrasound», *Clin. Phys. Physiol. Meas.*, vol. 11, n.º 1, p. 1, feb. 1990, doi: 10.1088/0143-0815/11/1/001.

[7] M. Czosnyka, «Monitoring and interpretation of intracranial pressure», *J. Neurol. Neurosurg. Psychiatry*, vol. 75, n.º 6, pp. 813-821, jun. 2004, doi: 10.1136/jnnp.2003.033126.

[8] K. R. Jat, «Spirometry in children», *Prim. Care Respir. J.*, vol. 22, n.º 2, Art. n.º 2, jun. 2013, doi: 10.4104/pcrj.2013.00042.

[9] D. Castro, S. M. Patil, y M. Keenaghan, «Arterial Blood Gas», en *StatPearls*, Treasure Island (FL): StatPearls Publishing, 2023. Accedido: 18 de septiembre de 2023. [En línea]. Disponible en: http://www.ncbi.nlm.nih.gov/books/NBK536919/

[10] P. D. Mannheimer, J. R. Cascini, M. E. Fein, y S. L. Nierlich, «Wavelength selection for low-saturation pulse oximetry», *IEEE Trans. Biomed. Eng.*, vol. 44, n.º 3, pp. 148-158, mar. 1997, doi: 10.1109/10.554761.

[11] A. J. Salacinski, M. Alford, K. Drevets, S. Hart, y B. E. Hunt, «Validity and Reliability of a Glucometer Against Industry Reference Standards», *J. Diabetes Sci. Technol.*, vol. 8, n.º 1, pp. 95-99, ene. 2014, doi: 10.1177/1932296813514315.

[12] V. Pecoraro *et al.*, «The diagnostic accuracy of digital, infrared and mercury-in-glass thermometers in measuring body temperature: a systematic review and network meta-analysis», *Intern. Emerg. Med.*, vol. 16, n.º 4, pp. 1071-1083, jun. 2021, doi: 10.1007/s11739-020-02556-0.

[13] G. Buzsáki, C. A. Anastassiou, y C. Koch, «The origin of extracellular fields and currents — EEG, ECoG, LFP and spikes», *Nat. Rev. Neurosci.*, vol. 13, n.º 6, Art. n.º 6, jun. 2012, doi: 10.1038/nrn3241.

[14] P. Singh, S. D. Joshi, R. K. Patney, y K. Saha, «Fourier-Based Feature Extraction for Classification of EEG Signals Using EEG Rhythms», *Circuits Syst. Signal Process.*, vol. 35, n.º 10, pp. 3700-3715, oct. 2016, doi: 10.1007/s00034-015-0225-z.

[15] F. Bager, F. D. Shaw, A. Tavener, M. P. F. Loeffen, y C. E. Devine, «Comparison of EEG and ECoG for detecting cerebrocortical activity during slaughter of calves», *Meat Sci.*, vol. 27, n.º 3, pp. 211-225, ene. 1990, doi: 10.1016/0309-1740(90)90052-8.

[16] R. A. Ramadan y A. V. Vasilakos, «Brain computer interface: control signals review», *Neurocomputing*, vol. 223, pp. 26-44, feb. 2017, doi: 10.1016/j.neucom.2016.10.024.

[17] A. Schlögl, C. Keinrath, D. Zimmermann, R. Scherer, R. Leeb, y G. Pfurtscheller, «A fully automated correction method of EOG artifacts in EEG recordings», *Clin. Neurophysiol.*, vol. 118, n.º 1, pp. 98-104, ene. 2007, doi: 10.1016/j.clinph.2006.09.003.

[18] A. Jafarifarmand, M.-A. Badamchizadeh, S. Khanmohammadi, M. A. Nazari, y B. M. Tazehkand, «Real-time ocular artifacts removal of EEG data using a hybrid ICA-ANC approach», *Biomed. Signal Process. Control*, vol. 31, pp. 199-210, ene. 2017, doi: 10.1016/j.bspc.2016.08.006.

[19] M. Miao, W. Hu, B. Xu, J. Zhang, J. J. P. C. Rodrigues, y V. H. C. de Albuquerque, «Automated CCA-MWF Algorithm for Unsupervised Identification and Removal of EOG Artifacts From EEG», *IEEE J. Biomed. Health Inform.*, vol. 26, n.º 8, pp. 3607-3617, ago. 2022, doi: 10.1109/JBHI.2021.3131186.

[20] K. Najarian y R. Splinter, *Biomedical Signal and Image Processing*, 2.ª ed. Boca Raton: CRC Press, 2016. doi: 10.1201/b11978.

[21] A. J. Casson, M. Abdulaal, M. Dulabh, S. Kohli, S. Krachunov, y E. Trimble, «Electroencephalogram», en *Seamless Healthcare Monitoring: Advancements in Wearable, Attachable, and Invisible Devices*, T. Tamura y W. Chen, Eds., Cham: Springer International Publishing, 2018, pp. 45-81. doi: 10.1007/978-3-319-69362-0_2.

[22] L. Brown, J. van de Molengraft, R. F. Yazicioglu, T. Torfs, J. Penders, y C. Van Hoof, «A low-power, wireless, 8-channel EEG monitoring headset», en *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, ago. 2010, pp. 4197-4200. doi: 10.1109/IEMBS.2010.5627393.

[23] A. Estraneo *et al.*, «Standard EEG in diagnostic process of prolonged disorders of consciousness», *Clin. Neurophysiol.*, vol. 127, n.º 6, pp. 2379-2385, jun. 2016, doi: 10.1016/j.clinph.2016.03.021.

[24] G. F. Ayala, M. Dichter, R. J. Gumnit, H. Matsumoto, y W. A. Spencer, «Genesis of epileptic interictal spikes. New knowledge of cortical feedback systems suggests a neurophysiological explanation of brief paroxysms», *Brain Res.*, vol. 52, pp. 1-17, mar. 1973, doi: 10.1016/0006-8993(73)90647-1.

[25] I. Timofeev y S. Chauvette, «Sleep slow oscillation and plasticity», *Curr. Opin. Neurobiol.*, vol. 44, pp. 116-126, jun. 2017, doi: 10.1016/j.conb.2017.03.019.

[26] W. F. Haupt y J. Rudolf, «European brain death codes: a comparison of national guidelines», *J. Neurol.*, vol. 246, n.º 6, pp. 432-437, jun. 1999, doi: 10.1007/s004150050378.

[27] J. Muthuswamy y N. V. Thakor, «Spectral analysis methods for neurological signals», *J. Neurosci. Methods*, vol. 83, n.º 1, pp. 1-14, ago. 1998, doi: 10.1016/S0165-0270(98)00065-X.

[28] S. Devuyst, T. Dutoit, P. Stenuit, y M. Kerkhofs, «Automatic K-complexes detection in sleep EEG recordings using likelihood thresholds», en *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, ago. 2010, pp. 4658-4661. doi: 10.1109/IEMBS.2010.5626447.

[29] D. H. Nieman, J. H. T. M. Koelman, D. H. Linszen, L. J. Bour, P. M. Dingemans, y B. W. Ongerboer de Visser, «Clinical and neuropsychological correlates of the P300 in schizophrenia», *Schizophr. Res.*, vol. 55, n.º 1, pp. 105-113, may 2002, doi: 10.1016/S0920-9964(01)00184-0.

[30] T. Picton, O. Lins, y M. Scherg, «The recording and analysis of event-related potentials», *Handb. Neuropsychol*, vol. 10, ene. 1995.

[31] S. Sur y V. K. Sinha, «Event-related potential: An overview», *Ind. Psychiatry J.*, vol. 18, n.º 1, p. 70, jun. 2009, doi: 10.4103/0972-6748.57865.

[32] V. Jurcak, D. Tsuzuki, y I. Dan, «10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems», *NeuroImage*, vol. 34, n.º 4, pp. 1600-1611, feb. 2007, doi: 10.1016/j.neuroimage.2006.09.024.

[33] V. Asadpour, Ed., *Brain-Computer Interface*, vol. 9. en Artificial Intelligence, vol. 9. IntechOpen, 2022. doi: 10.5772/intechopen.94618.

[34] X. Jiang, G.-B. Bian, y Z. Tian, «Removal of Artifacts from EEG Signals: A Review», *Sensors*, vol. 19, n.º 5, p. 987, feb. 2019, doi: 10.3390/s19050987.

[35] O. G. Lins, T. W. Picton, P. Berg, y M. Scherg, «Ocular artifacts in EEG and event-related potentials I: Scalp topography», *Brain Topogr.*, vol. 6, n.º 1, pp. 51-63, sep. 1993, doi: 10.1007/BF01234127.

[36] R. J. Kobler, A. I. Sburlea, C. Lopes-Dias, A. Schwarz, M. Hirata, y G. R. Müller-Putz, «Corneo-retinal-dipole and eyelid-related eye artifacts can be corrected offline and online in electroencephalographic and magnetoencephalographic signals», *NeuroImage*, vol. 218, p. 117000, sep. 2020, doi: 10.1016/j.neuroimage.2020.117000.

[37] A. S. Keren, S. Yuval-Greenberg, y L. Y. Deouell, «Saccadic spike potentials in gamma-band EEG: Characterization, detection and suppression», *NeuroImage*, vol. 49, n.º 3, pp. 2248-2263, feb. 2010, doi: 10.1016/j.neuroimage.2009.10.057.

[38] P. Anderer, H. V. Semlitsch, B. Saletu, y M. J. Barbanoj, «Artifact processing in topographic mapping of electroencephalographie activity in neuropsychopharmacology», *Psychiatry Res. Neuroimaging*, vol. 45, n.º 2, pp. 79-93, ago. 1992, doi: 10.1016/0925-4927(92)90002-L.

[39] Y. Zhang, X. Zheng, W. Xu, y H. Liu, «RT-Blink: A Method Toward Real-Time Blink Detection From Single Frontal EEG Signal», *IEEE Sens. J.*, vol. 23, n.º 3, pp. 2794-2802, feb. 2023, doi: 10.1109/JSEN.2022.3232176.

[40] J. Heo, H. Yoon, y K. S. Park, «A Novel Wearable Forehead EOG Measurement System for Human Computer Interfaces», *Sensors*, vol. 17, n.º 7, Art. n.º 7, jul. 2017, doi: 10.3390/s17071485.

[41] A. G. Correa, E. Laciar, H. D. Patiño, y M. E. Valentinuzzi, «Artifact removal from EEG signals using adaptive filters in cascade», *J. Phys. Conf. Ser.*, vol. 90, n.º 1, p. 012081, nov. 2007, doi: 10.1088/1742-6596/90/1/012081.

[42] P. Berg y M. Scherg, «A multiple source approach to the correction of eye artifacts», *Electroencephalogr. Clin. Neurophysiol.*, vol. 90, n.º 3, pp. 229-241, mar. 1994, doi: 10.1016/0013-4694(94)90094-9.

[43] E. Santamaría-Vázquez, V. Martínez-Cagigal, F. Vaquerizo-Villar, y R. Hornero, «EEG-Inception: A Novel Deep Convolutional Neural Network for Assistive ERP-Based Brain-Computer Interfaces», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 28, n.º 12, pp. 2773-2782, dic. 2020, doi: 10.1109/TNSRE.2020.3048106.

[44] A. M. Ladda, F. Lebon, y M. Lotze, «Using motor imagery practice for improving motor performance – A review», *Brain Cogn.*, vol. 150, p. 105705, jun. 2021, doi: 10.1016/j.bandc.2021.105705.

[45] B. Parsons y J. Faubert, «Enhancing learning in a perceptual-cognitive training paradigm using EEG-neurofeedback», *Sci. Rep.*, vol. 11, n.º 1, Art. n.º 1, feb. 2021, doi: 10.1038/s41598-021-83456-x.

[46] M. Arns, H. Heinrich, y U. Strehl, «Evaluation of neurofeedback in ADHD: The long and winding road», *Biol. Psychol.*, vol. 95, pp. 108-115, ene. 2014, doi: 10.1016/j.biopsycho.2013.11.013.

[47] J. D. Kelleher, *Deep learning*. en The MIT press essential knowledge series. Cambridge, Massachusetts London: The MIT Press, 2019.

[48] Y. LeCun, Y. Bengio, y G. Hinton, «Deep learning», *Nature*, vol. 521, n.º 7553, Art. n.º 7553, may 2015, doi: 10.1038/nature14539.

[49] J. Zou, Y. Han, y S.-S. So, «Overview of Artificial Neural Networks», en *Artificial Neural Networks: Methods and Applications*, D. J. Livingstone, Ed., en Methods in Molecular Biology™. Totowa, NJ: Humana Press, 2009, pp. 14-22. doi: 10.1007/978-1-60327-101-1_2.

[50] M. Hajinoroozi, Z. Mao, T.-P. Jung, C.-T. Lin, y Y. Huang, «EEG-based prediction of driver's cognitive performance by deep convolutional neural network», *Signal Process. Image Commun.*, vol. 47, pp. 549-555, sep. 2016, doi: 10.1016/j.image.2016.05.018.

[51] C. Li, X. Huang, R. Song, R. Qian, X. Liu, y X. Chen, «EEG-based seizure prediction via Transformer guided CNN», *Measurement*, vol. 203, p. 111948, nov. 2022, doi: 10.1016/j.measurement.2022.111948.

[52] Y. Lecun, L. Bottou, Y. Bengio, y P. Haffner, «Gradient-based learning applied to document recognition», *Proc. IEEE*, vol. 86, n.º 11, pp. 2278-2324, nov. 1998, doi: 10.1109/5.726791.

[53] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks», en *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2012. Accedido: 7 de julio de 2023. [En línea]. Disponible en: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html

[54] K. Simonyan y A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition». arXiv, 10 de abril de 2015. doi: 10.48550/arXiv.1409.1556.

[55] A. Bosch Rué, J. Casas-Roma, y T. Lozano Bagén, *Deep learning: principios y fundamentos*, Primera edición digital. Barcelona: Editorial UOC, 2020.

[56] A. Salami, J. Andreu-Perez, y H. Gillmeister, «EEG-ITNet: An Explainable Inception Temporal Convolutional Network for Motor Imagery Classification», *IEEE Access*, vol. 10, pp. 36672-36685, 2022, doi: 10.1109/ACCESS.2022.3161489.

[57] R. Verleger, «Valid identification of blink artefacts: are they larger than 50 μV in EEG records?», *Electroencephalogr. Clin. Neurophysiol.*, vol. 87, n.º 6, pp. 354-363, dic. 1993, doi: 10.1016/0013-4694(93)90148-O.

[58] E. Shachar, A. Lev, y O. Rosen, «MED: Muse™-based Eye-blink Detection Algorithm Using a Single EEG Channel», en *2022 IEEE Signal Processing in Medicine and Biology Symposium (SPMB)*, dic. 2022, pp. 1-5. doi: 10.1109/SPMB55497.2022.10014708.

[59] A. Egambaram, N. Badruddin, V. S. Asirvadam, E. Fauvet, C. Stolz, y T. Begum, «Unsupervised Eye Blink Artifact Identification in Electroencephalogram», en *TENCON 2018 - 2018 IEEE Region 10 Conference*, oct. 2018, pp. 2148-2152. doi: 10.1109/TENCON.2018.8650467.

[60] H. Nolan, R. Whelan, y R. B. Reilly, «FASTER: Fully Automated Statistical Thresholding for EEG artifact Rejection», *J. Neurosci. Methods*, vol. 192, n.º 1, pp. 152-162, sep. 2010, doi: 10.1016/j.jneumeth.2010.07.015.

[61] W.-D. Chang y C.-H. Im, «Enhanced Template Matching Using Dynamic Positional Warping for Identification of Specific Patterns in Electroencephalogram», *J. Appl. Math.*, vol. 2014, p. e528071, abr. 2014, doi: 10.1155/2014/528071.

[62] M. Lo Giudice *et al.*, «1D Convolutional Neural Network approach to classify voluntary eye blinks in EEG signals for BCI applications», en *2020 International Joint Conference on Neural Networks (IJCNN)*, jul. 2020, pp. 1-7. doi: 10.1109/IJCNN48605.2020.9207195.

[63] L. Frølich, T. S. Andersen, y M. Mørup, «Classification of independent components of EEG into multiple artifact classes», *Psychophysiology*, vol. 52, n.º 1, pp. 32-45, 2015, doi: 10.1111/psyp.12290.

[64] L. Pion-Tonachini, K. Kreutz-Delgado, y S. Makeig, «ICLabel: An automated electroencephalographic independent component classifier, dataset, and website», *NeuroImage*, vol. 198, pp. 181-197, sep. 2019, doi: 10.1016/j.neuroimage.2019.05.026.

[65] E. Santamaría-Vázquez *et al.*, «MEDUSA©: A novel Python-based software ecosystem to accelerate brain-computer interface and cognitive neuroscience research», *Comput. Methods Programs Biomed.*, vol. 230, p. 107357, mar. 2023, doi: 10.1016/j.cmpb.2023.107357.

[66] T. C. Ferree, «Spherical splines and average referencing in scalp electroencephalography», *Brain Topogr.*, vol. 19, n.º 1-2, pp. 43-52, 2006, doi: 10.1007/s10548-006-0011-0.

[67] R. Ghosh, N. Sinha, y S. K. Biswas, «Automated eye blink artefact removal from EEG using support vector machine and autoencoder», *IET Signal Process.*, vol. 13, n.º 2, pp. 141-148, 2019, doi: 10.1049/iet-spr.2018.5111.

[68] S. Rihana, P. Damien, y T. Moujaess, «EEG-Eye Blink Detection System for Brain Computer Interface», en *Converging Clinical and Engineering Research on Neurorehabilitation*, J. L. Pons, D. Torricelli, y M. Pajaro, Eds., en Biosystems & Biorobotics. Berlin, Heidelberg: Springer, 2013, pp. 603-608. doi: 10.1007/978-3-642-34546-3_98.

[69] S. Pérez-Velasco, E. Santamaría-Vázquez, V. Martínez-Cagigal, D. Marcos-Martínez, y R. Hornero, «EEGSym: Overcoming Inter-Subject Variability in Motor Imagery Based BCIs With Deep Learning», *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 30, pp. 1766-1775, 2022, doi: 10.1109/TNSRE.2022.3186442.

[70] T.-T. Wong, «Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation», *Pattern Recognit.*, vol. 48, n.º 9, pp. 2839-2846, sep. 2015, doi: 10.1016/j.patcog.2015.03.009.

[71] M. Agarwal y R. Sivakumar, «Blink: A Fully Automated Unsupervised Algorithm for Eye-Blink Detection in EEG Signals», en *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, sep. 2019, pp. 1113-1121. doi: 10.1109/ALLERTON.2019.8919795.

[72] D. Marcos-Martínez *et al.*, «ITACA: An open-source framework for Neurofeedback based on Brain–Computer Interfaces», *Comput. Biol. Med.*, vol. 160, p. 107011, jun. 2023, doi: 10.1016/j.compbiomed.2023.107011.

[73] S. Sadiya, T. Alhanai, y M. M. Ghassemi, «Artifact Detection and Correction in EEG data: A Review», en *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, may 2021, pp. 495-498. doi: 10.1109/NER49283.2021.9441341.