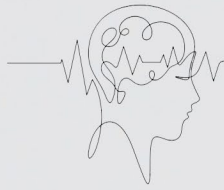
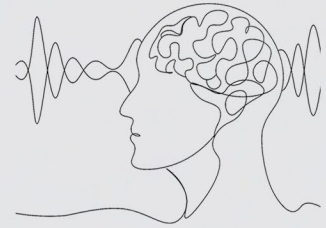


TESIS DOCTORAL



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

DOCTORADO EN EMPRESA, INTERNET Y TECNOLOGÍA DE LAS COMUNICACIONES
(EMITIC)



ANALYSIS OF MOTOR INTENTION THROUGH EEG SIGNAL PROCESSING AND ARTIFICIAL INTELLIGENCE MODELS

AUTOR

NABIL ISAAC AJALI HERNANDEZ

DIRECTOR

CARLOS MANUEL TRAVIESO GONZÁLEZ

NOVIEMBRE 2024,
LAS PALMAS DE G.C.



ULPGC
Universidad de
Las Palmas de
Gran Canaria



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA DE DOCTORADO

Programa de doctorado **Empresa, Internet y Tecnologías de las Comunicaciones (EmITIC)**

Título de la Tesis:

“Analysis of motor intention through EEG signal processing and artificial intelligence models”

Tesis Doctoral presentada por **Nabil Isaac Ajali Hernández**

Dirigida por el **Dr. Carlos Manuel Travieso González**

Las Palmas de Gran Canaria, a 13 de noviembre de 2024

El director,

Carlos Manuel Travieso González

El Doctorando,

Nabil Isaac Ajali Hernández

*“Analysis of Motor Intention Through EEG Signal Processing and
Artificial Intelligence Models”*

Nabil Isaac Ajali Hernández

November, 2024

Agradecimientos

A ti **mami**, por ser mi luz, quererme, apoyarme y haber hecho de mi lo que soy, sé
que estás orgullosa de tu niño

A mi **padre** por haberme cuidado y querido siempre de esa forma tan especial.
Por ser un guerrero y luchar por estar conmigo

A **Miriam** por confiar siempre en mí, ser mi apoyo incondicional, estar a mi lado y
quererme como nunca me han querido

A **Carmen y Vicente** por hacerme sentir uno más y cuidarme

A **mis amigos, amigas** que son mi verdadera familia. Sin ellos no soy quien soy.
(David, Andoni, Cachucho, Carlos, Mónica, Mario, Rami, Yoli, Álvaro, Rubén, Gus, Sofi)

A **Carlos Travieso** porque además de ser buen tutor e investigador es mejor
persona y siempre me ha dado el 110%, no he podido elegir mejor

A **todos los colaboradores** que he tenido por hacerme el camino más fácil.

A los que están y a los que se fueron

Por último, quiero agradecer a mi “**yo del pasado**”

por hacerme estar donde quiero estar. “

«Los problemas van a llegar, como los afrontes depende de ti»

Índice

1. Introducción	1
1.1. El Cerebro y el electroencefalograma	3
1.1.1. Estructura y funciones	4
1.1.2. Las neuronas	6
1.1.3. La señal y las ondas cerebrales	7
1.2. Inteligencia artificial y <i>Brain-Computer Interfaces</i>	9
1.2.1. Componentes Fundamentales de una <i>BCI</i> . Aplicaciones	11
1.3. Estado del arte	14
1.3.1. Bases de datos famosas	14
1.3.2. Mejores resultados	17
1.4. Hipótesis y objetivos	21
1.5. Novedad de la propuesta y contribuciones	21
1.6. Estructura de la memoria	25
2. Materiales y métodos	27
2.1. Base de datos	27
<i>Physionet</i>	27
Estructura de los Experimentos	27
Selección de Tareas y Configuración Experimental	29
Formato de los Datos	30
Lectura de la señal	30
2.2. Procesado de la señal	30
Tratamiento de datos EDF+ a vectores	31
Filtro de la señal	33
Tratamiento de datos EDF+ a matrices	35
2.3. Extracción de características	36
Transformada <i>Wavelet</i> Discreta	36
Parámetros estadísticos	38
Reducción de dimensionalidad	40
Análisis de Componentes Principales (<i>PCA</i>)	40
Análisis de Componentes Independientes (<i>ICA</i>)	41
2.4. Clasificadores	42
2.4.1. <i>Machine Learning</i>	43
Máquinas de Soporte Vectorial (<i>SVM</i>)	43
K-Nearest Neighbor (<i>k-NN</i>)	43
Árboles de Decisión	45

Regresión Logística	46
Análisis Discriminante Lineal (<i>LDA</i>).....	46
Perceptrón Multicapa (<i>MLP</i>)	47
Clasificador <i>Naive Bayes</i>	48
Ensamble (Métodos de Ensamble).....	50
2.4.2. <i>Deep Learning</i>	51
Redes Neuronales Convolucionales (<i>CNN</i>).....	51
Redes Neuronales Recurrentes (<i>RNN</i>)	53
Long Short-Term Memory (<i>LSTM</i>).....	54
Unidades <i>GRU</i>	57
<i>LSTM</i> Bidireccional (<i>BiLSTM</i>)	58
Capas de Atención (<i>AL</i>).....	59
<i>Transformers</i>	61
3. Metodología experimental	69
3.1. Esquema General.....	69
3.2. MATLAB. Arquitecturas propuestas	70
3.2.1. Transformada <i>Wavelet</i> Discreta	72
3.2.2. Parámetros estadísticos.....	74
3.2.3. <i>IPMS</i>	74
3.2.4. Clasificadores de <i>Machine Learning</i>	76
3.2.5. Cambio de paradigma. Matrices 2D	78
3.2.6. Clasificadores de <i>Deep Learning</i>	79
3.3. Migración a Python.....	84
3.4. Hiperparámetros y condiciones.....	91
3.5. Métricas	94
3.6. Librerías de Python	99
4. Resultados	105
4.1. <i>Machine Learning</i>	105
4.2. <i>Deep Learning</i>	117
5. Discusión.....	129
6. Conclusions and future works	149
6.1. Conclusions	149
6.2. Future research directions.....	152
7. Bibliografía.....	155
8. Anexos	169
8.1. Anexo I. Publicación capítulo libro <i>IntechOpen (Springer)</i>	170

8.2. Anexo II. Publicación capitulo libro <i>Sustainable Computing: Transforming Industry 4.0 to Society 5.0</i> , 31-47	173
8.3. Anexo III. Publicación <i>IRBM</i>	175
8.4. Anexo IV. Publicación bajo revisión <i>BCI. Scientific Reports</i>	176
8.5. Anexo V. Publicación <i>Desalination</i>	177
8.6. Anexo VI. Publicación <i>Scientific Reports COVID</i>	178
8.7. Anexo VII. Publicación colaboración en <i>Sensors</i>	179
8.8. Anexo VIII. Publicación bajo revisión <i>Scientific Reports</i> , clasificación de emociones	180
8.9. Anexo IX. Publicación bajo revisión en <i>IEEE</i> . Firma offline.....	181
8.10. Anexo X. Otras acciones de difusión.....	182
Anexo X.1. Publicación RTVC	182
Anexo X.2. Difusión en la web y redes sociales de la ULPGC	183
8.11. Anexo IX. Conclusiones y líneas futuras en español.....	184
Conclusiones	184
Líneas futuras.....	187

Índice de figuras

Figura 1.1. Cerebro y esquema de una neurona.	4
Figura 1.2. Disposición de las zonas más relevantes del encéfalo.	5
Figura 1.3. Vista detallada de una neurona y sus principales componentes.	7
Figura 1.4. Actividad eléctrica captada por un casco EEG fruto de la sinapsis neuronal.....	8
Figura 1.5. Esquema realista de un dispositivo <i>BCI</i> en funcionamiento.	10
Figura 2.1. Ejemplo gráfico de la estructura de los experimentos.	29
Figura 2.2. R003S001. Prueba 3 sujeto 1. Movimiento real de la mano izquierda y derecha visualizado en <i>EDF Browser</i>	31
Figura 2.3. Ejemplo de un experimento cualquiera entre las tareas 3 a 14..	32
Figura 2.4. Filtro <i>Butterworth</i> paso banda de diferentes órdenes.....	34
Figura 2.5. Esquema simplificado del funcionamiento del algoritmo generador de matrices 2D..	35
Figura 2.6. Descomposición de una señal en varios niveles utilizando la <i>DWT</i>	37
Figura 2.7. Esquema de reducción de la dimensionalidad aplicando <i>PCA</i>	41
Figura 2.8. Ejemplos de <i>SVM</i> con diferentes hiperplanos o <i>kernels</i>	43
Figura 2.9. Representación de una clasificación con algoritmo <i>K-NN</i> con $k=10$	44
Figura 2.10. Esquema del funcionamiento interno de un perceptrón con una sola capa.....	47
Figura 2.11. Aplicación del algoritmo de la <i>CNN</i> a una matriz de datos de 2 dimensiones..	53
Figura 2.12. Esquema desglosado de una red <i>RNN</i>	53
Figura 2.13. Esquema de una celda <i>LSTM</i>	56
Figura 2.14. Esquema de una capa de atención.	60
Figura 2.15. Esquema de la arquitectura de un <i>transformer</i> obtenida del trabajo de Nyandwi..	62
Figura 3.1. Diagrama de flujo general de la metodología experimental seguida en la tesis.	70
Figura 3.2. Fragmento del primer código desarrollado en <i>MATLAB</i> para tratar los datos en formato txt.....	71
Figura 3.3. Selección de canales utilizados. Zona del lóbulo frontal asociada a la intención motora.....	72
Figura 3.4. Fragmento del primer código desarrollado en <i>MATLAB</i> para tratar los datos en formato txt.	73
Figura 3.5. Explicación gráfica de la aplicación del <i>IPMS</i> . A cada evento T1 o T2 se le resta la media de su evento de descanso previo (T0).	75
Figura 3.6. Algoritmos de <i>Machine Learning</i> de la herramienta <i>Classification Learner</i> de <i>MATLAB</i> utilizados en la clasificación de la señal EEG.	77
Figura 3.7. Fragmento del código donde se implementa la arquitectura de la <i>CNN</i>	79

Figura 3.8. Fragmento del código donde se muestra la arquitectura de la <i>BiLSTM</i> . Válida para <i>LSTM</i> y <i>GRU</i> .	80
Figura 3.9. Fragmento del código donde puede observarse la parte de la arquitectura del Autoencoder arriba. Abajo la parte del <i>BiLSTM</i> .	81
Figura 3.10. Fragmento del código donde se implementa la arquitectura de la <i>CNN</i> .	82
Figura 3.11. Fragmento del código inicial donde se leen las señales y se establece el entorno de trabajo.	85
Figura 3.12. Fragmento del código donde se implementa la arquitectura de la <i>CNN-LSTM</i> en Python.	86
Figura 3.13. Esquema del proceso completo desde la toma de datos hasta la clasificación final utilizando la arquitectura híbrida <i>CNN-LSTM</i> .	87
Figura 3.14. Fragmento del código donde se define <i>la attention layer</i> y sus propiedades.	88
Figura 3.15. Fragmento del código donde se muestra la arquitectura del <i>vision transformer</i> .	89
Figura 3.16. Diagrama resumen de desarrollo de la tesis. Se muestran las metodologías seguidas y las decisiones tomadas.	90
Figura 3.17. Ejemplo de una matriz de confusión.	96
Figura 3.18. Línea de tiempo de las técnicas y los experimentos llevados a cabo para cumplir con la hipótesis y los objetivos.	103
Figura 4.1. Porcentajes de éxito en la clasificación de la intención motora de los 10 primeros sujetos.	106
Figura 4.2. Gráficas de los éxitos alcanzados en la clasificación independiente de sujeto con el subconjunto.	109
Figura 4.3. Gráficas de los mejores experimentos para cada sujeto y sus porcentajes de éxito.	110
Figura 4.4. Matriz de confusión de la primera clasificación utilizando <i>CNN-LSTM</i> .	117
Figura 4.5. Despliegue de la arquitectura <i>CNN-LSTM</i> e hiperparámetros en MATLAB. 79,64% de éxito en la clasificación.	118
Figura 4.6. Éxitos en las iteraciones de las clasificaciones y media total en rojo.	120
Figura 4.7. Matriz de confusión del experimento 4 de la tabla 4.9. <i>CNNx2-maxpooling CNN-attention-LSTMx2-attention</i> .	123
Figura 4.8. Matriz de confusión del experimento 5 de la tabla 4.9. <i>CNNx2-maxpooling-CNN-attention-LSTMx2</i> .	123
Figura 4.9. Matriz de confusión del experimento 6 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> .	124
Figura 4.10. Matriz de confusión del experimento 7 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> iteración adicional.	124
Figura 4.12. Matriz de confusión del experimento 9 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> doble entreno.	125

Figura 4.11. Matriz de confusión del experimento 8 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM epoch 100</i>	125
Figura 4.14. Matriz de confusión del experimento 11 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM batch (62) y epoch (150)</i>	126
Figura 4.13. Matriz de confusión del experimento 10 de la tabla 4.9. <i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM K FOLD 5</i>	126
Figura 4.15. Matriz de confusión del experimento utilizando <i>vision transformer</i> y una capa <i>LSTM</i> final para complementar el modelo.	127
Figura 4.16. Matriz de confusión del experimento utilizando el <i>vision transformer</i> de la figura 3.15..	128
Figura 4.17. Evolución de la función de pérdida en cada iteración (<i>epoch</i>).	128
Figura 5.1. Ejemplo de la no correlación entre eventos T1 y T2 aplicando AAC en el canal FC1 de los sujetos S001 y S002.	132

Índice de tablas

Tabla 1.1. Características de las ondas cerebrales básicas y frecuencia asociadas.	9
Tabla 1.2. Resumen del uso de las diferentes bases de datos en publicaciones sobre <i>BCI</i>	16
Tabla 1.3. Resumen del uso de las diferentes bases de datos en publicaciones sobre <i>BCI</i>	19
Tabla 2.1. Resumen de los diferentes experimentos contenidos en la base de datos <i>Physionet</i> . 28	
Tabla 4.1. Exactitud (éxito) en la clasificación de los 3 experimentos por sujeto de intención motora por usuario utilizando diferentes clasificadores.	108
Tabla 4.2. Clasificación de la intención motora utilizando diferentes clasificadores de <i>Machine Learning</i> y teniendo en cuenta la hipótesis del <i>IPMS</i>	111
Tabla 4.3. Clasificación de la intención motora utilizando diferentes clasificadores de <i>Deep Learning</i> , aplicando <i>DWT</i> y teniendo en cuenta la hipótesis del <i>IPMS</i>	113
Tabla 4.4. Resultados de la clasificación utilizando un ensamble compuesto por una capa <i>BiLSTM</i> y diferentes capas de clasificadores de <i>Machine Learning</i>	114
Tabla 4.5. Resultados de la clasificación del ensamble utilizando los experimentos 1 y 2 para entrenar y el experimento 3 para testear.	115
Tabla 4.6. Resultados de la clasificación con ensambles utilizando diferentes ratios. Los datos corresponden a experimentos de movimiento real en conjunto con los de intención motora. 116	
Tabla 4.7. Exactitud de la clasificación variando condiciones en la arquitectura previa.	119
Tabla 4.8. Resumen de las métricas de cada iteración en la clasificación de la señal de los 105 usuarios.	120
Tabla 4.9. Resumen de las métricas de cada iteración en la clasificación de la señal de los 105 usuarios.	122
Tabla 5.1. Porcentajes de éxito medios en la clasificación con y sin <i>IPMS</i> de 10 sujetos.	135
Tabla 5.2. Comparativa con otros trabajos del estado del arte que buscan la clasificación binaria de intenciones motoras utilizando EEG y un entrenamiento global generalizado.	144
Tabla 5.3. Comparativa con otros trabajos del estado del arte que buscan la clasificación binaria de intenciones motoras utilizando EEG y capas de atención.	147
Table 6.1. <i>Publications in the development of the doctoral thesis.</i>	151
Tabla 8.1. Publicaciones en el desarrollo de la tesis doctoral.	186

Acrónimos

Acronimo	Definición en inglés	Definición en español
AAC	<i>Average Amplitude Change</i>	Cambio de Amplitud Medio
Adam	<i>Adaptive Moment Estimation</i>	–
AI	<i>Artificial Intelligence</i>	Inteligencia Artificial
AL	<i>Attention Layers</i>	Capa de Atención
Avg	<i>Average</i>	Media aritmética
BiLSTM	<i>Bidirectional Long Short-Term Memory</i>	–
BCI	<i>Brain-Computer Interface</i>	Interfaz Cerebro-Ordenador
BPF	<i>Band-Pass Filter</i>	Filtro Paso-Banda
CNN	<i>Convolutional Neural Network</i>	Red Neuronal Convolutiva
CWT	<i>Continuous Wavelet Transform</i>	Transformada <i>Wavelet</i> Continua
DWT	<i>Discrete Wavelet Transform</i>	Transformada <i>Wavelet</i> Discreta
ECG	<i>Electrocardiogram</i>	Electrocardiograma
ECoG	<i>Electrocorticography</i>	Electrocorticografía
EDF	<i>European Data Format</i>	–
EEG	<i>Electroencephalography</i>	Electroencefalograma
EMG	<i>Electromyogram</i>	Electromiograma
ERP	<i>Event-Related Potential</i>	Potencial cerebral asociado a evento
FC	<i>Fully Connected</i>	Capa totalmente conectada
FN	<i>False Negative</i>	Falso Negativo
FP	<i>False Positive</i>	Falso Positivo
fNIRS	<i>Functional Near-Infrared Spectroscopy</i>	Espectroscopía Funcional por Infrarrojo Cercano
GPU	<i>Graphics Processing Unit</i>	Unidad de Procesado Gráfico
GRU	<i>Gated Recurrent Unit</i>	Unidades Recurrentes de Compuerta
ICA	<i>Independent Component Analysis</i>	Análisis de Componentes Independientes
IoT	<i>Internet of Things</i>	Internet de las Cosas
IPMS	<i>Immediate Previous Mental State</i>	Estado Mental Inmediatamente Previo
K-NN	<i>K-Nearest Neighbors</i>	–
KURT	<i>Kurtosis</i>	Curtosis
LDA	<i>Linear Discriminant Analysis</i>	Análisis Discriminante Lineal
LSTM	<i>Long Short-Term Memory</i>	–
MAV	<i>Mean Absolute Value</i>	Media Absoluta
MEG	<i>Magnetoencephalography</i>	Magnetoencefalografía
MLP	<i>Multi-Layer Perceptron</i>	Perceptrón Multi-Capa
MSE	<i>Mean Squared Error</i>	Error Cuadrático Medio
NLP	<i>Natural Language Processing</i>	Procesamiento del Lenguaje Natural

PCA	<i>Principal Component Analysis</i>	Análisis de Componentes Principales
RBM	<i>Restricted Boltzmann Machine</i>	Máquina de Boltzmann
RMS	<i>Root Mean Square</i>	Media Cuadrática
RNN	<i>Recurrent Neural Network</i>	Red Neuronal Recurrente
SKEW	<i>Skewness</i>	Oblicuidad
SMA	<i>Supplementary Motor Area</i>	Área Motora Suplementaria
SNC	<i>Central Nervous System</i>	Sistema Nervioso Central
SSI	<i>Signal Square Integral</i>	Suma del Cuadrado de la Señal
STD	<i>Standard Deviation</i>	Desviación Estándar
SVM	<i>Support Vector Machine</i>	Máquina Soporte Vector
TN	<i>True Positive</i>	Verdadero Positivo
TN	<i>True Negative</i>	Verdadero Negativo
VEP	<i>Visual Evoked Potential</i>	Potencial Evocado Visualmente
ViT	<i>Vision Transformer</i>	-
VAR	<i>Variance</i>	Varianza

1.Introducción

A lo largo del siglo XX, avances significativos en campos de la ciencia como la computación, la ingeniería y la medicina han destacado entre otros muchos por haber conducido a una actualidad en la que la tecnología está presente en el día a día de las personas y profundamente integrada y arraigada en la sociedad y las empresas. Siendo indispensable en diversos sectores como el de la tecnología de las señales y la comunicación (internet o radio) o en la integración del Internet de las Cosas (*IoT*), el cual permite escenarios tan futuristas como la domótica, conducción autónoma o el automatizado de procesos complejos en campos como el de la ingeniería o incluso la medicina (en cirugías asistidas por IA).

Este desarrollo se debe al aumento de la potencia de cálculo y la reducción de costos generada por la mejora de los procesadores y las Unidades Graficas de Procesado (*GPU's* por sus siglas en inglés). Esto ha democratizado el acceso a equipos de alto rendimiento, que ahora se utilizan en hogares, hospitales, instituciones y centros de investigación, generando y procesando grandes volúmenes de datos mediante técnicas y algoritmos avanzados de *Deep y Machine Learning*. Dando como resultado la generación de la *Big Data*. En definitiva, la sociedad se adentra en un paradigma donde la inteligencia artificial (IA), está presente en todos los ámbitos.

En este contexto, la presente tesis doctoral se enfoca en el análisis de señales de electroencefalograma (EEG) y su aplicación en interfaces cerebro-ordenador (*BCI* por sus siglas en inglés). El EEG, es una herramienta que registra la actividad eléctrica cerebral, la cual ha sido fundamental para comprender, estudiar el cerebro y poder diagnosticar patologías neurológicas desde su invención en 1924 por Hans Berger.

No obstante, su potencial va más allá del diagnóstico, extendiéndose a la investigación de la mente y de sus potenciales aplicaciones en áreas como la rehabilitación, el control mental de dispositivos físicos (como interruptores lógicos o brazos biónicos), o incluso el control de videojuegos mediante la señal cerebral.

La investigación en el campo de las interfaces cerebro-ordenador ha experimentado un crecimiento notable en las últimas décadas y actualmente diversos equipos de investigación, así como famosas empresas (*Neuralink*, de Elon Musk, *OpenAI* de Sam Altman o *Microsoft* y *Google*) están aprovechando este nuevo auge de la IA (con los *Transformers*, los algoritmos *GPT* y los modelos de transferencia), para tratar de desarrollar sistemas inteligentes que buscan optimizar la comunicación entre el cerebro y el ordenador mediante el monitoreo de la actividad cerebral y el uso de estos algoritmos. El objetivo final de esta línea de investigación es lograr una interpretación precisa de las señales cerebrales y su traducción para obtener un sistema fluido entre la mente y el ordenador, que conectado a diferentes dispositivos externos cumplirán un objetivo específico.

La importancia de esta línea radica en su potencial para mejorar no solo la calidad de vida de personas con discapacidades motoras o de comunicación, sino también en la capacidad para introducir en la sociedad una herramienta capaz de penetrar prácticamente en cualquier campo para mejorarlo y optimizarlo. También tiene el potencial para ayudar a personas con riesgo de exclusión social o para reintroducir a personas con accidentes motrices o cerebrovasculares a una nueva normalidad. Las *BCI* podrían permitirles interactuar con el mundo de una manera más natural e independiente, brindándoles nuevas oportunidades de participación y autonomía.

Por lo tanto, se espera un futuro donde las interfaces cerebro-ordenador tengan un impacto significativo en diversas áreas. En el ámbito médico, podrían utilizarse para el diagnóstico y tratamiento de enfermedades neurológicas, como la epilepsia o el Parkinson. En el campo de la rehabilitación, podrían ayudar a pacientes con lesiones cerebrales o medulares a recuperar funciones motoras perdidas. En la industria del entretenimiento permitiría nuevas formas de interacción con videojuegos y experiencias de realidad virtual y así con un ilimitado número de posibilidades.

Sin embargo, el desarrollo de interfaces cerebro-ordenador plantea desafíos importantes. Uno de ellos es la necesidad de mejorar la precisión y fiabilidad de la detección e interpretación de las señales cerebrales. La variabilidad individual en los patrones de EEG y la presencia de ruido e interferencias son obstáculos que deben

superarse. Además, es fundamental garantizar la seguridad y ética en el uso de estas tecnologías, protegiendo la privacidad y autonomía de los usuarios.

La presente tesis doctoral propone contribuir al avance del campo de las interfaces cerebro-ordenador mediante el desarrollo de nuevos algoritmos y técnicas de procesamiento de señales de EEG. Se exploran métodos de aprendizaje automático para mejorar la detección y clasificación de patrones cerebrales asociados a movimientos e intenciones motoras. Asimismo, se investigarán estrategias para optimizar la comunicación entre el cerebro y el ordenador, buscando una interacción más fluida y natural. Los resultados de esta investigación contribuyen al desarrollo de interfaces cerebro-ordenador más precisas, fiables y accesibles, que puedan mejorar la calidad de vida de personas con discapacidades y a abrir nuevas posibilidades en campos como la medicina, la rehabilitación y el entretenimiento.

1.1. El Cerebro y el electroencefalograma

El cerebro es el órgano rector del Sistema Nervioso Central (SNC), situado en el interior de la cavidad craneal, representa la parte más conocida y con mayor volumen del encéfalo. Presente en todos los seres vivos vertebrados se encuentra suspendido en líquido cefalorraquídeo, un líquido transparente cuyas funciones incluyen la protección del cerebro, tanto en el aspecto físico (golpes, movimientos) como en el inmunológico. [1, 2], ver figura 1.1. El cerebro desempeña un papel fundamental en la homeostasis del organismo y en la interacción del individuo con su entorno, siendo el órgano de mayor importancia entre la familia de los vertebrados.

Entre sus múltiples funciones destacan:

- Regulación de funciones vitales [2]
- Procesamiento de la información sensorial [3]
- Control motor [4]
- Regulación de emociones y conductas [2]
- Funciones cognitivas superiores [5]

En conjunto, estas funciones hacen del cerebro un órgano complejo del que se desconocen muchos aspectos y cuya comprensión es esencial para desentrañar los

misterios de la mente humana, desarrollar tratamientos efectivos para enfermedades neurológicas y psiquiátricas y modelar sistemas en donde pueda expresarse su potencial.

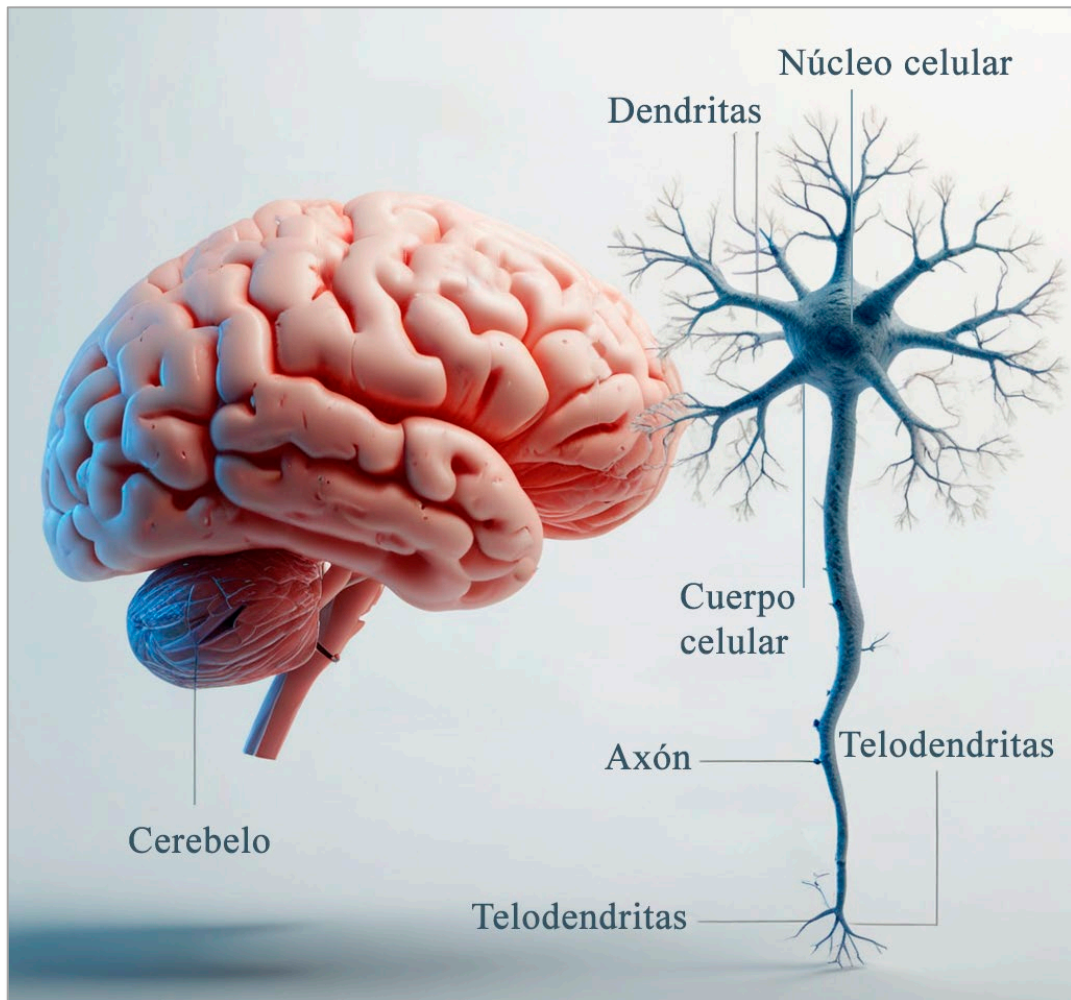


Figura 1.1. Cerebro y esquema de una neurona.

1.1.1. Estructura y funciones

El cerebro humano, presenta una estructura compleja compuesta por elementos corticales y subcorticales, ver figura 1.2. Se divide en dos hemisferios, derecho e izquierdo, interconectados por el cuerpo calloso (1). Bajo la corteza cerebral residen estructuras complejas como las estructuras subcorticales, el tálamo (2), los ganglios basales, la amígdala, el hipocampo y los cuerpos mamilares (6), las cuales permiten llevar a cabo las funciones superiores de los seres humanos, como el procesamiento de la información, la regulación de conductas, la abstracción, el ego o la consciencia [1, 2,

6]. El neurocirujano Jesús Martín-Fernández explica en [7] algunos de estos conceptos y profundiza en el funcionamiento del cerebro, en cómo se comportan las distintas partes de este. Trata temas como el alma, las conexiones neuronales y el futuro de este campo.

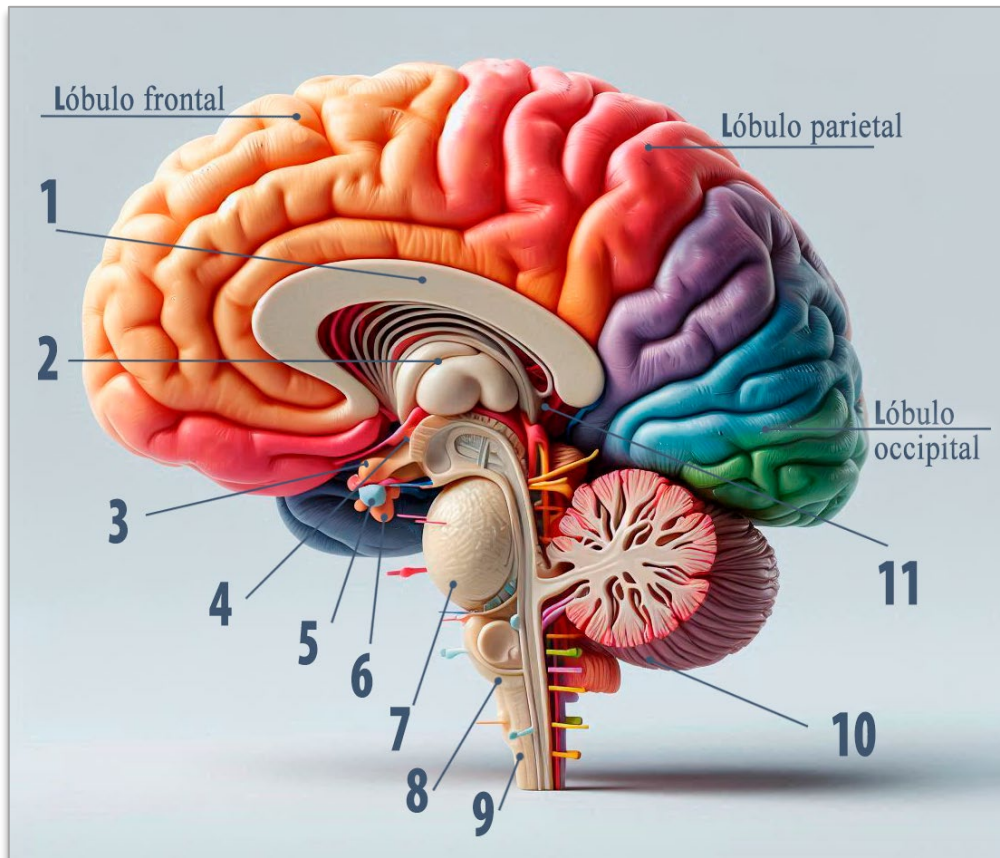


Figura 1.2. Disposición de las zonas más relevantes del encéfalo.

Otras de las partes más importantes del encéfalo son:

- **El cerebelo (10)** es la segunda estructura encefálica en tamaño. Destaca por su papel crucial en la coordinación motora [8].
- **El hipotálamo (4), la glándula pituitaria e hipófisis (5)** actúan como nexo entre los sistemas nervioso y endocrino. Su actividad es fundamental en la Termorregulación, regulación de conductas básicas (alimentación, respuesta sexual, agresividad) y comportamientos de alto grado como las emociones.
- **Glándula pineal (11) y quiasma óptico (3)**. Liberación de melatonina, hormona de regulación de los ciclos circadianos de sueño y vigilia.

- **El tronco encefálico** está compuesto por la médula espinal (9), el bulbo raquídeo (8), la protuberancia (7) y el mesencéfalo. Se encarga de las funciones vitales primitivas autónomas (presión arterial, frecuencia cardíaca, movimientos respiratorios, la micción y la digestión).

El control del movimiento y su respuesta imaginaria

El control del movimiento y su respuesta imaginaria, es decir, pensar en ese movimiento, son procesos complejos que involucran una red distribuida de áreas cerebrales. La corteza motora, los ganglios basales y el cerebelo son actores clave en la ejecución y coordinación de los movimientos, mientras que la corteza parietal posterior y la prefrontal contribuyen a la planificación y toma de decisiones. La imaginación del movimiento activa muchas de estas áreas, permitiendo simular y anticipar acciones en nuestra mente.

La zona del lóbulo frontal, concretamente, la corteza premotora y el área motora suplementaria (SMA por sus siglas en inglés) en especial, son particularmente importantes en la generación de estas imágenes motoras, ya que permiten visualizar y anticipar las consecuencias de las acciones antes de ejecutarlas [9- 11].

1.1.2. Las neuronas

Las células especializadas que conforman el cerebro son las neuronas. Estas son las encargadas de transmitir señales electroquímicas, conocidas como potenciales de acción, los cuales permiten procesar y transmitir información tanto a nivel intercelular como intracelular y cuya actividad se puede registrar obteniendo señales eléctricas [12]. Estos procesos son posibles debido a la estructura peculiar de las neuronas, la cual les permite interconectarse entre ellas mediante axones y dendritas, ver figura 1.1.

Siendo (1) el soma o cuerpo que contiene el núcleo celular con el ADN, y es el centro metabólico de la neurona [13]. **(2) los axones**, que conducen los impulsos nerviosos desde el soma hacia otras células **(3) la mielina**, que es una sustancia que acelera la transmisión de la información [2, 14, 15]. Los axones presentan botones terminales o **varicosidades y puntos de contacto sináptico (4)** que permiten la comunicación entre

neuronas (5). Por último, las **dendritas (6)** reciben los impulsos nerviosos provenientes de otras neuronas, facilitando la comunicación interneuronal [16].

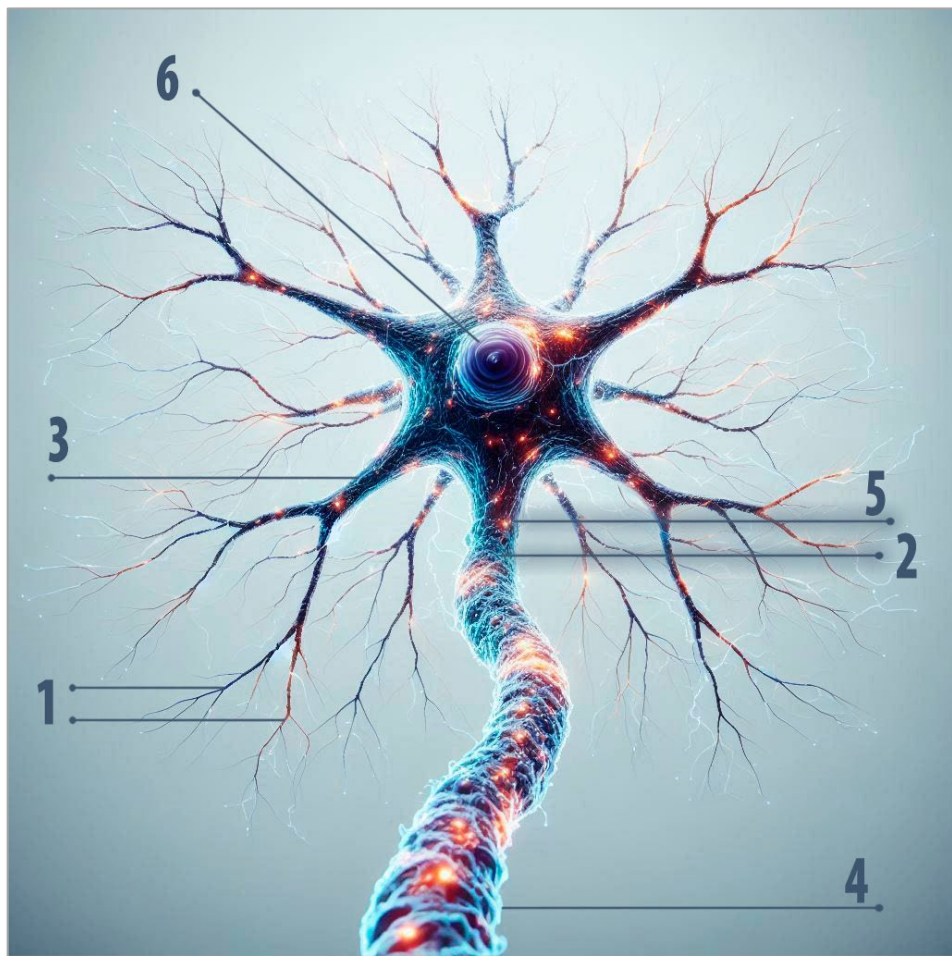


Figura 1.3. Vista detallada de una neurona y sus principales componentes.

1.1.3. La señal y las ondas cerebrales

La actividad cerebral eléctrica es fruto de la sinapsis neuronal en diversas regiones del cerebro, en donde grandes grupos de neuronas se sincronizan y emiten potenciales de acción de manera organizada. Esta actividad puede ser medida de manera no invasiva mediante el EEG, que registra los potenciales de acción de estas neuronas y los grafica en forma de microvoltios [12]. En la figura 1.4 se muestra de izquierda a derecha el potencial de acción entre dos neuronas, donde se produce un intercambio de neurotransmisores y en consecuencia una actividad eléctrica, seguido de la captación de este potencial a través del casco de EEG y su posterior interpretación gráfica.



Figura 1.4. Actividad eléctrica captada por un casco EEG fruto de la sinapsis neuronal.

La actividad cerebral es extremadamente compleja, sin embargo, los ritmos u ondas cerebrales se agrupan en distintas bandas de frecuencia, cada una asociada a un tipo particular de actividad neuronal y a estados cognitivos y emocionales específicos. En la tabla 1.1, se muestran los diferentes tipos de ondas cerebrales, su rango de frecuencia y su estado neurológico asociado. Aunque no existe consenso claro sobre la frecuencia exacta a la que se limita cada tipo de onda, en este caso se utiliza una de las más conocidas [17, 18].

Tabla 1.1. Características de las ondas cerebrales básicas y frecuencia asociadas.

Banda de frecuencia	Frecuencia	Estado neurológico
Delta (δ)	0.5 – 4 Hz	Dormido
Theta (ϑ)	4 – 8 Hz	Relajación profunda
Alpha (α)	8 – 12 Hz	Atención pasiva relajada – tarea automática
Beta (β)	12 – 35 Hz	Activo– atento– ansioso
Gamma (γ)	>35 Hz	Concentración máxima/ actividad excesiva

El análisis detallado de las ondas cerebrales y de la señal cerebral mediante técnicas como el EEG revela la compleja dinámica de la actividad neuronal que refleja estados cognitivos y funcionales del cerebro. La identificación y caracterización de las distintas bandas de frecuencia permiten discernir patrones específicos asociados a procesos como la memoria, la atención, la relajación y la intención motora. Sin embargo, la naturaleza intrincada y multidimensional de estas señales plantea desafíos significativos en términos de procesamiento y análisis eficiente.

1.2. Inteligencia artificial y *Brain-Computer Interfaces*

Para extraer información relevante de las señales EEG, es imprescindible recurrir a métodos avanzados de procesamiento de datos que puedan manejar la alta dimensionalidad y la baja relación señal-ruido inherentes a estas mediciones.

En este contexto, la IA, y más específicamente los algoritmos de *Machine* y *Deep Learning*, emergen como herramientas poderosas capaces de modelar y aprender los patrones complejos y las relaciones no lineales presentes en las señales cerebrales. Estos métodos permiten no solo la clasificación precisa de estados neuronales, sino también la predicción y traducción de intenciones motoras en comandos utilizables para dispositivos externos.

La integración de técnicas de IA con el análisis de las señales cerebrales abre nuevas posibilidades en el desarrollo de los llamados interfaces cerebro-computadora o en inglés "*brain-computer interfaces*". Este enfoque no solo mejora la precisión y la velocidad de la interpretación de las señales EEG, sino que también facilita la

creación de estos sistemas adaptativos que pueden aprender y ajustarse continuamente a las particularidades individuales de cada usuario.

A continuación, se definen los sistemas *BCI's*, sus componentes fundamentales, así como sus potenciales aplicaciones, beneficios e implicaciones:

Los *BCI's* son sistemas que permiten una comunicación directa entre el cerebro y un dispositivo externo, sin la necesidad de utilizar los canales convencionales de salida; como el neuromuscular, los movimientos corporales o el habla. Estos sistemas se basan en la captación, interpretación y traducción de las señales cerebrales, en comandos que puedan controlar dispositivos externos, como prótesis robóticas, computadores o sistemas domóticos, ver figura 1.5 [19, 20].



Figura 1.5. Esquema realista de un dispositivo *BCI* en funcionamiento.

Estos sistemas pueden ser invasivos o no invasivos:

- **BCI Invasivas:** Requieren la implantación de electrodos directamente en el tejido cerebral. Aunque ofrecen una mayor resolución en la adquisición de señales, tienen riesgos asociados a la cirugía y a la posible degradación del dispositivo con el tiempo. Ejemplo: *Neuralink*.
- **BCI No Invasivas:** Utilizan métodos como el EEG, que implica la colocación de electrodos en el cuero cabelludo. Aunque la resolución espacial es menor, son más seguros y prácticos para el uso diario. Ejemplo: La presente tesis.

1.2.1. Componentes Fundamentales de una BCI. Aplicaciones

El desarrollo de una BCI implica varias etapas técnicas, que incluyen la adquisición de señales cerebrales (generalmente mediante EEG o una base de datos), el preprocesamiento y limpieza de estas señales para eliminar ruido, la extracción de características relevantes y, finalmente, la clasificación de estas características para transformar la intención del usuario en comandos específicos [21 – 26].

- **Adquisición de Señales:** Las BCI capturan la actividad neuronal mediante diversas técnicas de neuroimagen, siendo el EEG la más utilizada por su naturaleza no invasiva, bajo coste y alta resolución temporal. Otras técnicas incluyen la magnetoencefalografía (MEG), la espectroscopía funcional por infrarrojo cercano (fNIRS) y la electrocorticografía (ECoG), en sistemas más especializados o invasivos.
- **Preprocesamiento de Señales:** Las señales cerebrales capturadas presentan una baja relación señal-ruido debido a la presencia de artefactos electromagnéticos del entorno, la propia respiración o actividad muscular y otras interferencias. En esta etapa se aplican filtros y técnicas matemáticas para eliminar estos artefactos y mejorar la calidad de la señal.
- **Extracción de Características:** La señal cerebral contiene una cantidad masiva de datos, pero no toda esa información es relevante para la tarea en cuestión o no está optimizada. Por ello, se extraen características que representan aspectos específicos de la actividad cerebral, como las frecuencias de las ondas cerebrales o cambios en el potencial cerebral relacionados con eventos (*ERP, Event-Related Potentials*).

- Clasificación e Interpretación: El corazón del sistema *BCI* es el modelo de clasificación, el cual interpreta las características extraídas para detectar patrones específicos de actividad cerebral asociados a intenciones motoras, comandos específicos o estados cognitivos. Aquí, los algoritmos de IA, y más concretamente el *Deep Learning*, han mostrado un gran potencial en la mejora de la precisión y adaptabilidad de las *BCI*, gracias a su capacidad de procesar grandes volúmenes de datos y aprender patrones complejos.
- Salida y Control de Dispositivos: Finalmente, los patrones detectados son convertidos en comandos que controlan dispositivos externos, como brazos robóticos, sillas de ruedas, o software de asistencia. Los sistemas de retroalimentación permiten que el usuario reciba información del dispositivo controlado, mejorando así la precisión y la experiencia de uso.

Esto trae consigo una serie de aplicaciones y beneficios potenciales para la sociedad y los individuos y un gran potencial para mejorar escenarios en una amplia gama de ámbitos, desde la salud hasta la industria [27, 28]. Algunos ejemplos son:

- Rehabilitación Neurofisiológica y Prótesis Controladas por la Mente: Una de las aplicaciones más prometedoras es en la rehabilitación de personas con discapacidades motoras graves, como parálisis o amputaciones. Las *BCI* permiten que los usuarios controlen directamente dispositivos como sillas de ruedas o prótesis robóticas, mediante la interpretación de sus intenciones motoras a partir de la actividad cerebral. Para personas con lesiones medulares o enfermedades neurodegenerativas, estas interfaces ofrecen una vía de comunicación y movilidad que no depende de los sistemas motrices convencionales, mejorando su independencia y calidad de vida.
- Comunicación Asistida: En casos de pacientes con trastornos como el síndrome de enclaustramiento (*Locked-in Syndrome*), donde los individuos están completamente paralizados, pero cognitivamente intactos, las *BCI* ofrecen una forma de comunicación basada en la interpretación de sus señales cerebrales. Esto permite a los pacientes expresar sus necesidades y deseos, restaurando su capacidad de interactuar con el entorno y reduciendo el aislamiento social.

- Control de Dispositivos Domóticos y Tecnológicos: Las *BCI* también pueden integrarse en sistemas domóticos, permitiendo a las personas controlar diversos dispositivos en su hogar (luces, puertas, electrodomésticos) simplemente utilizando su actividad cerebral. Esto es especialmente útil para personas con movilidad limitada.
- Neurorrehabilitación y Plasticidad Cerebral: Las *BCI* también se están utilizando en terapias de rehabilitación para mejorar la neuroplasticidad y recuperar funciones motoras perdidas. Por ejemplo, mediante el uso de una *BCI*, los pacientes que han sufrido un accidente cerebrovascular pueden recibir retroalimentación visual o auditiva mientras intentan mover una extremidad, lo que facilita la reorganización neuronal y la recuperación motora.
- Entrenamiento Cognitivo y Neurofeedback: Las *BCI* pueden emplearse en el entrenamiento cognitivo mediante el uso de neurofeedback, una técnica en la que los usuarios reciben retroalimentación en tiempo real de su actividad cerebral para aprender a regular su propio estado mental. Esto tiene aplicaciones en el tratamiento de trastornos como el trastorno por déficit de atención e hiperactividad (TDAH), la ansiedad o la depresión, mejorando la regulación emocional y el control atencional.
- Exoesqueletos: Las *BCI* podrían utilizarse tanto a nivel lúdico, como a nivel bélico o como prótesis en las que un exoesqueleto es controlado por las ondas cerebrales.

Aunque las *BCI* ofrecen grandes beneficios potenciales, también plantean importantes desafíos éticos, sociales y técnicos que deben ser considerados [29]:

- Privacidad y Seguridad de los Datos Cerebrales
- Autonomía e interacciones no deseadas del Usuario
- Precisión y la capacidad de respuesta en tiempo real
- Desigualdad en el Acceso (Coste)
- Responsabilidad Legal, Ética y responsabilidad jurídica

1.3. Estado del arte

El desarrollo de los *BCI* ha avanzado de manera exponencial en las últimas décadas, impulsado por el progreso en áreas como la neurociencia, el procesamiento de señales, la IA y sobre todo gracias al desarrollo del hardware y software necesarios para ello [19].

El estado del arte actual en *BCI* se caracteriza por la integración de técnicas avanzadas de aprendizaje automático, especialmente el *Deep Learning*, para mejorar la interpretación de señales neuronales y superar las limitaciones tradicionales de los métodos más simples de procesamiento de señales, como los basados en filtrado de frecuencias o en la descomposición de componentes principales (*PCA*). Además, los avances en la miniaturización e integración de dispositivos y en la mejora de las interfaces no invasivas están acercando las *BCI* a una etapa en la que pueden ser utilizadas de manera cotidiana en aplicaciones clínicas y de consumo.

Este apartado tiene como objetivo ofrecer una visión detallada y crítica del estado actual de la tecnología de *BCI*, analizando las principales bases de datos, metodologías, algoritmos, y aplicaciones, al mismo tiempo que se identifican las barreras que persisten y las oportunidades futuras para mejorar su desempeño y aplicabilidad.

1.3.1. Bases de datos famosas

Las *BCI*'s son sistemas que pretenden funcionar en tiempo real con el objetivo de alcanzar los escenarios anteriormente mencionados. Sin embargo, en la mayoría de los casos, los sistemas comienzan siendo probados mediante simulaciones de software. Para ello se utilizan bases de datos, normalmente públicas (por su reproducibilidad), en las que decenas o centenas de sujetos realizan ciertas tareas o acciones mientras se registra su señal cerebral.

Existen muchos tipos de tareas que pueden ser monitorizadas mediante EEG y presentan un alto interés por parte de la comunidad científica. Uno de los temas que más interés despierta en este contexto de las *BCI* es la clasificación de tareas mentales, como, por ejemplo, la clasificación con éxito del pensamiento del movimiento de las manos o los pies. Es decir, de pensar en mover un brazo y que se clasifique ese

movimiento como correcto. De aquí en adelante la tesis se centrará en esta cuestión, dado que la hipótesis, que se planteará posteriormente, abarca la aplicación de los sistemas *BCI* en tareas mentales de clasificación de movimiento de las extremidades superiores.

En los últimos años, uno de los conjuntos de bases de datos más famosas, que han sido utilizadas en decenas de estudios relacionados con esta cuestión, es el conjunto de “*BCI competition datasets*”. Esto es una serie de bases de datos, que se utilizan y desarrollan anualmente en un concurso internacional sobre sistemas *BCI*, y que tienen como objetivo proporcionar datos neurocientíficos de alta calidad de libre acceso, que puedan utilizarse para evaluar el grado actual de los avances técnicos en *BCI* [30–34].

La consecuencia directa de un conjunto de datos tan famoso es que se genera un sesgo, asociado al uso único de esta familia de bases de datos en lugar de a diferentes bases de datos.

Arpaia *et al.* [35] realizaron un estudio en el que utilizan el criterio PRISMA para analizar la literatura y obtuvieron 89 artículos de *Scopus* y *PubMed* relacionados con los sistemas *BCI* aplicados clasificación de tareas mentales. Sus hallazgos se muestran en la tabla 1.2, en donde se puede ver que más del 75% de todos los artículos publicados han sido utilizando estas bases de datos de “*BCI-competition*”.

Como puede observarse en la tabla 1.2, existe una gran tendencia a utilizar las bases de datos de “*BCI-competition*”. Estas utilizan una pequeña cantidad de usuarios, lo cual conlleva a unos resultados más precisos y una tasa de acierto mayor.

Por ejemplo, en un estudio publicado por Yu *et al.* [41], se propuso un marco para identificar tareas mentales utilizando algoritmos de *Machine Learning*, y se compararon sus resultados con los ya existentes. Los resultados demostraron altas precisiones en sujetos específicos, llegando en ciertos casos a un 99,8% (*BCI competition IV-a*), 93,3% (*BCI competition IV-b*), 91,9% (*BCI competition III*) y 88,1% (*BCI competition V*).

Tabla 1.2. Resumen del uso de las diferentes bases de datos en publicaciones sobre *BCI*

Base de datos	Usuarios	Canales EEG	Número de usos en
<i>BCI competition III-3a</i> [31]	3	64	17/89 (19%)
<i>BCI competition III-4a</i> [31]	5	118	27/89 (30%)
<i>BCI competition IV-1</i> [32]	7	59	14/89 (15%)
<i>BCI competition IV-2a</i> [33]	9	22	52/89 (58%)
<i>BCI competition IV-2b</i> [34]	9	3	36/89 (40%)
<i>Dataset GigaScience</i> [36]	52	64	3/89 (3%)
<i>High-Gamma Dataset</i> [37]	14	128	3/89 (3%)
<i>Physionet EEG</i> [30]	109	64	2/89 (2%)
<i>MAMEN Phase</i> [38]	34	61	2/89 (2%)
Otras [39, 40]	5–15	15–62	3/89 (3%)
Bases de datos privadas	1–12	2–64	9/89 (10%)

Pese a tener una alta tasa de acierto existe un problema de sesgo, ya que estos conjuntos se entrenan con pocos usuarios (3-9 sujetos) y con unos protocolos muy estrictos en sus pruebas y muestreo. Creando herramientas muy específicas que funcionan muy bien para casos concretos. Esto se ve reflejado en una tasa de participación más baja para otros conjuntos de datos que utilizan un número mayor de usuarios, incluidos el conjunto de datos *GigaScience* o *Physionet*, con tasas de uso en las publicaciones de cerca del 3% y el 2%, respectivamente.

Por esta razón y con el objetivo de evitar un sistema sesgado, en la presente tesis se busca utilizar una base de datos que abarque una mayor cantidad de sujetos con el objetivo de buscar un sistema más generalizado, que pueda ser implantado de una manera masiva, siendo independiente del sujeto, o muy poco dependiente. Mejorando así la fiabilidad, a costa del riesgo de disminuir el índice de éxito.

Por ejemplo, en la clasificación binaria utilizando el conjunto de datos de la *BCI competition IV-2b*, la precisión para la clasificación de tareas mentales oscila entre el 70% y el 90%, debido al uso de sólo 10 usuarios en el desarrollo del modelo. Sin embargo, al utilizar 109 sujetos (conjunto *Physionet*) el sistema es más generalizado, pero no pasa del 80% de éxito en la mayoría de los casos.

1.3.2. Mejores resultados

Muchos han sido los trabajos que se han centrado en este paradigma de la clasificación de tareas mentales de movimiento de las manos. Alcanzando rangos que van desde el 60% hasta el 99% en el éxito de la clasificación [35], [42]. Sin embargo, como se expresó anteriormente los resultados pueden variar en función de diversos factores:

- Base de datos
- Número de sujetos
- Especificidad del sistema (generalizado o individuo a individuo)
- Tipo y número de electrodos utilizados
- Tipo de muestreo y técnicas (invasiva/no invasiva)
- Metodología (procesado, algoritmos de clasificación)

En la última década, el campo de las *BCI*'s ha ido creciendo y muchos trabajos se han centrado en la parte de la clasificación, aplicando diferentes arquitecturas y modelos con el objetivo de mejorar los resultados de esta. Por ejemplo, Xu *et al.* [43] alcanzaron una tasa de éxito del 74.2% en la clasificación de movimientos imaginados utilizando una arquitectura de redes neuronales convolucionales (*CNN*) basada en *VGG-16* (algoritmo pre-entrenado) para la clasificación de señales EEG. Por otro lado, el equipo de Lu *et al.* [44] desarrolló un esquema de *Deep Learning* basado en una máquina de Boltzmann restringida (*RBM*) para este tipo de clasificaciones y logró una precisión del 84,2%. Por su parte, Li *et al.* [45] consiguieron hasta un 83.2% en el éxito de la clasificación utilizando un algoritmo de red neuronal convolucional simplificada (*SCNN*) de transformada *wavelet* continua (*CWT*) para la identificación de estos patrones del EEG. Tayed *et al.* [46] propusieron un *BCI* no invasivo basado en EEG y obtuvieron una precisión media del 80,8% con 20 participantes. Por último, Ha *et al.* [47] lograron una tasa de acierto del 78.4% de éxito gracias a utilizar *CapsNet* para la clasificación de señales EEG de dos clases (mano izquierda vs mano derecha).

Como puede observarse hay una amplia y diversa cantidad de trabajos que han llevado este ámbito a ser cada vez más competitivo y exitoso, utilizando todo tipo de enfoques basados en estadística, redes neuronales recurrentes (*RNN*), convolucionales,

multidominio y todo tipo de arquitecturas basadas en *Deep y Machine Learning*. Pero, de nuevo, utilizando la limitada base de datos del concurso *BCI*.

En la tabla 1.3 se muestra un resumen de los trabajos más relevantes, junto con la base de datos utilizada, número de sujetos de esta y la tasa de éxito alcanzada en la clasificación.

Tabla 1.3. Resumen del uso de las diferentes bases de datos en publicaciones sobre BCI.

Estudio	Base de datos	Usuarios	Tasa de acierto
Ko <i>et al.</i> [48]	<i>BCI competition IV-2a</i>	9	45,00%
Amira <i>et al.</i> [49]	<i>BCI competition IV-2a</i>	9	58,42%
Murcide <i>et al.</i> [50]	<i>BCI competition IV-2a</i>	9	61,86%
Sakhavi <i>et al.</i> [51]	<i>BCI competition IV-2a</i>	9	70,60%
Kim <i>et al.</i> [52]	<i>BCI competition III-4a</i>	3	74,30%
Sakhavi <i>et al.</i> [53]	<i>BCI competition III-4a and BCI competition IV-2a</i>	3 & 9	74,40%
Jiaqi <i>et al.</i> [54]	<i>BCI competition III-a- BCI competition IV-2a</i>	3 & 9	77,70%
Jiao <i>et al.</i> [55]	<i>BCI competition IV-2b</i>	9	78,00± 2,30%
Lee and Choi [56]	<i>BCI competition IV-2b</i>	9	Morlet- 78,90%, Bump-77,30%
Ai <i>et al.</i> [57]	<i>BCI competition IV 2a</i>	9	79,70%
Kim <i>et al.</i> [58]	<i>Physionet</i>	109	80,05%

Dose <i>et al.</i> [59]	<i>Physionet</i>	109	Global: 80,30 ± 12,50%
Luo and Chao [60]	<i>BCI competition IV-2b</i>	9	82,75%
Olivas-Padilla and Chacon-Murguia [61]	<i>BCI competition IV 2a</i>	9	85,04%
Miao <i>et al.</i> [62]	<i>BCI competition III 4a</i>	9	89,20%

En resumen, existen diferentes líneas de investigación para abordar los desafíos de las *BCI* y la clasificación de la intención motora. La precisión de los sistemas *BCI* puede variar en función del usuario, como se especifica en [63], lo que limita la generalidad del sistema. Además, muchos estudios con altas tasas de éxito, como los anteriores, se han centrado en entrenar y clasificar las señales de cada usuario individualmente para realizar una media y no con el conjunto completo, lo que puede limitar aún más la creación de un sistema generalizado. Esto, unido al problema de utilizar bases de datos con pocos usuarios da como resultado una tasa de éxito que contienen sesgo o está sobreestimada. Por el contrario, este trabajo pretende desarrollar un sistema *BCI* utilizando la base de datos *Physionet*, que contiene un gran número de usuarios (109 sujetos) y permite obtener resultados más generalizados a diferencia de los estudios que han utilizado los conjuntos de datos *BCI competition*.

Los estudios que utilizan la base de datos *Physionet* han alcanzado precisiones de hasta el 80,0% en la clasificación binaria de intenciones motoras, sin centrarse en usuarios individuales [58]. Por otra parte, algunos estudios han logrado una mayor precisión en clasificaciones utilizando la base de datos *Physionet*, sin embargo, fue clasificando las señales de cada usuario individualmente [64].

Así, este trabajo pretende desarrollar un sistema más generalizado que pueda funcionar para cualquier usuario y evitar posibles sesgos por buscar especificidad.

1.4. Hipótesis y objetivos

Se plantea la siguiente hipótesis:

“La información extraída de la señal cerebral mediante el EEG contiene información valiosa relacionada con la intención motora. El correcto análisis y tratamiento de esta, permite clasificar los pensamientos y llevar a cabo una acción motora a partir de una intención”

Dada la hipótesis, los objetivos a cumplir son los siguientes:

- 1) Estudiar el campo de las señales cerebrales y las *Brain–Computer interfaces*.
- 2) Profundizar en los algoritmos de *Machine y Deep Learning* aplicados a este paradigma.
- 3) Analizar el estado del arte en lo relacionado con la intención motora y su correcta clasificación.
- 4) Proponer un sistema que supere el estado del arte.
- 5) Aportar al tejido investigador mediante publicaciones en revistas indexadas.

1.5. Novedad de la propuesta y contribuciones

Novedad de la propuesta

La presente tesis doctoral introduce una serie de innovaciones clave en el campo de las *BCI*, particularmente en la utilización de datos del *EEG* para la detección y clasificación de la intención motora, con el objetivo de controlar dispositivos robóticos mediante técnicas avanzadas de inteligencia artificial. La principal novedad de esta investigación radica en el enfoque generalizable y no invasivo, buscando superar los sesgos presentes en estudios previos que dependen de un número reducido de sujetos o datos muy específicos.

En lugar de recurrir a las bases de datos más frecuentemente utilizadas, que tienden a generar modelos altamente específicos para individuos particulares, esta tesis apuesta por una base de datos más amplia y generalizada, como *Physionet*, que contiene datos de 109 sujetos. Esto permitirá un enfoque más robusto y adaptable a una mayor diversidad de usuarios. Además, la propuesta incluye la exploración de diferentes arquitecturas de *Machine Learning* y *Deep Learning* aplicadas a señales EEG, combinando modelos de redes neuronales convolucionales (*CNN*) con redes *Long-Short Term Memory (LSTM)* y capas de Atención, configurando un marco novedoso para la clasificación de intenciones motoras. Esto ha derivado en que las técnicas desarrolladas que han surgido a partir de la investigación han sido publicadas en revistas indexadas por su novedoso enfoque.

Contribuciones

1. Desarrollo de arquitecturas híbridas innovadoras

Se implementan metodologías novedosas que combinan arquitecturas avanzadas de *Deep Learning*, integrando modelos *CNN* con *LSTM* y capas de atención. Este enfoque híbrido, junto con un preprocesamiento optimizado de las señales EEG, mejora notablemente la precisión en la clasificación de intenciones motoras y permite una interpretación más confiable de los datos.

2. Aportes significativos en el ámbito de las BCI y procesamiento de señales

Este trabajo, ha dado como fruto la creación de dos capítulos de libro especializados en *BCI*, uno para *IntechOpen* y otro para *Sustainable Computing*, que abordan el análisis de modelos de *Machine Learning* y *Deep Learning*, así como la implementación de técnicas de clasificación que consideran el estado mental previo de los usuarios en las pruebas de la base de datos.

3. Publicaciones científicas en revistas indexadas:

- **Artículo publicado en *IRBM* (Q1, 2023):** Se desarrolló una herramienta de clasificación de señales EEG para evaluar la capacidad de generar señales

eficientes de intención motora de los sujetos, utilizando la base de datos *Physionet*, aplicando algoritmos de *Machine Learning* y *Deep Learning*.

- **Artículo publicado en *Desalination* (Q1, 2023):** Publicación enfocada en la predicción de permeabilidad a partir de series temporales, empleando los modelos desarrollados en este trabajo.
- **Artículo publicado en *Scientific Reports* (Q1, 2023):** Sobre el análisis de series temporales relacionadas con el COVID-19. Se utilizó la metodología desarrollada en esta tesis para aplicar los algoritmos híbridos de clasificación en el tratamiento de datos de salud.
- **Artículo bajo revisión en *Scientific Reports* (Q1, 2023):** Se centra en la clasificación de intenciones motoras mediante modelos híbridos de *Machine Learning* y *Deep Learning*, resaltando las mejoras en precisión y aplicabilidad.
- **Artículo bajo revisión en *Scientific Reports*:** Sobre clasificación de emociones. Utiliza algoritmos híbridos para la clasificación de estados emocionales, basados en la metodología de la tesis.

4. Colaboraciones científicas relevantes:

- **Colaboración en *Transactions on Information Forensics and Security* (Q1, 2023):** Envío en revisión sobre la interpretación de firmas 3D en el aire a través de algoritmos de *Machine Learning*.
- **Colaboración publicada en *Sensors* (Q2, 2023):** Artículo publicado sobre análisis meteorológico, aplicando algoritmos de series temporales similares a los utilizados en *BCI*.

5. Optimización en procesamiento y evaluación de señales EEG

Durante la tesis se ha desarrollado un tipo de preprocesamiento avanzado que consiste en un sistema de empaquetamiento de datos innovador para reducir el ruido en las señales EEG y mejorar la precisión del sistema *BCI*, logrando una clasificación precisa sin necesidad de entrenamientos específicos para cada usuario.

6. Publicaciones locales

Se han hecho diversas entrevistas en radio y se ha publicado en Radio Televisión Canaria (RTVC) un artículo de prensa que fomenta la divulgación científica y da prestigio al tejido investigador canario. (<https://rtvc.es/ulpgc-algoritmo-para-predecir-el-movimiento/>).

7. Refuerzo de las relaciones internacionales con la Universidad de Swansea (Gales).

Se ha realizado una estancia de investigación en la Universidad de Swansea, fomentando la colaboración y la relación internacional entre la ULPGC y esta universidad. Creando un precedente para futuras investigaciones y/o estancias.

1.6. Estructura de la memoria

La presente memoria de tesis doctoral cuenta con 6 capítulos, la bibliografía y 8 anexos. A continuación, se describe brevemente el contenido de cada uno de estos puntos mencionados:

1. **Introducción:** Se contextualiza el problema de estudio, introduciendo conceptos fundamentales sobre el cerebro, el EEG, la inteligencia artificial y las *BCI*. Se destacan los avances recientes y los desafíos en el campo. Planteándose así la hipótesis de trabajo.
2. **Materiales y Métodos:** Este capítulo describe las bases de datos utilizadas, incluyendo *Physionet*, los algoritmos de *Machine Learning* y *Deep Learning* empleados, y las técnicas de preprocesamiento de señales EEG. También se detallan los modelos desarrollados y los clasificadores aplicados.
3. **Metodología Experimental:** Se explican los experimentos realizados, variaciones en los clasificadores y en los hiperparámetros, condiciones experimentales, así como las optimizaciones aplicadas para mejorar los resultados del sistema propuesto.
4. **Resultados:** En este capítulo se presentan los resultados obtenidos en los experimentos, analizando la precisión, fiabilidad y desempeño de los algoritmos implementados para la clasificación de las señales EEG y el control de dispositivos robóticos.
5. **Discusión:** Se discuten los hallazgos obtenidos en relación con el estado del arte, comparando los resultados de esta tesis con estudios previos y destacando las principales contribuciones e implicaciones de los resultados obtenidos.
6. **Conclusiones y Líneas Futuras:** Se validan las hipótesis propuestas y se discuten las posibles líneas de investigación futura, sugiriendo mejoras adicionales y aplicaciones potenciales en campos como la neurorrehabilitación y el control robótico mediante *BCI*.
7. **Bibliografía:** Se incluye un listado exhaustivo de todas las referencias utilizadas a lo largo de la investigación.
8. **Anexos:** Publicaciones científicas derivadas de la tesis y material adicional relevante.

2. Materiales y métodos

Este capítulo, en el cual se explican y detallan los materiales y métodos utilizados para el desarrollo de la tesis, se divide en 5 apartados:

En el primero de ellos se detalla la base de datos utilizada y sus características. Seguidamente se exponen y detallan los diferentes algoritmos de *Machine* y *Deep Learning* que han sido utilizados a lo largo del desarrollo de la tesis. Posteriormente, se narra de qué forma ha sido procesada la señal cerebral obtenida de la base de datos y que clasificadores han sido propuestos para los experimentos. Y, por último, se detallan las arquitecturas o modelos finales obtenidos como conjunto de lo anterior.

2.1. Base de datos

Physionet

Este trabajo se ha desarrollado utilizando la base de datos pública *Physionet* [30], [65], que consiste en una serie de registros EEG de dos minutos de duración obtenidos de 109 individuos. Esta base de datos recoge las señales EEG capturadas mientras los sujetos realizan una variedad de tareas motoras, tanto reales como imaginarias, grabadas mediante un sistema EEG de 64 canales.

Estructura de los Experimentos

El protocolo experimental está compuesto por 14 experimentos por sujeto divididos en seis tipos de tareas o sesiones. Estas tareas están organizadas de la siguiente manera:

- 1) **Fase de referencia o línea base:** Consiste en la adquisición de EEG en reposo, con una duración de un minuto para cada fase, es decir, una adquisición con los ojos cerrados y luego lo mismo, pero con los ojos abiertos. Esto son los experimentos 1 y 2 (de los 14 totales).
- 2) **Tareas motoras:** Se les solicita a los sujetos realizar o imaginar movimientos específicos. Esto es, desde el experimento 3 hasta el 14 de cada sujeto.

Dentro de las tareas motoras existen cuatro tipos de tareas en total.

- **Tarea 1:** Apertura y cierre del puño derecho o izquierdo, según la indicación visual en pantalla.
- **Tarea 2:** Imaginación de la apertura y cierre del puño derecho o izquierdo, siendo la misma tarea que la tarea 1, pero imaginando el movimiento en lugar de realizándolo.
- **Tareas 3 y 4:** Estas consisten en el movimiento, por un lado y en la imaginación, por otro lado, de ambos puños y ambas piernas. Sin embargo, no han sido consideradas en este estudio ya que se optó por un enfoque binario centrado únicamente en los movimientos de las manos.

Cada tipo de tarea se repite 3 veces, dando los 12 experimentos que sumados a las líneas base dan los 14 experimentos por sujeto. En la tabla 2.1, se recoge a modo de resumen el conjunto de experimentos y sus características, resaltando los que han sido de utilidad en esta tesis.

Tabla 2.1. Resumen de los diferentes experimentos contenidos en la base de datos *Physionet*.

<i>Physionet</i> 160Hz 14 tareas					
	Linea base	Tarea 1	Tarea 2	Tarea 3	Tarea 4
Descripción	Ojos abiertos /cerrados	Real mano izq/dcha	Imaginado mano izq/dcha	Real manos y pies	Imaginado manos y pies
Duración	60s	121-123s	121-123s	121-123s	121-123s
Número experimento	1 y 2	3, 7 y 11	4, 8 y 12	5, 9 y 13	6, 10 y 14

Selección de Tareas y Configuración Experimental

Como se ha especificado anteriormente, solo se han tenido en cuenta las tareas motoras del tipo 1 y 2 (mano izquierda y derecha, tanto real como imaginada), que repetidas cada una 3 veces dan un resultando en un conjunto de seis experimentos para cada individuo. Cada experimento tiene una duración de aproximadamente dos minutos (121-123s) y las señales fueron registradas utilizando el sistema estándar 20-10 con 64 electrodos. A su vez, se excluyeron algunos canales del EEG, como Nz, F9, F10, FT10, A1, A2, TP9, TP10, P9 y P10, para evitar artefactos de movimiento y mejorar la calidad de los datos. La tasa de muestreo de las señales EEG fue de 160 Hz, lo que asegura una captura detallada de la actividad cerebral durante las tareas.

Así mismo, en cada experimento de aproximadamente 2 minutos, se alternan eventos de tipo “movimiento real” o “pensamiento de movimiento” (según el experimento) con eventos de descanso.

A los eventos de movimiento o pensamiento se les asigna una anotación del tipo T1 cuando el movimiento o pensamiento es del brazo izquierdo y T2 para el brazo derecho. Entre cada evento T1 o T2 está el mencionado periodo de 4.2s que corresponde al descanso. A este se le denota como T0. Los eventos T1 y T2 duran 4.1 segundos, seguidos siempre por ese periodo T0, proporcionando una alternancia regular en la actividad registrada, ver figura 2.1.

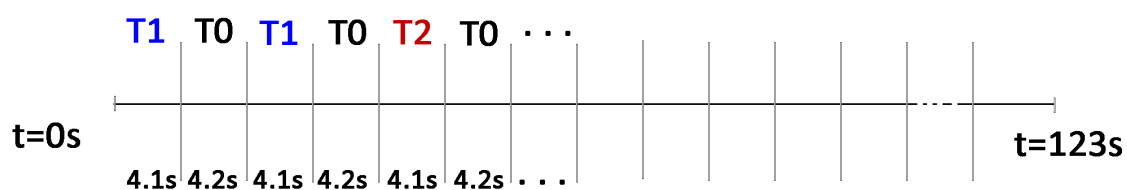


Figura 2.1. Ejemplo gráfico de la estructura de los experimentos.

Para mantener la uniformidad en los datos y simplificar el análisis, se eliminaron los eventos T0, centrando el análisis únicamente en los momentos de ejecución o imaginación de los movimientos.

Formato de los Datos

Los datos originales son proporcionados en formato *European Data Format (EDF+)*. Estos archivos *EDF+* han sido convertidos a formato numérico, facilitando su procesamiento para ser transformados en matrices que son más fácilmente operables por los *softwares* utilizados. Esto asegura que los datos están en un formato que cumple un estándar de partida para la posterior aplicación de modelos de *Machine* y *Deep Learning*.

Este análisis exhaustivo de la base de datos *Physionet* asegura un enfoque riguroso y bien fundamentado para el procesamiento y clasificación de señales EEG en el contexto de las *BCI*.

Lectura de la señal

La lectura, visualización y exportación de los **archivos** que contenían los experimentos con la señal EEG en formato *EDF+* fue llevada a cabo mediante los *softwares* *EDF Viewer*, la herramienta de MATLAB, *BCILAB* y posteriormente con Python (*Spyder*).

2.2. Procesado de la señal

En este apartado se describen los diversos métodos de procesamiento de señales EEG utilizados a lo largo de la tesis. Dado que las señales EEG contienen ruido y artefactos, teniendo una relación señal-ruido compleja, es crucial aplicar técnicas de preprocesamiento que optimicen la calidad de los datos antes de ser analizados.

A continuación, se detallan las diferentes técnicas aplicadas, incluyendo la conversión de los archivos en formato EDF a vectores, la aplicación de filtros Butterworth de diferentes órdenes, la selección de canales relevantes para el estudio, la normalización de las señales para asegurar su homogeneidad, y finalmente el algoritmo de tratamiento de datos que transforma los archivos EDF a matrices optimizadas para su posterior análisis.

Estas técnicas han formado parte del conjunto de experimentos y no han sido probadas todas en uno solo, ya que algunas demostraron no ser eficientes a diferencia

de otras que si lo fueron. Por lo tanto, **la mejor versión de la arquitectura final no es un conjunto de todos juntos sino una selección de las mejores técnicas.**

Tratamiento de datos EDF+ a vectores

Debido a la naturaleza de los datos estos son tratados a través de diferentes *softwares* con el objetivo de llevarlos a un formato cómodo. En una etapa inicial tanto "MATLAB" como "Python" carecían de un comando para la conversión de EDF+ a vectores, matrices, txt, etc... A medida que avanzó la tesis se pudo realizar este proceso de una manera más cómoda.

En un primer lugar, se utilizó el software "EDF-Browser" para visualizar datos, ver figura 2.2, y exportarlos a formato "txt" y "csv".

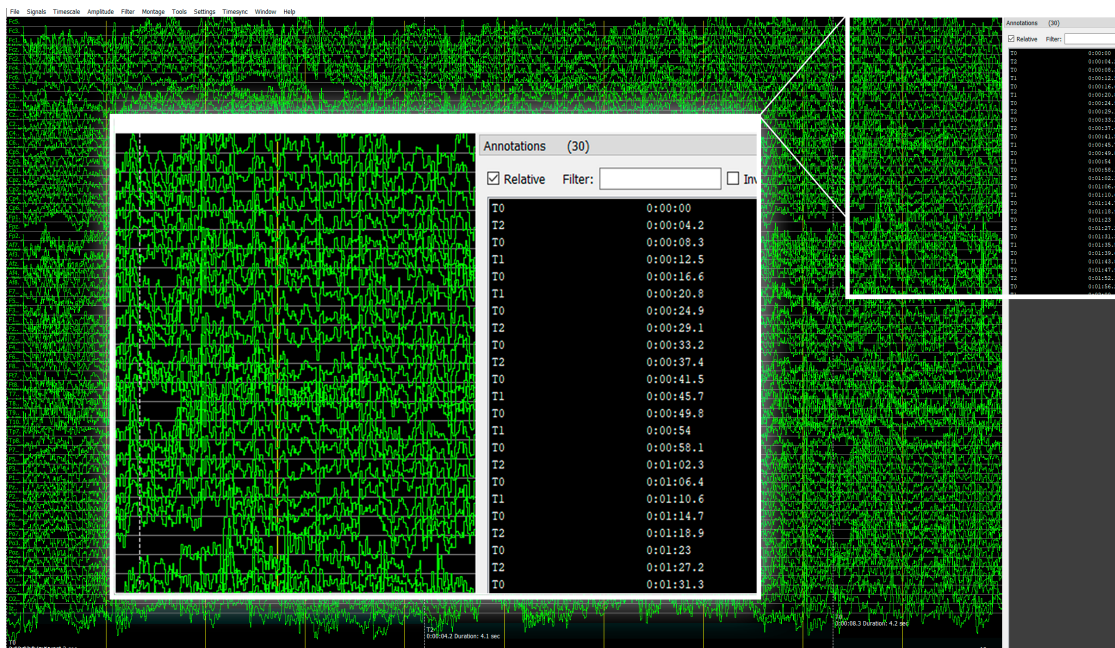


Figura 2.2. R003S001. Prueba 3 sujeto 1. Movimiento real de la mano izquierda y derecha visualizado en EDF Browser.

Posteriormente se desarrolló un algoritmo en MATLAB que lee los datos en formato ".txt" y genera matrices con los datos de interés de manera selectiva de cada uno de los experimentos [66]. Este algoritmo, funciona importando los archivos "txt", separando las anotaciones o etiquetas (vector que contiene información sobre donde están los T0,

T1 y T2) y eliminando los eventos T0. Dejando así una matriz de datos con los eventos T1 y T2 y su vector de etiquetas asociado.

Al tener una frecuencia de muestreo de 160Hz realmente la matriz de datos presenta una forma como la de la figura 2.3. En donde se tiene una matriz de 64 filas, coincidente con el número de canales del EEG y 9840 muestras por cada uno de los experimentos. Estas muestras suponen un evento T1 o T2 cada 656 valores (4,1s).

Y por otra parte el vector anotación es un vector de 1x9840 muestras donde simplemente se indica para cada una de las muestras si pertenece a un evento T1 o T2.

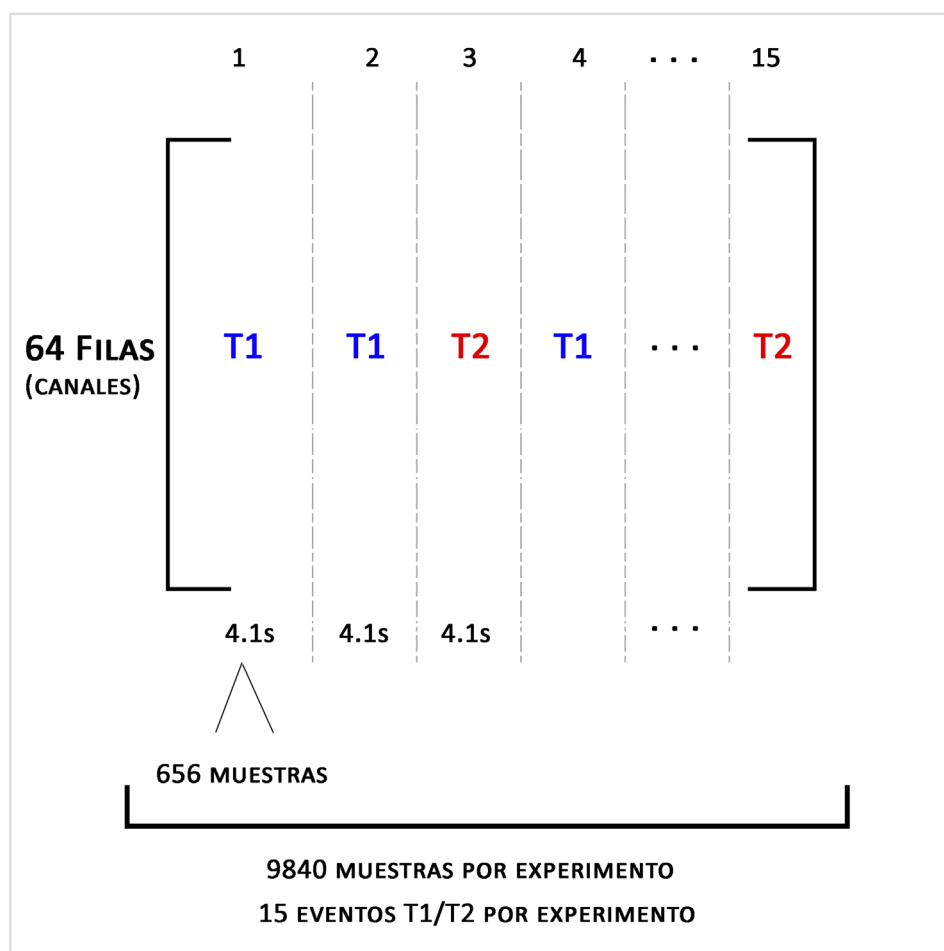


Figura 2.3. Ejemplo de un experimento cualquiera entre las tareas 3 a 14. En este ejemplo los eventos T0 ya han sido eliminados.

Selección de canales

Según algunos estudios, como los llevados a cabo por Sleight *et al.* [67] y Deecke *et al.* [68], de los muchos canales que componen el electroencefalograma (normalmente 64 o 128) muchos contienen información específica de una región del cerebro asociada a algún tipo de campo, como podría ser el lenguaje, el movimiento o la lógica. Sin embargo, según estos estudios existe también un problema de ruido. Esto se debe a que cuando se utilizan todos los canales la información que se obtiene puede ser redundante por ser el cerebro un órgano multitarea.

Según estas fuentes, los electrodos que proporcionan la información sobre el movimiento de las manos (real o imaginado) se encuentran en la zona del lóbulo frontal.

Concretamente en los canales 3, 4, 7, 8, 11 y 12 que corresponden en el casco de 64 electrodos con FC1, FCz, FC6, C5, Cz y C2 respectivamente.

En la presente tesis se realizaron pruebas con un sistema que contenía estos canales específicos, así como con el conjunto completo para corroborar si efectivamente la literatura estaba en lo cierto o no.

Filtro de la señal

En el procesamiento de señales cerebrales provenientes del EEG, es fundamental aplicar técnicas de filtrado que mejoren la calidad de la señal y eliminen el ruido no deseado ya que la relación señal-ruido de estas es muy baja.

Dado que las señales cerebrales en sujetos sanos oscilan típicamente entre 1 y 40-50 Hz, correspondientes a los ritmos delta, theta, alfa, beta y gamma, se hace necesario aplicar filtros que eliminen las componentes de frecuencia que no aportan información relevante o que puedan enmascarar las señales de interés [69, 70].

1) Frecuencias no Deseadas

- Frecuencias bajas (<0.5 Hz): Estas frecuencias suelen estar asociadas a artefactos fisiológicos como movimientos lentos, tales como respiración, movimientos corporales o ajustes posturales durante la medición de la señal. Estos artefactos

pueden introducir fluctuaciones de baja frecuencia que interfieren con la correcta detección de las ondas cerebrales de interés.

- Frecuencias altas (>50 Hz): Las frecuencias superiores a 50 Hz generalmente están relacionadas con ruido eléctrico, como el generado por la red eléctrica (50/60 Hz) o por interferencias electromagnéticas de dispositivos cercanos. Estas frecuencias se encuentran fuera del rango de interés para el análisis de señales EEG y pueden distorsionar los resultados al enmascarar las ondas cerebrales.

2) Aplicación del Filtro *Butterworth*

Para eliminar estos componentes de frecuencia no deseados, se ha implementado un filtro *Butterworth* paso banda con un rango de corte entre 0.5 Hz y 50 Hz. Este tipo de filtro es uno de los más utilizados en el procesamiento de señales biomédicas debido a su respuesta suave y fase lineal, lo que permite preservar la forma de la señal original sin introducir distorsiones significativas. El filtro *Butterworth* es especialmente adecuado en este contexto porque tiene una respuesta plana en la banda de paso, lo

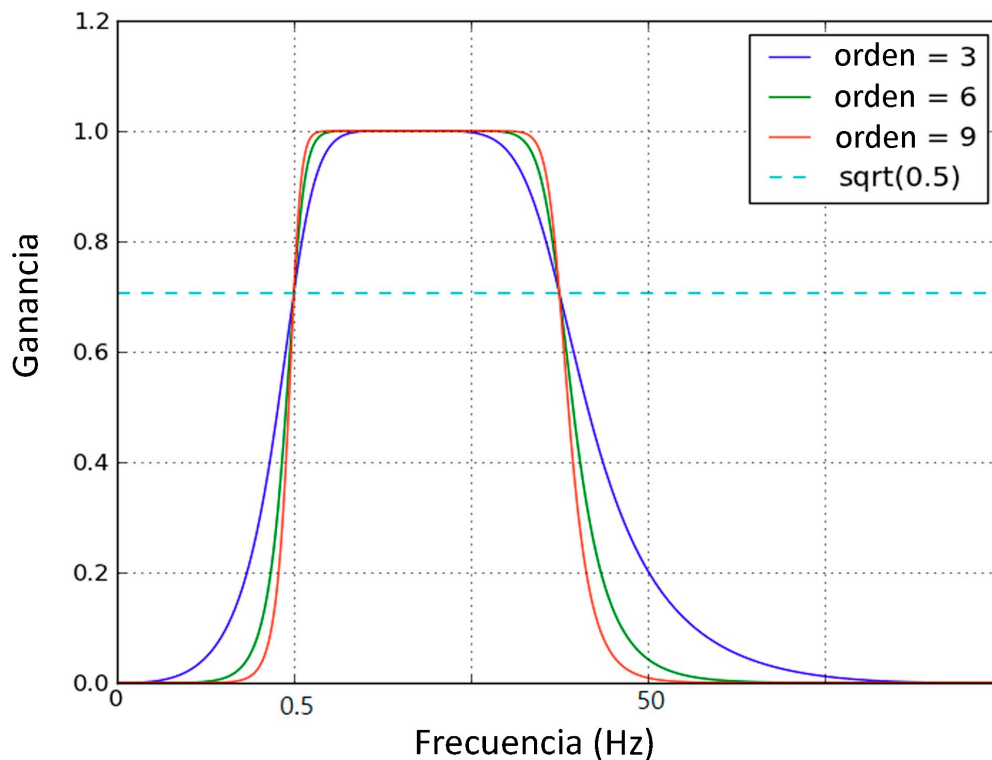


Figura 2.4. Filtro *Butterworth* paso banda de diferentes órdenes.

que significa que no introduce fluctuaciones adicionales en las frecuencias dentro del rango de interés, lo cual mantiene la integridad de las señales EEG.

Se probaron diferentes órdenes para el filtro. Finalmente, el filtro utilizado es un *Butterworth* orden 1. Se puede observar en la figura 2.4, un ejemplo de diferentes órdenes para el filtro. Se observa cómo las señales filtradas se encuentran contenidas dentro de los límites definidos por el filtro paso banda, lo que asegura la eliminación de frecuencias no deseadas y mejora la precisión de los resultados del análisis posterior.

Tratamiento de datos EDF+ a matrices

Debido a la naturaleza de los experimentos llevados a cabo, además de convertir los datos *EDF+* a vectores, es necesario utilizarlos en forma matricial. Por ello se desarrolla un algoritmo que, tras recibir los datos, pasa el EEG a un vector de matrices 2D. Esto se debe a que, durante el desarrollo de la tesis, se ha implementado una arquitectura que emula la conformación de los electrodos en el casco, pero en forma de matriz, ver figura 2.5.

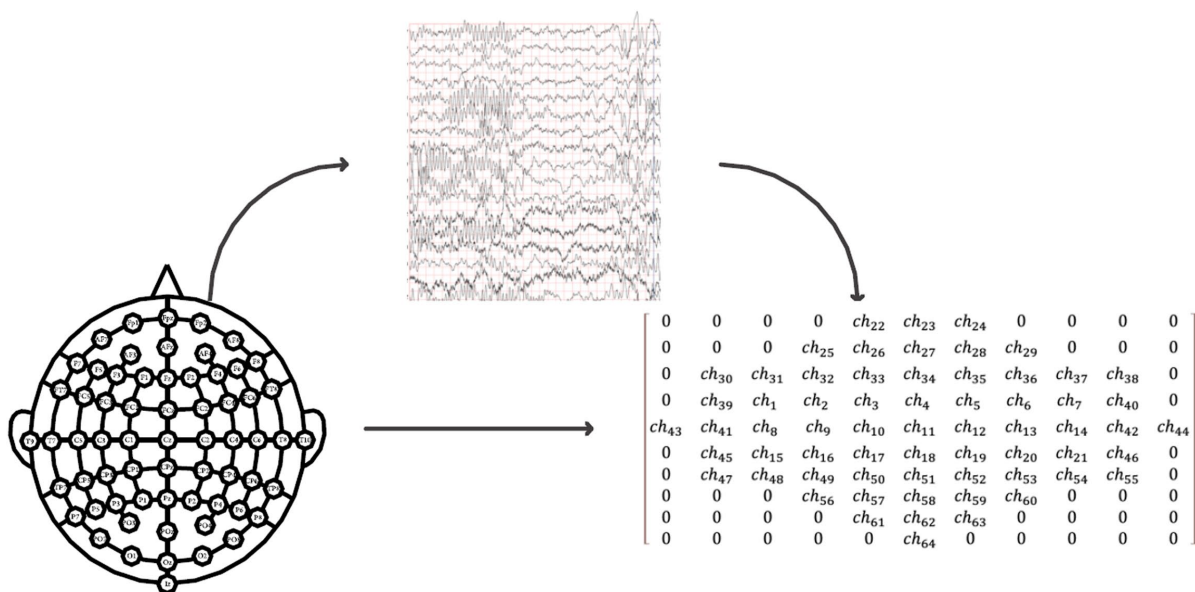


Figura 2.5. Esquema simplificado del funcionamiento del algoritmo generador de matrices 2D.

El objetivo de este preprocesamiento es maximizar las características extraíbles de la señal EEG, optimizando la conformación espacial para alimentar al algoritmo

clasificador de la manera más eficiente y precisa posible. Este método ha mostrado ser más eficaz que el uso de enfoques separados, dado que integra de manera sinérgica tanto la información espacial como temporal, lo que resulta en un mejor rendimiento en la clasificación final de los eventos [71].

2.3. Extracción de características

Transformada *Wavelet* Discreta

La Transformada *Wavelet* Discreta (*DWT*) es una herramienta matemática ampliamente utilizada en el procesamiento de señales no estacionarias, es decir, aquellas cuya frecuencia varía con el tiempo. A diferencia de la Transformada de Fourier, que tiene limitaciones al analizar señales con cambios abruptos o transitorios, la *DWT* ofrece una representación localizada en el tiempo y la frecuencia. Esto se consigue mediante el análisis de multiresolución, el cual permite descomponer la señal en diferentes bandas de frecuencia adaptadas al comportamiento temporal de la señal, proporcionando una visión detallada tanto de los componentes de alta como de baja frecuencia en una misma señal [72, 73], ver figura 2.6.

La *DWT* se basa en el uso de filtros paso bajo y paso alto que dividen la señal en dos componentes: uno que contiene las frecuencias altas y otro que incluye las frecuencias bajas.

Al aplicar estos filtros de manera jerárquica y repetitiva, es posible descomponer la señal en múltiples niveles, cada uno de los cuales ofrece una versión más detallada o general de la señal original. En cada nivel, el componente de baja frecuencia contiene las características más generales de la señal, mientras que el componente de alta frecuencia conserva la información más transitoria o detallada. Este enfoque de descomposición se puede realizar en varios niveles, dependiendo del grado de resolución deseado, lo que facilita un análisis profundo de la señal.

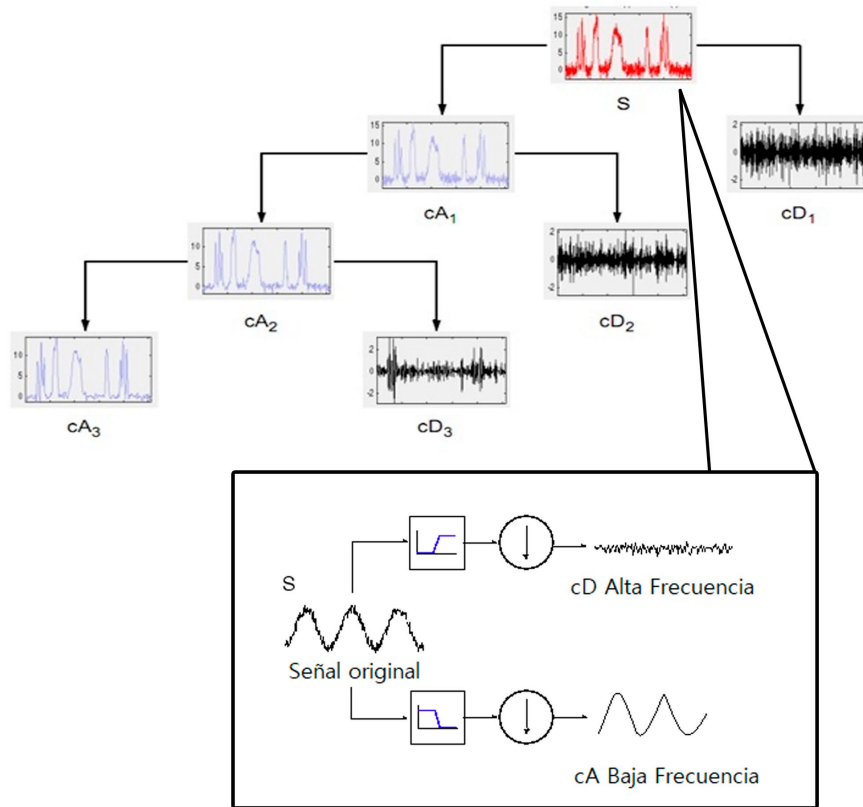


Figura 2.6. Descomposición de una señal en varios niveles utilizando la DWT.

En este trabajo, la *DWT* se emplea para descomponer las señales EEG en diferentes niveles de frecuencia, con el fin de identificar patrones tanto en las bajas como en las altas frecuencias que podrían ser indicativos de eventos de interés (como movimientos imaginarios de la mano izquierda o derecha). La familia de *wavelets* seleccionada es la *Coiflet*, que ha demostrado un rendimiento superior en la clasificación de señales EEG en estudios previos [71–73].

La expresión matemática para la ecuación de la *DWT* viene dada por [74–77]:

$$f(t) = \sum_k \sum_j a_{j,k} \varphi_{j,k}(t) + \sum_k \sum_j d_{j,k} \psi_{j,k}(t) \quad (2.1)$$

Donde:

- $\varphi_{j,k}(t)$ es la función de escalamiento (aproximación o bajas frecuencias)

- $\psi_{j,k}(t)$ es la *wavelet* madre, escalada y trasladada, que captura los detalles de la señal o altas frecuencias.
- $a_{j,k}$ son los coeficientes de aproximación en el nivel j y traslación k .
- $d_{j,k}$ son los coeficientes de detalle en el nivel j y traslación k .

La *DWT* está expresada en términos de dos índices, la traslación temporal k y el índice de escala j .

- Índice j (escala): El índice j representa el nivel de escala o resolución. En cada nivel de descomposición, el índice j controla cuántas veces la *wavelet* se expande o se contrae. Al aumentarla, la *wavelet* se comprime y detecta detalles más finos (frecuencias altas). En niveles más bajos de j , la *wavelet* se expande, capturando las componentes de baja frecuencia.
- Índice k (traslación): El índice k representa la traslación temporal de la *wavelet*. Controla la posición de la *wavelet* en el dominio temporal. Dependiendo del valor de k , la *wavelet* se desplaza sobre la señal para analizar diferentes regiones de esta.

Ambos índices j y k son enteros en la *DWT*, y las *wavelets* están diseñadas para formar un conjunto ortogonal, lo que significa que son mutuamente ortogonales. Esto garantiza que las funciones base (*wavelets*) no se superpongan de manera redundante al descomponer la señal, lo que permite una descomposición única y eficiente.

Parámetros estadísticos

Tal y como se propone en el trabajo de Noguera, M. *et al.* [21], ciertos parámetros estadísticos podrían mejorar la clasificación de la señal cerebral de las intenciones motoras.

Por ello, durante la tesis se han realizado experimentos aplicando estos parámetros y teniendo en cuenta uno o varios de ellos, con el fin de analizar su aportación en el éxito de la clasificación de la intención motora.

Estos parámetros, son la media absoluta (*MAV*), la varianza (*VAR*), el error cuadrático medio (*RMS*), la integración o suma del valor absoluto (*IEEG*) y su cuadrado (*SSI*), el cambio de amplitud medio (*AAC*) y las medidas de asimetría, curtosis (*KURT*) y skewness (*SKEW*). Las abreviaciones corresponden a sus siglas en inglés.

En la tabla 2.2, puede verse un resumen de estos parámetros, así como la expresión matemática que los define [78–80]:

Tabla 2.2. Parámetros estadísticos

<i>RMS</i>	$\sqrt{\frac{1}{N} \sum_{n=1}^N D_i^2(n)}$	(2.2)
<i>MAV</i>	$\frac{1}{N} \sum_{n=1}^N D_i(n) $	(2.3)
<i>IEEG</i>	$\sum_{n=1}^N D_i(n) $	(2.4)
<i>SSI</i>	$\sum_{n=1}^N D_i(n) ^2$	(2.5)
<i>VAR</i>	$\frac{1}{N-1} \sum_{n=1}^N D_i^2(n)$	(2.6)
<i>AAC</i>	$\frac{1}{N} \sum_{n=1}^N D_i(n+1) - D_i(n) $	(2.7)
<i>SKEW</i>	$\sum_{i=1}^N \frac{(x_i - \bar{x})^3 n_i}{NS_x^3}$	(2.8)
<i>KURT</i>	$\sum_{i=1}^N \frac{(x_i - \bar{x})^4 n_i}{NS_x^4}$	(2.9)

Donde, N es el número total de muestras, D_i es el dato de la iteración enésima (n), n_i es la frecuencia absoluta y S_x es la desviación típica

Reducción de dimensionalidad

La extracción de características es un paso crucial en el procesamiento de señales, ya que, permite reducir la dimensionalidad de los datos preservando la información relevante. Los métodos de extracción de características son ampliamente utilizados para hacer frente a problemas de alta dimensionalidad, redundancia y ruido en los datos, mejorando así el rendimiento de los algoritmos de clasificación y regresión.

En este trabajo se han utilizado dos técnicas conocidas para evaluar si la reducción de la dimensionalidad afecta a la clasificación posterior de la intención motora.

Estas son el Análisis de Componentes Principales (*PCA*) y el Análisis de Componentes Independientes (*ICA*), por sus siglas en inglés. Son técnicas estadísticas que transforman los datos originales en nuevas representaciones basadas en principios diferentes: ortogonalidad y no-gaussianidad, respectivamente [81–84].

Análisis de Componentes Principales (*PCA*)

El método *PCA*, es una técnica estadística de reducción de dimensionalidad que proyecta los datos en un espacio de menor dimensión, conservando la mayor cantidad posible de la varianza original. Es decir, el *PCA* busca una combinación lineal de las características originales que maximice la varianza, lo que equivale a encontrar los vectores propios de la matriz de covarianza de los datos. Estos vectores propios o componentes principales definen un nuevo sistema de coordenadas en el cual las dimensiones más importantes corresponden a aquellas con mayor varianza, ver figura 2.7, [81, 82].

Matemáticamente, siendo \mathbf{X} una matriz de datos con n observaciones, filas, y p variables, columnas, donde cada columna está centrada en su media (media cero). El *PCA* encuentra una nueva base ortogonal para representar los datos proyectándolos en los componentes principales.

$$\mathbf{Z} = \mathbf{XW} \quad (2.10)$$

Donde \mathbf{Z} es la representación de los datos proyectados en un espacio de menor dimensionalidad y \mathbf{W} se obtiene calculando los vectores propios de la matriz de covarianza \mathbf{C} .

$$\mathbf{C} = \frac{1}{n} \mathbf{X}^T \mathbf{X}. \quad (2.11)$$

Siendo n el número de observaciones (filas de la matriz \mathbf{X}), \mathbf{X} la matriz de datos centrada en la media y \mathbf{X}^T su transpuesta.

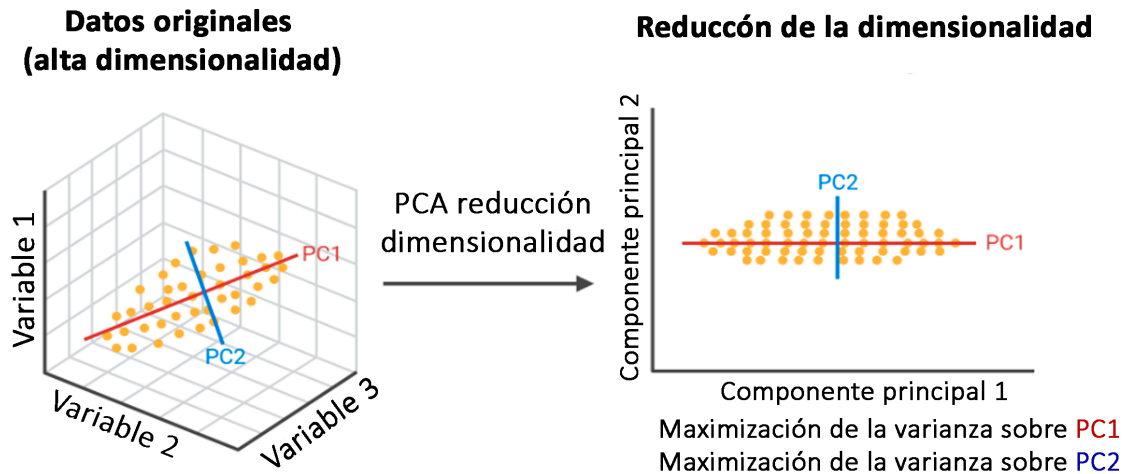


Figura 2.7. Esquema de reducción de la dimensionalidad aplicando PCA.

Análisis de Componentes Independientes (ICA)

El ICA es una técnica de extracción de características más avanzada que el PCA. Está diseñada para separar señales mezcladas en componentes estadísticamente independientes. Mientras que PCA se basa en la ortogonalidad y varianza, el ICA se enfoca en maximizar la independencia estadística entre las señales, haciendo especial énfasis en explotar la no-gaussianidad de los datos. Esta técnica es especialmente útil cuando los datos provienen de una combinación lineal de fuentes subyacentes desconocidas, como en problemas de separación de señales o "*blind source separation*" (BSS), [83, 84].

Matemáticamente, siendo \mathbf{X} una matriz de datos observada (mezcla lineal de las fuentes) y \mathbf{S} una matriz de señales fuente (que queremos hallar). El modelo ICA es:

$$\mathbf{X} = \mathbf{A} \mathbf{S} \quad (2.12)$$

Donde \mathbf{A} es la matriz de mezcla que representa cómo las señales \mathbf{S} se mezclan para formar las observaciones \mathbf{X} .

Para estimar \mathbf{A} y \mathbf{S} a partir de \mathbf{X} se puede buscar la matriz de transformación \mathbf{W} y expresar \mathbf{S} en función de esta:

$$\mathbf{S} = \mathbf{W} \mathbf{X} \quad (2.13)$$

Donde \mathbf{W} es la matriz inversa de \mathbf{A} obtenida mediante la maximización de la independencia de las componentes.

EL *ICA* ha sido especialmente útil en aplicaciones como el análisis de señales biomédicas (*ECG*, *EMG* o *EEG*), donde se requiere separar señales superpuestas como las originadas por diferentes fuentes neuronales.

2.4. Clasificadores

En este apartado, se analizan los principales algoritmos de procesamiento de series temporales, tanto de *Machine Learning* como de *Deep Learning* que han sido utilizados en esta tesis para el procesamiento y clasificación de las señales del EEG.

Estas señales, que representan la actividad eléctrica cerebral a lo largo del tiempo, requieren técnicas especializadas capaces de capturar características no lineales. Tanto en lo referido a las características espaciales como a las dependencias temporales de los datos.

Se exploran modelos tradicionales como los *SVM* y redes neuronales de base amplia, junto con enfoques más avanzados como las redes *CNN* y las *RNN*, incluyendo variantes como *LSTM* y capas de atención. El objetivo es identificar las metodologías más eficientes para la clasificación de la intención motora y la predicción de patrones en señales EEG, optimizando la precisión y generalización del sistema *BCI*.

Los algoritmos de *Machine Learning* fueron probados en el software *MATLAB* con su herramienta integrada "*Classification learner*" mientras que los algoritmos de *Deep Learning* fueron implementados con código tanto en *MATLAB* como en *Python 3.9*.

2.4.1. Machine Learning

Máquinas de Soporte Vectorial (SVM)

Las SVM son algoritmos de aprendizaje supervisado ampliamente utilizados en el ámbito de las BCI por su capacidad para manejar problemas de alta dimensionalidad y realizar una clasificación binaria o multiclase eficazmente.

Las SVM funcionan encontrando un hiperplano que maximiza el margen entre las clases de datos, utilizando vectores de soporte, que son los puntos de datos más cercanos al hiperplano de separación [85, 86]. Ver figura 2.8.

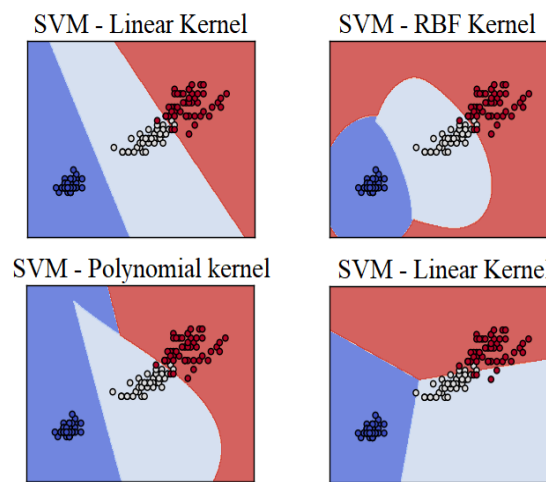


Figura 2.8. Ejemplos de SVM con diferentes hiperplanos o kernels.

La ecuación del hiperplano con el que se realiza la clasificación se da como:

$$w^T x + b = 0 \quad (2.14)$$

Donde w^T es el vector de pesos traspuesto, x es una instancia de los datos de entrada que se quieren clasificar y b es el sesgo.

K-Nearest Neighbor (k -NN)

El algoritmo KNN es un método de clasificación no paramétrico. Clasifica una nueva instancia en función de la mayoría de las clases de sus k vecinos más cercanos, donde la proximidad se mide generalmente con distancias como la *Euclidiana* o la *Manhattan* para variables continuas y la distancia *Hamming* para el caso de las categóricas.

Aunque simple, es efectivo en tareas de clasificación de EEG, especialmente para señales de baja dimensionalidad [87–89].

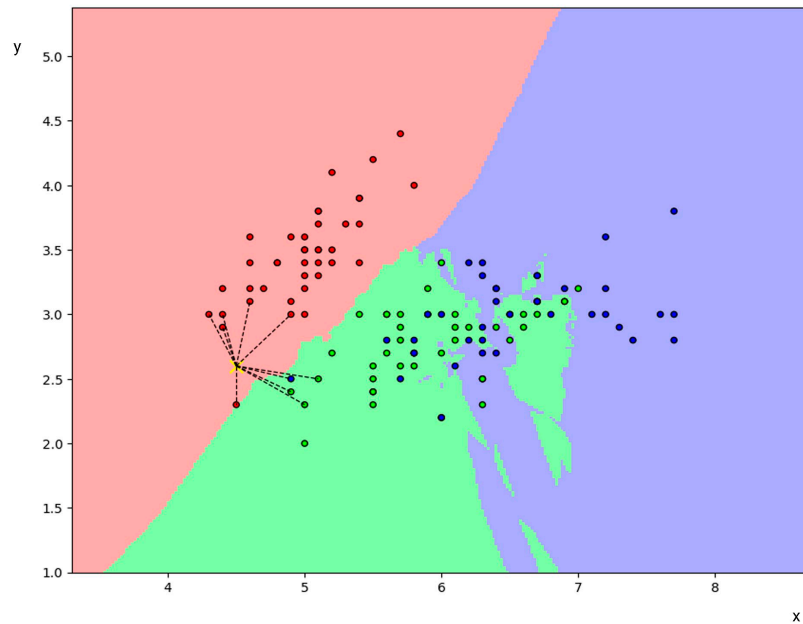


Figura 2.9. Representación de una clasificación con algoritmo K -NN con $k=10$ (iris dataset) [90].

Estas 3 distancias han sido utilizadas y sus ecuaciones son:

Euclidean distance (d_E)

$$d_E(x, y) = \sqrt{\sum_{i=1}^k (x_i - y_i)^2} \quad (2.15)$$

Manhattan distance (d_M)

$$d_M(x, y) = \sum_{i=1}^k |x_i - y_i| \quad (2.16)$$

Hamming distance (d_H)

$$d_H(x, y) = \sum_{i=1}^k |x_i - y_i| \quad (2.17)$$

$$x = y \Rightarrow d_H = 0$$

$$x \neq y \Rightarrow d_H = 1$$

Donde \mathbf{x} e \mathbf{y} son dos vectores de características que se desean comparar. Cada uno representa una instancia en el espacio de características y k el número de dimensiones o características de los vectores \mathbf{x} e \mathbf{y} .

Árboles de Decisión

Los árboles de decisión (*Decision Trees*) son modelos predictivos de aprendizaje supervisado que se utilizan tanto para clasificación como para regresión. Están compuestos por nodos de decisión que aplican una serie de reglas lógicas para dividir un conjunto de datos en subconjuntos homogéneos. Entre los más conocidos están el “*random forest*” o el “*bootstrap*”. Cada nodo del árbol elige la característica que mejor divide los datos según una métrica como la entropía o el índice de *Gini* [91–93].

Las ecuaciones para estas métricas vienen dadas por:

Entropía (E)

$$E(S) = - \sum_{i=1}^n p_i * \log(p_i) \quad (2.18)$$

Donde p_i es la proporción de ejemplos de la clase i en el conjunto de datos. i son las clases. Y n es el número de observaciones.

Índice Gini (G), mide la impureza de un conjunto de datos:

$$G(S) = 1 - \sum_{i=1}^n p_i^2 \quad (2.19)$$

Los árboles de decisión son útiles para la clasificación de señales EEG debido a su interpretabilidad. Estos algoritmos se emplean en tareas de clasificación cuando se busca entender claramente las decisiones del modelo. En este trabajo, aparte de los ya mencionados, se han utilizado también; *Fine-Tree*, *Medium Tree* y *Coarse Tree* de la herramienta “*Classification learner*” de MATLAB.

Regresión Logística

La regresión logística es un modelo probabilístico utilizado para modelar la probabilidad de que un evento ocurra, particularmente en problemas de clasificación binaria. La relación entre las variables independientes \mathbf{X} y la probabilidad de la clase objetivo p , se modela a través de la función logística o sigmoide [94, 95]:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right) = \beta_0 + \sum_{i=1}^n \beta_i x_i \quad (2.20)$$

Donde p es la probabilidad del evento, x_i son las variables predictoras y β_i son los coeficientes estimados.

Aunque más simple, es un modelo estadístico efectivo en la clasificación de señales EEG cuando se busca una solución interpretable y eficiente en términos computacionales.

Análisis Discriminante Lineal (LDA)

El LDA es una técnica de clasificación supervisada que busca proyectar las variables independientes en un subespacio de menor dimensión que maximiza la separación entre diferentes clases. Se basa en encontrar una combinación lineal de variables que maximice la relación entre la varianza entre clases y la varianza dentro de las clases [96, 97].

La fórmula del LDA busca maximizar la razón de la varianza entre clases S_B y la varianza dentro de las clases S_W :

$$LDA = \max\left(\frac{w^T S_B w}{w^T S_W w}\right) \quad (2.21)$$

Donde:

- S_B es la matriz de dispersión entre clases (*between-class scatter matrix*). Mide la dispersión entre la media de cada clase y la media general.

- S_W es la matriz de dispersión dentro de las clases (*within-class scatter matrix*). Se calcula sumando las matrices de covarianza de cada clase ponderadas por el número de muestras de cada clase.
- w es el vector de pesos o proyección lineal que estamos buscando para maximizar la separación entre las clases. Añadir el superíndice T es para indicar la traspuesta.

Este clasificador es popular para la clasificación de señales EEG debido a su simplicidad y capacidad para separar clases linealmente en escenarios de clasificación binaria o multclasificación.

Perceptrón Multicapa (MLP)

El perceptrón es el modelo de red neuronal más simple, con una única capa de salida. Calcula una combinación lineal de las entradas y luego aplica una función de activación binaria para determinar la clase [98]. Ver figura 2.10.

En tipo de red neuronal *feedforward* que se usa comúnmente en el procesamiento de señales EEG. Aunque más básico que los modelos de *Deep Learning*, sigue siendo útil para problemas sencillos de clasificación.

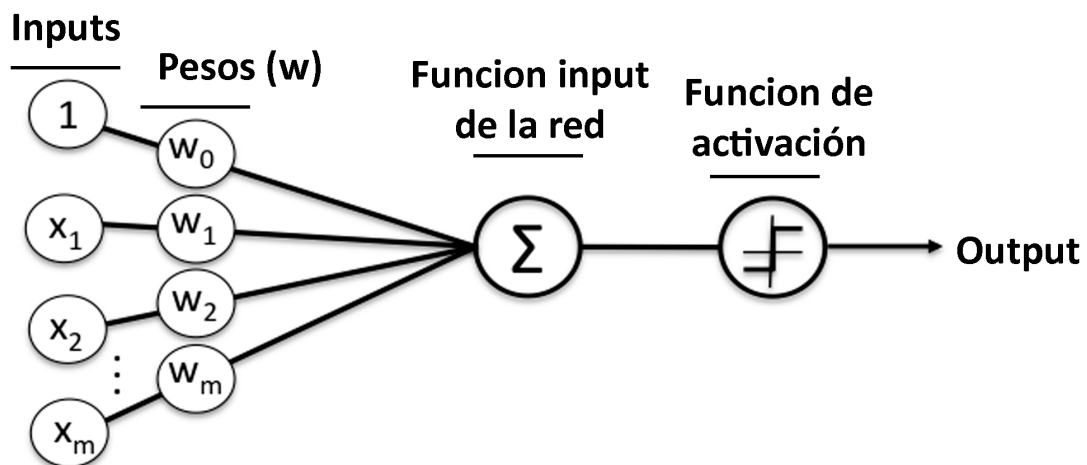


Figura 2.10. Esquema del funcionamiento interno de un perceptrón con una sola capa.

Clasificador *Naive Bayes*

El clasificador Naive Bayes es un modelo probabilístico basado en el Teorema de Bayes, ampliamente utilizado en problemas de clasificación, particularmente en tareas de procesamiento de lenguaje natural (*NLP*), filtrado de spam, y otras áreas donde la independencia entre las variables es una suposición razonable. Este clasificador es "naive" (ingenuo) porque asume que todas las características (atributos) son independientes entre sí, dado el valor de la clase. Esta suposición simplifica enormemente el cálculo de probabilidades, aunque puede ser una simplificación extrema en muchas aplicaciones [99–101].

Las señales EEG suelen presentar mucho ruido y artefactos. Los clasificadores *Naive Bayes* pueden tener un buen desempeño ante el ruido, proporcionando buenos resultados en la práctica, incluso cuando las características no son completamente independientes. Además, este tipo de clasificadores se comportan bien frente a la clasificación multiclase, lo que puede ser útil en problemas de EEG, como la clasificación de estados cognitivos o el control de dispositivos mediante *BCI*'s.

También presenta facilidad para modelar distribuciones al usar una distribución gaussiana. Esto puede ser útil para modelar características continuas en señales EEG (como amplitud y frecuencia), lo que le permitiría capturar patrones cerebrales de manera efectiva. Las ecuaciones detrás de este modelo son de forma simplificada las siguientes [99–101]:

- **Teorema de Bayes**

$$P(\mathbf{C} | \mathbf{X}) = \frac{P(\mathbf{X} | \mathbf{C}) \cdot P(\mathbf{C})}{P(\mathbf{X})} \quad (2.22)$$

Se calcula la probabilidad \mathbf{P} posterior de la clase \mathbf{C} dado el conjunto de características \mathbf{X} .

Donde $P(\mathbf{X} | \mathbf{C})$ es la probabilidad condicional de observar los datos \mathbf{X} dados que pertenecen a la clase \mathbf{C} . $P(\mathbf{C})$ es la probabilidad previa de la clase (\mathbf{C}), (de que un dato pertenezca a esta). Y, por último, $P(\mathbf{X})$ es la probabilidad marginal de observar los datos \mathbf{X} sin que pertenezcan a ninguna clase específica.

- **Suposición de Independencia Condicional.**

El clasificador Naive Bayes asume que las características son independientes entre sí, lo que simplifica el cálculo de $P(X | C)$ en un producto de probabilidades individuales.

$$P(X | C) = \prod_{i=1}^n P(x_i | C) \quad (2.23)$$

Donde X es el conjunto de características $\{x_1, x_2 \dots x_n\}$ y $P(x_i | C)$ es la probabilidad condicional de la característica x_i dado que la clase es C . n es el número de observaciones de i

- **Regla de la clasificación.**

Para clasificar una nueva instancia, se selecciona la clase C que maximice la probabilidad posterior. Como $P(X)$ es constante para todas las clases, se puede ignorar y usar la fórmula:

$$C_{predicho} = \mathop{\text{arg max}}_C P(C) \prod_{i=1}^n P(x_i | C) \quad (2.24)$$

Donde *argmax* es la abreviatura de «argumento del máximo», y se utiliza en estadística para indicar el valor que maximiza la función dada (el valor de x que maximiza la función). Y « \prod » es el análogo al símbolo de sumatorio, pero para el producto.

Esta fórmula resume el proceso de clasificación en Naive Bayes; se calcula el producto de las probabilidades individuales y se selecciona la clase con la mayor probabilidad posterior.

Para la clasificación realizada en esta tesis se prueba con los clasificadores Naives de tipo *Gaussian*, *Kernel* y el método “*optimizable*” de MATLAB.

Ensamble (Métodos de Ensamble)

Los métodos de ensamble combinan múltiples modelos base para mejorar la precisión predictiva. Entre los más conocidos están *Bagging* y *Boosting*.

En *Boosting*, por ejemplo, los modelos base se entrenan secuencialmente y cada uno trata de corregir los errores de su predecesor [102, 103].

Estos algoritmos que combinan múltiples clasificadores para mejorar el rendimiento global fueron ampliamente probados en *BCI* por su capacidad de reducir el sobreajuste y manejar variaciones en los datos EEG.

En este trabajo se utilizan los *ensambles* de la herramienta "*classification learner*" de *MATLAB*. Estos son; *Boosted Trees*, *Baged Trees*, *Subspace Discriminant*, *Subspace KNN* y *RUSBoosted Tree*.

Posteriormente, se diseñan ensambles personalizados basados en combinación de modelos de *Machine* y *Deep Learning*.

2.4.2. Deep Learning

Los algoritmos de *Deep Learning* utilizados en esta tesis han sido desarrollados tanto en el entorno de *MATLAB* como en el entorno *Python*.

Redes Neuronales Convolucionales (CNN)

Las *CNN* son un tipo de algoritmo de *Deep Learning* que se utiliza comúnmente en tareas de reconocimiento de imágenes y vídeos ya que es capaz de interpretar características espaciales y en consecuencia reconocer patrones y procesar datos de los píxeles de una imagen o de una matriz de 2 dimensiones.

Esta herramienta utiliza convoluciones para analizar y extraer características de los datos de entrada. Siendo esta una de sus principales ventajas, ya que le permite tener la capacidad de aprender representaciones jerárquicas de los datos de entrada. Este aprendizaje puede ser llevado a cabo gracias al uso de múltiples capas de operaciones convolucionales y de *pooling* (agrupamiento de las características). Que sumado al uso de capas totalmente conectadas o *fully connected*, en inglés, permite que las *CNN* pueden aprender automáticamente a detectar características cada vez más complejas en las imágenes de entrada, como bordes, texturas y objetos, véase la figura 2.11, [104–107].

La expresión matemática de la convolución en una *CNN* viene dada por:

$$(f * g)(x) = \int_{-\infty}^{\infty} f(t)g(x - t)dt \quad (2.25)$$

Donde f es la función de entrada (como una imagen), t es una variable de integración, que representa un desplazamiento o posición en el dominio de f , g es el *kernel* (o filtro) que se aplica a la imagen, y x es la posición en la imagen donde se realiza la operación de convolución.

La expresión anterior se refiere a la convolución en una dimensión. Sin embargo, en las *CNN* las operaciones de convolución normalmente se utilizan en múltiples dimensiones, sobre todo en imágenes que son matrices bidimensionales como es el caso de la presente tesis.

En una *CNN*, los filtros se deslizan sobre la imagen de entrada y se aplican multiplicaciones por elementos; a continuación, los resultados de estas multiplicaciones se suman para producir una nueva imagen denominada mapa de características o mapa de activación. Esto se hace varias veces con distintos filtros para extraer características cada vez más complejas de la imagen de entrada.

Además, las operaciones de *pooling* anteriormente mencionadas, permiten reducir la dimensionalidad de la imagen, lo que ayuda a evitar el sobreajuste y aumenta la eficiencia de la red [19].

Por lo tanto, la expresión matemática de las *CNNs* se basa en operaciones de convolución y *pooling*, que se aplican múltiples veces en diferentes capas de la red para extraer características cada vez más complejas de la imagen de entrada, véase figura 17.

La expresión matemática para una de estas capas 2D en una *CNN* vendría dada por [108]:

$$O_{i,j,k} = f \left(\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1} I_{i+m,j+n,c} * K_{n,m,c,k} + b_k \right) \quad (2.26)$$

Donde, \mathbf{O} es el tensor de salida de dimensiones (I_0, J_0, K) . I_0, J_0 son dimensiones espaciales de salida como alto y ancho y k es el número de filtros o mapas de características.

« $\sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \sum_{c=0}^{C-1}$ » son sumas anidadas **que recorren** los índices m (altura), n (anchura) y c (los canales de entrada, profundidad)

I , es el tensor de entrada (por ejemplo, una imagen) de dimensiones (I_i, J_i, c) , donde C son canales de entrada. $K_{m,n,c,k}$ es el tensor del filtro de convolución de dimensiones (m, n, c, k) , f es la función de activación (por ejemplo, *ReLU*), b_k es el tensor de sesgo asociado al filtro k . La suma se realiza sobre los índices m, n y c para recorrer el tensor de entrada y el tensor del núcleo.

Las *CNN* son uno de los enfoques más exitosos en *BCI* y EEG, ya que han sido ampliamente utilizadas para la extracción automática de características espaciales, siendo capaces de detectar patrones específicos en las señales EEG y logrando, por lo tanto, una clasificación más eficiente.

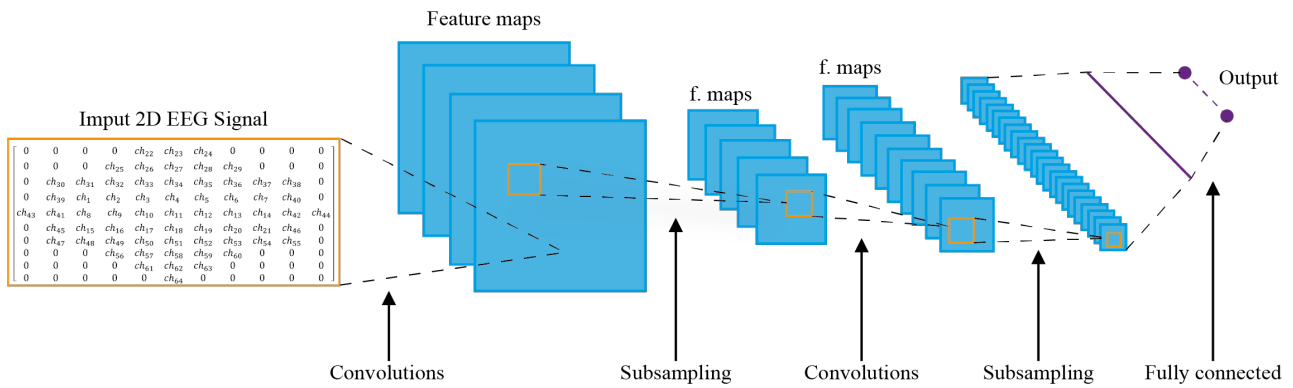


Figura 2.11. Aplicación del algoritmo de la CNN a una matriz de datos de 2 dimensiones. Se puede observar cómo se realizan varias convoluciones que crean los mapas de características. Finalmente, una capa *fully connected* permite generar un solo output [19].

Redes Neuronales Recurrentes (RNN)

Las RNN son un tipo de arquitectura de red neuronal diseñada para manejar datos secuenciales o series temporales. A diferencia de las redes *feedforward*, vistas en el apartado 2.3, las RNN introducen conexiones recurrentes, lo que les permite "recordar" información de estados previos, capturando dependencias temporales en los datos. Cada neurona en una RNN tiene una conexión hacia sí misma, lo que permite que la información fluya no solo hacia adelante, sino también hacia atrás a través del tiempo [109, 110], ver figura 2.12.

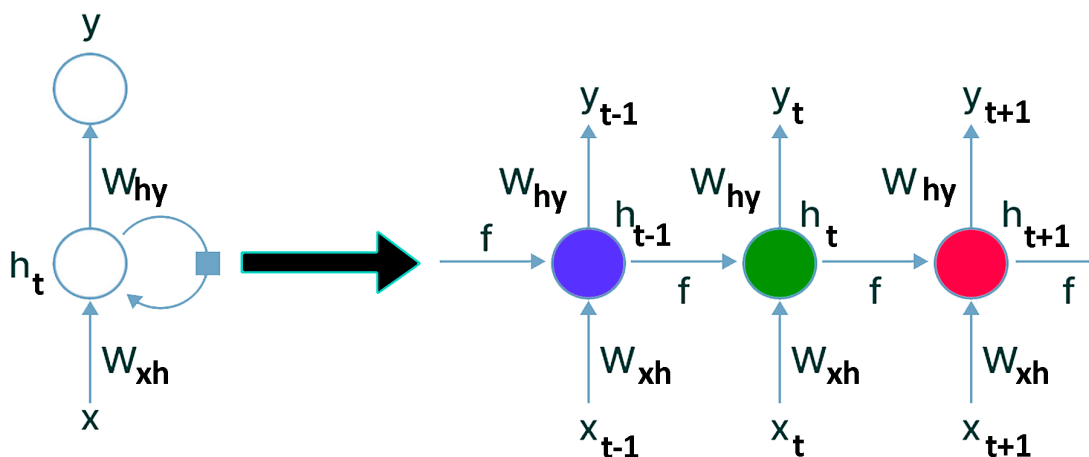


Figura 2.12. Esquema desglosado de una red RNN. Las flechas representan la matriz de pesos W_h

La fórmula básica para el cálculo del estado oculto de una RNN puede expresarse de la siguiente manera:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (2.27)$$

Donde:

- h_t es el estado oculto en el tiempo t
- x_t es la entrada en el tiempo t
- W_{xh} es la matriz de pesos entre la entrada y el estado oculto
- W_{hh} es la matriz de pesos recurrentes entre el estado oculto anterior h_{t-1} y el estado actual
- b_h es el sesgo asociado al estado oculto
- f es la función de activación (típicamente \tanh o $ReLU$).

Si se desea conocer la salida y_t en algún punto la ecuación matemática a utilizar es:

$$y_t = g(W_{hy}h_t + b_y) \quad (2.28)$$

Donde W_{hy} corresponde a la matriz de pesos que conecta el estado oculto h_t con la salida y_t y g es una función de activación como softmax o una función lineal, b_y es el sesgo asociado.

El estado oculto h_t captura la información relevante del pasado, lo que permite que las RNN sean efectivas para tareas que requieren memoria, como el procesamiento del lenguaje natural y las series temporales. También son especialmente útiles para procesar señales EEG como series temporales, ya que pueden mantener una memoria a corto plazo y modelar dependencias temporales en los datos.

Long Short-Term Memory (LSTM)

Las LSTM son una de las redes neuronales más utilizadas en el campo de la IA para el procesamiento o la predicción de series temporales. Es un tipo avanzado de RNN que está diseñado para manejar secuencias de datos más largas ya que utilizan una célula de memoria (C_k), que puede almacenar y transmitir información de estados anteriores y

actuales para su uso futuro. Adicionalmente, las *LSTM* utilizan una serie de puertas algorítmicas capaces de retener información útil de estados anteriores o actuales y actuar en función del objetivo.

Existen variantes de estas redes, como lo son las *LSTM*-bidireccionales (*BiLSTM*) o las unidades recurrentes de compuerta (*GRU*) que presentan funciones ligeramente diferentes.

Las *LSTM* se aplican a tareas como el reconocimiento de la escritura a mano, el reconocimiento del habla, la epidemiología, los videojuegos y la predicción meteorológica, entre otras. También han demostrado ser eficaces en el manejo de las señales del EEG, al tener la habilidad de capturar las dependencias temporales del largo plazo.

A su vez, es común ver a las redes *LSTM* combinadas con otras técnicas para mejorar la arquitectura general de la red, como es el caso de la presente tesis.

El algoritmo *LSTM* se compone principalmente de tres partes (puerta de entrada o *input gate*, puerta de salida u *output gate* y puerta de olvido o *forget gate*) y una célula de memoria. Sus expresiones matemáticas son las siguientes [111–113]:

- **Input gate** (i_t): es la capa de entrada a la neurona, encargada de actualizar el estado de la red mediante la función sigmoideal.

$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2.29)$$

W_i es la representación del peso de la entrada, b_i representa el sesgo correspondiente, x_t es el paso de tiempo actual y h_{t-1} es la salida del paso de tiempo anterior. σ tendrá un valor $\in [0, 1]$, que representa descartar completamente o guardar completamente los datos, respectivamente [111–114].

- **Forget gate** (f_t): es la capa encargada de decidir si guardar o descartar la información. Este es el primer paso de la *LSTM*.

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \quad (2.30)$$

W_f es la representación del peso de la entrada, b_f representa el sesgo correspondiente, x_t es el paso de tiempo actual y h_{t-1} es la salida del paso de tiempo anterior.

- **Output gate (o_t):** aquí es donde se determina la salida de información. Esta salida se basa en la versión filtrada del estado de la célula. La capa sigmoidea decide el valor de salida y lo multiplica por el estado de la célula [113].

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (2.31)$$

$$h_t = o_t \cdot \tanh(C_t) \quad (2.32)$$

W_o es la representación del peso de la entrada, b_o representa el sesgo correspondiente y x_t es el paso de tiempo actual. h_{t-1} es la salida de la capa LSTM en el paso de tiempo actual y σ fue definido en «2.29». Por último, el estado anterior de la celda C_{t-1} , debe ser actualizado (C_t). Esto se calcula a través del olvido de una puerta de entrada, como se muestra en la figura 2.13.

$$C_t = f_t \cdot C_{t-1} + i_t g_t \quad (2.33)$$

Donde g_t es una capa *tanh*

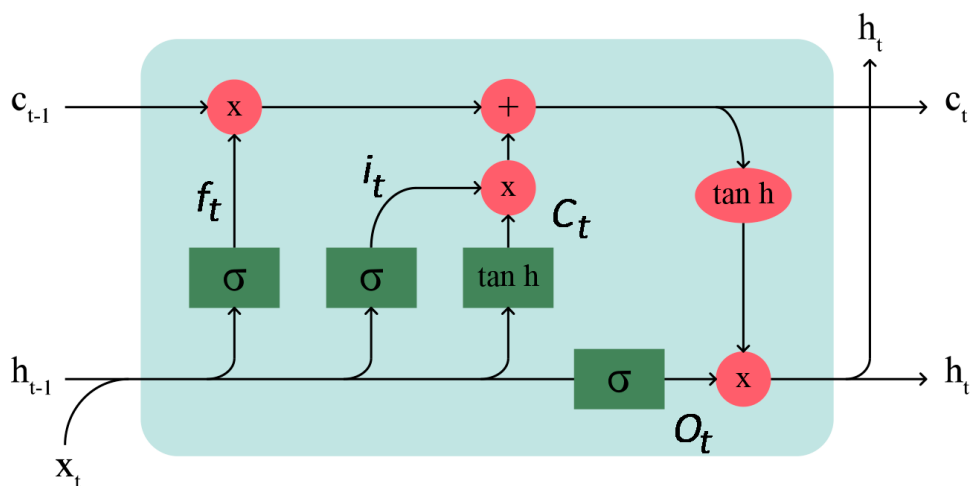


Figura 2.13. Esquema de una celda LSTM.

Unidades GRU

Las unidades *GRU* (*Gated Recurrent Units*) son una variante más ligera de las *LSTM*. Fueron introducidas para resolver problemas como el desvanecimiento del gradiente que dificulta el aprendizaje en secuencias largas. Las *GRU* son más simples que las *LSTM* y tienen menos parámetros, ya que combinan la celda de entrada y de olvido en una sola unidad, lo que las hace computacionalmente más eficientes [115–117].

La *GRU* utiliza dos compuertas principales; la compuerta de actualización y la compuerta de reinicio. Estas se definen matemáticamente como:

- **Compuerta de actualización (z_t):**

$$z_t = \sigma (W_z \cdot x_t + U_z h_{t-1}) \quad (2.34)$$

Donde z_t controla cuánto del estado anterior (h_{t-1}) debe ser conservado en el estado actual. W_z es la matriz de pesos que conecta la entrada x_t , con la compuerta de actualización. U_z es la matriz de pesos que conecta el estado oculto anterior (h_{t-1}), con la compuerta de actualización.

- **Compuerta de reinicio (r_t):** es la capa de entrada a la neurona, encargada de actualizar el estado de la red mediante la función sigmoïdal (σ).

$$r_t = \sigma (W_r \cdot x_t + U_r h_{t-1}) \quad (2.35)$$

Donde r_t controla cuánto de la información pasada se olvida. W_r es la matriz de pesos para la entrada x_i en la compuerta de reinicio y U_r es la matriz de pesos para el estado oculto en la compuerta de reinicio.

- **El estado oculto (h_t) se actualiza como:**

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tilde{h}_t \quad (2.36)$$

Donde:

- « \circ » es el producto de *Hadamard* [118].

- $\tilde{h}_t = \tanh(W_h \cdot x_t + U_h \cdot (r_t \circ h_{t-1})) \quad (2.37)$

$\tilde{\mathbf{h}}_t$ es el nuevo estado candidato a ser \mathbf{h}_t calculado usando la puerta de reinicio. \mathbf{W}_h es la matriz de pesos para la entrada \mathbf{x}_t y \mathbf{U}_h la del estado oculto (\mathbf{h}_{t-1}). Por último, « $\mathbf{r}_t \circ \mathbf{h}_{t-1}$ » es la multiplicación elemento a elemento entre el estado oculto y la puerta de reinicio controlando cuánto del estado anterior se considera para el cálculo del nuevo estado ($\tilde{\mathbf{h}}_t$).

Las *GRU* al igual que las *LSTM* también son utilizadas en EEG para modelar dependencias temporales, con la ventaja de tener menos parámetros lo que podría llegar a significar una ventaja en ciertos casos.

LSTM Bidireccional (*BiLSTM*)

Las Redes *BiLSTM* son una extensión de las *LSTM* tradicionales que tienen la capacidad de procesar secuencias de datos en ambas direcciones, es decir, desde el principio hacia el final y desde el final hacia el principio. Esto permite que el modelo tenga en cuenta tanto el contexto pasado como el futuro en cada paso de la secuencia, lo que mejora significativamente el rendimiento en tareas donde la información de los extremos de la secuencia es importante [119–121].

La arquitectura *BiLSTM* consiste en dos capas *LSTM*: una que procesa la secuencia de manera **forward** y otra que la procesa de manera **backward**. La fórmula para el cálculo de las salidas en ambas direcciones es:

$$\vec{h}_t = LSTM(x_t, \vec{h}_{t-1}) \quad (2.38)$$

$$\overleftarrow{h}_t = LSTM(x_t, \overleftarrow{h}_{t+1}) \quad (2.39)$$

Las salidas se combinan para formar la representación final:

$$h_t = [\vec{h}_t, \overleftarrow{h}_t] \quad (2.40)$$

Dónde:

- \vec{h}_t es el estado oculto en la dirección *forward*.
- \overleftarrow{h}_t es el estado oculto en la dirección *backward*.
- x_t vector de entrada

Capas de Atención (AL)

Las capas de atención o *Attention Layers* son mecanismos de redes neuronales diseñados para enfocarse en las partes más relevantes de una secuencia de datos, permitiendo que el modelo aprenda a ponderar ciertos elementos de la secuencia más que otros.

Este mecanismo ha demostrado ser particularmente útil en tareas que involucran secuencias largas o datos en los que no todas las partes de la secuencia son igualmente importantes.

El mecanismo de estas capas de atención permite que la red neuronal aprenda relaciones dependientes entre diferentes partes de la secuencia, lo que mejora el rendimiento en tareas como la traducción automática, el reconocimiento de patrones o el *NLP*. Siendo el mecanismo de autoatención (*self-attention*) uno de los que se encuentra detrás del famoso “Chat-GPT” [122–124].

En la presente tesis se aplica de manera exitosa al ámbito de las señales EEG, donde el patrón de la señal varía a lo largo del tiempo, y no todo el muestreo es relevante.

Para entender su funcionamiento se toma una sola capa de atención. Esta, toma como entrada una matriz de datos o una secuencia de vectores y produce una secuencia de vectores de atención. A estos vectores de atención se les asigna un peso calculado mediante una serie de operaciones, ponderándolos. Luego se realizan una serie de transformaciones (productos escalares, *scores*, concatenaciones...) para definir los valores resultantes y finalmente se obtiene la salida en forma lineal [122]. Véase la figura 2.14.

Donde X es una matriz de datos, Q (*query o consulta*), K (*key o clave*) y V (*value o valor*) son las subredes de la *attention head* (zona donde se separan y ponderan los vectores o datos mediante diferentes operaciones), w indica que las subredes están ponderadas y, finalmente, tras los productos escalares se realiza una concatenación mediante *softmax* para obtener un vector lineal.

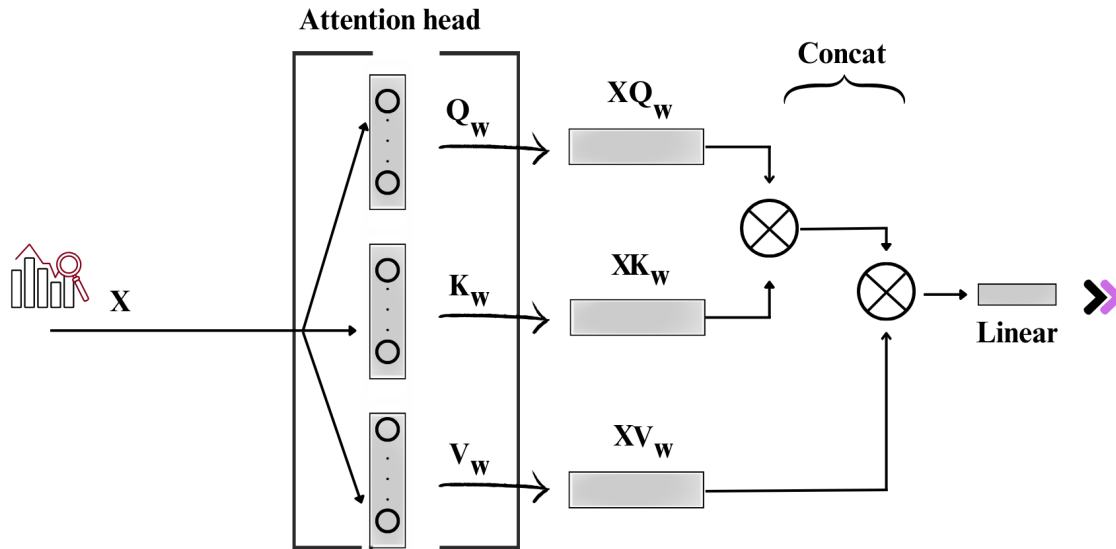


Figura 2.14. Esquema de una capa de atención.

Matemáticamente esto se expresa de la siguiente manera. Tras recibir la matriz de datos o vectores, se mide la similitud entre Q y K con un score, donde T sería el índice del token o vector específico que se esté evaluando:

$$Score(Q, K) = QK^T \quad (2.41)$$

Seguidamente se calculan los pesos de atención utilizando la función **softmax** y el producto escalar (a_{ij}).

$$a_{ij} = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (2.42)$$

Donde d_k es la dimensionalidad de las claves K , lo que evita que los productos escalares se vuelvan demasiado grandes.

Finalmente se calcula el valor ponderado utilizando los valores V y se concatenan para dar una salida

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.43)$$

Como se expuso anteriormente, en el análisis de **señales EEG**, el mecanismo de atención ha demostrado ser útil para **identificar las características temporales más relevantes de la señal** en lugar de tratar todas las partes de la secuencia de EEG por igual. Por ello, son una herramienta a tener en cuenta, ya que las señales EEG pueden tener picos de actividad en momentos específicos que son de mayor importancia para la clasificación o el análisis (como en tareas de intención motora o identificación de patrones específicos). En la presente tesis se utilizan estas capas para conformar arquitecturas tanto en solitario como combinadas con otro tipo de algoritmos y probar si su uso está justificado o no.

Transformers

Un *Transformer* es una arquitectura completa de *Deep Learning* que usa varias capas de atención combinadas con otras capas (como redes *feedforward*) y trabaja de manera paralela para procesar secuencias de datos de manera eficiente y escalable. Ha sido diseñada para manejar tareas secuenciales como el procesamiento del lenguaje natural, la traducción automática y otras aplicaciones de series temporales. Siendo la base de modelos como **GPT** y **BERT** [125, 126].

Los *Transformers* son altamente paralelizables, lo que mejora considerablemente la eficiencia computacional y el rendimiento en tareas de secuencia larga. El *Transformer* se basa en el mecanismo de **autoatención (*self-attention*)**, que permite a la red ponderar diferentes partes de la secuencia de entrada para aprender dependencias a largo plazo, algo que las redes recurrentes tradicionales no manejan de manera eficiente [122].

La arquitectura *Transformer* está compuesta principalmente de dos bloques, ver figura 2.15:

1. **Encoder**: Procesa la secuencia de entrada y genera una representación interna.
2. **Decoder**: Toma la representación interna generada por el *encoder* y produce la secuencia de salida.

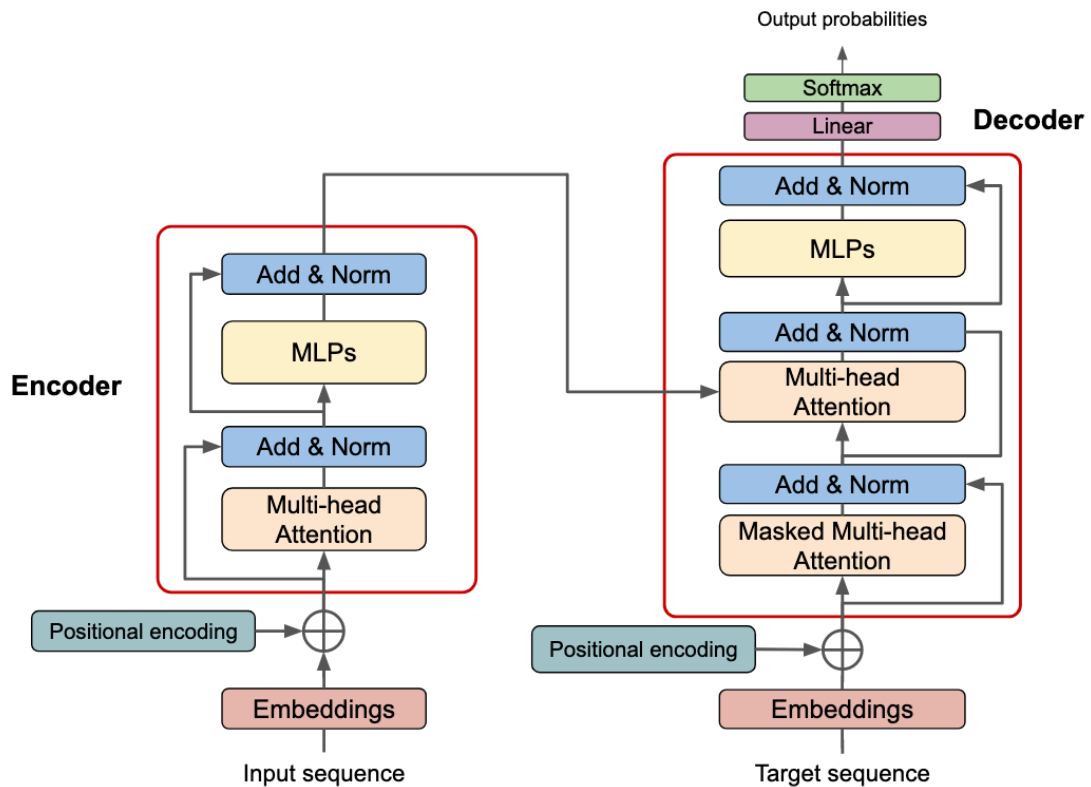


Figura 2.15. Esquema de la arquitectura de un *transformer* obtenida del trabajo de Nyandwi [128].

Ambos bloques están compuestos por múltiples capas, con subcomponentes que se explican a continuación [122], [125–127]:

1. **Mecanismo de Autoatención.** La base del *transformer* es el mecanismo de atención ya visto en la ecuación (2.40), y que se aplica a la capa *encoder* y a la *decoder*. Cada palabra (o *token*) en la secuencia de entrada se representa mediante una consulta (Q) una clave (K) y un valor (V).
2. **Multi-head Attention:** Para mejorar la capacidad del modelo de capturar diferentes aspectos de la secuencia, los *Transformers* utilizan múltiples cabezas de atención ("multi-head attention"). Esto significa que, en lugar de aplicar una única capa de atención, se aplican varias capas en paralelo, cada una enfocada en diferentes partes de la secuencia.

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^0 \quad (2.44)$$

Donde W^0 es una matriz de proyección de salida y cada cabeza (head_i) se define como una capa de atención:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2.45)$$

Siendo W_i^Q, W_i^K, W_i^V las matrices de proyección aprendidas por cada cabeza. Una **matriz de proyección** transforma o reorganiza los datos en un espacio de menor dimensión, donde ciertas propiedades son más fáciles de capturar.

3. **Feedforward Network (FFN)**: Después del mecanismo de atención, se utiliza una red *feedforward* en cada capa del *Transformer*. Esta red *feedforward* es aplicada de manera independiente a cada posición en la secuencia y es la misma para todas las posiciones. Se aplica una transformación no lineal a cada *token* de la secuencia.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \quad (2.46)$$

La primera capa aplica, al vector de entrada x , una transformación lineal usando W_1 (matriz de pesos de la primera capa) y b_1 (vector de sesgos de la primera capa) usando una función ReLU para introducir no linealidad (devuelve « $xW_1 + b_1$ », si es positivo y 0 si es negativo). La salida de la primera capa (no lineal) es transformada nuevamente con W_2 (matriz de pesos de la segunda capa) y b_2 (vector de sesgos de la segunda capa) para obtener el vector de salida final.

4. **Positional Encoding (PE)** :: Dado que el *Transformer* no tiene componentes recurrentes o convolucionales, no puede tener una noción explícita del orden de los *tokens*. Para resolver esto, se añaden **codificaciones posicionales** a los vectores de entrada. Estas codificaciones, que se suman a las representaciones de entrada, permiten que el modelo aprenda relaciones posicionales, es decir, da un orden a los componentes de entrada o *tokens*. Esta, se expresa según:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right), PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \quad (2.47)$$

Donde pos es la posición del token, i es el índice de la dimensión, y d_{model} es la dimensionalidad del modelo.

En el campo del análisis de señales EEG, los *Transformers* se están utilizando cada vez más para identificar patrones temporales y espaciales complejos, ya que pueden capturar dependencias a largo plazo en las señales cerebrales. Los modelos basados en *Transformers* han mostrado resultados prometedores en tareas como la clasificación de estados cognitivos y control basado en EEG, superando en algunos casos a las RNN tradicionales.

Vision Transformer (ViT)

El *Vision Transformer* adapta el mecanismo de atención para trabajar con imágenes en lugar de texto. Mientras que el *transformer* original procesa secuencias de *tokens* de texto, el *ViT* procesa imágenes. Sin embargo, las imágenes no son tratadas directamente como un conjunto de píxeles; en su lugar, se dividen en pequeños parches o segmentos (subimágenes), que se tratan como si fueran "*tokens*" en una secuencia, similar a cómo se tratan las palabras en *NLP* [129, 130].

Por ejemplo, una imagen de tamaño 224×224 píxeles, podría dividirse en segmentos de tamaño 16×16. Así, una imagen completa se convertiría en una secuencia de 196 "*tokens*" (segmentos) donde cada token representa un pequeño bloque de la imagen.

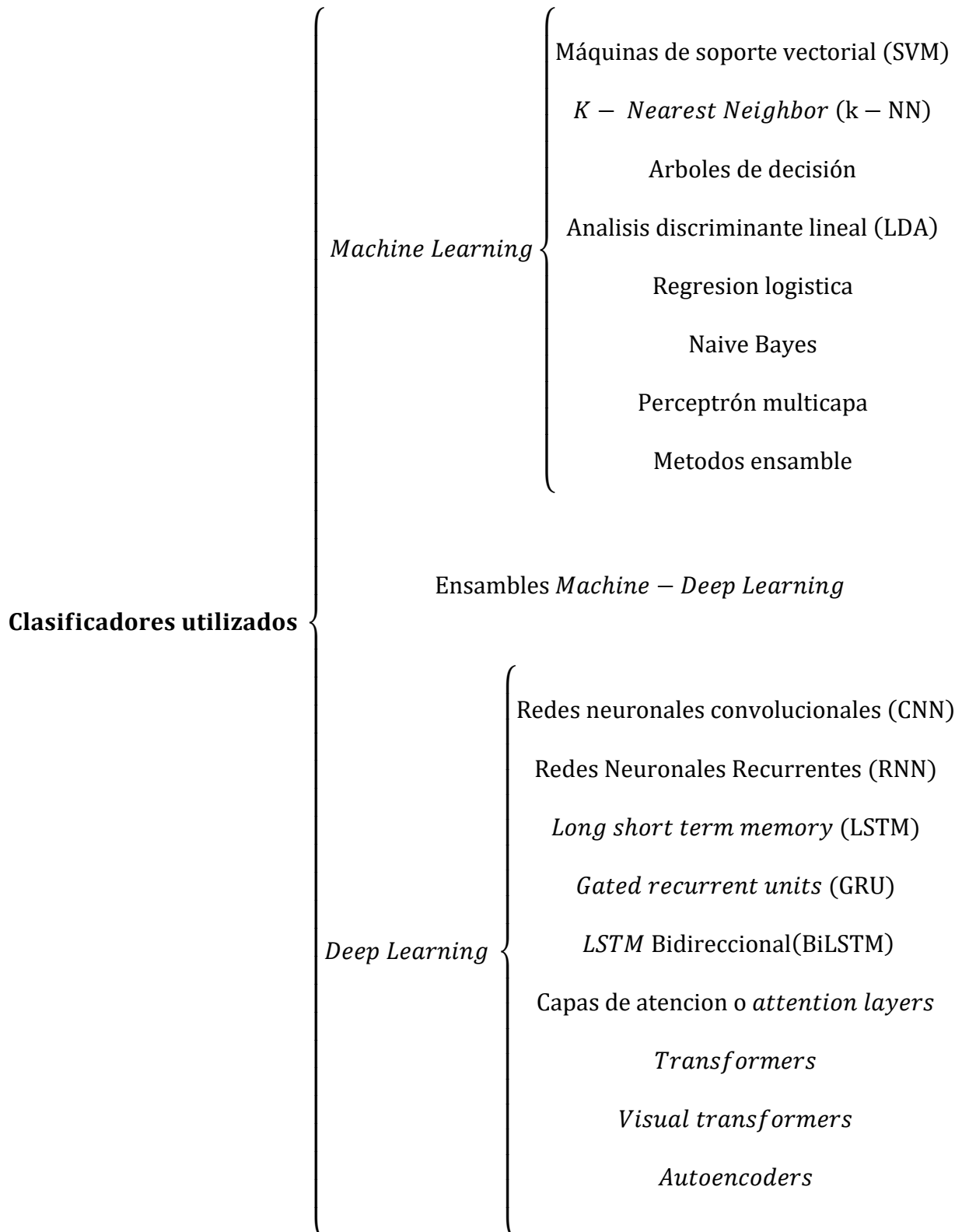
Al igual que en los *transformers* de texto, se utiliza un *positional encoding* para indicar la posición de cada segmento dentro de la imagen y se aplica el mismo mecanismo de autoatención para que cada segmento de la imagen pueda atender a cualquier otro segmento. Esto permite que el modelo capture relaciones globales en la imagen, lo que es útil para tareas como la clasificación de imágenes o las relaciones entre las señales 2D del EEG.

Autoencoders

Los *Autoencoders* son un tipo de red neuronal no supervisada diseñada para aprender una representación comprimida de los datos de entrada. Están compuestos de dos partes: el **encoder**, que comprime la entrada en una representación latente, y el **decoder**, que intenta reconstruir la entrada original a partir de esta representación comprimida. Esto es especialmente útil para **reducir el ruido** de una serie de datos, **detectar anomalías** en estos o captar patrones latentes o subyacentes en la señal del conjunto de datos [131, 132].

En este trabajo los *autoencoders* han sido utilizados de forma sencilla pero no se profundiza en su uso, por lo que no se entra en el detalle matemático subyacente tras estos.

Se muestra a continuación un resumen con todos los algoritmos de *Machine* y *Deep Learning* utilizados a lo largo de la tesis, ya sea como arquitectura en solitario o formando parte de un conjunto de mayor complejidad:



Capas de apoyo en la red

En la aplicación de todas las arquitecturas de *Deep Learning* anteriores **es muy común el uso de ciertas capas que permiten la fluidez entre todas las partes del modelo y el correcto funcionamiento de este**. Algunas de las más relevantes y que han sido ya mencionadas son:

- **Capas *Fully-connected***

Una capa *fully connected*, o totalmente conectada en español, es **una capa neuronal en la que cada neurona está conectada a todas las neuronas de la capa anterior**. Su función principal es **combinar de manera no lineal las características aprendidas por capas anteriores** y poder así ayudar a realizar una predicción final. La salida de una capa FC se define matemáticamente como [108], [133]:

$$\mathbf{y} = f \left(\sum_{i=1}^n (W_i \cdot x_i) + b \right) \quad (2.48)$$

Donde x_i es el vector de entrada, W_i es la matriz de pesos, b es el vector de sesgos, f es una función de activación (como *ReLU*, *Sigmoid*, etc.), n es el número de observaciones total e \mathbf{y} es la salida de la capa.

La principal ventaja de las capas *fully connected* es que **permiten capturar relaciones complejas entre las características**, pero a costa de un mayor número de parámetros, lo que aumenta el riesgo de sobreajuste si no se gestionan adecuadamente.

- **Capas *Dropout***

La capa *dropout* es una técnica de regularización que se utiliza para prevenir el sobreajuste en redes neuronales. En cada iteración de entrenamiento, se eliminan (o "desconectan") aleatoriamente un porcentaje de las neuronas, lo que evita que el modelo dependa demasiado de neuronas específicas y **obliga a la red a aprender patrones más robustos** [134].

El mecanismo de una capa *dropout* es bastante sencillo, **consiste en multiplicar el vector de entrada a una capa por una máscara**. Esta máscara es una máscara binaria que **se genera aleatoriamente para cada iteración de entrenamiento**, tiene la misma forma que la entrada y **cada elemento es 0 o 1**. La probabilidad de que cada elemento de la máscara sea 1 se denomina probabilidad de abandono. La probabilidad de que cada elemento de la máscara sea 1 se denomina tasa de abandono. **La tasa de abandono es un hiperparámetro que suele fijarse entre 0,2 y 0,5**, dependiendo de la aplicación específica y de la complejidad del modelo [135, 136].

Durante la fase de prueba, es normal utilizar una tasa de abandono de 0, lo que significa que **todas las neuronas están activas**. Esto se debe a que **el abandono sólo se aplica durante la fase de entrenamiento** y no se utiliza durante la fase de prueba.

▪ Capas *Softmax*

La capa *softmax* se utiliza principalmente en la salida de una red neuronal para problemas de clasificación multicategoría. **Esta capa convierte las salidas de la red en probabilidades, asegurando que la suma de las probabilidades de todas las clases sea igual a 1**. La función *softmax* se define como [137, 138]:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}} \quad (2.49)$$

Donde z_i es la salida de la neurona i , N es el número total de clases y $\sigma(z_i)$ es la probabilidad predicha para la clase i

3. Metodología experimental

A lo largo del desarrollo de la tesis y para cumplir la hipótesis y los objetivos descritos en el apartado “1.4”, se han comprobado decenas de artículos científicos indexados en *JCR* y se han conseguido plantear diversas metodologías basadas en el estado del arte.

A continuación, se describen estas múltiples metodologías y las diferentes arquitecturas que han surgido fruto de esta investigación. Estas han sido utilizadas para construir sistemas de clasificación de la señal EEG, que contiene la información de la intención motora. Con ello se plantea superar el del éxito en la clasificación del estado arte actual y demostrar la hipótesis propuesta.

3.1. Esquema General

La metodología de trabajo sigue un **esquema común** en todos los experimentos y metodologías que se probaron a lo largo de la tesis. Los puntos de este esquema general son:

- 1- **Se parte siempre de la base de datos *Physionet***, vista en el apartado 2.1. «Base de datos», de donde se extrae la señal EEG.
- 2- Posteriormente se hace un procesado de la señal cruda del EEG, en donde **se transforma la señal de formato *EDF+* a un formato interpretable** por los lenguajes de programación. En este caso MATLAB o Python. También se realizan otro tipo de procesados, según el caso. Estos han sido detallados en el apartado 2.2. «Procesado de la señal».
- 3- A continuación, se lleva a cabo la **extracción de características de la señal «vistas en el apartado 2.3.»**, según si el caso lo requiere o no. En este punto hay casos donde la extracción se realiza en conjunto con el procesado de la señal.
- 4- Por último, se implementa la arquitectura mediante distintos tipos de algoritmos «apartado 2.4.» y **se realiza la clasificación de la intención motora.**

Puede verse un diagrama de flujo de la metodología general en la figura 3.1.

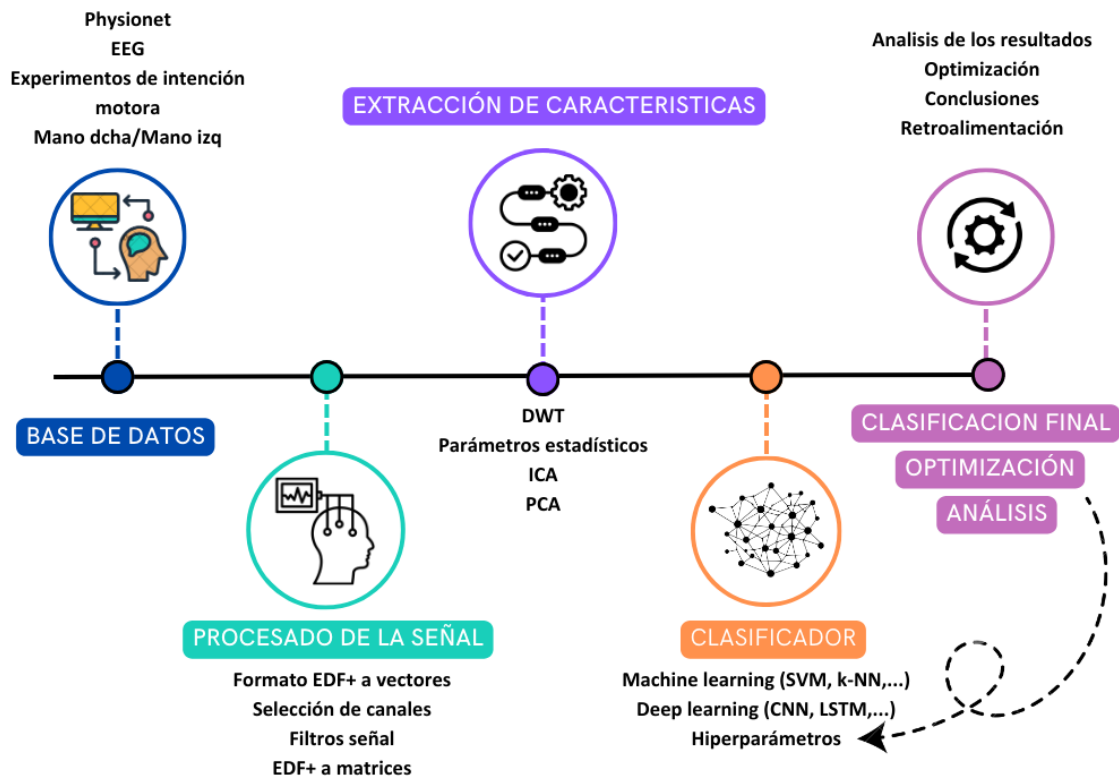


Figura 3.1. Diagrama de flujo general de la metodología experimental seguida en la tesis.

3.2. MATLAB. Arquitecturas propuestas

En una etapa inicial, el *software* elegido para llevar a cabo el desarrollo de los experimentos fue *MATLAB*. Esto se debe a que **presenta herramientas intuitivas y muy útiles** como *BCILAB*, *Deep Network Designer* o *classification learner*. Estas permiten visualizar datos para su análisis, crear arquitecturas y llevar a cabo clasificaciones en paralelo usando *Machine Learning*.

En dicha etapa inicial de la tesis, no existían comandos de conversión de formato de *EDf+* a vectores o matrices en *MATLAB*. La solución fue utilizar el software de visualización de EEG, *EDF reader* (visto en el apartado 2.1) para exportar los datos de *EDF+* a formato “.txt”.

Una vez **teniendo los datos en formato txt**, un algoritmo de lectura y tratamiento de los datos es diseñado, para **obtener vectores de datos de los experimentos sin los valores T0** (descansos), como fue explicado en el apartado 2.2 (figura 8). Estos vectores se juntan en matrices de vectores que **son arregladas a un tamaño uniforme** ya que,

durante el tratamiento de los datos, se comprobó que **algún sujeto presentaba datos corruptos** o se excedió en el protocolo de muestreo dando matrices con mayor tamaño. De esta manera, se consigue que **todos los vectores y matrices tengan la misma dimensión**, un factor clave en el acondicionamiento para utilizar algoritmos de *Machine* y *Deep Learning*.

Además, este vector va **acompañado de su vector de etiquetas asociado** que contiene las etiquetas T1 y T2, correspondientes con los movimientos de las manos izquierda y derecha o de las intenciones de mover estas.

A continuación, se muestra en la figura 3.2, un ejemplo del inicio del código, que contiene 3725 líneas en total y en el que, **tras obtener los datos y los vectores se aplica la selección de canales y el filtro Butterworth** anteriormente descritos en el apartado 2.2. «procesado de la señal».

```

% ----PONER CONTADOR PARA QUE NOMBRE CORRECTAMENTE EL EXP----
%Anotaciones
Folder = pwd;
filename = dir(fullfile(Folder, '*.txt')); %Cargamos TODOS los nombres de las pruebas de los 109 usuarios
filename(3:4:end,:) = []; %Eliminamos las filas "header"
filename(3:3:end,:) = []; %Eliminamos las filas signals
delimiterIn = ',';
headerlinesIn = 1;

filename1 = filename;
filename1(2:2:end,:) = []; %SOLO NOMBRE DE LOS EVENTOS ANOTACIONES
filename2 = filename;
filename2(1:2:end,:) = []; %SOLO NOMBRE DE LOS EVENTOS DATOS

EXPE = []; %Matriz vacía para crear la matriz de características
ANOT = []; %vector vacío para crear la matriz de etiquetas
m = 1;
j = 1;

%-----ANOTACIONES-----

while j <= 10 && m <= 10

    aFile = (filename1(j).name);

    An = importdata(aFile, delimiterIn); %Importar datos
    An(1,:) = []; %eliminar primera fila
    An(1:2:end,:) = []; %eliminar todos los eventos T0 quedando T1 y T2]
    AN = string(An);
    t1 = contains(AN, 'T1');
    t2 = contains(AN, 'T2');
    T1 = find(t1 == 1);
    T2 = find(t2 == 1);

    %El vector t1 tendrá un 1 en donde es t1 y un 0 donde es t2

    %TRAS CADA LOOP se obtienen los eventos T1 y T2 del sujeto y prueba (109 sujetos y x pruebas por sujeto)
    %-----DATOS-----

    bFile = (filename2(m).name);

    A = importdata(bFile, delimiterIn, headerlinesIn); %Importar datos quitando la cabecera
    aux = A.data; %Convertirlo a una matriz "Double"
    aux2 = aux(:, 1); %obtener la primera columna que será un eje

```

Figura 3.2. Fragmento del primer código desarrollado en MATLAB para tratar los datos en formato txt.

La selección de canales se muestra en la figura 3.3, en donde, según la bibliografía, se concentra la información necesaria relacionada con la intención motora.

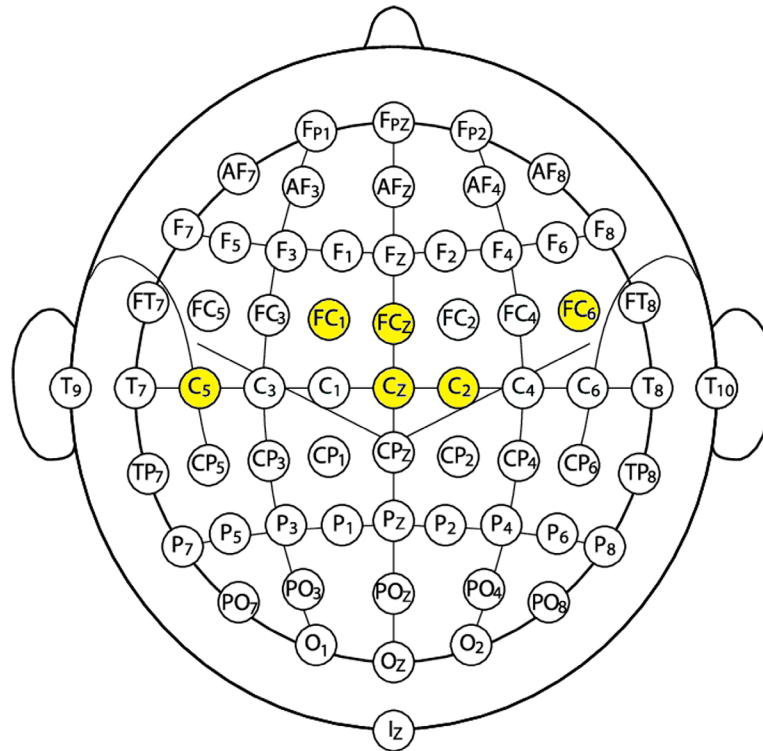


Figura 3.3. Selección de canales utilizados. Zona del lóbulo frontal asociada a la intención motora [67, 68].

Además, en este algoritmo desarrollado, se incluyen las técnicas de la extracción de características mencionadas en el apartado 2.3. Estas técnicas son aplicadas tras haber hecho la selección de canales y haber aplicado el filtro Butterworth, preparándolos para que, posteriormente, se lleven a cabo las clasificaciones.

A continuación, se detalla la aplicación y metodología llevada a cabo en la extracción de características.

3.2.1. Transformada *Wavelet* Discreta

Teniendo los vectores con los eventos T1 y T2, la selección de canales y, el filtro paso banda aplicado, se aplican hasta cuatro niveles de descomposición y se utilizan todas las bajas frecuencias resultantes y la última alta frecuencia.

Esta descomposición en cuatro niveles **permite captar información a distintos grados de resolución. Las bajas frecuencias generalmente corresponden a patrones generales de la señal, asociados con el comportamiento global del cerebro, mientras que las altas frecuencias capturan variaciones rápidas y transitorias que podrían estar relacionadas con eventos neuronales específicos.** Al conservar las aproximaciones generales de la señal y, el detalle de la última alta frecuencia es posible maximizar la cantidad de información disponible para los algoritmos de clasificación.

El uso de la *DWT* en este contexto no solo facilita la extracción de características útiles para la clasificación, sino que también **mejora el análisis al dividir la señal en componentes más manejables.** Al hacerlo, **se eliminan posibles fuentes de ruido o información no relevante,** mejorando así la precisión del algoritmo de clasificación.

A continuación, se muestra un fragmento del código, en donde se puede observar el caso en el que se aplica una *DWT* de la familia *Daubechies* en 4 niveles [74], [138].

```

be=1;
c=656;
for k=1:1:15

E=TX(be:c,:);
if k==1

%Probando con la wavelet Daubechies
E1=E;
E1_1=E1(:,1); %cada columna es un vector CANAL FC1
[cAE1_1_1,cDE1_1_1] = dwt(E1_1,'db5'); % se aplica la DWT a cada vector en 5 niveles generando
[cAE1_1_2,cDE1_1_2] = dwt(cAE1_1_1,'db5'); %para el vector 1 del usuario 1 5 vector
[cAE1_1_3,cDE1_1_3] = dwt(cAE1_1_2,'db5'); %uya nomenclatura es E1_1_1 , E1_1_2 ...
[cAE1_1_4,cDE1_1_4] = dwt(cAE1_1_3,'db5'); %es la imagen del paper
[cAE1_1_5,cDE1_1_5] = dwt(cAE1_1_4,'db5');

E1_2=E1(:,2); %Vector 1 columna o canal 2 FCz

[cAE1_2_1,cDE1_2_1] = dwt(E1_2,'db5'); % se aplica la DWT a cada vector en 5 niveles generando
[cAE1_2_2,cDE1_2_2] = dwt(cAE1_2_1,'db5'); %para el vector 1 del usuario 1 5 vector
[cAE1_2_3,cDE1_2_3] = dwt(cAE1_2_2,'db5'); %uya nomenclatura es E1_1_1 , E1_1_2 ...
[cAE1_2_4,cDE1_2_4] = dwt(cAE1_2_3,'db5'); %es la imagen del paper
[cAE1_2_5,cDE1_2_5] = dwt(cAE1_2_4,'db5');

E1_3=E1(:,3);%Vector 1 ch3 FC6

[cAE1_3_1,cDE1_3_1] = dwt(E1_3,'db5'); % se aplica la DWT a cada vector en 5 niveles generando
[cAE1_3_2,cDE1_3_2] = dwt(cAE1_3_1,'db5'); %para el vector 1 del usuario 1 5 vector
[cAE1_3_3,cDE1_3_3] = dwt(cAE1_3_2,'db5'); %uya nomenclatura es E1_1_1 , E1_1_2 ...
[cAE1_3_4,cDE1_3_4] = dwt(cAE1_3_3,'db5'); %es la imagen del paper
[cAE1_3_5,cDE1_3_5] = dwt(cAE1_3_4,'db5');

E1_4=E1(:,4);%Vector 1 ch 4 C5

[cAE1_4_1,cDE1_4_1] = dwt(E1_4,'db5'); % se aplica la DWT a cada vector en 5 niveles generando
[cAE1_4_2,cDE1_4_2] = dwt(cAE1_4_1,'db5'); %para el vector 1 del usuario 1 5 vector
[cAE1_4_3,cDE1_4_3] = dwt(cAE1_4_2,'db5'); %uya nomenclatura es E1_1_1 , E1_1_2 ...
[cAE1_4_4,cDE1_4_4] = dwt(cAE1_4_3,'db5'); %es la imagen del paper
[cAE1_4_5,cDE1_4_5] = dwt(cAE1_4_4,'db5');

```

Figura 3.4. Fragmento del primer código desarrollado en MATLAB para tratar los datos en formato txt.

3.2.2. Parámetros estadísticos

Tras aplicar *DWT* multinivel, se toma en cuenta el trabajo de Noguera, M. *et al.* [21], en el que se teoriza sobre la **aplicación de parámetros estadísticos en la clasificación de tareas en los BCI**. Según este artículo, estos **podrían mejorar la clasificación de la señal cerebral** en intenciones motoras. Por ello, se **plantean 2 escenarios**:

- El primero, en el que **se clasifica la señal proveniente de la *DWT* directamente** en los clasificadores expuestos en el apartado 2.4. «clasificadores» **sin parámetros estadísticos**.
- El segundo, en el que, tras la *DWT*, **se aplica el cálculo de los parámetros estadísticos y se realiza la clasificación** teniéndolos en cuenta.

El desafío planteado es **observar si el resultado mejora** al utilizar los parámetros estadísticos, **o si, por el contrario, se mantiene igual o empeora**.

Adicionalmente, se busca superar el éxito de los trabajos propuestos en el estado del arte (ver tabla 1.3).

3.2.3. IPMS

Una vez se realizan los experimentos con *DWT* y en los escenarios antes descritos (sin y con parámetros estadísticos), **se observan ciertos usuarios con una mejor tasa de clasificación que otros**, independientemente del escenario. Esto lleva al razonamiento de que **no solo influye el experimento y sus condiciones sino el estado mental del sujeto, así como el propio sujeto**.

Como proponen Roc A. *et al.* en [139] y es ratificado en [140], no sólo importa el estado mental, sino el propio usuario en sí. Por este motivo, el resultado de una clasificación puede diferir mucho entre usuarios.

El motivo principal es la predisposición de los sujetos, **hay sujetos que tienen una mejor respuesta a los sistemas BCI** por su capacidad de concentración o su control neuronal. Además, **el estado mental inmediatamente anterior al momento de la intención motora afecta directamente al resultado de su clasificación**. El más mínimo estímulo o confusión podría afectar a la clasificación consecuente y, por tanto, al diseño del sistema *BCI*.

Teniendo todo lo anterior en cuenta, **se propone un método novedoso denominado «estado mental previo» o IPMS** por sus siglas en inglés que ha sido publicado por el autor de esta tesis en un capítulo de libro [140].

En esta metodología, **a cada evento de intención motora (T1 / T2) se le resta una media del evento de reposo previo (T0) para reducir el ruido de la actividad cerebral de base y tener en cuenta el estado mental previo inmediato**. En la figura 3.5 se muestra sobre el gráfico del EEG un ejemplo de la aplicación de esta teoría en uno de los eventos T2 de un sujeto.

Una vez se aplica a todos los datos este cálculo, se repite la clasificación para comprobar si efectivamente, el estado mental previo de los sujetos influye o no en el éxito final de la clasificación de la intención motora.

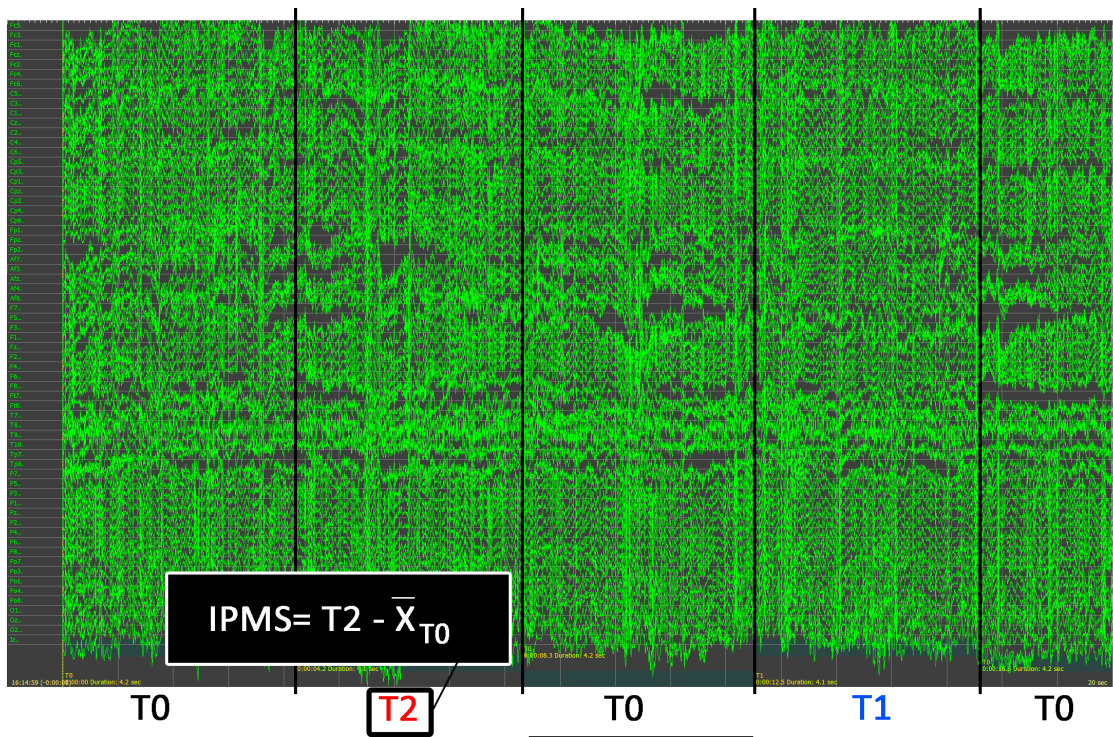


Figura 3.5. Explicación gráfica de la aplicación del IPMS. A cada evento T1 o T2 se le resta la media de su evento de descanso previo (T0).

3.2.4. Clasificadores de *Machine Learning*

La fase de clasificación es la que corresponde con la última etapa de los escenarios antes descritos (3.2.1, 3.2.1. y 3.2.3). Es decir, tras la extracción de las características y el procesamiento de los datos en cada uno de los apartados, **la última fase es la fase de clasificación mediante técnicas de *Machine Learning*. Los experimentos de *Machine Learning* se realizan primero con un subconjunto de 10 sujetos y posteriormente con la totalidad, añadiendo ensambles con algoritmos más avanzados.**

Para ello, se utiliza la herramienta «*Classification Learner*» de MATLAB, que permite probar diversos clasificadores en paralelo de forma rápida y eficiente.

Los clasificadores empleados en este análisis son los mostrados en la figura 3.6 y corresponden a los siguientes métodos de clasificación, **todos explicados previamente en el apartado 2.4.** del documento:

- **Clasificadores Naive Bayes:** Modelos que emplean el teorema de Bayes asumiendo independencia entre las características.
- **Árboles de decisión:** Árboles de decisión que permiten dividir los datos en subconjuntos basándose en atributos relevantes.
- **SVM:** Modelos que optimizan un hiperplano separador entre clases.
- **LDA y regresiones:** Métodos para maximizar la separación entre clases.
- **K-NN:** Modelos que asignan una clase basándose en las instancias más cercanas en el espacio de características.
- **Métodos ensemble:** Métodos que combinan múltiples modelos de clasificación (por ejemplo, *Bagged Trees*, *Boosted Trees*) para mejorar el rendimiento.
- **Perceptrón y redes neuronales simples:** Redes neuronales multicapa con distintos niveles de complejidad.
- ***Ensamblados híbridos de *Deep* y *Machine Learning*:** Se prueban diferentes clasificadores combinados, variando condiciones para probar su eficacia conjunta.

* Estos métodos se realizan fuera de la herramienta *Classification Learner*

Con la herramienta *Classification Learner* se evalúan distintos parámetros y configuraciones de estos clasificadores, ajustando los modelos y optimizando su rendimiento para seleccionar el que mejor se ajuste a los datos generados a partir de los parámetros obtenidos en los análisis previos. **Este enfoque permite identificar el algoritmo más eficiente para los distintos casos antes expuestos.**

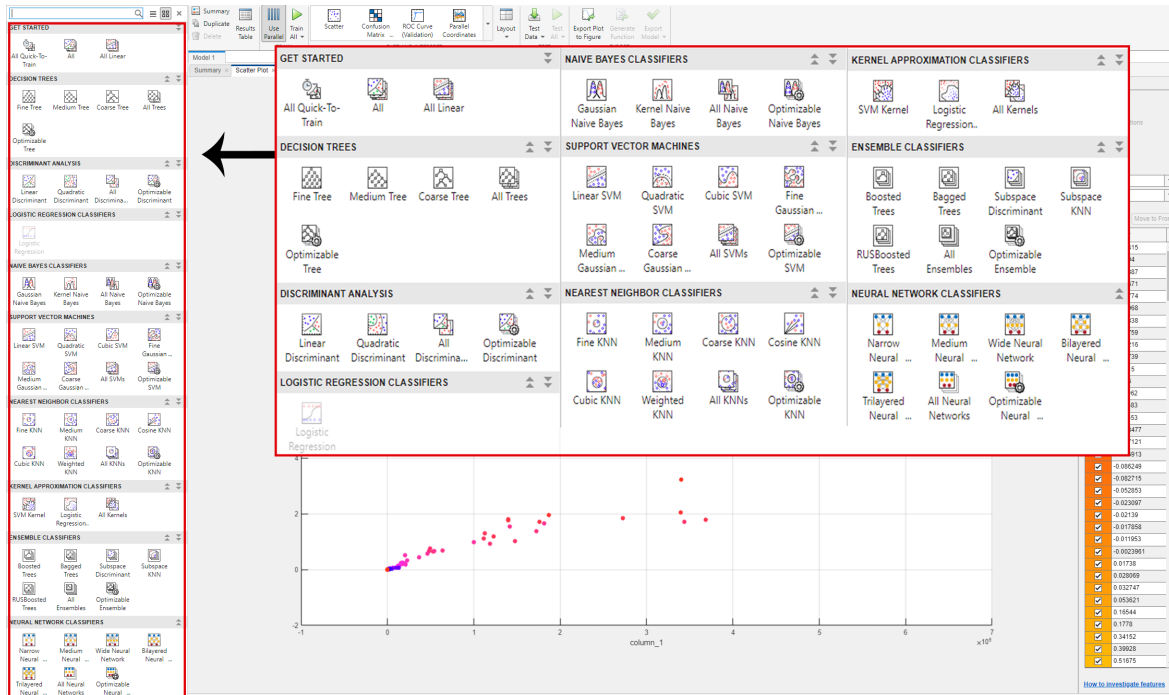


Figura 3.6. Algoritmos de *Machine Learning* de la herramienta *Classification Learner* de MATLAB utilizados en la clasificación de la señal EEG.

En el caso de los ensambles de *Deep* y *Machine Learning* se tiene un enfoque distinto:

1. Se analizan algoritmos de *Deep Learning* con el subconjunto, variando condiciones.
2. Se realizan pruebas con el ensamble algoritmos variando condiciones.
3. Se prueba con la base de datos completa.
4. Se realizan clasificaciones con los ensambles utilizando 2 experimentos de cada usuario para entrenar y se testea con el tercero.
5. Se realizan clasificaciones con los ensambles combinando los experimentos reales y pensados para tener un mayor número de datos. Variando ratios para obtener un rango óptimo y evaluando los resultados.

3.2.5. Cambio de paradigma. Matrices 2D

Tras realizar las clasificaciones con los algoritmos de *Machine Learning*, y **observando la tendencia de los últimos años**, se hizo necesario aplicar algoritmos más avanzados. **Capaces de captar mejor las relaciones no-lineales**, con una **mayor eficiencia** y sin necesidad de una extracción de características previa. Algoritmos que son capaces de **manejar una gran cantidad de datos y capturar las señales espaciales y temporales de manera más precisa**, algoritmos de *Deep Learning*.

Tras observar el trabajo planteado por Zhang, D. *et al.* [71] **se decidió cambiar el enfoque de *Machine Learning*, a un enfoque donde en lugar de generar vectores se generan matrices 2D que luego son la entrada para un clasificador de *Deep Learning*.**

Por ello, en la presente tesis se ha implementado una arquitectura novedosa que busca analizar el desempeño de los algoritmos de *Deep Learning* para clasificaciones tanto en solitario como combinados, con el objetivo de evaluar si el sistema es capaz de mejorar lo anteriormente visto.

En este nuevo enfoque, **de manera similar** al anteriormente visto en el apartado 2.2 «procesado de la señal», **los datos son separados por sujetos y sesiones, y luego se eliminan los eventos de reposo (T0), dejando únicamente los eventos de interés**, es decir, aquellos etiquetados como T1 o T2. Cada evento, como se describe en la sección 2.1, tiene una duración de 4.1 segundos, con una frecuencia de muestreo de 160 Hz, lo que genera 656 muestras por evento. En este punto, **se obtiene un vector de dimensiones 64x656 para cada evento, representando las mediciones de los 64 canales a lo largo de 4.1 segundos.**

El siguiente paso en el procesamiento consiste en la **reestructuración de estos datos en matrices de 10x11, replicando la disposición espacial de los electrodos en el casco EEG**, lo que **permite preservar** no solo la información temporal como hasta ahora sino **la información espacial de las señales cerebrales**. Las posiciones en las que no hay señal es rellenada con ceros, dando lugar a una matriz bidimensional de 10x11 por evento. Esto fue **ilustrado en la figura 2.5**, del apartado 2.2.

Posteriormente, **estos datos se reordenan por eventos y se crea su matriz de etiquetas, resultando en un tensor de 10x11x656, donde 656 corresponde al número**

de muestras de un evento completo (T1 o T2). Esta estructuración permite capturar la información espacial, crítica para una clasificación robusta.

3.2.6. Clasificadores de *Deep Learning*

A continuación, teniendo los datos acondicionados para su correcta clasificación, se plantean las diferentes arquitecturas y se realiza la clasificación de la señal.

Las arquitecturas elegidas para su evaluación se implementan en MATLAB con ayuda del *Deep Network Designer*. Estas arquitecturas propuestas, se apoyan en los clasificadores antes definidos en el apartado 2.4.2. «*Deep Learning*». Los detalles de las arquitecturas implementadas, así como un fragmento de sus códigos, son las siguientes:

Modelos individuales

- **CNN**

Se utiliza una secuencia de entrada 2D con 656 muestras por entrada (pertenecientes a cada evento). Se establecen 3 capas convolucionales en serie y a la salida de estas se utiliza una capa *fully connected* unida. De manera contigua a la anterior, se establece una capa *dropout*, y, por último, una capa *softmax* y una capa de clasificación final, ver figura 3.7. En MATLAB las capas *CNN* necesitan de los comandos «*sequence folding*» y «*sequence unfolding*» para tratar las matrices en forma de secuencia 2D.

- **LSTM y BiLSTM**

```

lgraph = layerGraph();
CNN = [

sequenceInputLayer([10 11 656], "Name", "inputcnn", "Normalization", "zscore") %'Mean', 'averageImage, ...

sequenceFoldingLayer("Name", "seqfold")
convolution2dLayer([3 3], 32, "Name", "conv_2", "Padding", "same", "PaddingValue", "replicate")
convolution2dLayer([3 3], 64, "Name", "conv_22", "Padding", "same", "PaddingValue", "replicate")
convolution2dLayer([3 3], 128, "Name", "conv_3", "Padding", "same", "PaddingValue", "symmetric-include-edge")
fullyConnectedLayer(2, "Name", "fc1")
dropoutLayer(0.5, "Name", "dropout1")
sequenceUnfoldingLayer("Name", "sequunfold")
fullyConnectedLayer(2, "Name", "fc2")
dropoutLayer(0.5, "Name", "dropout2")
softmaxLayer("Name", "softmax")
classificationLayer("Name", "OUT");

lgraph = addLayers(lgraph, CNN);

% CONEXIONES

lgraph = connectLayers(lgraph, 'seqfold/miniBatchSize', 'sequunfold/miniBatchSize');

```

Figura 3.7. Fragmento del código donde se implementa la arquitectura de la CNN.

De manera análoga a las *CNN* se establece una arquitectura donde se tiene un esquema bastante similar tanto para *LSTM*, como para *BiLSTM* y *GRU*.

La única diferencia al tratar estos algoritmos de manera individual es elegir el comando correcto, ya sea «*LSTMLayer*», «*biLSTMLayer*» o «*gruLayer*» respectivamente. Se añade una capa de entrada con el número de características. Luego, las capas *LSTM*/*BiLSTM* o *GRU* en serie deseadas, seguidas de una capa *fully connected*, una capa *softmax*, y, por último, la capa de clasificación, ver figura 3.8.

```
%-----
%LSTM arquitectura
numFeatures = 8184;
numHiddenUnits = 50;
numClasses = 2;
layers = [ ...
    sequenceInputLayer(numFeatures) % se puede poner tambien convolutional layer
    biLstmLayer(numHiddenUnits,'OutputMode','sequence','StateActivationFunction','tanh','GateActivationFunction','sigmoid')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];
```

Figura 3.8. Fragmento del código donde se muestra la arquitectura de la *BiLSTM*. Válida para *LSTM* y *GRU*.

Modelos combinados

Tras comprobar que los métodos de *Deep Learning* individuales (*CNN*, *LSTM*, *BiLSTM*, *GRU*) logran resultados satisfactorios en la clasificación de las señales EEG, **se decide explorar ensambles de los modelos, así como combinaciones híbridas de estos**. Esta estrategia busca **aprovechar las fortalezas complementarias de cada arquitectura, lo que puede mejorar la robustez y generalización del modelo** al combinar las predicciones de múltiples capas.

Por un lado, **los ensambles permiten capturar distintas características de las señales EEG que pueden ser ignoradas por modelos individuales**, mejorando así la precisión del sistema en escenarios más complejos.

Por otro lado, **las arquitecturas híbridas como *Autoencoder-BiLSTM*, *CNN-LSTM*, y *CNN-BiLSTM***, pueden aprovechar la capacidad de extracción automática de características espaciales de las *CNN*, con la modelización secuencial y temporal de largo plazo de las *LSTM* y *BiLSTM*. Esta combinación es especialmente útil para señales

EEG, donde se necesita capturar tanto patrones espaciales como temporales de manera eficiente.

Debido a que los ensamblados utilizados son combinación de varias capas *BiLSTM* o *LSTM* no se detalla su arquitectura por ser similar a la de la figura 3.9 pero con varias capas. La metodología llevada a cabo con los modelos híbridos se muestra a continuación:

- **Autoencoder – BiLSTM**

En la arquitectura *Autoencoder-BiLSTM*, el **autoencoder** se utiliza inicialmente para aprender una representación comprimida de los datos de entrada (señales EEG), permitiendo una **reducción de dimensionalidad y preservación de características relevantes**. Posteriormente, el **BiLSTM** procesa las secuencias reconstruidas por el *autoencoder*, **capturando patrones temporales y dependencias a largo plazo**.

Se establece para ello la **arquitectura del Autoencoder**, que consta de un **encoder**, un **decoder** y **capas de regularización**. Seguidamente se define la **capa BiLSTM** en donde su entrada es la salida del **decoder**, ver figura 3.9.

```

%Autoencoder
hiddenSize=50;
autoenc = trainAutoencoder(XTrain,hiddenSize,...
    'EncoderTransferFunction','logsig',...
    'DecoderTransferFunction','logsig',...
    'L2WeightRegularization',0.001,...
    'SparsityRegularization',1,...
    'SparsityProportion',0.05);

xReconstructed = predict(autoenc,XTrain);
xReconstructed=xReconstructed';
XTrain=xReconstructed;

%LSTM architecture
numFeatures = 8184;
numHiddenUnits = 50;
numClasses = 2;
layers = [ ...
    sequenceInputLayer(numFeatures) % se puede poner tambien convolutional layer
    lstmLayer(numHiddenUnits,'OutputMode','sequence','StateActivationFunction','tanh','GateActivationFunction','sigmoid')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];

```

PARTE DEL
AUTOENCODER

Figura 3.9. Fragmento del código donde puede observarse la parte de la arquitectura del Autoencoder arriba. Abajo la parte del *BiLSTM*.

Esta combinación busca ser especialmente **eficaz para analizar señales EEG debido a la capacidad del *autoencoder* de extraer características latentes y del *BiLSTM* de modelar la dinámica temporal de las señales.**

- ***CNN-LSTM***

Finalmente, se utiliza la arquitectura implementada, basada en *CNN-LSTM*, la cual combina dos enfoques que se complementan para el análisis de las señales EEG. Las *CNN* son capaces de capturar las relaciones espaciales inherentes en los datos EEG, fundamentales para identificar patrones en los electrodos distribuidos espacialmente sobre el cuero cabelludo. Y, por otro lado, las *LSTM* son adecuadas para modelar la naturaleza temporal de la señal, dada su capacidad para aprender dependencias a largo plazo en series temporales. Esta combinación permite capturar de manera efectiva tanto la estructura espacial como las dependencias temporales de los datos, lo que podría conducir a una mejora significativa en la precisión de la clasificación de eventos T1 y T2.

En la figura 3.10 puede verse la arquitectura desarrollada.

```

lgraph = layerGraph();
CNNLSTM = [

sequenceInputLayer([10 11 656],"Name","inputcnn", "Normalization","zscore") %'Mean'averagelImage, ...

% CNN
sequenceFoldingLayer("Name","seqfold")
convolution2dLayer([3 3],32,"Name","conv_2","Padding","same","PaddingValue","replicate")
convolution2dLayer([3 3],64,"Name","conv_22","Padding","same","PaddingValue","replicate")
convolution2dLayer([3 3],128,"Name","conv_3","Padding","same","PaddingValue","symmetric-include-edge")
fullyConnectedLayer(2,"Name","fc1")
dropoutLayer(0.5,"Name","dropout1")
sequenceUnfoldingLayer("Name","sequnfold")
flattenLayer("Name","flatten")
% LSTM
lstmLayer(64,'OutputMode','last','name','lstm1')
lstmLayer(64,'OutputMode','sequence','name','lstm2')
fullyConnectedLayer(2,"Name","fc2")
dropoutLayer(0.5,"Name","dropout2")

softmaxLayer("Name","softmax")
classificationLayer("Name","OUT");

lgraph = addLayers(lgraph,CNNLSTM);

% CONEXIONES

lgraph=connectLayers(lgraph,'seqfold/miniBatchSize','sequnfold/miniBatchSize');
```

Figura 3.10. Fragmento del código donde se implementa la arquitectura de la *CNN*.

Esta combina capas **CNN con capas LSTM**, y se estructura de la siguiente manera:

▪ **Capas CNN:**

- Se utiliza una capa de entrada adecuada al tamaño de las matrices y también el comando *folding* para poder utilizar las CNN en este modo.
- Seguidamente tres capas de convolución 2D con diferentes tamaños de filtros (32, 64 y 128 canales), utilizando *padding* y normalización.
- Una capa *fully connected* y una *dropout* que previenen el sobreajuste, reduciendo la dimensionalidad y regularizando la red.
- Finalmente se cierra con un *unfolding* que permite la conexión con el LSTM.

▪ **Capas LSTM:**

- Se añade una capa *flatten* para aplanar los datos provenientes de la CNN, ya que estos son de 2 dimensiones.
- Se despliegan 2 capas **LSTM** en serie que procesan las características extraídas por la CNN. Estas capas están configuradas, una en modo "*last*", lo que significa que se toma el último estado oculto para las predicciones finales y la otra en modo "*sequence*", por lo que toma toda la última secuencia.
- Por último, se establece una capa *fully connected* para ayudar a la clasificación, seguida de *dropout*, la capa *softmax* para generar las probabilidades de las clases y la capa de clasificación, que da el output final del algoritmo.

También fue probado el caso con *BiLSTM* y con *GRU* en este tipo de modelo híbrido, sin embargo, se explica este caso por ser el de mejor desempeño.

3.3. Migración a Python

Justificación

El cambio de Matlab a Python (Spyder) en el desarrollo de la tesis se justificó principalmente por la **flexibilidad y versatilidad que ofrece Python**, ampliamente **utilizado en la comunidad científica**. Python permite un **mayor control y personalización de los algoritmos**, permitiendo afinar **detalles a niveles más profundos** que en Matlab, que es más hermético. Además, ofrece **compatibilidad nativa** con bibliotecas de vanguardia como **Keras, TensorFlow y CUDA**, optimizando el procesamiento de datos y el uso de GPUs.

Otro factor crucial fue la **implementación del formato EDF+** en Python, lo que simplificó enormemente el manejo y análisis de señales EEG, integrando de manera eficiente los datos en el flujo de trabajo. Python, además, **facilita la integración de diferentes fuentes de datos y herramientas**, permitiendo una mayor capacidad de ajuste y exploración en los modelos de *Deep Learning*, algo clave en el análisis, procesamiento de señales complejas como el EEG y desarrollo de algoritmos híbridos.

Código y arquitecturas

El primer paso tras migrar a Python consistió en **replicar el algoritmo de creación de matrices 2D utilizado previamente** en Matlab, con el objetivo de mantener la estructura de los experimentos desarrollados en la tesis. Este proceso, junto con la inclusión del comando de lectura *EDF+* en las versiones de Python, **garantiza que los experimentos partan desde la base de datos de una manera protocolaria, controlada y optimizada** al estar todo el proceso automatizado.

En la figura 3.11 puede verse un fragmento del código en su inicio, en donde se establece el directorio de la base de datos, se configura el entorno para que se ejecute en la unidad de procesamiento gráfico (*GPU*), por ser mucho más rápida, y se extraen los datos. Realizando un cálculo de los valores de EEG para hacer comprobaciones y que posteriormente se procesen estos datos para crear las matrices 2D.


```

5 @author: Nabil I. Ajali-Hernández ""
6
7
8
9 import keras
10 import os
11 os.chdir("E:\Physionet\Physionet_BD\BD_Todosmezclados") #Directorio del pendrive con BD
12 #Uso de CPU
13 # os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
14 # GPU
15 import tensorflow as tf
16 print("GPU disponibles:")
17 print(tf.config.list_physical_devices("GPU"))
18 tf.config.experimental.set_visible_devices(tf.config.list_physical_devices("GPU")[0], "GPU") #Configurar para usar
19 print("Dispositivo de cálculo actual:") #Comprobar que se esta usando
20 print(tf.config.list_logical_devices("GPU"))
21
22
23
24 #Leer EDF
25
26 import numpy as np
27 import mne
28 import pyedflib
29
30 #Obtener una lista con todos los archivos EDF
31 edf_files = [f for f in os.listdir() if f.endswith('.edf')]
32 #Crea un diccionario para almacenar los datos de las señales de todos los archivos
33 all_signals = {}
34 # Leer todos los archivos EDF y almacenar los datos en el diccionario
35 for file in edf_files:
36     with pyedflib.EdfReader(file) as f:
37         # Obtener información del archivo
38         num_signals = f.signals_in_file
39         signal_labels = f.getSignalLabels()
40
41         # Leer todas las señales del archivo
42         signals = []
43         for i in range(num_signals):
44             signals.append(f.readSignal(i))
45
46         # Almacenar las señales del archivo con sus etiquetas
47         all_signals[file] = {label: signal for label, signal in zip(signal_labels, signals)}
48
49 # Ahora tienes todos los datos de los archivos EDF en el diccionario `all_signals`
50
51 # Ejemplo de cómo acceder a los datos de una señal específica de un archivo:
52 if "S001R04.edf" in all_signals:
53     eeg_fpz_cz = all_signals["S001R04.edf"]["EEG Fpz-Cz"] # Reemplaza 'EEG Fpz-Cz' con la etiqueta real
54     print(eeg_fpz_cz[:10]) # Imprime los primeros 10 valores
55
56 # Concatenar todas las arrays de señales en una sola (considerando múltiples señales por archivo)
57 concatenated_signals = np.concatenate([signal for signals in all_signals.values() for signal in signals.values()])
58
59 # Calcular y mostrar el mínimo y máximo
60 minimo_global = np.min(concatenated_signals)
61 maximo_global = np.max(concatenated_signals)
62 media_global = np.mean(concatenated_signals)
63 std_global = np.std(concatenated_signals)
64 mediana_global = np.median(concatenated_signals)
65 varianza_global = np.var(concatenated_signals)
66 percentil95 = np.percentile(concatenated_signals, 95)
67 histograma_global = np.histogram(concatenated_signals)
68
69 print(f'Mínimo global: {minimo_global}; Máximo global: {maximo_global}; Media global: {media_global}; STD global: {std_global}; Mediana global: {mediana_global}; Varianza global: {varianza_global}')
70

```

Elección del directorio
Configuraciones de la GPU

Manejo de la base de datos
Obtención de señales, anotaciones, etc...
Creación de variables necesarias...

Ejemplo con el Sujeto 001
Cálculo de parámetros de la señal

Figura 3.11. Fragmento del código inicial donde se leen las señales y se establece el entorno de trabajo.

Una vez replicadas las matrices 2D del conjunto de datos, **se procede a replicar el mejor caso observado en MATLAB** para asegurar que los algoritmos desarrollados funcionan adecuadamente y se obtienen resultados comparables en Python, garantizando que la migración no afecta a la calidad del rendimiento. **En la figura 3.12** puede verse un fragmento de código en donde se muestra la arquitectura **CNN-LSTM** que fue la que mejor desempeño tuvo en MATLAB

```

import numpy as np
from sklearn.model_selection import KFold
from keras.models import Sequential
from keras.layers import Conv2D, LSTM, Dense, Flatten, Dropout, BatchNormalization, Lambda
import keras
from sklearn.metrics import confusion_matrix

# Definir Los valores de entrenamiento y test
XX_train = XX
YY_train = YY
timesteps=4410
data_dim=(11,10)
# Definir Los parámetros para k-fold
k = 10

# Crear las divisiones de k-fold
kf = KFold(n_splits=k)

# Inicializar una lista para almacenar los resultados de evaluación en cada fold
evaluations = []
m=1
# Iterar sobre las divisiones generadas por k-fold
for train_index, val_index in kf.split(XX_train):

    X_train_fold, X_val_fold = XX_train[train_index], XX_train[val_index]
    y_train_fold, y_val_fold = YY_train[train_index], YY_train[val_index]

    # Crear la arquitectura de la red
    model = Sequential()
    model.add(BatchNormalization(input_shape=(11, 10, 1)))
    model.add(Conv2D(128, (3,3), activation='relu', padding='same', input_shape=(11, 10, 1)))
    model.add(Conv2D(64, (3,3), activation='relu', padding='same'))
    model.add(Conv2D(32, (3,3), activation='relu', padding='same'))
    model.add(Dense(2))
    model.add(Dropout(0.5))

    def remove_channel(x):
        return x[... , 0]
    model.add(Lambda(remove_channel))

    model.add(LSTM(units=64, return_sequences=True, input_shape=(timesteps, data_dim)))
    model.add(LSTM(units=64, return_sequences=False))
    model.add(Dense(2))
    model.add(Dropout(0.5))
    model.add(Dense(1, activation='sigmoid'))

    # Compilar el modelo
    options = keras.optimizers.Adam(learning_rate=0.0001, epsilon=1e-8)
    model.compile(optimizer=options, loss='binary_crossentropy', metrics=['accuracy'])

    print("Iteración:", m)
    m=m+1

    # Entrenar el modelo
    model.fit(X_train_fold, y_train_fold, batch_size=82, shuffle=True, epochs=10)

    # Evaluar el modelo en los datos de validación
    evaluation = model.evaluate(X_val_fold, y_val_fold)
    evaluations.append(evaluation)

```

CNN

LSTM

Figura 3.12. Fragmento del código donde se implementa la arquitectura de la CNN-LSTM en Python.

Esto implicó un ajuste fino de los hiperparámetros y la arquitectura de la red para maximizar la reproducibilidad de los resultados. Esta arquitectura fue objeto de publicación en una revista indexada [19]. El diagrama de flujo del proceso completo se muestra en la figura 3.13.

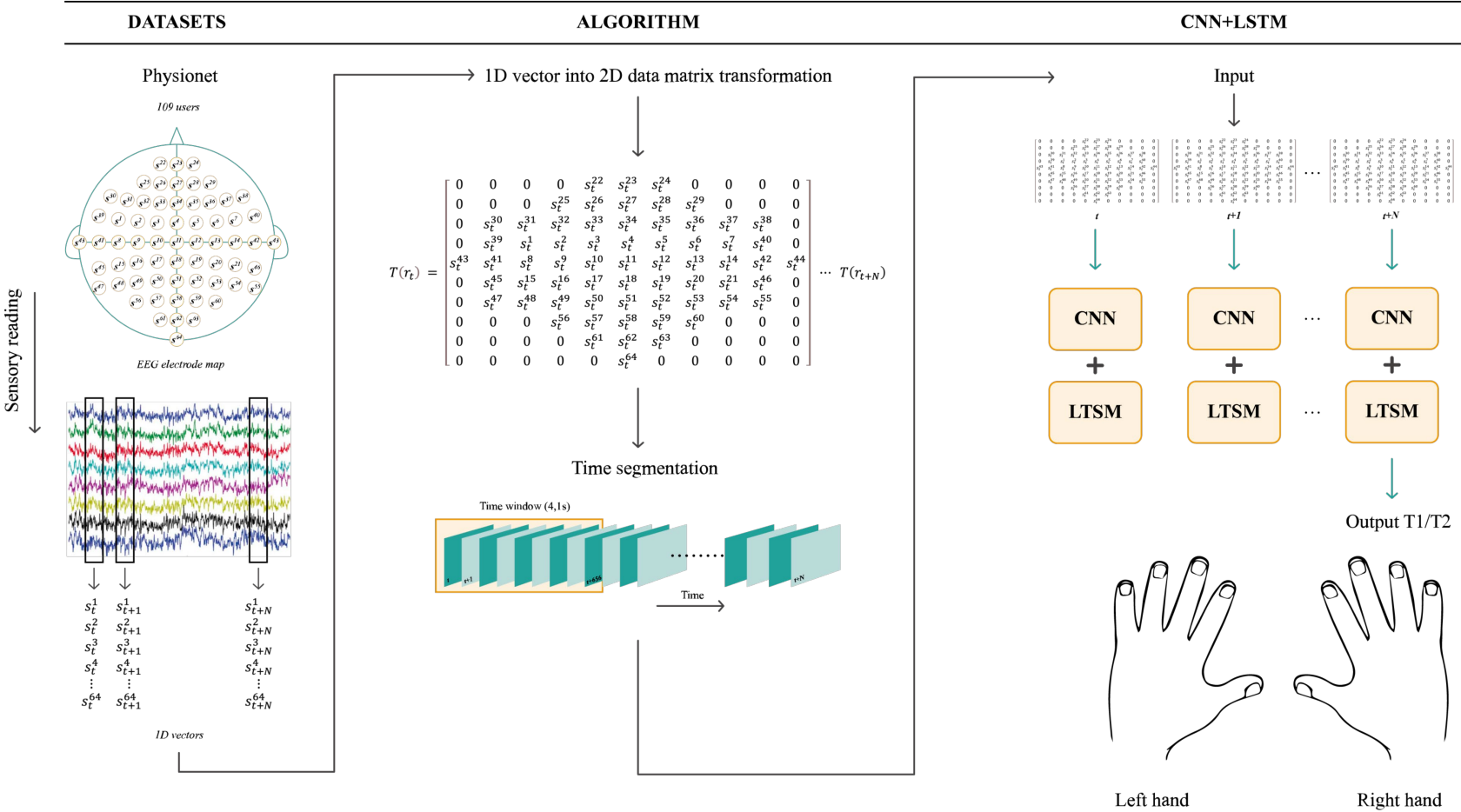


Figura 3.13. Esquema del proceso completo desde la toma de datos hasta la clasificación final utilizando la arquitectura híbrida CNN-LSTM.

Una vez completada la replicación, se exploraron nuevas combinaciones como **CNN-BiLSTM**, **CNN-LSTM con Attention Layer**, **Transformer** y **Vision Transformer (ViT)**, aprovechando las capacidades avanzadas de Python para implementar arquitecturas más complejas y novedosas.

Estas combinaciones permitieron explorar el rendimiento de arquitecturas híbridas, combinando procesamiento espacial (*CNN*) con secuencial (*LSTM/BiLSTM*) y mecanismos de atención, proporcionando muy buenos resultados en algunos casos y futuras líneas de investigación en otros.

Finalmente, se llevaron a cabo decenas de comparaciones entre los dos mejores modelos obtenidos, variando condiciones, número de capas ocultas, y ajustando hiperparámetros como el *learning rate* y el tamaño de *batch*. Este proceso de comparación fue clave para optimizar el rendimiento del modelo y seleccionar la arquitectura más robusta para el análisis de las señales EEG.

Se muestran a continuación, en las figuras 3.14 y 3.15, los fragmentos de código para la definición de la *attention layer* y la arquitectura del *vision transformer*, respectivamente, por ser consideradas de interés.

Attention Layer

```

from keras import backend as K

class AttentionLayer(Layer):
    def __init__(self):
        super(AttentionLayer, self).__init__()

    def build(self, input_shape):
        # Define los pesos entrenables para calcular la atención
        self.W = self.add_weight(shape=(input_shape[-1], 1), initializer='random_normal', trainable=True)
        super(AttentionLayer, self).build(input_shape)

    def call(self, inputs):
        # Calcula los pesos de atención
        attention_weights = K.squeeze(K.dot(inputs, self.W), axis=-1)
        attention_weights = K.softmax(attention_weights, axis=1)

        # Expande las dimensiones para permitir la multiplicación
        attention_weights = K.expand_dims(attention_weights, axis=-1)

        # Aplica la atención ponderando las características de entrada
        weighted_inputs = attention_weights * inputs

        # Suma ponderada de las características de entrada
        attention_output = K.sum(weighted_inputs, axis=1)

        return attention_output

    def compute_output_shape(self, input_shape):
        return input_shape[0], input_shape[-1]

```

Figura 3.14. Fragmento del código donde se define la *attention layer* y sus propiedades.

ViT

```

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset, TensorDataset
import torch.optim as optim
import matplotlib.pyplot as plt
from tqdm import tqdm

class PatchEmbedding(nn.Module):
    def __init__(self, in_channels, patch_size, emb_size, img_size):
        super().__init__()
        self.patch_size = patch_size
        self.emb_size = emb_size
        self.n_patches = (img_size[0] // patch_size) * (img_size[1] // patch_size)
        self.projection = nn.Conv2d(in_channels, emb_size, kernel_size=patch_size, stride=patch_size)

    def forward(self, x):
        x = self.projection(x) # (B, emb_size, n_patches*0.5, n_patches*0.5)
        x = x.flatten(2) # (B, emb_size, n_patches)
        x = x.transpose(1, 2) # (B, n_patches, emb_size)
        return x

class PositionalEncoding(nn.Module):
    def __init__(self, n_patches, emb_size):
        super().__init__()
        self.pos_embedding = nn.Parameter(torch.randn(1, n_patches, emb_size))

    def forward(self, x):
        return x + self.pos_embedding

class TransformerEncoder(nn.Module):
    def __init__(self, emb_size, num_heads, hidden_dim, num_layers, dropout):
        super().__init__()
        encoder_layer = nn.TransformerEncoderLayer(d_model=emb_size, nhead=num_heads, dim_feedforward=hidden_dim, dropout=dropout)
        self.encoder = nn.TransformerEncoder(encoder_layer, num_layers=num_layers)

    def forward(self, x):
        return self.encoder(x)

class VisionTransformer(nn.Module):
    def __init__(self, in_channels, patch_size, emb_size, img_size, num_heads, hidden_dim, num_layers, num_classes, dropout=0.1):
        super().__init__()
        self.patch_embedding = PatchEmbedding(in_channels, patch_size, emb_size, img_size)
        self.pos_encoding = PositionalEncoding(self.patch_embedding.n_patches, emb_size)
        self.transformer_encoder = TransformerEncoder(emb_size, num_heads, hidden_dim, num_layers, dropout)
        self.classification_head = nn.Linear(emb_size, num_classes)

    def forward(self, x):
        x = self.patch_embedding(x)
        x = self.pos_encoding(x)
        x = self.transformer_encoder(x)
        x = x.mean(dim=1) # Pooling ----- reduciendo la salida a 1 D ----- si no reduces y deas 2D
        x = self.classification_head(x)
        return x

# Inicializar el modelo y moverlo al dispositivo adecuado
model = VisionTransformer(
    in_channels=656,
    patch_size=3,
    emb_size=144,
    img_size=(10, 11),
    num_heads=6,
    hidden_dim=510,
    num_layers=12,
    num_classes=2
).to(dispositivo) # Mover el modelo a La GPU si está disponible

model.to(dispositivo)

```

Figura 3.15. Fragmento del código donde se muestra la arquitectura del *vision transformer*. Se define en Python utilizando la biblioteca *Pytorch*, por ser más sencilla en el tratamiento de estos algoritmos.

A continuación, a modo de resumen, se muestran en la figura 3.16, la totalidad de arquitecturas explicadas en la metodología. Ordenadas cronológicamente de izquierda a derecha.

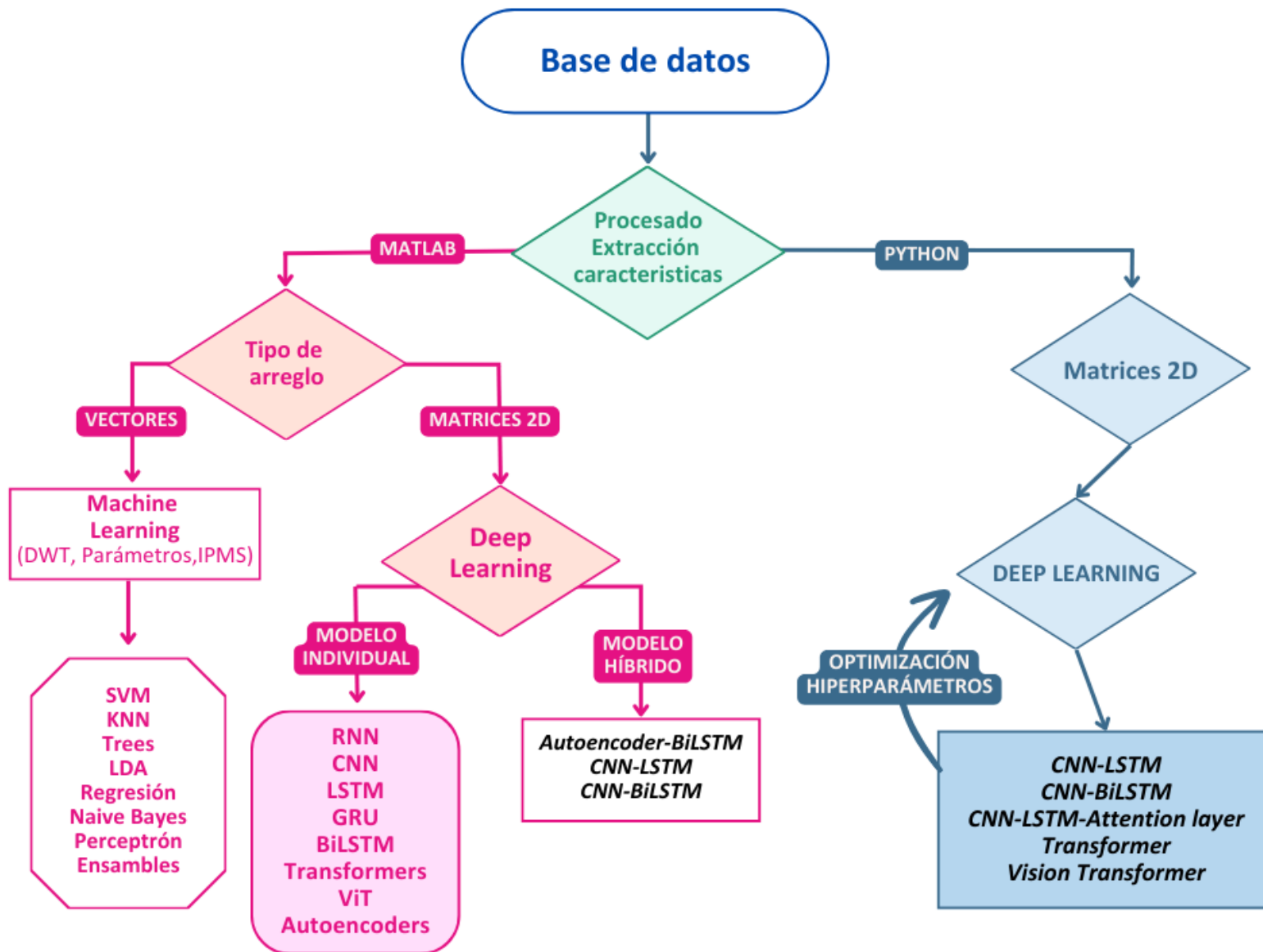


Figura 3.16. Diagrama resumen de desarrollo de la tesis. Se muestran las metodologías seguidas y las decisiones tomadas.

3.4. Hiperparámetros y condiciones

En esta sección se detallan los hiperparámetros que se emplearon en el proceso de **entrenamiento** y ajuste de los modelos de clasificación de señales EEG. Estos hiperparámetros **fueron optimizados de manera experimental** para maximizar el rendimiento del modelo, variando cada uno en **diferentes rangos para encontrar la configuración más adecuada**. A continuación, se definen los principales hiperparámetros utilizados, detallando los rangos explorados.

Método Adam como Optimizador

El algoritmo **Adam** (*Adaptive Moment Estimation*) es uno de los optimizadores más utilizados en redes neuronales debido a su **capacidad de adaptarse dinámicamente a los gradientes**. Este optimizador ajusta las tasas de aprendizaje de los parámetros de forma independiente, combinando las ventajas de los métodos **RMSProp y Momentum**. Adam realiza actualizaciones en los parámetros del modelo basándose en estimaciones del primer (m_t) y segundo momento (v_t) de los gradientes, ajustando la tasa de aprendizaje según la siguiente ecuación [141, 142]:

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (3.1)$$

Donde:

- θ_t son los parámetros del modelo en el paso t .
- η es la tasa de aprendizaje (*learning rate*).
- \hat{m}_t y \hat{v}_t son el primer y segundo momento del gradiente, respectivamente.
- ϵ es un valor pequeño que se añade para evitar divisiones por cero y estabilizar.

Las actualizaciones de m_t y v_t se definen como:

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \quad (3.2)$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \quad (3.3)$$

Donde \mathbf{g}_t es el gradiente, β_1 es el factor de decaimiento del primer momento (rango testeado $\beta_1 = [0.5 - 0.99]$, **valor óptimo $\beta_1 = 0.9$** como mejor valor), y β_2 es el factor de decaimiento del segundo momento (rango testeado $\beta_2 = [0.8-0.999]$, **valor óptimo $\beta_2 = 0.999$**) y *Epsilon* (rango testeado $\epsilon = [10^{-4} - 10^{-9}]$, **valor óptimo $\epsilon = 10^{-8}$**)

El optimizador *Adam* es ampliamente utilizado debido a su capacidad para converger rápidamente en problemas complejos, lo que lo convierte en una opción adecuada para modelos de *Deep Learning* aplicados a EEG.

Initial Learning Rate

El *Initial Learning Rate* (η) controla la magnitud de los ajustes que se aplican a los parámetros del modelo en cada iteración. **Un valor demasiado alto puede causar inestabilidad, mientras que uno demasiado bajo puede ralentizar el entrenamiento.** En este trabajo, se encontró tras experimentar con amplio rango de $[1 \cdot 10^{-1} - 1 \cdot 10^{-6}]$ que un **valor óptimo para η es $1 \cdot 10^{-4}$.**

Este valor permitió una convergencia eficiente y rápida, evitando caer en mínimos locales de la función de pérdida y evitando además sobreajustes prematuros del modelo.

MiniBatchSize

El *Mini Batch Size* determina el **número de muestras procesadas antes de actualizar los pesos del modelo.**

Se experimentó con tamaños de mini-lotes en rangos desde 1 hasta 656, observando que tamaños pequeños aumentaban la variabilidad en las actualizaciones, mientras que tamaños grandes reducían el ruido en los gradientes.

Este parámetro afecta el rendimiento de la *GPU* y la memoria, por lo que se ajustó en función del *hardware* disponible. El valor óptimo depende de la arquitectura que se utilice, siendo **82 ± 18 un rango que demostró buen funcionamiento.**

Max Epochs y Shuffle

El número máximo de *Epochs* es el **número de veces que el modelo pasa por todo el conjunto de entrenamiento**. Se experimentó con valores de 1 a 1000 *epochs*. Se usó el **modo shuffle para mezclar los datos en cada epoch**, evitando que el modelo se sobreajuste a secuencias específicas del conjunto de datos. También se realizaron pruebas sin mezclar los datos o mezclándolos solo en la primera ronda (*single shuffle*).

Se encuentra que **aleatorizar en cada epoch (Shuffle) ayuda a mejorar la generalización, siendo los 100 epoch suficiente en la mayoría de los casos**.

Gradient Threshold Method

El *Gradient Threshold Method* es un método de regularización que puede usarse, por un lado, para evitar sobreajustes debido a grandes pesos, y, por otro lado, para evitar que los gradientes colapsen al volverse demasiado grandes durante el entrenamiento [108].

En la presente tesis, **se empleó la norma L_2 (L2Norm)** como método de umbral de gradiente. Su definición es la que sigue.

En términos generales, la norma L_2 mide la distancia de un vector desde el origen del espacio hasta su posición final, lo que corresponde a su longitud o magnitud en el espacio Euclidiano. Se define como la **raíz cuadrada de la suma de los cuadrados de los componentes de un vector**. Esto es, dado un vector $\|X\| = [x_1, x_2, \dots, x_n]$, la norma L_2 se calcula como [108], [136]:

$$\|X\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (3.4)$$

Presenta una serie de propiedades que son:

- 1- **No negatividad:** $\|X\|_2 \geq 0$ y $\|X\|_2 = 0$, solo si $x=0$
- 2- **Homogeneidad:** Para cualquier escalar α , $\|\alpha x\|_2 = |\alpha| \|x\|_2$
- 3- **Desigualdad triangular:** $\|x + y\|_2 \leq \|x\|_2 + \|y\|_2$

Como se expresó anteriormente, la norma L_2 se utiliza tanto para regularizar, como para controlar el gradiente. En el caso de la regularización L_2 , su función de costo puede expresarse en ese caso como:

$$J(\theta) = \text{Coste original} + \lambda \|\theta\|_2^2 \quad (3.5)$$

Donde λ es un parámetro que controla la regularización y θ son los pesos del modelo.

Entrenamiento y Test: *Holdout* y *K-Fold*

Para la evaluación de los modelos, se utilizó el método *Holdout*, dividiendo los datos en diferentes proporciones de entrenamiento y prueba, probando todos los rangos desde 90-10 (entrenamiento-test) hasta 30-70 (entrenamiento-test), variando siempre de 5 en 5 las ratios para no hacer cambios bruscos e ir minuciosamente analizando el comportamiento del sistema.

El método *Holdout* es simple y permite una rápida validación del modelo con un conjunto ciego con el que no ha estado en contacto. Es decir, **el modelo entrena con una serie de datos y luego se testea con otro conjunto, con el que no ha interactuado, y que es completamente ajeno al primer conjunto**, ofreciendo así una **garantía de que el sistema responde bien ante nuevos datos y evita el sesgo**. En algunos casos con pocos datos, este tipo de validación puede sufrir de alta varianza.

Por ello, **también se probó el método *k-fold cross-validation***, dividiendo el conjunto de datos en $n=10$ particiones y entrenando el modelo con diferentes subconjuntos. Este método reduce el sesgo y ofrece una evaluación más robusta del rendimiento del modelo.

3.5. Métricas

Para evaluar de forma rigurosa el rendimiento de modelos de clasificación binaria, se suelen emplear una variedad de métricas con el fin de proporcionar una visión más completa del comportamiento de los modelos. A continuación, se definen las métricas clave utilizadas en este trabajo, así como sus expresiones matemáticas [143–145].

Estas métricas **han sido fundamentales para comparar y entender tanto los modelos en sí, como los hiperparámetros y ajustes que se han ido haciendo a lo largo de esta tesis** para llegar demostrar la hipótesis propuesta en el apartado 1.5.

Exactitud (*accuracy*)

La **exactitud** es una medida de la **proporción de predicciones correctas realizadas** por el modelo, tanto para la clase positiva como para la clase negativa. Sin embargo, cuando las clases están desbalanceadas puede no ser una métrica confiable, ya que un modelo podría predecir la clase mayoritaria con alta exactitud, pero fallar en la clase minoritaria.

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN} \quad (3.6)$$

Donde a partir de ahora, **TP** es verdadero positivo (*True Positive*), **TN** es verdadero negativo (*True Negative*), **FP** es falso positivo (*False Positive*) y **FN** es falso negativo (*False Negative*)

Precisión

La **precisión** mide la **proporción de predicciones positivas verdaderas sobre todas las predicciones positivas** realizadas por el modelo. Es especialmente útil cuando el coste de los falsos positivos es elevado, como en aplicaciones médicas o en detección de fraude.

$$\text{Precisión} = \frac{TP}{TP + FP} \quad (3.7)$$

Especificidad

La especificidad, también conocida como la tasa de verdaderos negativos, mide la **proporción de predicciones negativas verdaderas sobre todas las predicciones negativas** realizadas por el modelo. Es crucial cuando el coste de los falsos negativos es alto.

$$\text{Especificidad} = \frac{TN}{TN + FP} \quad (3.8)$$

Matriz de Confusión

La matriz de confusión es una tabla o matriz que **permite visualizar todo el rendimiento del modelo**. Correctamente programada muestra en pantalla los valores anteriormente vistos (TP, TN, FP y FN) además de las métricas de exactitud y precisión.

Es una **herramienta ampliamente utilizada en este campo y más que suficiente a la hora de comparar dos clasificadores, para ver sus fortalezas y debilidades**. Por ello, es la métrica que más ha sido utilizada y con mayor peso en la tesis. Ver figura 3.17.

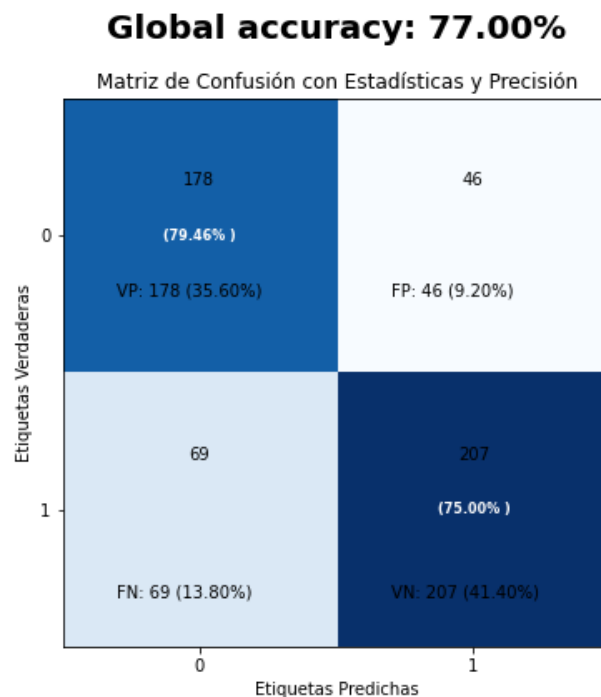


Figura 3.17. Ejemplo de una matriz de confusión.

Las dos siguientes métricas no fueron tan ampliamente utilizadas, ya que con las tres anteriores (exactitud, precisión y especificidad) y la matriz de confusión es suficiente para evaluar los modelos y comparar con el estado del arte.

Sin embargo, a la hora de enviar el artículo [19] y otros dos artículos que se encuentran bajo revisión fue de interés añadirlos para darle más robustez a la justificación de ciertos casos.

Coeficiente *kappa*

El **coeficiente *kappa* (κ)** mide el **grado de acuerdo entre las predicciones del modelo y las etiquetas verdaderas**, teniendo en cuenta el acuerdo que podría ocurrir por casualidad. Se utiliza comúnmente en el contexto de clasificación binaria y multiclase. Los valores de *Kappa* oscilan entre -1 y 1. Para $\kappa = 1$ **se tiene un acuerdo perfecto** entre el modelo y las etiquetas reales, **para $\kappa = 0$** se tiene un acuerdo igual al esperado por el azar (el **resultado es aleatorio**) y para una $\kappa < 0$ **el acuerdo es incluso peor que el azar** [146]. Matemáticamente se define como:

$$\kappa = \frac{P_0 - P_e}{1 - P_e} \quad (3.9)$$

Donde P_0 es la proporción de acuerdo observado y P_e es la proporción de acuerdo esperado por azar

Índice de *Youden*

El índice de *Youden* (J) **equilibra la sensibilidad y la especificidad**, y es particularmente útil cuando las clases están desbalanceadas. Mide la **diferencia entre la tasa de verdaderos positivos y la de falsos positivos**, proporcionando una única métrica que resume el rendimiento del modelo. Sus valores, al igual que con el coeficiente *kappa*, oscilan entre -1 y 1. **Siendo $J=1$ una clasificación perfecta, un $J=0.5$ un buen rendimiento y siendo mala o pésima la clasificación $J \leq 0$** [147].

$$J = \text{Sensibilidad} + \text{Especificidad} - 1 \quad (3.10)$$

Media aritmética (Avg)

La media aritmética o *average* (avg) en inglés, es una medida de tendencia central que se utiliza para determinar el valor promedio de un conjunto de datos. Es muy útil para describir la "ubicación" central de los datos y es una de las medidas básicas en el análisis de datos de cualquier estudio o población. Es ampliamente utilizada en el campo del análisis de datos, todo tipo de estadísticos y otros muchos entornos [148].

Matemáticamente, la media aritmética μ de un conjunto de datos $X = \{x_1, x_2, \dots, x_n\}$ se expresa como:

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i \quad (3.11)$$

Siendo n el número total de observaciones.

Desviación estándar (STD)

La desviación estándar o STD por sus siglas en inglés, es una medida estadística que indica el grado de dispersión o variabilidad de un conjunto de datos respecto a su media aritmética. Cuanto mayor sea la desviación estándar, más dispersos estarán los datos; cuanto menor sea, más concentrados estarán alrededor de la media. Se calcula como la raíz cuadrada de la varianza, la cual es el promedio de las diferencias al cuadrado entre cada valor del conjunto de datos y la media del conjunto [149].

Matemáticamente, para un conjunto de datos $X = \{x_1, x_2, \dots, x_n\}$, la desviación estándar σ se calcula como:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2} \quad (3.12)$$

Siendo n el número total de observaciones.

3.6. Librerías de Python

A continuación, se muestran las librerías de Python utilizadas en el desarrollo de la tesis y que han sido de mayor relevancia:

- **keras**: Utilizada ampliamente para construir redes neuronales y modelos de aprendizaje profundo. En este trabajo se utiliza en contables ocasiones para definir capas, modelos y optimizadores dentro de las arquitecturas.
- **os**: Se utiliza para cambiar el directorio de trabajo «*os.chdir*» y configurar variables de entorno, como en el caso de «*CUDA_VISIBLE_DEVICES*» para definir el dispositivo de cálculo, como la GPU.
- **tensorflow**: Usada para configurar la disponibilidad y uso de GPU, además de verificar dispositivos de cálculo. Funciona como «*backend*» de Keras en este caso.
- **h5py**: Utilizada para leer y manipular archivos en formato *HDF5*, cargando datos de EEG desde archivos «*.h5*».
- **scipy.io**: Permite la carga de datos en formato MATLAB «*.mat*», facilitando el acceso a conjuntos de datos específicos de EEG.
- **numpy**: Fundamental para el procesamiento y manipulación de matrices numéricas. Es utilizado ampliamente para preparar los datos de entrenamiento y test.
- **tqdm**: Es una herramienta que facilita la visualización de barras de progreso en Python. En este código, se utiliza para mostrar el avance durante el entrenamiento de cada época, proporcionando un seguimiento en tiempo real.
- **matplotlib.pyplot**: Utilizada para la visualización de datos, en particular para crear y personalizar la matriz de confusión.
- **seaborn**: Biblioteca para visualización de datos que trabaja sobre *Matplotlib*, especialmente útil para gráficos estadísticos. Aquí se usa para visualizar la matriz de confusión en un formato más legible con «*heatmap*».
- **sklearn.metrics**: Se usa para calcular y mostrar la matriz de confusión, evaluando así el rendimiento de clasificación del modelo.

- **torch:** *PyTorch* es una biblioteca de aprendizaje profundo ampliamente utilizada, especialmente en investigación. Aquí se utiliza para definir y entrenar modelos de redes neuronales, manejar tensores, y aprovechar el procesamiento en GPU. Es la alternativa a *tensorflow* de *keras*.
- **torch.nn:** Este submódulo de *PyTorch* contiene clases para construir redes neuronales, como «*nn.Conv2d*», «*nn.LSTM*», «*nn.Linear*», entre otras capas que forman las redes convolucionales y recurrentes del modelo.
- **torch.nn.functional:** Proporciona funciones para aplicar operaciones matemáticas y funciones de activación, como *reLu* en redes neuronales, que no requieren definición explícita de parámetros entrenables.
- **torch.utils.data:** Incluye utilidades para manejar y cargar datos de manera eficiente. Aquí se utiliza «*DataLoader*», «*Dataset*», y «*TensorDataset*» para empaquetar y cargar los datos en lotes durante el entrenamiento.
- **torch.optim:** Contiene optimizadores de entrenamiento, como *Adam*, que ajustan los pesos de la red durante el aprendizaje para reducir el error.

Resumen de la experimentación

Para alcanzar los objetivos planteados y validar la hipótesis, se llevaron a cabo varios experimentos. **Las Figuras 3.16 y 3.18 ilustran el diagrama de flujo de estos procedimientos** y decisiones clave, destacando los siguientes experimentos y técnicas:

1- Preprocesamiento de la señal EEG:

Conversión de formato y filtrado: Los datos en formato EDF+ fueron convertidos a vectores, facilitando el manejo computacional de la señal para distintos experimentos. Este paso incluyó el uso de un filtro Butterworth para mantener solo las frecuencias relevantes para el análisis de EEG (0.5 a 50 Hz).

Selección de canales y eliminación de eventos irrelevantes: Se llevo a cabo una selección de canales del lóbulo frontal (asociado a la intención motora) y se eliminaron eventos T0 (descanso), enfocados en los eventos de intención motora T1 y T2. Probando tanto la selección de canales como la «no-selección», en los experimentos.

2- Extracción de características y reducción de dimensionalidad:

Transformada Wavelet Discreta (DWT): Para captar información a distintos niveles de resolución temporal y frecuencial, se utilizó una descomposición en cuatro niveles, con un enfoque en frecuencias bajas y la última alta frecuencia. Los experimentos en MATLAB utilizando vectores son llevados a cabo con este procesado.

Métodos de reducción dimensional: Incluyeron la eliminación de artefactos, con *PCA* e *ICA*. Técnicas que no aportaron mejora en la clasificación.

Parámetros estadísticos: Se llevó a cabo el cálculo de parámetros estadísticos asociados a una mejora en la clasificación. Realizando experimentos que no validaron esta idea planteada en la bibliografía.

3- Clasificación de intención motora mediante algoritmos de Machine y Deep Learning:

Machine Learning: Se aplicaron métodos de clasificación iniciales en MATLAB, incluyendo *Support Vector Machines (SVM)*, *K-Nearest Neighbors (KNN)*, Perceptrones multicapa y otros, a un grupo representativo de la base de datos (10 sujetos). Este primer enfoque buscaba evaluar los algoritmos y optimizar las metodologías.

IPMS: Se repitieron los experimentos anteriores teniendo en cuenta el estado mental previo de los sujetos.

Deep Learning: Se aplicaron métodos de clasificación en MATLAB, incluyendo *BiLSTM*, *LSTM*, *GRU* y otros, con el objetivo de analizar su desempeño en este subgrupo y teniendo en cuenta la *DWT* y el *IPMS*.

Modelos híbridos MATLAB: Se implementaron arquitecturas híbridas en MATLAB, combinando los algoritmos de *Machine Learning* con *(Bi)-LSTM* y variando condiciones. Llevando a cabo clasificaciones con toda la base de datos.

- Se utilizaron solo los datos de tareas pensadas para entrenar y testear (utilizando los experimentos 1 y 2 de cada sujeto para entrenar y el 3 para testear)
- Se utilizando datos reales para entrenar y pensados para testear (con diferentes ratios de entrenamiento y test)
- Se probó con distintos tipos de validación en los modelos (*Holdout* y *k-fold*).

Cambio a matrices 2D en MATLAB: Posteriormente, con el cambio de metodología, se crearon matrices 2D a como entradas y se implementaron arquitecturas híbridas en MATLAB. Combinando *CNN*'s con *(Bi)-LSTM* y variando condiciones. Llevando a cabo clasificaciones con toda la base de datos.

Modelos híbridos Python: En la etapa de migración a Python, se implementaron arquitecturas híbridas, combinando *CNN* con *LSTM* y se realizaron clasificaciones para replicar el modelo anterior. Posteriormente se añadieron los mecanismos de atención, con los que se realizaron diferentes pruebas y clasificaciones.

4- Optimización de modelos y evaluación con nuevas arquitecturas:

Vision Transformer (ViT): Para explorar el uso de arquitecturas más avanzadas, se implementaron modelos de *Vision Transformer* y se realizaron clasificaciones en solitario y añadiéndole redes *LSTM*.

5- Tuning de Hiperparámetros

En todos los experimentos se evaluaron diversos hiperparámetros (e.g., *learning rate*, tamaño de *batch*) variándolos para optimizar las clasificaciones.

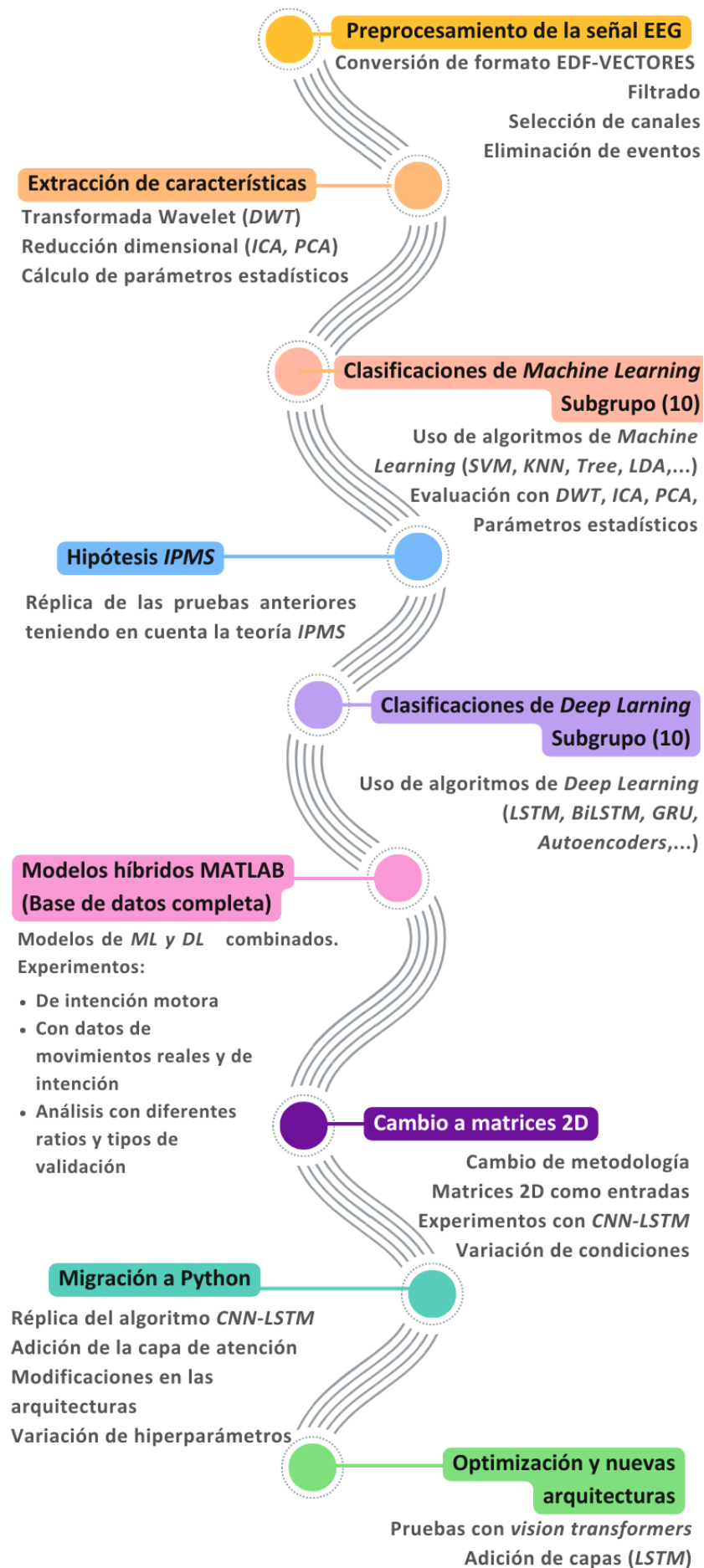


Figura 3.18. Línea de tiempo de las técnicas y los experimentos llevados a cabo para cumplir con la hipótesis y los objetivos. Con ello se busca clasificar la intención motora a través de la señal EEG.

4. Resultados

En esta sección **se presentan los resultados obtenidos a lo largo del desarrollo de la investigación**, los cuales han sido obtenidos mediante la aplicación de la metodología descrita en el apartado 3.

Los resultados aquí expuestos corresponden a las simulaciones más relevantes de entre los cientos que se llevaron a cabo durante el proceso de optimización y evaluación de los modelos.

El flujo de trabajo general que permitió la obtención de los resultados **se puede dividir en dos grandes enfoques**: el análisis inicial utilizando **MATLAB**, y la posterior migración y optimización en **Python**. En ambos entornos se trabajó aplicando modelos individuales y combinados.

Los resultados expuestos en esta sección permiten **evaluar y comparar el impacto de los modelos, la optimización de hiperparámetros y la combinación de diferentes arquitecturas**, tal y como se ilustra en el diagrama de flujo metodológico presentado en la figura 3.16. Posteriormente en el apartado de Discusión, se procederá a interpretar estos resultados, debatiendo su relevancia en relación con los objetivos propuestos y su impacto en el campo de estudio, además de comparar los modelos más efectivos y explorar sus limitaciones.

El hardware utilizado en las simulaciones es un ordenador con un procesador I7-9600-K, 16 GB de RAM, y una tarjeta gráfica *Gigabyte GeForce RTX-2060 Windforce OC-6GB-GDDR6* con 7,03 TFLOPS. Los tiempos de test oscilan entre 25 y 35 ms.

4.1. Machine Learning

Se muestran los **resultados asociados a la metodología del apartado 3.2**. Concretamente aquellos que van desde el punto 3.2.1 «DWT» hasta el 3.2.4 «Clasificadores de *Machine Learning*».

Como se especificó en el apartado 3.2.4 «clasificadores de *Machine Learning*» **al comenzar la experimentación se trabajó con un subconjunto de 10 usuarios** para probar los algoritmos, el entorno y optimizar el manejo de las herramientas de una

manera más simple. Posteriormente se trabaja con el conjunto completo. De este modo se puede probar la eficacia de la metodología planteada para, en caso de ineficiencias, cambiar de estrategia.

4.1.1. Sujetos S001-S010

Se muestran los resultados para la clasificación de la señal de la intención motora mediante EEG de estos 10 sujetos (no del movimiento real). Se utiliza una ratio entrenamiento-test de 75-25. El caso de mejor desempeño se muestra en la figura 4.1. En este no se utiliza ni *ICA*, ni *PCA*. Tampoco se utilizan los parámetros estadísticos, ni el *IPMS*. Se utilizan las *DWT* calculadas previamente en 4 niveles y el filtro butterworth.

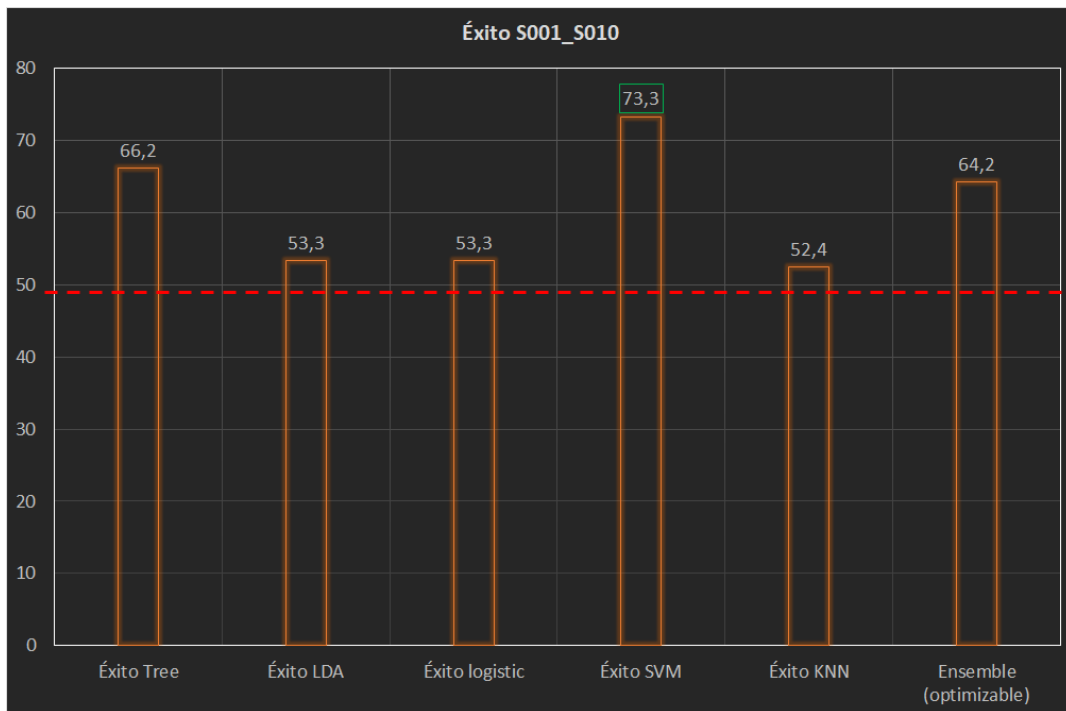


Figura 4.1. Porcentajes de éxito en la clasificación de la intención motora de los 10 primeros sujetos.

Los resultados en las mismas condiciones, pero sin aplicar *DWT*, se muestran en la figura 4.2. En donde aparece en rojo la línea del 50% de exactitud. Los resultados con *ICA* y *PCA* rondan el $50 \pm 6\%$ para ambos casos (con y sin *DWT*).

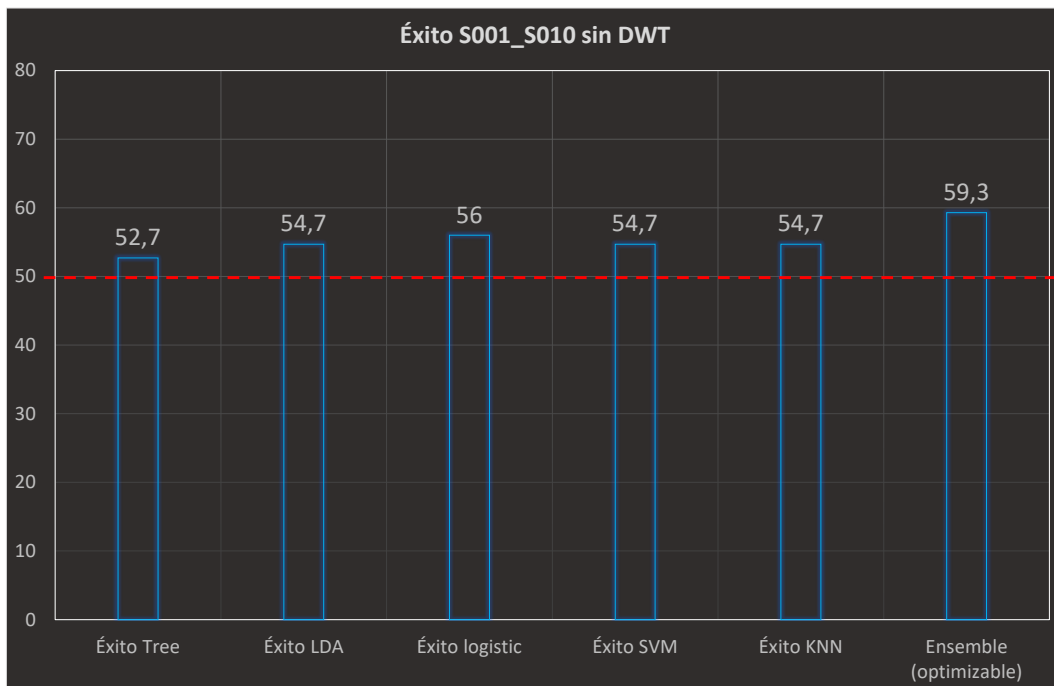


Figura 4.2. Porcentajes de éxito en la clasificación de la intención motora de los 10 primeros sujetos. No ICA. No PCA. No parámetros. No DWT.

4.1.2. Sujetos S001-S010 análisis independiente

Tras realizar la clasificación del subconjunto, se realiza el mismo análisis, pero de manera independiente. El objetivo es observar el comportamiento del sistema cuando el entrenamiento y el test se hacen de manera individual. A continuación, en la tabla 4.3, se muestran los **resultados de la mejor clasificación de los 3 experimentos de intención** (y no los de movimiento real). El sistema se entrena utilizando la configuración que mejor resultado dio en el apartado anterior. Esto es sin *ICA*, *PCA* y sin la utilización de los parámetros estadísticos. Tampoco se añade la hipótesis *IPMS* aún.

Tabla 4.1. Exactitud (éxito) en la clasificación de los 3 experimentos por sujeto de intención motora por usuario utilizando diferentes clasificadores. Ratio training-test: 75-25. No ICA. No PCA. No IPMS. No parámetros. DWT 4 niveles. Clasificación independiente de sujeto.

Usuario	Éxito Tree	Media_tree	Éxito LDA	Media_LDA	Éxito logistic	Media_LOG	Éxito SVM	Media_SVM	Éxito KNN	Media_KNN	Ensamble (optimizable)	Media
S001_exp1_1	33,3	51,1	40,0	44,4	40,0	51,1	60,0	62,2	60,0	71,1	46,7	66,7
S001_exp1_2	53,3		40,0		40,0		53,3		80,0		66,7	
S001_exp1_3	66,7		53,3		73,3		73,3		73,3		86,7	
S002_exp1_1	66,7	66,7	66,7	55,6	73,3	60,0	80,0	66,7	80,0	82,2	73,3	71,1
S002_exp1_2	80,0		66,7		46,7		66,7		80,0		80,0	
S002_exp1_3	53,3		33,3		60,0		53,3		86,7		60,0	
S003_exp1_1	20,0	26,4	40,0	53,3	53,3	48,9	60,0	62,2	53,3	64,4	53,3	60,0
S003_exp1_2	26,0		60,0		40,0		73,3		73,3		66,7	
S003_exp1_3	33,3		60,0		53,3		53,3		66,7		60,0	
S004_exp1_1	26,7	44,5	33,3	53,3	93,3	71,1	53,3	62,2	53,3	68,9	33,3	55,5
S004_exp1_2	66,7		73,3		66,7		73,3		80,0		73,3	
S004_exp1_3	40,0		53,3		53,3		60,0		73,3		60	
S005_exp1_1	73,3	51,1	60,0	48,9	60,0	53,3333333	93,3	75,5	80,0	71,1	73,3	62,2
S005_exp1_2	26,7		26,7		46,7		60,0		60,0		53,3	
S005_exp1_3	53,3		60,0		53,3		73,3		73,3		60,0	
S006_exp1_1	26,7	46,7	53,3	46,7	46,7	51,1	53,3	53,3	46,7	64,5	46,7	60,0
S006_exp1_2	53,3		40,0		86,7		53,3		80,0		66,7	
S006_exp1_3	60,0		46,7		20,0		53,3		66,7		66,7	
S007_exp1_1	40,0	51,1	53,3	51,1	60,0	53,3	60,0	64,4	73,3	73,3	73,3	66,6
S007_exp1_2	73,3		66,7		60,0		80,0		86,7		73,3	
S007_exp1_3	40,0		33,3		40,0		53,3		60,0		53,3	
S008_exp1_1	80,0	42,2	60,0	51,1	80,0	62,2	73,3	64,4	93,3	73,3	80,0	53,3
S008_exp1_2	13,3		46,7		86,7		66,7		53,3		53,3	
S008_exp1_3	33,3		46,7		20,0		53,3		73,3		26,7	
S009_exp1_1	66,7	66,7	46,7	46,7	66,7	53,3	53,3	53,3	66,7	66,7	73,3	71,1
S009_exp1_2	60,0		46,7		53,3		53,3		60,0		66,7	
S009_exp1_3	73,3		46,7		40,0		53,3		73,3		73,3	
S010_exp1_1	86,7	55,6	53,3	37,8	40,0	42,2	73,3	60,0	60,0	55,5	86,7	62,2
S010_exp1_2	40,0		33,3		46,7		53,3		53,3		53,3	
S010_exp1_3	40,0		26,7		40,0		53,3		53,3		46,7	
		TREE		LDA		REGRESIÓN		SVM		K-NN		ENSAMBLE
MEDIA TOTAL		50,2		48,9		54,7		62,4		69,1		62,9

A continuación, en la figura 4.3, puede verse gráficamente un resumen de las clasificaciones de los 10 sujetos de manera independiente. Se muestra resaltado el mejor resultado de cada clasificador.

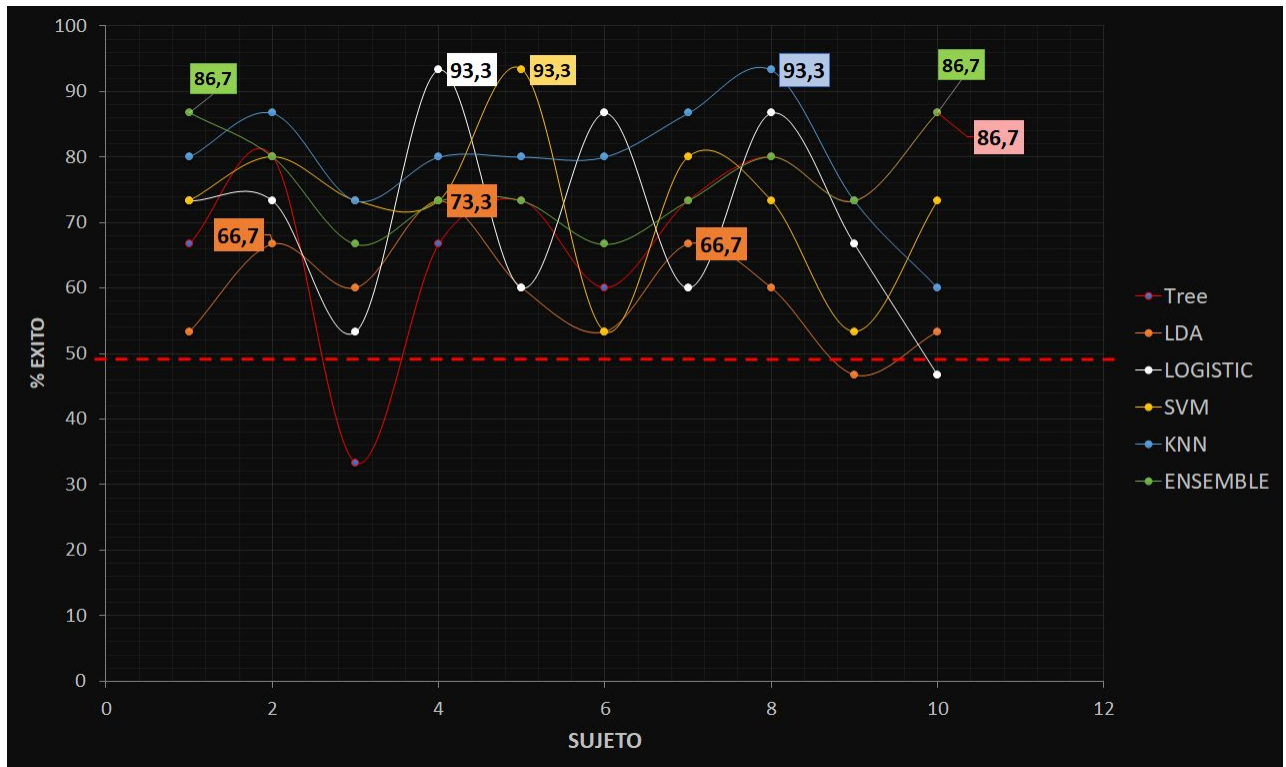


Figura 4.3. Gráficas de los éxitos alcanzados en la clasificación independiente de sujeto con el subconjunto.

Seguidamente, en la figura 4.4 se observan los resultados de los mejores experimentos de cada sujeto. Resaltándose para cada caso el clasificador que mejor desempeño tuvo.

Para ello, se ha tomado el experimento donde mayor éxito se alcanzó en cada sujeto. Mostrando los datos de ese experimento para cada clasificador.

Por lo tanto, como puede verse en los datos, esto solo implica que el experimento elegido es aquel en donde mayor éxito total se alcanzó para ese sujeto. Esto quiere decir, que, por ejemplo, hay casos donde muchos clasificadores presentan buenos resultados, pero se ha elegido aquel caso en donde uno de ellos es el que más destaca en el éxito total sobre el resto de los clasificadores (para ese sujeto).

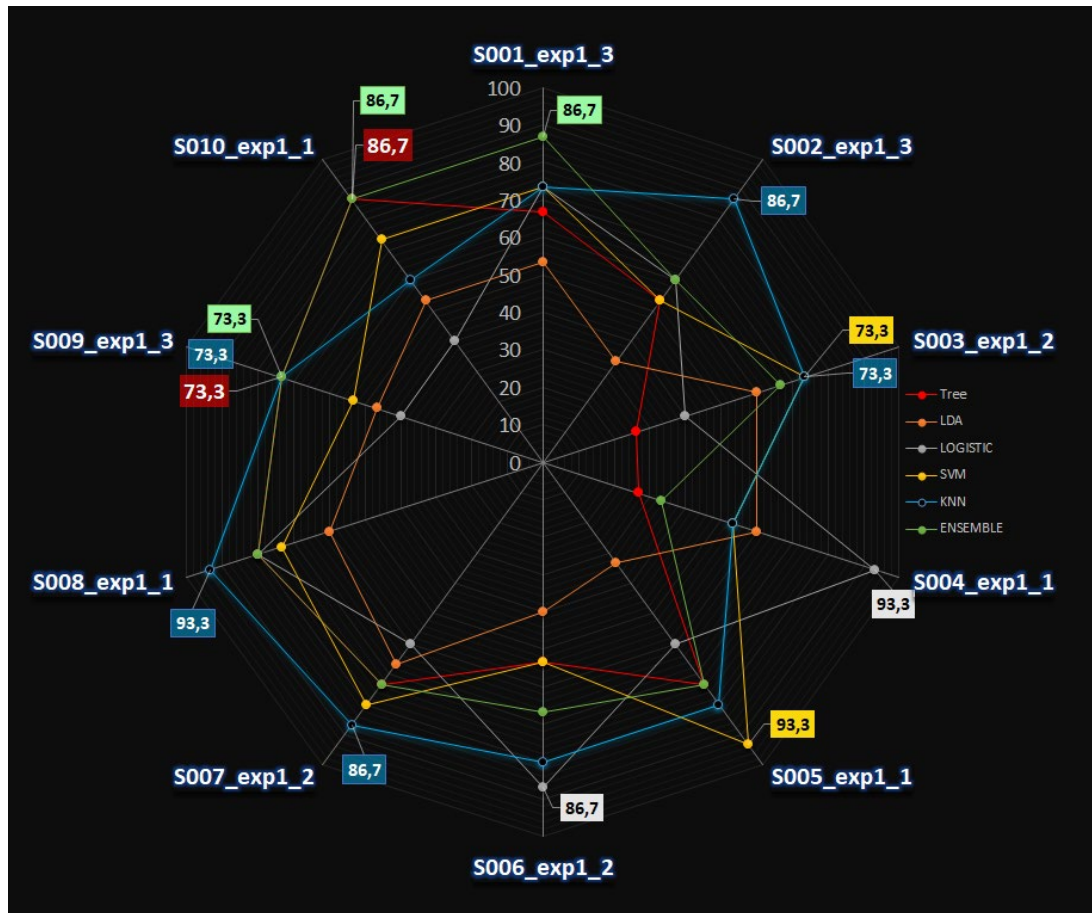


Figura 4.4. Gráficas de los mejores experimentos para cada sujeto y sus porcentajes de éxito.

4.1.3. IPMS

Se muestra a continuación, en la tabla 4.2, el resultado de la clasificación para este subconjunto teniendo en cuenta la hipótesis del estado previo (*IPMS*) y se añade entre paréntesis la clasificación obtenida para el caso sin el *IPMS*, antes visto en la tabla 4.1.

Se resaltan los casos que han sido favorables a la teoría en verde y en rojo los que han tenido un peor desempeño.

Las condiciones de la simulación son las mismas que las utilizadas para obtener los resultados del caso anterior (tabla 4.1).

Tabla 4.2. Clasificación de la intención motora utilizando diferentes clasificadores de *Machine Learning* y teniendo en cuenta la hipótesis del *IPMS*.

Usuario y prueba	Tree	LDA	Log	SVM	KNN	Ensamble
S001_exp1_1	40,0 (33,3)	33,3 (40,0)	53,3 (40,0)	66,7 (60,0)	53,3 (60,0)	53,3 (46,7)
S001_exp1_2	73,3 (53,3)	66,7 (40,0)	80,0 (40,0)	73,3 (53,3)	73,3 (80,0)	86,7 (66,7)
S001_exp1_3	53,3 (66,7)	33,3 (53,3)	73,3 (73,3)	60,0 (73,3)	66,7 (73,3)	73,3 (86,7)
S002_exp1_1	26,7 (66,7)	86,7 (66,7)	86,7 (73,3)	53,3 (80,0)	53,3 (80,0)	46,7 (73,3)
S002_exp1_2	73,3 (80,0)	60,0 (66,7)	46,7 (46,7)	86,7 (66,7)	86,7 (80,0)	73,3 (80,0)
S002_exp1_3	20,0 (53,3)	40,0 (33,3)	46,7 (60,0)	53,3 (53,3)	66,7 (86,7)	66,7 (60,0)
S003_exp1_1	46,7 (20,0)	46,7 (40,0)	20,0 (53,3)	53,3 (60,0)	73,3 (53,3)	53,3 (53,3)
S003_exp1_2	93,3 (26,0)	66,7 (60,0)	73,3 (40,0)	73,3 (73,3)	80,0 (73,3)	93,3 (66,7)
S003_exp1_3	73,3 (33,3)	46,7 (60,0)	60,0 (53,3)	53,3 (53,3)	53,3 (66,7)	80,0 (60,0)
S004_exp1_1	60,0 (26,7)	53,3 (33,3)	40,0 (93,3)	53,3 (53,3)	60,0 (53,3)	73,3 (33,3)
S004_exp1_2	60,0 (66,7)	46,7 (73,3)	53,3 (66,7)	53,3 (73,3)	60,0 (80,0)	60,0 (73,3)
S004_exp1_3	93,3 (40,0)	66,7 (53,3)	53,3 (53,3)	66,7 (60,0)	80,0 (73,3)	93,3 (60,0)
S005_exp1_1	60,0 (73,3)	60,0 (60,0)	53,3 (60,0)	66,7 (93,3)	73,3 (80,0)	53,3 (73,3)
S005_exp1_2	60,0 (26,7)	80,0 (26,7)	73,3 (46,7)	80,0 (60,0)	80,0 (60,0)	66,7 (53,3)
S005_exp1_3	40,0 (53,3)	40,0 (60,0)	40,0 (53,3)	53,3 (73,3)	53,3 (73,3)	46,7 (60,0)
S006_exp1_1	53,3 (26,7)	46,7 (53,3)	66,7 (46,7)	66,7 (53,3)	60,0 (46,7)	60,0 (46,7)
S006_exp1_2	80,0 (53,3)	73,3 (40,0)	80,0 (86,7)	80,0 (53,3)	80,0 (80,0)	86,7 (66,7)
S006_exp1_3	26,7 (60,0)	53,3 (46,7)	53,3 (20,0)	53,3 (53,3)	66,7 (66,7)	66,7 (66,7)
S007_exp1_1	60,0 (40,0)	46,7 (53,3)	66,7 (60,0)	53,3 (60,0)	86,7 (73,3)	60,0 (73,3)
S007_exp1_2	33,3 (73,3)	33,3 (66,7)	53,3 (60,0)	53,3 (80,0)	46,7 (86,7)	53,3 (73,3)
S007_exp1_3	66,7 (40,0)	40,0 (33,3)	46,7 (40,0)	60,0 (53,3)	86,7 (60,0)	66,7 (53,3)
S008_exp1_1	66,7 (80,0)	40,0 (60,0)	53,3 (80,0)	60,0 (73,3)	66,7 (93,3)	60,0 (80,0)
S008_exp1_2	73,3 (13,3)	73,3 (46,7)	93,3 (86,7)	86,7 (66,7)	80,0 (53,3)	93,3 (53,3)
S008_exp1_3	53,3 (33,3)	53,3 (46,7)	46,7 (20,0)	60,0 (53,3)	86,7 (73,3)	66,7 (26,7)
S009_exp1_1	66,7 (66,7)	60,0 (46,7)	53,3 (66,7)	73,3 (53,3)	73,3 (66,7)	66,7 (73,3)
S009_exp1_2	60,0 (60,0)	46,7 (46,7)	26,7 (53,3)	46,7 (53,3)	66,7 (60,0)	60,0 (66,7)
S009_exp1_3	60,0 (73,3)	33,3 (46,7)	33,3 (40,0)	53,3 (53,3)	53,3 (73,3)	60,0 (73,3)
S010_exp1_1	40,0 (86,7)	53,3 (53,3)	33,3 (40,0)	53,3 (73,3)	80,0 (60,0)	60,0 (86,7)
S010_exp1_2	73,3 (40,0)	46,7 (33,3)	46,7 (46,7)	60,0 (53,3)	73,3 (53,3)	73,3 (53,3)
S010_exp1_3	40,0 (40,0)	26,7 (26,7)	73,3 (40,0)	53,3 (53,3)	53,3 (53,3)	46,7 (46,7)

4.1.4. Ensamblados de *Deep y Machine Learning*

A continuación, como se explicó en el último punto del apartado 3.2.4, **se realiza una clasificación con diferentes ensamblados (o conjuntos) compuestos de algoritmos de *Deep y Machine Learning*.**

Para ello, se muestran **primero los resultados de los clasificadores de *Deep Learning* en solitario** y utilizando la **misma metodología que con los clasificadores de *Machine Learning*** (clasificar los vectores de intención motora a los cuales se les aplica *DWT* e *IPMS*). El objetivo es **analizar el porcentaje de éxito de estos y entender de qué manera podrían contribuir al ensamble.**

En la tabla 4.3, se muestran los **diferentes algoritmos analizados**, su **porcentaje de éxito** en la clasificación de la señal de la intención motora obtenida según la metodología previamente explicada y el **ajuste de hiperparámetros utilizado**. Se muestran los **casos de mayor porcentaje** de éxito con el objetivo de simplificar cientos de simulaciones en las que se han hecho diferentes combinaciones y se han variado decenas de parámetros.

Tabla 4.3. Clasificación de la intención motora utilizando diferentes clasificadores de *Deep Learning*, aplicando *DWT* y teniendo en cuenta la hipótesis del IPMS.

Algoritmo	<i>LSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>GRU</i>	Perceptrón	<i>Autoencoder**</i>
Numero características	8184	8184	8184	8184	8184	8184
Neuronas capa oculta	50	1000	50	50	400	50
HIPERPARÁMETROS DE ENTRENAMIENTO						
Método	<i>rmsprop</i>	<i>rmsprop</i>	<i>rmsprop</i>	<i>rmsprop</i>	-	<i>rmsprop</i>
Función de activación	<i>tanh</i>	<i>tanh</i>	<i>tanh</i>	<i>tanh</i>	-	<i>tanh</i>
Función de activación <i>gate</i>	<i>Sigmoid</i>	<i>Sigmoid</i>	<i>Sigmoid</i>	<i>Sigmoid</i>	-	<i>Sigmoid</i>
Modo <i>output</i>	<i>sequence</i>	<i>sequence</i>	<i>sequence</i>		-	<i>sequence</i>
<i>Max Epoch</i>	30	90	90	90	-	90
<i>Gradient Threshold Method</i>	l2norm	l2norm	l2norm	l2norm	-	l2norm
<i>Gradient Threshold Method</i>	2	1	1	1	-	1
Mostrar en pantalla	<i>True</i>	<i>True</i>	<i>True</i>	<i>True</i>	-	<i>True</i>
<i>Momentum</i>	-		-	-	-	-
<i>Gradient Decay Factor</i>	-		-	-	-	-
<i>Learn Rate Schedule</i>	<i>none</i>	<i>none</i>	<i>none</i>	<i>none</i>	-	<i>none</i>
<i>Learn Rate Drop Factor</i>	<i>none</i>	<i>none</i>	<i>none</i>	<i>none</i>	-	<i>none</i>
<i>Learn Rate Drop Period</i>	<i>none</i>	<i>none</i>	<i>none</i>	<i>none</i>	-	<i>none</i>
<i>Initial Learn Rate</i>	<i>Default</i>	<i>Default</i>	<i>Default</i>	<i>Default</i>	-	<i>Default</i>
<i>Normalized (matlab normalized default)</i>	<i>none</i>	<i>none</i>	<i>none</i>	<i>none</i>	-	Sí
Exactitud (%)	57,78	52,22	61,11	58,89	63,33	54,44

113 **El *autoencoder* presenta además los siguientes parámetros; Encoder Transfer Function (*logsig*), Decoder Transfer Function (*logsig*), L2Weight Regularization (0,001), Sparsity Regularization (1) y Sparsity Proportion (0,05)

Tras analizar los resultados, el siguiente paso consiste en probar un ensamble de los métodos de *Machine Learning* con los clasificadores anteriores, siendo el método ***BiLSTM*** el que mejor responde a esta combinación.

Se muestran en la tabla 4.4, los experimentos que mejor desempeño tuvieron en la clasificación de la señal utilizando estas combinaciones con el subgrupo de 10 usuarios y una ratio entrenamiento-test del 80-20.

Tabla 4.4. Resultados de la clasificación utilizando un ensamble compuesto por una capa *BiLSTM* y diferentes capas de clasificadores de *Machine Learning*.

Algoritmos DL	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>
Neuronas ocultas	50	50	50	50	50
	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>
	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>
Algoritmos ML	-	Perceptrón	Perceptrón	Perceptrón	Perceptrón
	-	-	-	<i>Tree</i>	<i>Tree</i>
	-	-	-	-	<i>LDA</i>
Iteraciones ensamble	No	10	5	5	5
Exactitud (%)	65,56	63,33	66,67	63,33	65,78

Puesto que las pruebas con el subgrupo han sido utilizadas para testear y probar las diferentes metodologías, el siguiente paso es probar **este ensamble en diferentes condiciones y con la base de datos completa (109 usuarios)**, a excepción de 4 sujetos que presentaba datos corruptos.

Las condiciones en las que se realizaron las simulaciones con este ensamble fueron:

- **Clasificación con la base de datos completa (105)** y diferentes hiperparámetros. Ratio de entreno 75-25.
- Clasificación utilizando los **experimentos 1 y 2 de los sujetos para entrenar y el experimento 3 para hacer el test ciego**
- Clasificación con los **datos de los experimentos reales para entrenar y las intenciones motoras para testear.**

Ensamble y base de datos completa (105)

El resultado es similar al de la tabla 4.4. Oscilando entre el 65% y el 68% de éxito utilizando el ensamble de un *BiLSTM* en conjunto con los métodos de *Machine Learning* (*KNN*, *SVM*, *Tree*, *LDA* y perceptrón).

Experimentos 1 y 2 vs 3 de intención motora

Para este caso, el entrenamiento se realiza con los experimentos 1 y 2 de cada sujeto y se testea con los experimentos 3. El resultado de las mejores clasificaciones viene dado en la tabla 4.5:

Tabla 4.5. Resultados de la clasificación del ensamble utilizando los experimentos 1 y 2 para entrenar y el experimento 3 para testear.

Algoritmos DL	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>	<i>BiLSTM</i>
Neuronas ocultas	20	50	50	500	2000
	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>	<i>KNN</i>
	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>
Algoritmos ML	Perceptrón	Perceptrón	Perceptrón	Perceptrón	Perceptrón
	<i>Tree</i>	<i>Tree</i>	<i>Tree</i>	<i>Tree</i>	<i>Tree</i>
	-	-	LDA	-	-
Iteraciones ensamble	50	5	5	50	5
Exactitud (%)	72,67	75,33	69,33	73,33	74,67

Experimentos movimiento real vs intención

Se realizan seguidamente clasificaciones utilizando los datos de movimiento real y los datos de intención motora para entrenar y testear. Se prueba con diferentes ratios de entrenamiento- test. Se utiliza el *BiLSTM* en el ensamble y se varían los clasificadores de para evaluar los resultados. En la tabla 4.6, se muestran los resultados para las diferentes ratios propuestos. Estas proporciones son:

- Real vs intención (50:50). Experimentos reales para entrenamiento, y, experimentos de intenciones para testear.

- Real vs intención (67:33). Experimentos reales y primer experimento de intención motora de cada sujeto para entrenamiento y los experimentos 2 y 3 de intenciones motoras para test.
- Real vs intención (83:17). Experimentos reales y experimentos 1 y 2 de intención motora para entrenamiento y los experimentos 3 de intención motora para testear.

Tabla 4.6. Resultados de la clasificación con ensambles utilizando diferentes ratios. Los datos corresponden a experimentos de movimiento real en conjunto con los de intención

Proporción 50:50					
Algoritmos DL Neuronas ocultas	<i>BiLSTM</i>	-	-	-	<i>BiLSTM</i>
	50	-	-	-	50
Algoritmos ML	<i>KNN</i>	-	-	-	-
	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>
	Perceptrón	Perceptrón	Perceptrón	-	Perceptrón
	<i>Tree</i>	-	-	-	-
	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>
Exactitud (%)	69,69	72,15	72,00	72,51	72,27
Proporción 67:33					
Algoritmos DL Neuronas ocultas	<i>BiLSTM</i>	-	-	-	<i>BiLSTM</i>
	50	-	-	-	50
Algoritmos ML	<i>KNN</i>	-	-	-	-
	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>
	Perceptrón	Perceptrón	Perceptrón	-	Perceptrón
	<i>Tree</i>	-	-	-	-
	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>
Exactitud (%)	70,44	71,56	71,65	72,06	71,97
Proporción 83:17					
Algoritmos DL Neuronas ocultas	<i>BiLSTM</i>	-	-	-	<i>BiLSTM</i>
	50	-	-	-	50
Algoritmos ML	<i>KNN</i>	-	-	-	-
	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>	<i>SVM</i>
	Perceptrón	Perceptrón	Perceptrón	-	Perceptrón
	<i>Tree</i>	-	-	-	-
	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>	<i>LDA</i>
Exactitud (%)	72,51	73,71	73,33	73,39	73,2

4.2. *Deep Learning*

A continuación, como se explicó en el apartado 3.2.5 «Cambio de paradigma. Matrices 2D» se utiliza el **conjunto completo de los datos y algoritmos más complejos**, como lo son los modelos de *Deep Learning*. Se realiza una transición en donde se **evalúan estos modelos y se combinan** para probar diferentes condiciones.

Se hace una distinción entre los resultados obtenidos en MATLAB de los resultados obtenidos en Python. Los resultados se muestran en los apartados siguientes:

4.2.1. Matrices 2D. Pruebas con *CNN* y *LSTM*

Siguiendo con la dinámica anterior, se realiza una **clasificación de la señal de la mano derecha-izquierda**, combinando datos de movimientos reales con intención motora, con el objetivo de comprobar si esta metodología presenta mejores resultados que los ensambles anteriormente vistos.

Se utilizan matrices de entradas de datos como se comentó previamente y se trabaja con una **arquitectura híbrida *CNN* – *LSTM***. Se utiliza una ratio de 70:30 para entrenamiento-test. Los resultados se muestran en la figura 4.5.

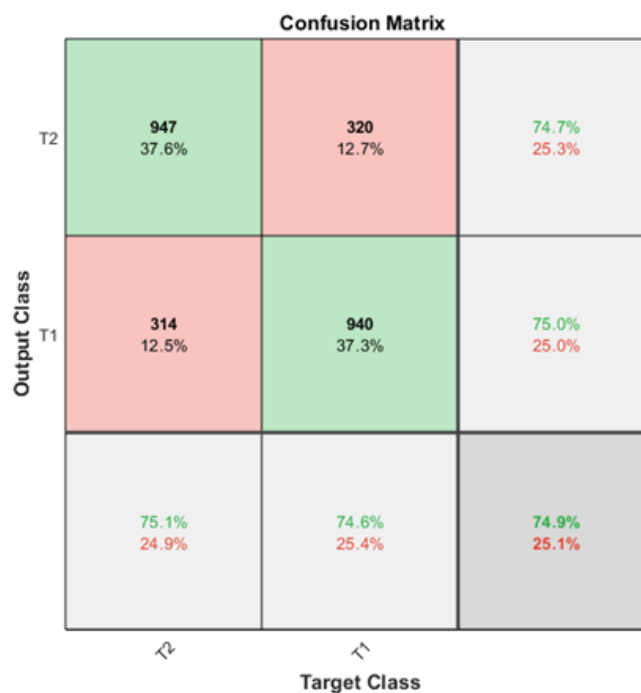


Figura 4.5. Matriz de confusión de la primera clasificación utilizando *CNN-LSTM*.

Seguidamente se realizan decenas de simulaciones variando condiciones tanto en la arquitectura (añadiendo capas o modificándolas), como variando hiperparámetros.

De este modo se obtienen diversos resultados. Uno de los más interesantes es el mostrado en la figura 42, en donde **se obtiene un 79,64% de éxito clasificando la señal de la intención motora** con un k-fold=10. Se puede observar la arquitectura del clasificador, con vectores 2D como entradas, así como, sus capas y parámetros.

```

lgraph = layerGraph();
cascademode = [
    sequenceInputLayer([1 1 10 1], "Name", "inputcnn", "Normalization", "zscore")
    sequenceFoldingLayer("Name", "seqfold")
    convolution2dLayer([3 3], 32, "Name", "conv_1", "Padding", "same", "PaddingValue", "replicate")
    convolution2dLayer([3 3], 64, "Name", "conv_2", "Padding", "same", "PaddingValue", "replicate")
    convolution2dLayer([3 3], 128, "Name", "conv_3", "Padding", "same", "PaddingValue", "symmetric-include-edge")
    fullyConnectedLayer(2, "Name", "fc1")
    dropoutLayer(0.5, "Name", "dropout1")
    sequenceUnfoldingLayer("Name", "sequnfold")
    flattenLayer("Name", "flatten")
    lstmLayer(64, 'OutputMode', 'last', 'name', 'lstm1')
    lstmLayer(64, 'OutputMode', 'sequence', 'name', 'lstm2')
    fullyConnectedLayer(2, "Name", "fc2")
    dropoutLayer(0.5, "Name", "dropout2")
    softmaxLayer("Name", "softmax")
    classificationLayer("Name", "OUT")];

lgraph = addLayers(lgraph, cascademode);
lgraph = connectLayers(lgraph, 'seqfold/miniBatchSize', 'sequnfold/miniBatchSize');
% CONEXIONES
options = trainingOptions('adam', ...
    'GradientDecayFactor', 0.8, ...
    'SquaredGradientDecayFactor', 0.990, ...
    'Epsilon', 1e-8, ...
    'LearnRateSchedule', 'piecewise', ...
    'InitialLearnRate', 0.0001, ...
    'GradientThresholdMethod', 'l2norm', ...
    'MaxEpochs', 90, ...
    'Shuffle', 'every-epoch', ...
    'MiniBatchSize', 128, ...
    'ExecutionEnvironment', 'gpu', ...
    'Verbose', true, ...
    'Plots', 'training-progress');
  
```

Figura 4.6. Despliegue de la arquitectura CNN-LSTM e hiperparámetros en MATLAB. 79,64% de éxito en la clasificación.

Tras obtener este resultado se realiza un mayor esfuerzo por encontrar la **configuración óptima** y en consecuencia se obtiene la tabla 4.7. En ella se muestran diversos experimentos, sus cambios más significativos con respecto a la arquitectura de la figura 4.6 y sus porcentajes de éxito en la clasificación de la señal de intención motora.

Tabla 4.7. Exactitud de la clasificación variando condiciones en la arquitectura previa.

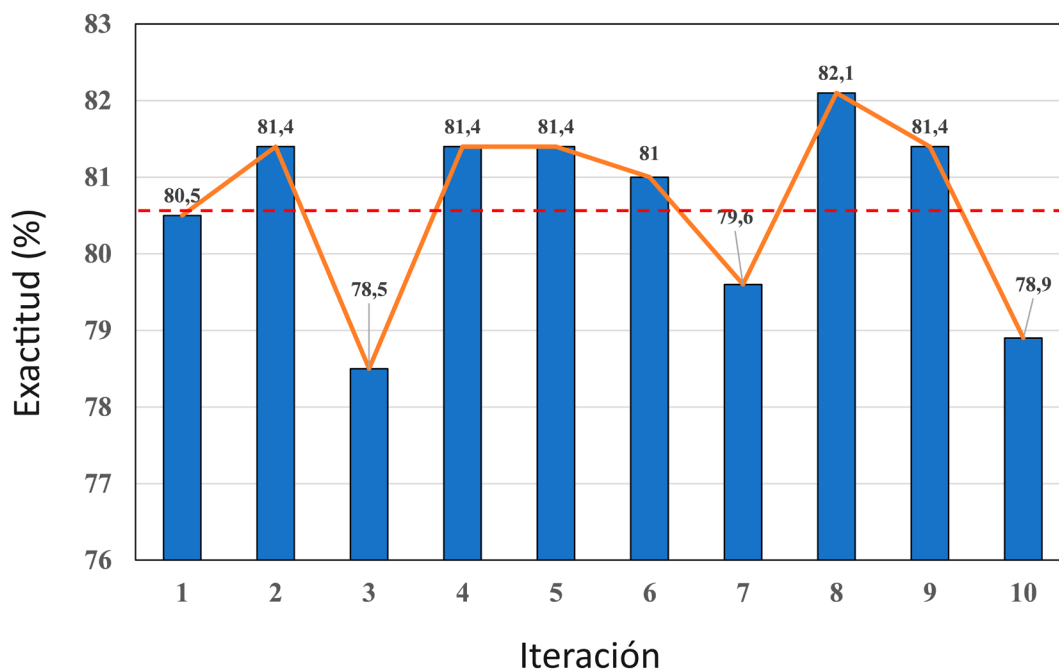
Condiciones	Exactitud (%)
Cambio de capa <i>softmax</i> por <i>Sigmoid</i>	50,00%
Adición de una normalización (<i>layerNormalizationLayer</i>)	63,00%
Cambio de la parte final de la <i>CNN</i> « <i>Fully connected</i> » y « <i>dropout</i> » por « <i>batchNormalizationLayer</i> », « <i>reluLayer</i> » y « <i>maxPooling2dLayer</i> ([3 3], Stride= [3 3]) » en serie.	76,00%
Cambio en el tamaño de filtro de las convoluciones a [10 11]	79,50 ± 2,30
Cambio en el tamaño de filtro de las convoluciones a [4 4]	79,60 ± 2,00
Adición de un « <i>maxpooling</i> [3]» después de las convoluciones	79,90% ± 2,60
Reestructuración de convoluciones por un total de 6: Dobles convoluciones de 32(x2), 64(x2) y 128(x2). Las primeras en modo « <i>replicate</i> » y las segundas en « <i>symmetric-include-edge</i> »	80,20 ± 3,30

Finalmente se realiza esta clasificación utilizando la arquitectura de la figura 4.5. Se establece el método k-fold y realizan 10 iteraciones para hacer una media de 10 clasificaciones. Los resultados se recogen los en la tabla 4.8.

Tabla 4.8. Resumen de las métricas de cada iteración en la clasificación de la señal de los 105 usuarios.

k-fold 10	Exactitud (%)		Especificidad (%)		κ	<i>Youden</i>	Exactitud total (%)
Iteración	T1	T2	T1	T2			
1	78,9	82,3	78,8	83,2	0,611	0,610	80,5
2	81,2	81,7	80,9	81,9	0,628	0,628	81,4
3	78,4	78,5	78,2	78,7	0,570	0,569	78,5
4	81,1	81,7	81,8	81,8	0,628	0,628	81,4
5	82,3	80,5	82,7	80,1	0,628	0,628	81,4
6	82,1	79,9	81,9	79,1	0,620	0,619	81,0
7	82,3	80,5	82,7	80,3	0,628	0,628	81,4
8	84,1	80,3	83,6	79,2	0,642	0,642	82,1
9	79,5	79,6	79,6	79,5	0,592	0,591	81,4
10	78,3	77,7	78,1	80,1	0,578	0,579	78,9
Media	80,8 ± 2,0	80,3 ± 1,4	81,0 ± 2,3	80,4 ± 1,5	0,6 ± 0,02	0,6 ± 0,02	80,6 ± 1,2

Los resultados del modelo muestran una exactitud media del $80,6\% \pm 1,2\%$ y una especificidad media del $80,7\% \pm 1,9\%$ (sumando las 2 clases), así como un coeficiente *kappa* del $0,6\% \pm 0,02\%$ y un índice de *Youden* con el mismo valor ($0,6\% \pm 0,02\%$). En la figura 4.7 se muestra gráficamente la exactitud total alcanzada en cada una de las simulaciones. En rojo se muestra la exactitud media total de 80,6.

**Figura 4.7.** Éxitos en las iteraciones de las clasificaciones y media total en rojo.

4.2.2. Migración a Python. Replica de pruebas.

Tras las pruebas llevadas a cabo en **MATLAB**, se observaron ciertas limitaciones inherentes a este entorno, principalmente debido a su naturaleza hermética en cuanto a personalización y flexibilidad en la manipulación detallada de los algoritmos.

Por tanto, tal como se explicó en el **apartado 3.3**, se tomó la decisión de migrar todo el proceso de desarrollo y experimentación a **Python**, lo que permitió ahondar con mayor profundidad en el ajuste de hiperparámetros y la creación de arquitecturas híbridas. Los experimentos realizados tras la migración incluyen:

- **Réplica del algoritmo de matrices 2D**, mostrado en la figura 3.11 y 3.12.
- **Evaluación de las capas de Atención (*Attention Layers*)**, figura 3.14.
- **Modelos híbridos (*CNN-BiLSTM*, *CNN-LSTM con capa de atención*)**, figura 3.13.
- **Nuevos modelos. *Transformers* y *ViT***, figura 3.15.

Los resultados experimentales de los 3 primeros puntos se muestran en la **tabla 4.9**. Donde se encuentra resumida la arquitectura y el éxito conseguido junto con los detalles que hacen diferir unas de las otras.

Posteriormente se añaden figuras, de la 4.8 a la 4.15, donde se muestran las **matrices de confusión** de algunos de los modelos que se han usado y que se recogen en la tabla 4.9.

Por último, se muestran los resultados obtenidos en la recta final de la investigación. Estos muestran los nuevos modelos como el *ViT* y su desempeño en la clasificación de las tareas de intención motora.

Tabla 4.9. Resumen de las métricas de cada iteración en la clasificación de la señal de los 105 usuarios.

Arquitectura	Exactitud (%)	Característica
<i>CNNx3-LSTMx2</i>	76,00	<i>CNNx3</i> combinada con <i>LSTMx2</i>
<i>CNNx3-Attention-LSTMx2</i> (1º Ronda)	58,00	Capa atención entre la <i>CNNx3</i> y <i>LSTMx2</i>
<i>CNNx3-Attention-LSTMx2</i> (2º Ronda)	57,00	<i>Kernel</i> gaussiano añadido en <i>CNN</i>
<i>CNNx2-maxpooling-CNN-attention-LSTMx2-attention</i>	75,28	Uso de <i>maxpooling</i> y capa atención en la <i>CNN</i>
<i>CNNx2-maxpooling-CNN-attention-LSTMx2</i>	75,81	Capa atención y <i>maxpooling</i> en capas <i>CNN</i>
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i>	77,09	Combinación de varias capas de <i>maxpooling</i> y atención
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> iteración adicional	82,88	Iteración adicional de entrenamiento en <i>LSTM</i>
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> epoch 100	83,66	Entrenado durante 100 épocas
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> doble entreno	84,63	Doble entrenamiento con mayor <i>learning rate</i>
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> K FOLD 5	81,29	K-Fold aplicado a la arquitectura <i>CNN-LSTM</i>
<i>CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM</i> batch (62) y epoch (150)	85,19	Ajuste en el tamaño del <i>batch</i> y mayor número de épocas

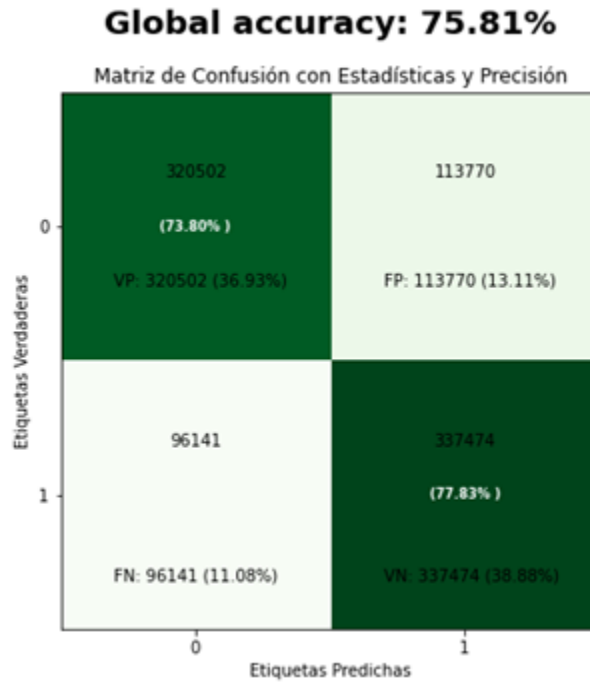


Figura 4.8. Matriz de confusión del experimento 4 de la tabla 4.9. *CNNx2-maxpooling CNN-attention-LSTMx2-attention*.

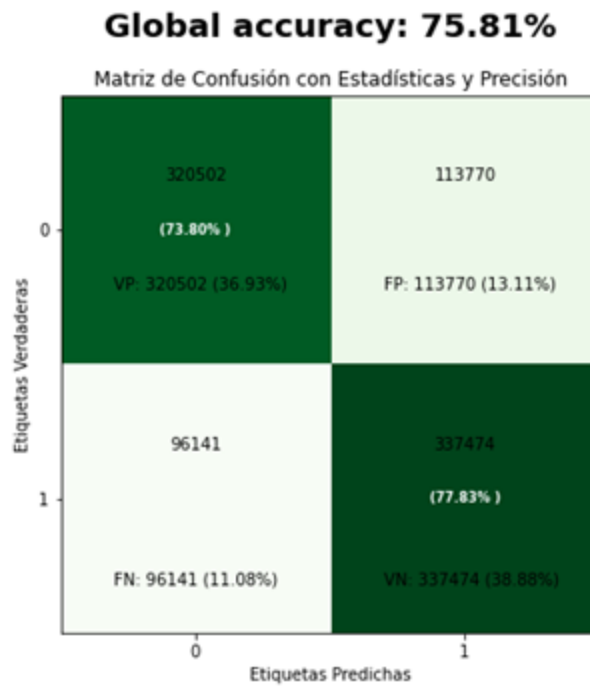


Figura 4.9. Matriz de confusión del experimento 5 de la tabla 4.9. *CNNx2-maxpooling-CNN-attention-LSTMx2*.

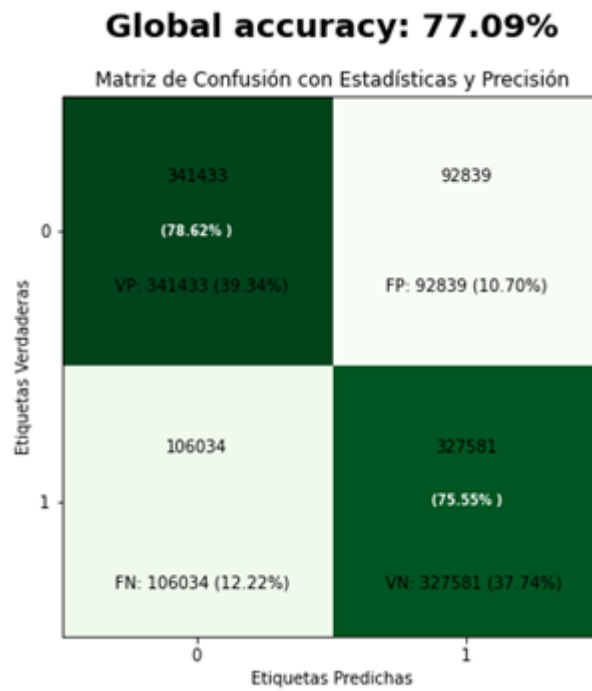


Figura 4.10. Matriz de confusión del experimento 6 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM*.

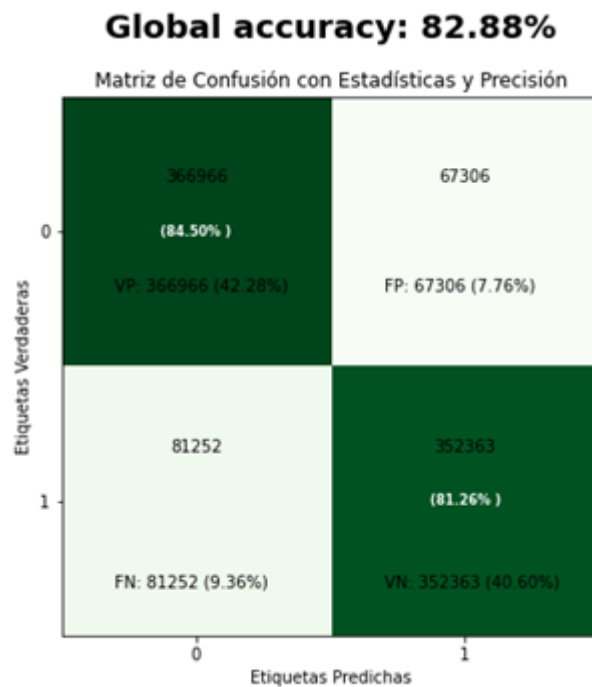


Figura 4.11. Matriz de confusión del experimento 7 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM* iteración adicional.

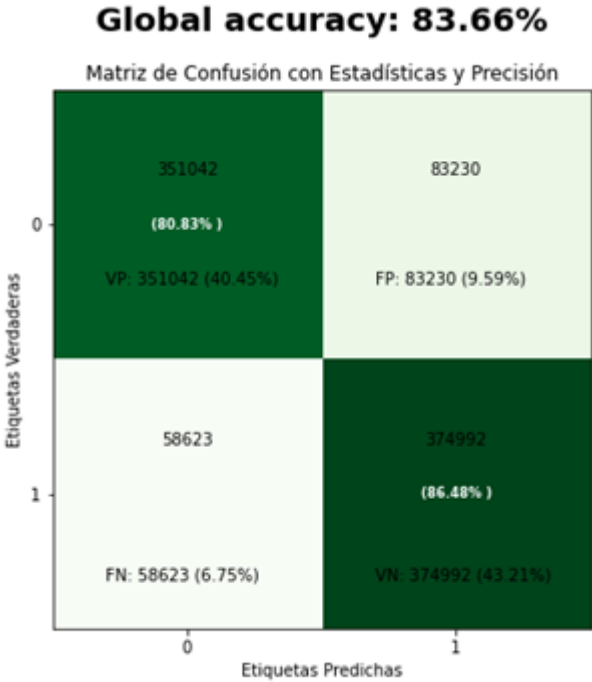


Figura 4.12. Matriz de confusión del experimento 8 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM epoch 100.*

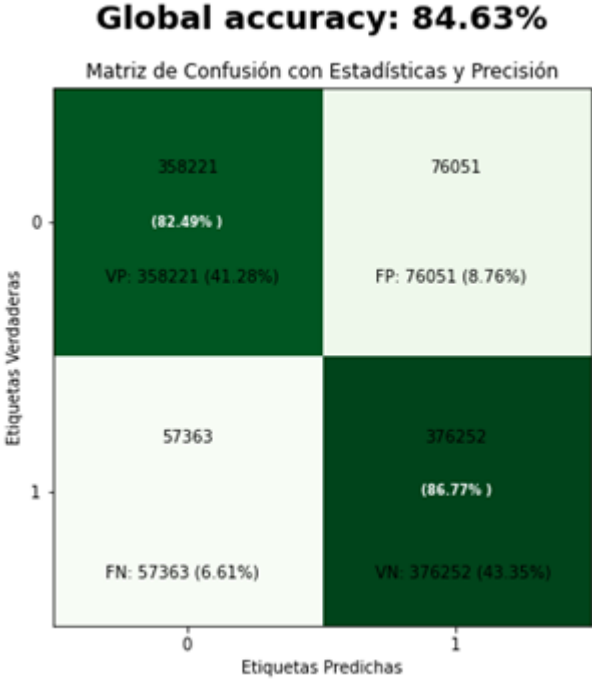


Figura 4.13. Matriz de confusión del experimento 9 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM doble entreno.*

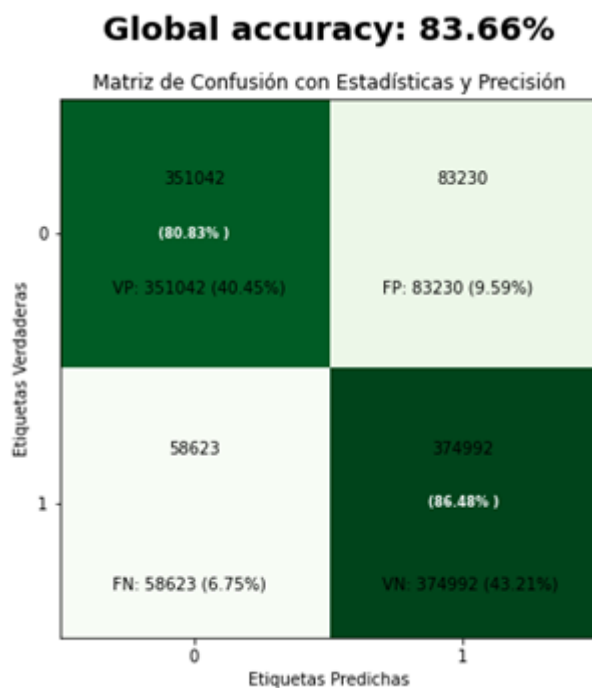


Figura 4.14. Matriz de confusión del experimento 10 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM K FOLD 5*.

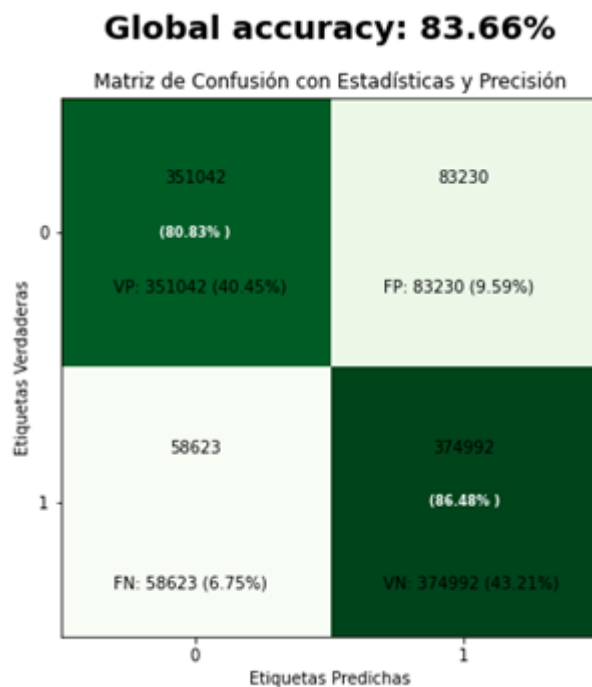


Figura 4.15. Matriz de confusión del experimento 11 de la tabla 4.9. *CNN-maxpooling-CNN-maxpooling-CNN-attention-LSTM batch (62) y epoch (150)*.

Vision transformer

Como se mencionó anteriormente, se muestran los resultados de aplicar la arquitectura del ViT (figura 3.15) a los datos y realizar la clasificación binaria.

En este caso se aplica el algoritmo con y sin la adición de una capa *LSTM* en el modelo. Dando lugar a la figura 4.16, donde si se le ha aplicado la *LSTM* y a la figura 4.17, donde NO se le ha aplicado *LSTM*.

- Experimento del ViT con *LSTM*:

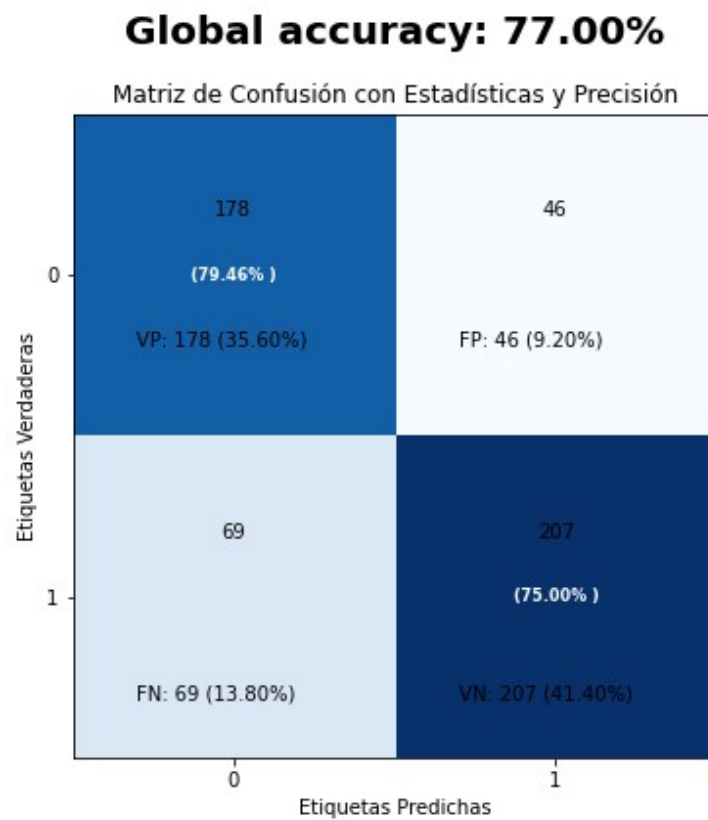


Figura 4.16. Matriz de confusión del experimento utilizando *vision transformer* y una capa *LSTM* final para complementar el modelo.

- Experimento del **ViT SIN LSTM**:

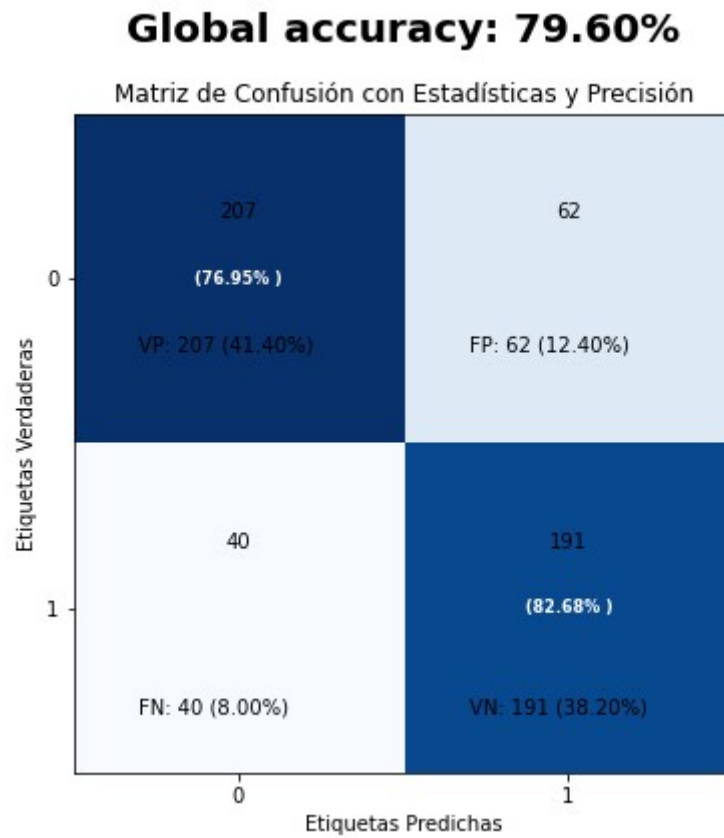


Figura 4.17. Matriz de confusión del experimento utilizando el *vision transformer* de la figura 3.15.

La **función de pérdida asociada al entrenamiento** se muestra en la figura 4.18.

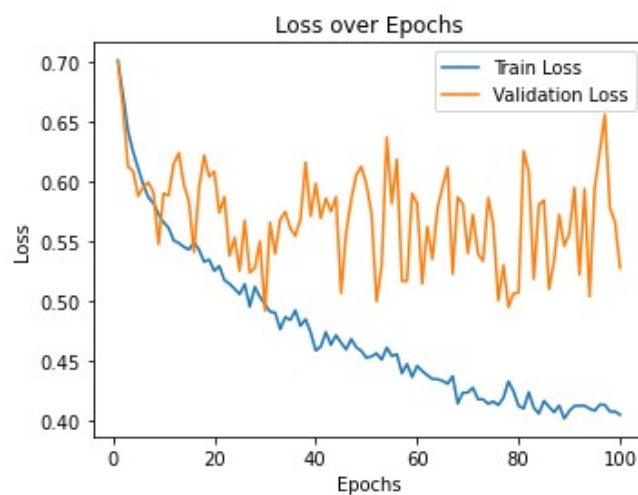


Figura 4.18. Evolución de la función de pérdida en cada iteración (*epoch*).

5. Discusión

Tras haber satisfecho el «objetivo 1» de la tesis y haber ahondado en el campo de las señales cerebrales y las BCI's, en esta sección **se presenta una discusión exhaustiva de los resultados obtenidos** en el análisis de señales EEG **para la clasificación de la intención motora.**

Cada subsección incluye un análisis detallado de las tablas y figuras presentadas en el apartado de resultados, así como una explicación profunda de los factores que influyen en cada uno de los hallazgos. El objetivo es ofrecer un **análisis riguroso que permita entender cada uno de los resultados y enlazarlos con los objetivos e hipótesis de esta tesis.**

5.1. Clasificaciones con *Machine Learning*

Se discuten en este apartado los resultados de las clasificaciones **utilizando *Machine Learning***, aplicado al subgrupo de 10 sujetos. Se analizan las clasificaciones del subconjunto entero y del análisis independiente de sujeto con los clasificadores.

Se hace hincapié en las técnicas antes mencionadas (*ICA, PCA, DWT*, parámetros estadísticos) que permiten explorar distintos escenarios y obtener respuesta al «objetivo 2» de la tesis.

5.1.1. Clasificación de la intención motora mediante EEG. Sujetos S001-S010

Los resultados con el subconjunto de diez sujetos (**S001-S010**) mostraron un **rendimiento aceptable, comparado con el estado del arte (tabla 1.3), de hasta un 73,3%**, como puede verse en la figura 4.1. Siendo el clasificador *SVM* el de mayor exactitud y los de peor exactitud *LDA, logistic* y *KNN*.

La combinación del filtro Butterworth y la *DWT* se ha demostrado eficaz en el procesamiento de señales EEG como cabía esperar. El filtro Butterworth, conocido por su capacidad de eliminar ruido de alta frecuencia sin distorsionar la señal en la banda de

paso, contribuye a mejorar la relación señal/ruido al restringir la señal a bandas de interés específicas (0.5–50 Hz) relevantes para la actividad motora.

Complementariamente, la *DWT*, como se explicó en el apartado 2.3 « Extracción de características», permite una descomposición multirresolución que aísla las bandas de frecuencia asociadas con la intención motora (e.g., bandas alfa y beta), facilitando la detección de patrones transitorios neuronales y destacando información crítica para los modelos de clasificación [150]. **El procesamiento llevado a cabo mejora la representación de las señales EEG y aporta un valor adicional en la predicción de la intención motora en sistemas BCI que utilizan *Machine Learning*, como puede observarse al no usar estos elementos**, figura 4.2.

No haber utilizado técnicas de reducción de dimensionalidad como *PCA* o *ICA*, pueden haber contribuido a este 73.3%, ya que **la ausencia de reducción dimensional podría interpretarse como una ventaja en este caso específico**, dado que el modelo logró aprovechar la complejidad de los datos originales sin perder información relevante. Los resultados utilizando estas técnicas de reducción de dimensionalidad (*ICA* y *PCA*) avalan esta afirmación al obtener una exactitud promedio casi aleatoria de un $50 \pm 6\%$.

5.1.2. Análisis independiente de sujetos S001-S010

A pesar de haber obtenido un resultado aceptable en promedio del 73,3%, **no se debe menospreciar el hecho de que los resultados de la figura 4.1 son una media de los mejores casos de cada sujeto por clasificador**.

Por ello, **el análisis debe realizarse observando la tabla 4.1**, en donde se recoge la clasificación para cada sujeto y cada uno de los 3 experimentos de estos, utilizando los diferentes clasificadores de *Machine Learning*.

De esta tabla se deduce:

- 1) **La clasificación** de un mismo sujeto **puede fluctuar ampliamente** de un experimento a otro (>65%).

- 2) **Existen sujetos con peor afinidad a este tipo de pruebas**, presentando valores casi aleatorios en sus clasificaciones, lastrando el resto de los resultados.
- 3) Clasificadores como el **SVM o el KNN** son capaces de generar **picos de exactitud muy altos** en ciertos sujetos, mientras que otros como *LDA* no son capaces de clasificar correctamente.
 - La causa de la primera deducción es la alta variabilidad en las señales EEG de cada individuo, la ratio señal-ruido de la señal cerebral y la reducida cantidad de datos que se utilizó.
 - La segunda deducción se explica en gran medida por las diferencias fisiológicas y de estado mental de los sujetos, tal como lo indican algunos estudios como el de Myrden, A *et al.* [151], en donde el estado del usuario que realiza la prueba condiciona el experimento, así como su capacidad para concentrarse. Por ejemplo, en la tabla 4.1, se observa que los sujetos S004 y S008, presentaron una precisión significativamente más baja. Esto posiblemente, es debido a la presencia de mayor ruido en sus señales o diferencias en su capacidad para generar señales EEG consistentes.
 - La deducción numero 3 es fácilmente observable al analizar las figuras 4.3 y 4.4. De estas puede observarse la variabilidad de las clasificaciones y los mejores desempeños de cada sujeto y cada clasificador, dando como consecuencia dicha deducción.

De lo anterior se extrae la importancia de los enfoques de personalización de modelos para cada sujeto, dado que las soluciones generalizadas pueden no ser adecuadas para todos los usuarios de sistemas *BCI*.

5.1.3. Parámetros estadísticos

En lo referido a la propuesta del apartado 3.2.2 «parámetros estadísticos», dónde **los parámetros estadísticos se aplican a los vectores de entrada utilizados en los clasificadores**. Los resultados han mostrado una **falta total de correlación**, siendo **totalmente aleatorios y añadiendo ruido** a los clasificadores. En consecuencia, **los resultados no se han mostrado por razones de simplicidad** en el trabajo.

Para explicar este hecho, **se ha realizado un estudio parámetro por parámetro, en cada uno de los canales del EEG, con el objetivo de ver la correlación** mostrada en los resultados.

Una de las observaciones clave es la falta de variación entre los eventos T1 y T2, lo que implica que **la adición o eliminación de estos parámetros estadísticos no afectó, o afectó de manera negativa a los resultados** de la clasificación.

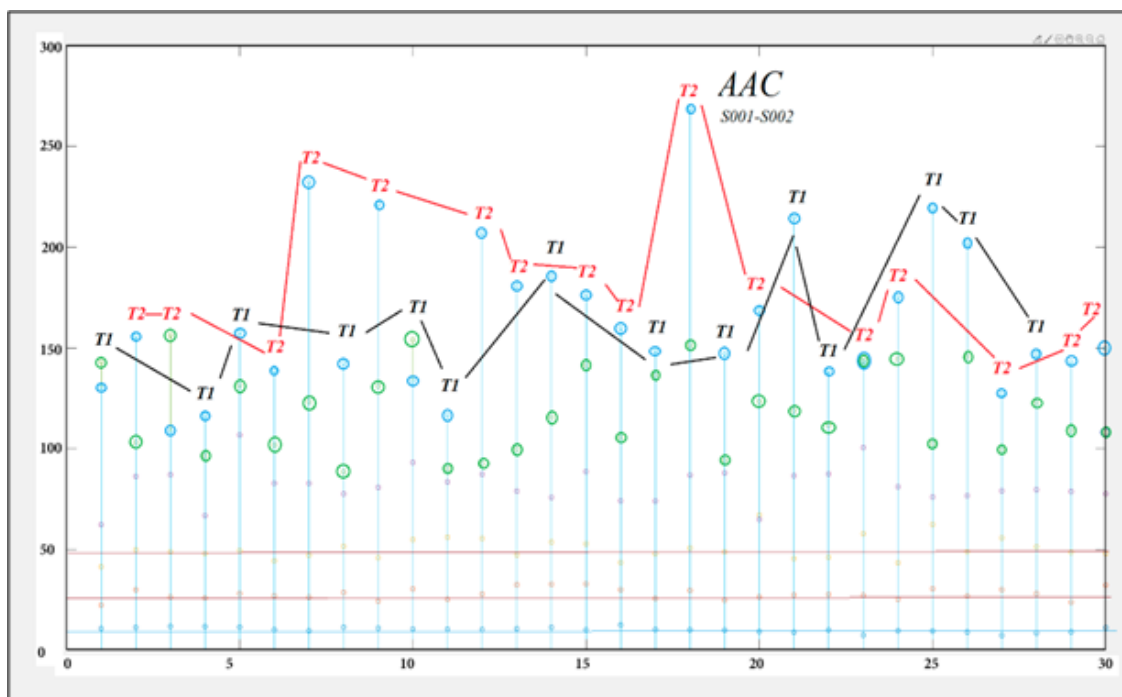


Figura 5.1. Ejemplo de la no correlación entre eventos T1 y T2 aplicando AAC en el canal FC1 de los sujetos S001 y S002.

En la Figura 5.1, se muestran los valores de cambio promedio de amplitud (AAC) para los eventos T1 y T2 en el canal FC1. Este gráfico revela que los valores de T1 y T2 son independientes y no presentan correlación, lo cual refuerza la hipótesis de que **los parámetros son una mala extracción de características o el uso de demasiados parámetros contribuye al ruido en los datos**. Este patrón de **falta de correlación** se observó de manera consistente en otros parámetros y canales, sin importar las unidades o el usuario, lo que sugiere que **los parámetros estadísticos aplicados no aportan una información** relevante para distinguir entre los dos eventos de intención motora.

Al eliminar los parámetros estadísticos y centrarse en el **tratamiento con DWT**, se **mejoró significativamente el rendimiento** de los clasificadores, como pudo verse en el apartado 5.1 y 5.2.

En este trabajo, **los resultados obtenidos** con los algoritmos de *Machine Learning* son notables, considerando que, en esta parte, **no se utilizaron algoritmos de Deep Learning ni se entrenó con un volumen tan grande de datos**. No se han encontrado estudios previos que hayan alcanzado una tasa de éxito tan alta utilizando la misma base de datos y los mismos usuarios con algoritmos de Machine Learning, lo que resalta la efectividad de la metodología planteada y concuerta con los «objetivos 3 y 4» de la tesis. Una de las razones del éxito, como se concluye en este trabajo, es la repetición de los experimentos tres veces por usuario, lo que permitió que los clasificadores aprendieran patrones más robustos a partir de los datos.

Se recomienda que **futuros estudios incluyan un periodo de entrenamiento para los sujetos, enfocado en la regulación de las ondas cerebrales, con el fin de optimizar el éxito de la clasificación**, como sugiere Roc A. *et al.* [63]. Este entrenamiento podría mejorar la consistencia de las señales EEG y, en consecuencia, la precisión de los modelos de clasificación.

5.2. *IPMS*. Sujetos S001-S010

Tras analizar los casos tanto de manera independiente, como de forma conjunta y aplicar procesamientos de diferentes naturalezas (*ICA*, *PCA*, *DWT* y parámetros) se procede a aplicar la hipótesis *IPMS* con el mejor de los casos (solo *DWT* y el filtro butterworth previo).

El uso de la técnica *IPMS* (en español, «estado mental inmediatamente previo») **mejoró el rendimiento del modelo** para la mayoría de los sujetos, como se observa en la tabla 4.2, en donde se muestran los resultados junto con la clasificación anterior en la que no se ha aplicado *IPMS*.

En la tabla 5.1, se extrae un resumen donde se muestra la clasificación media por sujeto en cada clasificador **con y sin *IPMS***. Esta tabla es fundamental para entender como **esta hipótesis mejora todos los resultados de media en los clasificadores** sin excepción. Mostrando **mejoras de hasta un 12,0%** en algunos casos como en el algoritmo *tree* y en otros un **mínimo de mejora del 0,7%** como al aplicarlo en el *SVM*. Además, también **reduce significativamente la desviación estándar**, lo que implica que se consigue estabilizar las clasificaciones, aumentando su precisión.

Esto se debe a que la técnica *IPMS* **permite mitigar el ruido en las señales EEG**, especialmente cuando las señales basales del sujeto antes de ejecutar una acción motora son erráticas.

Estos resultados indican que, si bien *IPMS* es una técnica poderosa, su efectividad depende en gran medida del perfil del sujeto y de la estabilidad de sus señales basales. Siendo evidente que **el ajuste del estado mental previo del sujeto es crucial para reducir artefactos y mejorar la consistencia de las señales EEG**.

De todo ello **se deduce que el estado previo a la realización de una tarea mental es significativo en el resultado final de la clasificación**.

No existen evidencias de estudios científicos que hayan tenido en cuenta el estado mental previo de esta forma. Por lo tanto, **se ha propuesto aquí un nuevo método con potencial para ser mejorado y lograr tasas de éxito mucho más altas**.

Fruto de lo anterior, **se han obtenido dos publicaciones en capítulos de libro, contribuyendo al estado del arte y al tejido investigador** [140], [152].

Tabla 5.1. Porcentajes de éxito medios en la clasificación con y sin *IPMS* de 10 sujetos

NO IPMS	Tree	LDA	Logistic	SVM	KNN	Ensamble
Average_S001	51,1	44,4	51,1	62,2	71,1	66,7
Average_S002	66,7	55,6	60,0	66,7	82,2	71,1
Average_S003	26,4	53,3	48,9	62,2	64,4	60,0
Average_S004	44,5	53,3	71,1	62,2	68,9	55,5
Average_S005	51,1	48,9	53,3	75,5	71,1	62,2
Average_S006	46,7	46,7	51,1	53,3	64,5	60,0
Average_S007	51,1	51,1	53,3	64,4	73,3	66,6
Average_S008	42,2	51,1	62,2	64,4	73,3	53,3
Average_S009	66,7	46,7	53,3	53,3	66,7	71,1
Average_S010	55,6	37,8	42,2	60,0	55,5	62,2
Media total (%)	51,1	48,6	53,8	62,2	68,6	63,9
STD (%)	11,2	5,0	7,6	6,1	6,7	5,7
IPMS	Tree	LDA	Logistic	SVM	KNN	Ensamble
Average_S001	55,5	44,4	68,9	66,7	64,4	71,1
Average_S002	48,9	62,2	60,0	64,4	68,9	62,2
Average_S003	73,3	53,4	51,1	62,2	71,1	75,5
Average_S004	80,0	55,6	51,1	57,8	66,7	75,5
Average_S005	57,8	60,0	55,5	66,7	68,9	57,8
Average_S006	64,4	62,2	66,7	71,1	68,9	71,1
Average_S007	57,8	40,0	55,6	55,5	73,4	68,9
Average_S008	68,9	57,8	66,7	68,9	77,8	73,3
Average_S009	64,4	46,7	48,9	60,0	68,9	62,2
Average_S010	60,0	42,2	53,3	55,5	71,1	60,0
Media total (%)	63,1	52,4	57,8	62,9	70,0	67,8
STD (%)	8,7	8,0	7,0	5,3	3,5	6,3
Éxito con IPMS	Tree	LDA	Logistic	SVM	KNN	Ensamble
Cambio en el éxito (%)	12,0	3,8	4,0	0,7	1,4	3,9
STD (%)	2,5	-3,0	0,6	0,8	3,2	-0,6

5.3. Clasificadores con *Deep Learning*

Se discuten a continuación los resultados del apartado 4.1.4 «Ensamblados de *Deep y Machine Learning*» en donde se ha hecho una clasificación utilizando diferentes clasificadores de *Deep Learning* con las condiciones ya descritas y el subgrupo de 10 usuarios. Posteriormente, se han diseñado modelos híbridos o ensamblados del mejor clasificador de *Deep Learning* en conjunto con los anteriormente vistos de *Machine Learning* para evaluar su desempeño en la clasificación de las señales del EEG.

En la **tabla 4.3**, se muestran los resultados de la clasificación de la intención motora utilizando los diferentes algoritmos de *Deep Learning* con el objetivo de **evaluar estos por separado** antes de realizar el ensamble. Los algoritmos utilizados incluyen ***LSTM*, *BiLSTM*, *GRU* y *Autoencoder***. Se añade el perceptrón por tener una referencia de una red neuronal simple. En este caso, los datos fueron procesados **utilizando *DWT* y la hipótesis del *IPMS***.

Se observa que:

- El algoritmo con **mejor rendimiento es el perceptrón**, con una capa oculta de 400 neuronas, alcanzando una exactitud del 63,33%. Sin embargo, otras **configuraciones como el *BiLSTM*, son de interés** pese a tener un rendimiento inferior (61,11%), ya que son algoritmos de *Deep Learning* especializados en las **series temporales** [152].
- El *GRU*, con una exactitud del 58,89%, muestra un rendimiento poco deseable, lo que concuerda con estudios previos que indican que ***GRU* tiende a ser menos efectivo en tareas de larga secuencia en comparación con *LSTM* y *BiLSTM***, especialmente en clasificaciones complejas de señales EEG [153].
- El ***Autoencoder* no parece aportar un valor significativo en este contexto**, alcanzando una exactitud del 54,44%. La combinación de un *autoencoder* con técnicas de extracción de características como *DWT* parece no ser suficiente para este tipo de clasificación, y estudios previos han sugerido que los *autoencoders* tienden a ser más efectivos en la reducción de dimensionalidad o reconstrucción de datos que en la clasificación de señales.

5.3.1. Modelos híbridos de *Deepy Machine Learning*

Como consecuencia de lo anterior, se construyen los ensambles con el clasificador *BiLSTM*. En la **tabla 4.3 se muestran los resultados obtenidos al combinar una capa *BiLSTM* con diferentes algoritmos de *Machine Learning*.**

De ella se puede concluir que:

- El mejor resultado se alcanza con la combinación de *BiLSTM* con *KNN*, *SVM* y Perceptrón, obteniendo una exactitud de 66,67%. Este valor es superior a los obtenidos en la Tabla 4.3, lo que refuerza la idea de que los ensambles de *Deep Learning* con *Machine Learning* mejoran el rendimiento global. El uso de algoritmos como el *SVM*, conocido por su capacidad para manejar problemas no lineales, junto con un modelo como *BiLSTM*, que puede captar dependencias temporales en los datos, parece haber contribuido a esta mejora.
- Al incrementar las iteraciones del ensamble de 5 a 10, la exactitud disminuye ligeramente, lo que puede sugerir que no siempre es beneficioso aumentar el número de iteraciones, posiblemente debido al sobreajuste o a la introducción de ruido.

La siguiente fase, como se explicó anteriormente, consiste en **analizar los ensambles utilizando la base de datos completa** (109), en la cual 105 sujetos son válidos y 4 presentan datos corruptos y por lo tanto una imposibilidad de ser añadidos en las clasificaciones. Los resultados se muestran en las tablas 4.5 y 4.6, que son discutidas a continuación:

En la **tabla 4.5, se analiza el rendimiento del ensamble utilizando los experimentos 1 y 2 de cada sujeto para entrenar y el experimento 3 para hacer el test ciego.**

- Los resultados muestran una variación de la **exactitud** en la clasificación con valor **mínimo de 69,33% y máximo de 75,33%.**
- El **mejor resultado** se obtiene con un ***BiLSTM* (50 neuronas ocultas) junto con una combinación de *KNN*, *SVM*, Perceptrón y *Tree***, lo que demuestra que la

combinación de estas técnicas es capaz de adaptarse de manera correcta a los datos de entrenamiento y testeo.

- Se observa que **el rendimiento no varía significativamente al aumentar el número de neuronas ocultas en el *BiLSTM* a 500 o 2000**, lo que indica que un mayor número de neuronas no mejora necesariamente la capacidad de generalización del modelo, sugiriendo que **el tamaño óptimo de las capas ocultas en *BiLSTM* se sitúa alrededor de las 50 neuronas**.

Por último, **en la tabla 4.6**, continuando con la línea de experimentación, **se analizan los resultados de la clasificación con ensambles utilizando diferentes proporciones de entrenamiento y testeo de los datos de movimiento real y de intención motora**.

- Se observa que la proporción 50:50 (real – intención) presenta una exactitud de hasta un 72,52% sin *BiLSTM*. Al aumentar la proporción de datos de entrenamiento a 67:33 la exactitud disminuye ligeramente hasta 72,06% (sin *BiLSTM*). Y, por último, al utilizar una proporción **83:17** (Experimentos reales y experimentos 1 y 2 de intención motora para entrenamiento y los experimentos 3 de intención motora para testear) **se obtiene una mayor exactitud (73,71%) pero sin *BiLSTM* también**. Esta correlación negativa puede deberse a diferencias estructurales o en las series temporales de los datos de movimientos reales utilizados para entrenar y que distorsionan el uso del *BiLSTM* que tiene memoria a largo plazo.

Se concluye lo siguiente:

Los resultados obtenidos en las tablas de la 4.3 a la 4.6 confirman que los ensambles que combinan técnicas de *Deep Learning*, como *BiLSTM*, con algoritmos de *Machine Learning*, como *KNN*, *SVM* y Perceptrón, logran una mejora significativa en la exactitud de la clasificación de señales de intención motora.

El uso de *BiLSTM* ha demostrado ser particularmente eficaz en estos ensambles, probablemente debido a su capacidad para modelar dependencias a largo plazo en los datos, lo que es crucial en la clasificación de señales EEG.

Sin embargo, es evidente que el aumento excesivo del número de neuronas o iteraciones no siempre conduce a mejores resultados, lo que sugiere que es crucial encontrar un equilibrio adecuado en la configuración de los hiperparámetros.

Así pues, **combinar datos reales y de intención motora para el entrenamiento y test parece no ser la estrategia óptima a la hora de configurar estos sistemas.**

5.4. Clasificaciones de *Deep Learning*. Matrices 2D

Con el propósito de seguir ahondando en los objetivos de esta tesis doctoral, y profundizar en el estudio de los *BCI* y sus algoritmos de clasificación de la señal cerebral se toma la decisión de avanzar del *Machine Learning* tradicional al *Deep Learning* exclusivamente.

En este trabajo el cambio se justifica por varias razones que están intrínsecamente relacionadas con las limitaciones de los métodos convencionales y las ventajas que ofrecen las redes neuronales profundas:

1) Capacidad de Representación Automática

Los algoritmos tradicionales de *Machine Learning*, como los árboles de decisión (*tree*), *SVM*, *KNN*, entre otros, dependen en gran medida de la extracción manual de características relevantes. En el caso de señales EEG, la extracción de características como la *DWT* o el *IPMS* han sido un paso crucial en este trabajo, pero no siempre es suficiente para capturar la complejidad subyacente de los datos. Con el tratamiento **utilizando *Deep Learning* se tiene la capacidad de aprender automáticamente las características relevantes a partir de los datos sin necesidad de un preprocesamiento intensivo.** Esto es especialmente útil cuando se trabaja con señales complejas como el EEG, donde las características óptimas pueden no ser fácilmente identificables *a priori*.

2) Manejo de Datos No Lineales y Complejos

Los métodos de *Machine Learning* tradicionales, como *LDA*, la regresión logística o *SVM* con *kernel* lineal, pueden tener dificultades para capturar relaciones complejas y no lineales en los datos. Algo que no sucede con el *Deep Learning*, que está diseñado para lidiar con estas complejidades debido a su estructura multicapa, que **permite**

modelar relaciones no lineales complejas. Algo que es crucial en el procesamiento de señales biológicas como el EEG.

3) Escalabilidad con Datos Masivos

A medida que aumenta la cantidad de datos disponibles, los métodos de *Machine Learning* tradicionales a menudo no escalan bien, mientras que **las redes neuronales profundas tienen la capacidad de manejar grandes volúmenes de datos**, mejorando su rendimiento con más datos. En el contexto de las señales EEG, esto es particularmente relevante ya que el uso de técnicas como *Deep Learning* puede aprovechar eficientemente datos masivos para entrenar modelos más robustos.

4) Mejor Rendimiento en Tareas Complejas

En tareas más avanzadas de clasificación y predicción, **las redes profundas han demostrado un rendimiento superior frente a los algoritmos tradicionales.** Al contar con un mayor número de parámetros, tienen una capacidad más avanzada para generalizar en diferentes tipos de tareas y dominios y de enfrentar escenarios con mayor variabilidad y ruido.

Este cambio trata de abordar con mayor eficiencia los objetivos 4 y 5 de la tesis, basando el sistema propuesto en el estado del arte, con el objetivo de superar los trabajos existentes y publicar los resultados obtenidos en revistas indexadas.

Por ello, se procede a discutir la nueva metodología de matrices 2D explicada en el apartado 3.2.5 «Cambio de paradigma. Matrices 2D» y los resultados de mayor relevancia obtenidos en la sección 4.2.1 «Matrices 2D. Pruebas con CNN y LSTM»

En este apartado, **se evaluó el rendimiento de un modelo híbrido CNN-LSTM aplicado a señales EEG, organizadas en matrices bidimensionales (2D)**, en la clasificación de intenciones motoras.

Las capas *BiLSTM* tuvieron un desempeño ligeramente inferior a las *LSTM* y por ello se realiza una simplificación, mostrando solamente los resultados de la combinación con las capas *LSTM*.

Los resultados indican que la arquitectura **CNN-LSTM** alcanza una precisión considerable incluso sin optimización previa, destacando su potencial para superar los ensambles anteriormente vistos.

A continuación, se discuten en detalle las figuras 4.5 y 4.6, así como las tablas 4.7 y 4.8, que ilustran los resultados alcanzados en estas pruebas.

Rendimiento inicial de clasificación de *CNN-LSTM*

La figura 4.5 presenta la matriz de confusión generada por la primera clasificación realizada con la arquitectura **CNN-LSTM**, alcanzando un rendimiento cercano al 75% de exactitud sin optimización. Este valor supera los alcanzados por los ensambles, que lograban aproximadamente un rango de entre un 65% y 73% de exactitud. Confirmando lo expuesto anteriormente.

La capacidad del modelo *CNN-LSTM* para lograr una precisión superior desde el inicio refleja su capacidad para capturar la naturaleza espaciotemporal de las señales EEG, aspecto crucial para clasificaciones que dependen de patrones en secuencias temporales.

Como se comentó esto es debido a que la arquitectura **CNN** permite extraer características espaciales de las señales EEG al identificar relaciones entre los distintos canales de registro. Mientras que las **LSTM** se encargan de capturar patrones de secuencia, lo que es esencial en datos temporales como los de EEG.

Este enfoque ha mostrado en estudios recientes [71] que, al combinar capas convolucionales y recurrentes, el modelo logra captar tanto la estructura espacial de los datos como su dependencia temporal, una combinación particularmente eficaz en la clasificación de señales cerebrales complejas.

Evaluación de diferentes configuraciones en la arquitectura *CNN-LSTM*

La tabla 4.7, muestra cómo la exactitud varía al modificar parámetros y condiciones en la arquitectura *CNN-LSTM*.

Los resultados sugieren que ciertas configuraciones y ajustes específicos en las capas *CNN*, tales como el uso de capas de normalización y *pooling*, mejoran significativamente

la precisión. Por ejemplo, ciertas configuraciones que combinan "*batch normalization*", activación "*ReLU*", y "*max pooling*" elevan el éxito en la clasificación hasta un 76,00%.

Un ajuste clave se observa en el tamaño del filtro de las capas convolucionales, donde tamaños de filtro menores permiten capturar detalles importantes en la señal, resultando en una exactitud del 79,60%. Esto es coherente con estudios previos [154], donde se ha demostrado que los filtros más pequeños en CNN son beneficiosos para preservar detalles finos en señales EEG.

Además, se probó el uso de diferentes estrategias de manejo de bordes en el padding, utilizando los modos "*replicate*" y "*symmetric-include-edge*". **Estos ajustes incrementaron el éxito en la clasificación hasta un 80,20%, sugiriendo que el manejo adecuado de los bordes de las convoluciones evita la pérdida de información en los límites de las secuencias, optimizando la captación de patrones espaciotemporales en las señales.**

Métricas de desempeño y análisis de iteraciones

Con el objetivo de analizar el desempeño del modelo de manera eficiente se realizaron 10 clasificaciones. Las métricas de evaluación después de realizar estas 10 iteraciones utilizando el método k-fold son presentadas en la tabla 4.8.

Los resultados destacan una **exactitud media de 80,6% ± 1,2% y una especificidad de 80,7% ± 1,9%**. Estos valores reflejan una **alta estabilidad en el modelo**, siendo capaces de identificar las clases objetivo con un nivel de error relativamente bajo.

Además, **el coeficiente *kappa* y el índice de Youden alcanzan valores elevados, lo que sugiere una gran consistencia en la clasificación y una capacidad robusta de discriminación entre clases.** La alta especificidad es particularmente relevante en aplicaciones de EEG, donde la capacidad del modelo para diferenciar eventos cerebrales específicos es fundamental. Esta consistencia ha sido observada en investigaciones similares, en las que se emplean arquitecturas híbridas *CNN-LSTM* para la clasificación de señales EEG, debido a la estabilidad que proporcionan frente a la variabilidad en las señales fisiológicas de los sujetos.

La **figura 4.7** ilustra gráficamente la exactitud promedio alcanzada en cada una de las iteraciones k-fold, con una media total del **80,60%** señalada en rojo. **Este rendimiento refleja la consistencia del modelo CNN-LSTM en clasificar correctamente las señales EEG a través de iteraciones estables y sugiere que el modelo tiene una fuerte capacidad de generalización, minimizando los riesgos de sobreajuste.** Este comportamiento es esencial en la clasificación de EEG, donde la variabilidad en las señales entre experimentos y sujetos puede afectar significativamente el rendimiento del modelo.

De todo lo anterior se concluye que los resultados obtenidos en este apartado confirman la efectividad de la arquitectura *CNN-LSTM* en la clasificación de señales EEG de intención motora, superando a los ensamblajes tradicionales de *Machine Learning* en precisión y consistencia.

Por lo tanto, **el modelo CNN-LSTM se posiciona como una arquitectura robusta y precisa para la clasificación de señales EEG de intención motora**, con resultados que sugieren su viabilidad para aplicaciones avanzadas de *BCI*'s. La arquitectura híbrida **no solo mejora la precisión, sino que también proporciona una estabilidad** que puede resultar crítica en escenarios de alta variabilidad fisiológica entre usuarios.

Elaboración de 2 artículos científicos en revistas Q1 (indexadas en JCR)

Fruto de la experimentación anterior **han sido elaborados 2 artículos científicos** del campo de los *BCI*'s, dando por cumplido el objetivo 5 de la tesis «Aportar al tejido mediante publicaciones en revistas indexadas». Además de estos artículos, se han conseguido publicar 3 artículos más en otras revistas Q1 indexadas en JCR , de esto se habla en secciones posteriores. **Los dos artículos relacionados con las BCI son los siguientes:**

- **El primero artículo ha sido publicado en la revista indexada IRBM [19], (Q1 en JCR)**
En este se desarrolla una herramienta que utiliza el algoritmo *CNN-LSTM* para diferenciar si un sujeto es apto o no apto a la hora de generar ondas cerebrales válidas en experimentos de intención motora. Esto implica que, tras eliminar a los

sujetos no aptos, el porcentaje de éxito de la clasificación general llega a superar valores del 81,00% con relativa facilidad. En el **ANEXO III se amplía información** por si fuera de interés.

- **El segundo artículo** (completo en el ANEXO IV), se encuentra **bajo revisión** en la revista indexada en JCR «*Scientific Reports*» (**Q1**), y cuyo título es «*A Hybrid Deep Learning Approach with 3D Temporal Data for User-Independent Classification of Imagined Motor Tasks in Brain-Computer Interfaces*». En él, se presentan los resultados de la tesis, correspondientes al apartado 4.2.1 «*Matrices 2D. Pruebas con CNN y LSTM*».

Si se hace una comparativa con el estado del arte, en la tabla 5.2, se observa que, en este segundo trabajo, al igual que en la tesis, utilizando la base de datos *Physionet* y un conjunto amplio de 109 usuarios (105), se logra una precisión de $80,60\% \pm 1,20\%$, superando significativamente a la mayoría de los estudios previos que emplearon otras bases de datos, como la *BCI Competition* y a los que usaron *Physionet*.

Tabla 5.2. Comparativa con otros trabajos del estado del arte que buscan la clasificación binaria de intenciones motoras utilizando EEG y un entrenamiento global generalizado.

Estudios	Conjunto de Datos	Usuarios	Exactitud
Ko et al. [48]	<i>BCI competition IV-2a</i>	9	45,00%
Amira et al. [49]	<i>BCI competition IV-2a</i>	9	58,42%
Murcide et al. [50]	<i>BCI competition IV-2a</i>	9	61,86%
Sakhavi et al. [51]	<i>BCI competition IV-2a</i>	9	70,60%
Kim et al. [52]	<i>BCI competition III-4a</i>	9	74,30%
Sakhavi et al. [53]	<i>BCI competition III-4a & IV-2a</i>	3 & 9	74,40%
Jiaqi et al. [54]	<i>BCI competition III-a, IV-2a</i>	3 & 9	77,70%
Jiao et al. [55]	<i>BCI competition IV-2b</i>	9	$78,00 \pm 2,30\%$
Lee and Choi [56]	<i>BCI competition IV-2b</i>	9	77,30% - 78,90%
Kim et al. [58]	<i>Physionet</i>	109	80,00%
Dose et al. [59]	<i>Physionet</i>	109	Global: $80,30 \pm 12,50\%$
Tesis/artículo Anexo IV	<i>Physionet</i>	109	$80,60 \pm 1,20\%$

Es relevante destacar, como se expresó en la tabla 1.2 del apartado 1.3.1 «Bases de datos famosas» que más del 75% de los estudios comparados utilizaron esta base de datos *BCI*, que es considerablemente más limitada en cuanto a la cantidad de usuarios y diversidad de señales.

La elección de la base de datos *Physionet* en este trabajo ofrece una clara ventaja, ya que se trata de un **conjunto de datos más amplio y diverso**, lo que permite entrenar modelos con una mayor variedad de señales EEG, lo cual es crucial para obtener **resultados más robustos y representativos**. Este enfoque está justificado en el apartado 1.5 «Novedad de la propuesta y contribuciones», donde se explica que la amplitud de la base de datos **permite evitar posibles sesgos** relacionados con la homogeneidad de los sujetos en estudios anteriores. Al incluir una muestra mucho mayor de usuarios, **se logra una mayor representatividad de las variaciones interpersonales que pueden presentarse en las señales EEG**.

Además, al utilizar un **modelo que busca ser independiente del usuario**, este trabajo ha desarrollado una **arquitectura capaz de generalizar correctamente** sin la necesidad de personalizar los modelos para cada sujeto. Esto contrasta con muchos de los estudios previos que entrenaron y validaron sus modelos basándose en conjuntos de usuarios limitados (generalmente entre 9 y 10), lo que puede inducir sesgos en los resultados, ya que estos modelos tienden a adaptarse a las características específicas de un pequeño número de individuos, comprometiendo su capacidad de generalización.

5.5. *Deep Learning Python*

Tras todos estos experimentos y simulaciones utilizando MATLAB, se decide dar el salto a Python como se justificó en el apartado 3.3 «Migración a Python».

Esta migración a **Python** abrió la puerta a realizar un análisis más detallado y exhaustivo de los modelos. Permitió entrar en detalles de la configuración de los algoritmos, así como también hizo posible la evaluación de capas de atención (**Attention Layer**) y la implementación de arquitecturas avanzadas como **transformers** y **Vision Transformers (ViT)**.

Tras analizar los resultados obtenidos en el apartado 4.2.2. «Migración a Python. Replica de pruebas» se observa en la tabla 4.9 que:

5.5.1. Réplica del algoritmo de creación de matrices 2D.

Tras **replicar el algoritmo** de MATLAB se obtiene el código de la arquitectura mostrada en la **figura 3.12. El resultado de la clasificación** de la señal de la intención motora con todos los sujetos de la base de datos en la réplica del algoritmo es del **76,00% utilizando 3 capas CNN y 2 LSTM.**

Esta replica no cumplió con las expectativas, seguramente por algún desajuste en la parametrización sumado a la inicialización de pesos y que probablemente en MATLAB existe algún elemento se ajusta automáticamente mientras que en Python no, al ser menos automático.

Por ello, se procedió a **analizar diferentes combinaciones** para tratar de mejorar el algoritmo. Entre las que se incluyeron capas de atención.

En la **tabla 4.9**, se presenta un **resumen** de las **diferentes arquitecturas** utilizadas y sus respectivas tasas de éxito tras la migración a Python. Por otro lado, las figuras de la 4.8 a la 4.15, complementan estos resultados mostrando las matrices de confusión que permiten visualizar el rendimiento de cada modelo, ayudando a identificar posibles áreas de mejora en la clasificación de las señales EEG.

En general, **los resultados reflejan una mejora significativa en términos de exactitud** debido a la mayor flexibilidad y capacidad de personalización de Python. Esto es particularmente notable en configuraciones como «*CNN-maxpooling-CNN-attention-LSTM*», donde el uso de técnicas avanzadas como "*maxpooling*" y las "*attention layers*" permitió optimizar la extracción de características y mejorar la robustez del modelo. El uso de una mayor cantidad de "*epoch*" y ajustes en el tamaño del "*batch*" contribuyó a incrementar la precisión de los modelos, como se observa en el modelo *CNN* con maxpooling y capa de atención entrenado durante 150 *epochs* con un "*batch size*" ajustado, alcanzando el **85,19% de precisión.**

De estos experimentos se concluye:

La arquitectura híbrida CNN-LSTM con capas de atención tiene potencial para capturar patrones tanto espaciales como secuenciales en los datos EEG, lo cual se alinea con estudios recientes que demuestran la efectividad de las capas de atención en la mejora de la precisión de clasificación en señales EEG.

En la tabla 5.3 se recoge una comparativa entre este resultado de la tesis utilizando las capas de atención (85,19%) con resultados de naturaleza similar en el estado del arte.

En estos trabajos mostrados, se han utilizado mecanismos de atención para la clasificación de señales cerebrales asociadas a la intención motora. El futuro objetivo es desarrollar un artículo científico utilizando los resultados obtenidos ya que tienen buen desempeño en comparación, no obstante, este se encuentra en desarrollo debido a que los resultados han sido recientes.

Tabla 5.3. Comparativa con otros trabajos del estado del arte que buscan la clasificación binaria de intenciones motoras utilizando EEG y capas de atención.

Estudios	Base de datos	Arquitectura	Exactitud
Li et al. [155]	BCI comp. IV-2a/2b	Dual attention based adversarial network	76,16% & 84,83%
Chen et al.[156]	BCI comp. IV-2a/2b	Multiattention CNN	81,48% & 82,54%
Kwak et al. [157]	Open Access IEEE [158]	Functional near-infrared spectroscopy guided attention network	78,59%
Sartipi et al. [159]	BCI comp. IV-2a &Physionet	Subject-independent semi-supervised deep architecture	61,00% & 83,00%
Dong Mei et al. [160]	Physionet	Modular temporal-spatial attention-based CNN	72,05%
Zhang et al. [161]	Physionet	LSTM - Attention	83,20%
Zhang et al. [162]	Physionet	Graph-based Hierarchical Attention Model	76,36%
La presente tesis	Physionet	CNN's-Attention-LSTM's	85,19 %

5.5.2. Nuevos modelos. *Transformers* y *ViT*

Por último, en la subsección dedicada al ***Vision Transformer*** se revela un **enfoque innovador** para el análisis de señales EEG. Esto se consigue al tratar los datos como imágenes divididas en parches, permitiendo que el modelo capture relaciones globales entre estos parches. Este método es especialmente **útil en tareas de clasificación complejas**, donde la correlación entre eventos espaciales juega un rol crucial.

La **figura 4.16** muestra el rendimiento del ***ViT* con una capa *LSTM* añadida (77,00%)**, mientras que la **figura 4.17** muestra el **rendimiento sin esta capa (79,60%)**. La inclusión de *LSTM* no mejora la capacidad del modelo para capturar dependencias temporales, lo cual es crítico en la clasificación de EEG donde las señales tienen una naturaleza secuencial. Por ello, queda abierta una ventana para indagar el porqué de este hecho.

Aunque el *ViT* sin *LSTM* ofrece buenos resultados, la pérdida de precisión global indica que **el modelo aún podría beneficiarse de un preprocesamiento más exhaustivo o de una mayor cantidad de datos de entrenamiento**. Estos resultados están alineados con estudios previos que resaltan cómo el *ViT* puede superar a los *CNN* tradicionales en tareas de clasificación de imágenes, y sugieren que, con una optimización adicional, el *ViT* podría ofrecer mejoras significativas en la clasificación de EEG [163].

6. Conclusions and future works

6.1. Conclusions

This research confirms and validates the central hypothesis proposed here: *"The information extracted from the brain signal via EEG contains valuable information related to motor intention. Correct analysis and processing of this data allows for the classification of thoughts and enables motor action based on an intention."*

This validation is based on the development of a precise and robust classification system for motor intentions, utilizing advanced EEG signal processing techniques and Machine and Deep Learning techniques. EEG signals, when are properly preprocessed and decomposed into their relevant components, reveal neuronal patterns that effectively identify motor intention, underscoring the importance of this information for BCI applications.

The success of classifying brain signals and motor intention has been gradually improved throughout the course of this research. This achievement has been made possible through a broad exploration of techniques, adjusted through different trials of various methods and architectures. This study has been enriched with diverse preprocessing methods and simple, complex, and hybrid Machine and Deep Learning techniques and architectures.

Regarding the stated objectives, the research has fully met them:

- 1) Study of the field of brain signals and Brain-Computer Interfaces (BCI):** An in-depth understanding of EEG signals, their nature, acquisition, and processing was achieved, which was fundamental in comprehending the type of signal to be processed and, in turn, designing an effective classification system.
- 2) Deepen knowledge of Machine and Deep Learning algorithms applied to this paradigm:** Various advanced architectures were explored and adapted, including Machine Learning models, hybrid models, and Deep Learning models like CNN-LSTM with attention layers or Vision Transformers, to improve accuracy in

classifying motor intentions. This objective enabled the design of the backbone of the experiments to validate the hypothesis.

- 3) Analysis of the state of the art in relation to motor intention and its proper classification:** An exhaustive review of previous studies in BCI was conducted, identifying areas for improvement and defining an innovative approach that combines signal processing and hybrid architectures for classification. This objective provided a starting point for the thesis and a final goal that has been achieved.
- 4) Propose a system that surpasses the state of the art:** The developed system with CNN-LSTM achieves an accuracy of 80.2%, improving the results of previous models in BCI and receiving validation from the scientific community. Furthermore, the combination of CNN-LSTM and attention layers for EEG analysis represents a significant advancement in motor intention classification, achieving a success rate above 85%, with a generalized approach and robustness intended to be published in a forthcoming article. This objective has therefore been largely achieved.
- 5) Contribute to the research community through publications in indexed journals:** This work has generated multiple publications in Q1 and Q2 journals, as well as book chapters, highlighting the acceptance of the results by the scientific community and underscoring their relevance in the BCI field. These are shown in Table 6.1 and fully listed in the Annexes. **Articles published in Q1 journals are marked in green, those in Q2 in blue, book chapters in yellow, and articles under review in orange.**

In summary, this thesis not only fulfills its objectives but also establishes an innovative and solid framework for advancing motor intention classification through deep learning and signal processing techniques, providing a precise, robust, and validated tool for future BCI system applications.

Table 6.1. Publications in the development of the doctoral thesis.

Title	Journal/ Chapter	Journal Impact Factor (2023)	Field	Status
Study of an Optimization Tool Avoided Bias for Brain-Computer Interfaces Using a Hybrid Deep Learning Model. [19]	IRBM	5,6 (Q1 «Engineering, Biomedical» 22/123)	BCI	Published
ANN based-model for estimating the boron permeability coefficient as boric acid in SWRO desalination plants using ensemble-based machine learning. [164]	Desalination	8,4 (Q1 «Water Resources» 3/128; Q1 «Engineering, Chemical» 13/170)	Time series in desalination	Published
Novel cost-effective method for forecasting COVID-19 and hospital occupancy using deep learning. [165]	Scientific Reports	3,8 (Q1 «Multidisciplinary Science» 25/134)	Time series in COVID	Published
Sky Image Classification Based on Transfer Learning Approaches [166]	Sensors	3,4 (Q2 en «Chemistry, Analytical» 34/106 y Q2 en «Engineering, Electrical & Electronic» 122/353)	Image classification	Published
Analysis of Brain Computer Interface Using Deep and Machine Learning [140]	IntechOpen (chapter)	-	BCI	Published
Analysis of Brain Signals to Forecast Motor Intentions Using Artificial Intelligence [152]	Sustainable Computing (chapter Springer)	-	BCI	Published
A Hybrid Deep Learning Approach with 3D Temporal Data for User-Independent Classification of Imagined Motor Tasks in Brain-Computer Interfaces	Scientific Reports	3,8 (Q1 «Multidisciplinary Science» 25/134)	BCI	Under Review
Emotions for Everyone: A Low-Cost, High-Accuracy Method for Emotion Classification	Scientific Reports	3,8 (Q1 «Multidisciplinary Science» 25/134)	Emotion classification	Under Review
Angular contour parametrization for the in-air signature-based identification	IEEE Transactions on Information Forensics and Security	6,3 (Q1 «Computer Science, Theory & Methods» 13/144; Q1 «Engineering, Electrical & Electronic» 42/353)	Pattern Recognition	Under Review

6.2. Future research directions

Brain-computer interfaces (BCI) present significant potential across various fields. Deep Learning algorithms are becoming increasingly advanced, and the hardware required to accompany this revolution is receiving more attention.

Some of the future research lines in this field include:

- Improving and implementing these algorithms to enable the control of external devices via brain signals. This field opens up various future research directions, both at the algorithmic level and in practical applications.
 - From an algorithmic perspective, a promising direction is the development of models based on hybrid transformers, self-attention algorithms, or even the use of generative AI, which could learn autonomously from the diverse readings gathered from the data.
 - The implementation of Vision Transformers (ViT) and similar models already utilized could be enhanced through data optimization techniques and more detailed preprocessing, making real-time analysis of EEG signals possible with greater accuracy and speed. This advancement could improve the quality of life for people with disabilities, neurodegenerative issues, or even find applications in rehabilitation, gaming, and education.
- Creating adaptive models that adjust their performance based on each individual's brain activity, addressing the physiological variability observed in EEG responses among users. This involves developing personalized BCI systems, which could enhance the effectiveness and accessibility of these technologies for individuals with specific needs. These personalized models would offer greater precision and reduce variability in classification results, proving particularly relevant for neurorehabilitation applications and prosthetic control.
- Exploring macroscopic applications of BCI beyond device control, such as treating neurological disorders, enhancing cognitive skills through neurofeedback, or providing advanced interaction in virtual reality applications.

- **Focusing on the security and ethical handling of brain data** in future research, ensuring privacy and the responsible use of these technologies in commercial and medical applications. BCI has the potential to be a foundational tool in neurotechnology, and its development could revolutionize both the clinical field and human-computer interaction in society.

7. Bibliografía

- [1] Latarjet, M., & Liard, A. R. (2004). *Anatomía humana* (Vol. 2). Ed. Médica Panamericana.
- [2] *CogniFit*. (s. f.). Sobre el Cerebro Humano - CogniFit. Recuperado 10 de junio de 2024, de <https://www.cognifit.com/es/cerebro>
- [3] Derrickson, T. (2006). Principios de anatomía y fisiología. *Editorial Médica Panamericana*
- [4] Department of Biochemistry and Molecular Biophysics Thomas Jessell, Siegelbaum, S., & Hudspeth, A. J. (2000). *Principles of neural science* (Vol. 4, pp. 1227-1246). E. R. Kandel, J. H. Schwartz, & T. M. Jessell (Eds.). New York: McGraw-hill.
- [5] Miller, E. K., & Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1), 167-202.
- [6] Bear, M. F., Connors, B. W., & Paradiso, M. A. (1998). Estructura del sistema nervioso. *Neurociencia explorando el cerebro*. Barcelona: Masson.
- [7] The Wild Project. (2024, 28 marzo). *The Wild Project #275 - Jesús Martín-Fdez | Superar la muerte gracias a la neurociencia, Neuralink* [Vídeo]. YouTube. <https://www.youtube.com/watch?v=oQYXSjQirk>
- [8] Wolf, U., Rapoport, M. J., & Schweizer, T. A. (2009). Evaluating the affective component of the cerebellar cognitive affective syndrome. *The Journal of neuropsychiatry and clinical neurosciences*, 21(3), 245-253.
- [9] Jeannerod, M. (2006). *Motor cognition: What actions tell the self* (Vol. 42). OUP Oxford.
- [10] Lotze, M., & Halsband, U. (2006). Motor imagery. *Journal of Physiology-paris*, 99(4-6), 386-395.
- [11] Héту, S., Grégoire, M., Saimpont, A., Coll, M. P., Eugène, F., Michon, P. E., & Jackson, P. L. (2013). The neural network of motor imagery: an ALE meta-analysis. *Neuroscience & Biobehavioral Reviews*, 37(5), 930-949.
- [12] Hammond, C. (2001). *Cellular and Molecular Neurobiology (Deluxe Edition)*. Academic Press.
- [13] Cooper, D. C. (2011). *Introduction to Neuroscience*. Donald C. Cooper Ph. D.
- [14] Levitan, I. B., & Kaczmarek, L. K. (2015). *The neuron: cell and molecular biology*. Oxford University Press, USA.

- [15] Paniagua, R., Nistal, M., Sesma, P., Álvarez, U. M., Fraile, B., Anadón, R., ... & De Miguel, M. P. (1998). *Citología e histología vegetal y animal*. McGraw-Hill Interamericana.
- [16] Moore, K. L., & Dalley, A. F. (2007). *Anatomía con orientación clínica*. Ed. Médica Panamericana.
- [17] Abhang, P. A., Gawali, B. W., & Mehrotra, S. C. (2016). Technological basics of EEG recording and operation of apparatus. *Introduction to EEG-and speech-based emotion recognition*, 19-50.
- [18] Buzsáki, G. (2006). *Rhythms of the Brain*. Oxford university press.
- [19] Ajali-Hernández, N. I., Travieso-González, C. M., Bermudo-Mora, N., Reino-Cacho, P., & Rodríguez-Saucedo, S. (2024). Study of an Optimization Tool Avoided Bias for Brain-Computer Interfaces Using a Hybrid *Deep Learning* Model. *IRBM*, 45(3), 100836.
- [20] Krucoff, M. O., Rahimpour, S., Slutzky, M. W., Edgerton, V. R., & Turner, D. A. (2016). Enhancing nervous system recovery through neurobiologics, neural interface training, and neurorehabilitation. *Frontiers in neuroscience*, 10, 584.
- [21] Noguera, M. A. P., Ortega, C. E. M., Castro, W., & peluffo Ordoñez, D. H. Análisis De Señales EEG Para Detección De Intenciones Motoras Aplicadas A Sistemas BCI.
- [22] Martínez Albaladejo, F. J. (2018). Evaluación experimental de aprendizaje máquina extremo aplicado a los sistemas de interfaz cerebro-ordenador basados en imaginación de movimiento.
- [23] De la Torre Abaitua, J. (2012). Procesado de senales eeg para un interfaz cerebro-máquina (bci). *Departamento de Teoría de la Señal y Comunicaciones. Universidad Carlos III de Madrid*.
- [24] Wolpaw, J. R. (2013). Brain-computer interfaces. In *Handbook of clinical neurology* (Vol. 110, pp. 67-74). Elsevier.
- [25] Nicolas-Alonso, L. F., & Gomez-Gil, J. (2012). Brain computer interfaces, a review. *sensors*, 12(2), 1211-1279.
- [26] Zhang, J., Li, J., Huang, Z., Huang, D., Yu, H., & Li, Z. (2023). Recent progress in wearable brain-computer interface (BCI) devices based on electroencephalogram (EEG) for medical applications: a review. *Health Data Science*, 3, 0096.
- [27] Värbu, K., Muhammad, N., & Muhammad, Y. (2022). Past, present, and future of EEG-based BCI applications. *Sensors*, 22(9), 3331.

-
- [28] Douibi, K., Le Bars, S., Lemontey, A., Nag, L., Balp, R., & Breda, G. (2021). Toward EEG-based BCI applications for industry 4.0: Challenges and possible applications. *Frontiers in Human Neuroscience*, *15*, 705064.
- [29] King, B. J., Read, G. J., & Salmon, P. M. (2024). The risks associated with the use of brain-computer interfaces: a systematic review. *International Journal of Human-Computer Interaction*, *40*(2), 131-148.
- [30] Schalk, G., McFarland, D. J., Hinterberger, T., Birbaumer, N., & Wolpaw, J. R. (2004). BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Transactions on biomedical engineering*, *51*(6), 1034-1043.
- [31] Blankertz, B., Müller, K. R., Krusienski, D. J., Schalk, G., Wolpaw, J. R., Schlogl, A., ... & Birbaumer, N. (2006). The BCI competition III: Validating alternative approaches to actual BCI problems. *IEEE transactions on neural systems and rehabilitation engineering*, *14*(2), 153-159.
- [32] Tangermann, M., Müller, K. R., Aertsen, A., Birbaumer, N., Braun, C., Brunner, C., ... & Blankertz, B. (2012). Review of the BCI competition IV. *Frontiers in neuroscience*, *6*, 55.
- [33] Brunner, C., Leeb, R., Müller-Putz, G., Schlögl, A., & Pfurtscheller, G. (2008). BCI Competition 2008–Graz data set A. *Institute for knowledge discovery (laboratory of brain-computer interfaces), Graz University of Technology*, *16*, 1-6.
- [34] Leeb, R., Lee, F., Keinrath, C., Scherer, R., Bischof, H., & Pfurtscheller, G. (2007). Brain-computer communication: motivation, aim, and impact of exploring a virtual apartment. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, *15*(4), 473-482.
- [35] Arpaia, P., Esposito, A., Natalizio, A., & Parvis, M. (2022). How to successfully classify EEG in motor imagery BCI: a metrological analysis of the state of the art. *Journal of Neural Engineering*, *19*(3), 031002.
- [36] Cho, H., Ahn, M., Ahn, S., Kwon, M., & Jun, S. C. (2017). EEG datasets for motor imagery brain-computer interface. *GigaScience*, *6*(7), gix034.
- [37] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggenberger, K., Tangermann, M., ... & Ball, T. (2017). Deep Learning with convolutional neural networks for EEG decoding and visualization. *Human brain mapping*, *38*(11), 5391-5420.
- [38] Nikolopoulos, S., Petrantonakis, P. C., Georgiadis, K., Kalaganis, F., Liaros, G., Lazarou, I., ... & Kompatsiaris, I. (2017). A multimodal dataset for authoring and editing multimedia content: the MAMEM project. *Data in brief*, *15*, 1048-1056.
-

- [39] Ofner, P., Schwarz, A., Pereira, J., & Müller-Putz, G. R. (2017). Upper limb movements can be decoded from the time-domain of low-frequency EEG. *PLoS one*, *12*(8), e0182578.
- [40] Steyrl, D., Scherer, R., Förstner, O., & Müller-Putz, G. R. (2014, September). Motor imagery brain-computer interfaces: random forests vs regularized LDA-non-linear beats linear. In *Proceedings of the 6th international brain-computer interface conference* (pp. 241-244).
- [41] Yu, X., Aziz, M. Z., Sadiq, M. T., Fan, Z., & Xiao, G. (2021). A new framework for automatic detection of motor and mental imagery EEG signals for robust BCI systems. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1-12.
- [42] Jaipriya, D., & Sriharipriya, K. C. (2024). Brain computer interface-based signal processing techniques for feature extraction and classification of motor imagery using EEG: A literature review. *Biomedical Materials & Devices*, *2*(2), 601-613.
- [43] Xu, G., Shen, X., Chen, S., Zong, Y., Zhang, C., Yue, H., ... & Che, W. (2019). A Deep transfer convolutional neural network framework for EEG signal classification. *IEEE Access*, *7*, 112767-112776.
- [44] Lu, N., Li, T., Ren, X., & Miao, H. (2016). A Deep Learning scheme for motor imagery classification based on restricted Boltzmann Machines. *IEEE transactions on neural systems and rehabilitation engineering*, *25*(6), 566-576.
- [45] Li, F., He, F., Wang, F., Zhang, D., Xia, Y., & Li, X. (2020). A novel simplified convolutional neural network classification algorithm of motor imagery EEG signals based on Deep Learning. *Applied Sciences*, *10*(5), 1605.
- [46] Tayeb, Z., Fedjaev, J., Ghaboosi, N., Richter, C., Everding, L., Qu, X., ... & Conradt, J. (2019). Validating Deep neural networks for online decoding of motor imagery movements from EEG signals. *Sensors*, *19*(1), 210.
- [47] Ha, K. W., & Jeong, J. W. (2019). Motor imagery EEG classification using capsule networks. *Sensors*, *19*(13), 2854.
- [48] Ko, W., Yoon, J., Kang, E., Jun, E., Choi, J. S., & Suk, H. I. (2018, January). Deep recurrent spatio-temporal neural network for motor imagery based BCI. In *2018 6th International Conference on Brain-Computer Interface (BCI)* (pp. 1-3). IEEE.
- [49] Ehtioui, A., Zouch, W., Ghorbel, M., Mhiri, C., & Hamam, H. (2024). Classification of BCI multiclass motor imagery task based on artificial neural network. *Clinical EEG and Neuroscience*, *55*(4), 455-464.

- [50] Degirmenci, M., Yuce, Y. K., Perc, M., & Isler, Y. (2023). Statistically significant features improve binary and multiple Motor Imagery task predictions from EEGs. *Frontiers in Human Neuroscience*, *17*, 1223307.
- [51] Sakhavi, S., Guan, C., & Yan, S. (2015, August). Parallel convolutional-linear neural network for motor imagery classification. In *2015 23rd European signal processing conference (EUSIPCO)* (pp. 2736-2740). IEEE.
- [52] Kim, Y. J., Kwak, N. S., & Lee, S. W. (2018, January). Classification of motor imagery for Ear-EEG based brain-computer interface. In *2018 6th International Conference on Brain-Computer Interface (BCI)* (pp. 1-2). IEEE.
- [53] Sakhavi, S., Guan, C., & Yan, S. (2018). Learning temporal information for brain-computer interface using convolutional neural networks. *IEEE transactions on neural networks and learning systems*, *29*(11), 5619-5629.
- [54] Wang, J., Chen, W., & Li, M. (2023). A multi-classification algorithm based on multi-domain information fusion for motor imagery BCI. *Biomedical Signal Processing and Control*, *79*, 104252.
- [55] Jiao, Y., Zhang, Y., Chen, X., Yin, E., Jin, J., Wang, X., & Cichocki, A. (2018). Sparse group representation model for motor imagery EEG classification. *IEEE journal of biomedical and health informatics*, *23*(2), 631-641.
- [56] Lee, H. K., & Choi, Y. S. (2018, January). A convolution neural networks scheme for classification of motor imagery EEG based on *wavelet* time-frequency image. In *2018 International Conference on Information Networking (ICOIN)* (pp. 906-909). IEEE.
- [57] Ai, Q., Chen, A., Chen, K., Liu, Q., Zhou, T., Xin, S., & Ji, Z. (2019). Feature extraction of four-class motor imagery EEG signals based on functional brain network. *Journal of neural engineering*, *16*(2), 026032.
- [58] Kim, Y., Ryu, J., Kim, K. K., Took, C. C., Mandic, D. P., & Park, C. (2016). Motor imagery classification using mu and beta rhythms of EEG with strong uncorrelating transform based complex common spatial patterns. *Computational intelligence and neuroscience*, *2016*(1), 1489692.
- [59] Dose, H., Møller, J. S., Iversen, H. K., & Puthusserypady, S. (2018). An end-to-end *Deep Learning* approach to MI-EEG signal classification for BCIs. *Expert Systems with Applications*, *114*, 532-542.
- [60] Luo, T. J., Zhou, C. L., & Chao, F. (2018). Exploring spatial-frequency-sequential relationships for motor imagery classification with recurrent neural network. *BMC bioinformatics*, *19*, 1-18.

- [61] Olivas-Padilla, B. E., & Chacon-Murguía, M. I. (2019). Classification of multiple motor imagery using *Deep* convolutional neural networks and spatial filters. *Applied Soft Computing*, *75*, 461-472.
- [62] Miao, M., Zhang, W., Hu, W., & Wang, R. (2020). An adaptive multi-domain feature joint optimization framework based on composite kernels and ant colony optimization for motor imagery EEG classification. *Biomedical Signal Processing and Control*, *61*, 101994.
- [63] Roc, A., Pillette, L., Mladenovic, J., Benaroch, C., N'Kaoua, B., Jeunet, C., & Lotte, F. (2021). A review of user training methods in brain computer interfaces based on mental tasks. *Journal of Neural Engineering*, *18*(1), 011002.
- [64] Lun, X., Yu, Z., Chen, T., Wang, F., & Hou, Y. (2020). A simplified *CNN* classification method for MI-EEG via the electrode pairs signals. *Frontiers in Human Neuroscience*, *14*, 338.
- [65] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., ... & Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *circulation*, *101*(23), e215-e220.
- [66] Ajali Hernández, N. I. (2019). *Análisis de la acción/intención motora mediante el estudio de las señales EEG* (Master's thesis).
- [67] Sleight, J., Pillai, P., & Mohan, S. (2009). Classification of executed and imagined motor movement EEG signals. *Ann Arbor: University of Michigan*, *110*, 54.
- [68] Deecke, L., Weinberg, H., & Brickett, P. (1982). Magnetic fields of the human brain accompanying voluntary movement: Bereitschaftsmagnetfeld. *Experimental brain research*, *48*, 144-148.
- [69] Daud, S. S., & Sudirman, R. (2015, February). Butterworth bandpass and stationary *wavelet* transform filter comparison for electroencephalography signal. In *2015 6th international conference on intelligent systems, modelling and simulation* (pp. 123-126). IEEE.
- [70] Elahi, M. A., Glavin, M., Jones, E., & O'Halloran, M. (2017). Adaptive artifact removal for selective multistatic microwave breast imaging signals. *Biomedical Signal Processing and Control*, *34*, 93-100.
- [71] Zhang, D., Yao, L., Chen, K., Wang, S., Chang, X., & Liu, Y. (2019). Making sense of spatio-temporal preserving representations for EEG-based human intention recognition. *IEEE transactions on cybernetics*, *50*(7), 3033-3044.

- [72] Mallat, S. G. (1989). A theory for multiresolution signal decomposition: the *wavelet* representation. *IEEE transactions on pattern analysis and Machine intelligence*, 11(7), 674-693.
- [73] Nieto, N., & Orozco, D. M. (2008). El uso de la transformada *wavelet* discreta en la reconstrucción de señales senosoidales. *Scientia et technica*, 14(38), 381-386.
- [74] Daubechies, I. (1992). CBMS-NSF regional conference series in applied mathematics. *Ten lectures on wavelets*, 61.
- [75] Inan, O. T., Giovangrandi, L., & Kovacs, G. T. (2006). Robust neural-network-based classification of premature ventricular contractions using *wavelet* transform and timing interval features. *IEEE transactions on Biomedical Engineering*, 53(12), 2507-2515.
- [76] Rai, H. M., Trivedi, A., & Shukla, S. (2013). ECG signal processing for abnormalities detection using multi-resolution *wavelet* transform and Artificial Neural Network classifier. *Measurement*, 46(9), 3238-3246.
- [77] Zhang, Z. (2023). The improvement of the discrete *wavelet* transform. *Mathematics*, 11(8), 1770.
- [78] Toledo-Pérez, D. C., Rodríguez-Reséndiz, J., Gómez-Loenzo, R. A., & Jauregui-Correa, J. C. (2019). Support vector *Machine*-based EMG signal classification techniques: A review. *Applied Sciences*, 9(20), 4402.
- [79] Krishnan, S., & Athavale, Y. (2018). Trends in biomedical signal feature extraction. *Biomedical Signal Processing and Control*, 43, 41-63.
- [80] Abbaspour, S., Lindén, M., Gholamhosseini, H., Naber, A., & Ortiz-Catalan, M. (2020). Evaluation of surface EMG-based recognition algorithms for decoding hand movements. *Medical & biological engineering & computing*, 58, 83-100.
- [81] Jolliffe, I. T., & Cadima, J. (2016). Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065), 20150202.
- [82] Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3), 37-52.
- [83] Hyvärinen, A., & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5), 411-430.
- [84] Calhoun, V. D., Liu, J., & Adalı, T. (2009). A review of group ICA for fMRI data and ICA for joint inference of imaging, genetic, and ERP data. *Neuroimage*, 45(1), S163-S172.

- [85] Cortes, C. (1995). Support-Vector Networks. *Machine Learning*.
- [86] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992, July). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144-152).
- [87] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1), 21-27.
- [88] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3), 175-185.
- [89] Sayad, S. (2010). An introduction to data science. *Toronto: University of Toronto*.
- [90] *Deep Dive on KNN: Understanding and Implementing the K-Nearest Neighbors Algorithm*. (2023, 9 junio). Arize AI. <https://arize.com/blog-course/knn-algorithm-k-nearest-neighbor/>
- [91] Safavian, S. R., & Landgrebe, D. (1991). A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3), 660-674.
- [92] Rokach, L., & Maimon, O. (2005). Decision trees. *Data mining and knowledge discovery handbook*, 165-192.
- [93] Strobl, C., Malley, J., & Tutz, G. (2009). An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4), 323.
- [94] Cox, D. R. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 20(2), 215-232.
- [95] Hosmer Jr, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). *Applied logistic regression*. John Wiley & Sons.
- [96] Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2), 179-188.
- [97] McLachlan, G. J. (2005). *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons.
- [98] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- [99] Zhang, H. (2004). The optimality of naive Bayes. *Aa*, 1(2), 3.

- [100] Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th international conference on Machine Learning (ICML-03)* (pp. 616-623).
- [101] McCallum, A., & Nigam, K. (1998, July). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization* (Vol. 752, No. 1, pp. 41-48).
- [102] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 24, 123-140.
- [103] Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2), 337-407.
- [104] Chauhan, R., Ghanshala, K. K., & Joshi, R. C. (2018, December). Convolutional neural network (CNN) for image detection and recognition. In *2018 first international conference on secure cyber computing and communication (ICSCCC)* (pp. 278-282). IEEE.
- [105] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with Deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- [106] Valueva, M. V., Nagornov, N. N., Lyakhov, P. A., Valuev, G. V., & Chervyakov, N. I. (2020). Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and computers in simulation*, 177, 232-243.
- [107] Venkatesan, R., & Li, B. (2017). *Convolutional neural networks in visual computing: a concise guide*. CRC Press.
- [108] Goodfellow, I. (2016). *Deep Learning*.
- [109] Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2), 179-211.
- [110] Lipton, Z. C. (2015). A Critical Review of Recurrent Neural Networks for Sequence Learning. *arXiv Preprint, CoRR, abs/1506.00019*.
- [111] Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7), 1235-1270.
- [112] Long Short-Term Memory Networks - MATLAB & Simulink. <https://www.mathworks.com/help/Deeplearning/ug/long-short-term-memory-networks.html>
- [113] Hochreiter, S. (1997). Long Short-term Memory. *Neural Computation MIT-Press*.

-
- [114] Schmidhuber, J., & Hochreiter, S. (1997). Long short-term memory. *Neural Comput*, 9(8), 1735-1780.
- [115] Cho, K. (2014). Learning phrase representations using RNN encoder-decoder for statistical *Machine* translation. *arXiv preprint arXiv:1406.1078*.
- [116] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.
- [117] Shen, G., Tan, Q., Zhang, H., Zeng, P., & Xu, J. (2018). *Deep Learning* with gated recurrent unit networks for financial sequence predictions. *Procedia computer science*, 131, 895-903.
- [118] Horn, R. A., & Johnson, C. R. (1994). *Topics in matrix analysis*. Cambridge university press.
- [119] Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11), 2673-2681.
- [120] Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional *LSTM* and other neural network architectures. *Neural networks*, 18(5-6), 602-610.
- [121] Faust, O., Hagiwara, Y., Hong, T. J., Lih, O. S., & Acharya, U. R. (2018). *Deep Learning* for healthcare applications based on physiological signals: A review. *Computer methods and programs in biomedicine*, 161, 1-13.
- [122] Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.
- [123] Bahdanau, D. (2014). Neural *Machine* translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.
- [124] Zhang, X., Chen, K., Fu, W., & Huang, H. (2019). Neural network-based stochastic adaptive attitude control for generic hypersonic vehicles with full state constraints. *Neurocomputing*, 351, 228-239.
- [125] Devlin, J. (2018). Bert: Pre-training of *Deep* bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [126] Brown, T. B. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.
- [127] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8), 9.
-

-
- [128] Nyandwi, J. (2023). The Transformer Blueprint: A Holistic Guide to the Transformer Neural Network Architecture. *Deep Learning Revision*.
- [129] Ali, A. M., Benjdira, B., Koubaa, A., El-Shafai, W., Khan, Z., & Boulila, W. (2023). Vision transformers in image restoration: A survey. *Sensors*, 23(5), 2385.
- Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786), 504-507.
- [130] Kim, H., & Ko, B. C. (2024). Rethinking Attention Mechanisms in Vision Transformers with Graph Structures. *Sensors*, 24(4), 1111.
- [131] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine Learning* (pp. 1096-1103).
- [132] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2008, July). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine Learning* (pp. 1096-1103).
- [133] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
- [134] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning research*, 15(1), 1929-1958.
- [135] Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in *Deep Learning*. In *international conference on Machine Learning* (pp. 1050-1059). PMLR.
- [136] Bishop, C. M., & Nasrabadi, N. M. (2006). *Pattern recognition and Machine Learning* (Vol. 4, No. 4, p. 738). New York: springer.
- [137] Bridle, J. S. (1990). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In *Neurocomputing: Algorithms, architectures and applications* (pp. 227-236). Berlin, Heidelberg: Springer Berlin Heidelberg.
- [138] Vonesch, C., Blu, T., & Unser, M. (2007). Generalized Daubechies wavelet families. *IEEE transactions on signal processing*, 55(9), 4415-4429.
- [139] Roc, A., Pillette, L., Mladenovic, J., Benaroch, C., N’Kaoua, B., Jeunet, C., & Lotte, F. (2021). A review of user training methods in brain computer interfaces based on mental tasks. *Journal of Neural Engineering*, 18(1), 011002.
-

-
- [140] Ajali-Hernández, N., & Travieso-Gonzalez, C. M. (2022). Analysis of Brain Computer Interface Using *Deep and Machine Learning*. In *Artificial Intelligence Annual Volume 2022*. IntechOpen.
- [141] Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [142] Loshchilov, I. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- [143] Tharwat, A. (2021). Classification assessment methods. *Applied computing and informatics*, 17(1), 168-192.
- [144] Powers, D. M. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*.
- [145] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4), 427-437.
- [146] Viera, A. J., & Garrett, J. M. (2005). Understanding interobserver agreement: the kappa statistic. *Fam med*, 37(5), 360-363.
- [147] Youden, W. J. (1950). Index for rating diagnostic tests. *Cancer*, 3(1), 32-35.
- [148] Kotz, S., Balakrishnan, N., Read, C. B., Vidakovic, B., & Johnson, N. L. (2005). *Encyclopedia of Statistical Sciences, Volume 1*. John Wiley & Sons.
- [149] Hyndman, R. J., & Fan, Y. (1996). Sample quantiles in statistical packages. *The American Statistician*, 50(4), 361-365.
- [150] Pattnaik, S., Dash, M., & Sabut, S. K. (2016, January). DWT-based feature extraction and classification for motor imaginary EEG signals. In *2016 International Conference on Systems in Medicine and Biology (ICSMB)* (pp. 186-201). IEEE.
- [151] Myrden, A., & Chau, T. (2015). Effects of user mental state on EEG-BCI performance. *Frontiers in human neuroscience*, 9, 308.
- [152] Ajali, N. I., & Travieso, C. M. (2023). Analysis of Brain Signals to Forecast Motor Intentions Using Artificial Intelligence. In *Sustainable Computing: Transforming Industry 4.0 to Society 5.0* (pp. 31-47). Cham: Springer International Publishing.
- [153] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., & Yger, F. (2018). A review of classification algorithms for EEG-based brain-computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3), 031005.
-

-
- [154] Li, Y., Zhang, X. R., Zhang, B., Lei, M. Y., Cui, W. G., & Guo, Y. Z. (2019). A channel-projection mixed-scale convolutional neural network for motor imagery EEG decoding. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(6), 1170-1180.
- [155] Li, H., Zhang, D., & Xie, J. (2023). MI-DABAN: A dual-attention-based adversarial network for motor imagery classification. *Computers in Biology and Medicine*, 152, 106420.
- [156] Chen, P., Gao, Z., Yin, M., Wu, J., Ma, K., & Grebogi, C. (2021). Multiattention adaptation network for motor imagery recognition. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(8), 5127-5139.
- [157] Kwak, Y., Song, W. J., & Kim, S. E. (2022). FGANet: fNIRS-guided attention network for hybrid EEG-fNIRS brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30, 329-339.
- [158] Shin, J., von Lühmann, A., Blankertz, B., Kim, D. W., Jeong, J., Hwang, H. J., & Müller, K. R. (2016). Open access dataset for EEG+ NIRS single-trial classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(10), 1735-1745.
- [159] Sartipi, S., & Cetin, M. (2024). Subject-independent deep architecture for EEG-based motor imagery classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*.
- [160] Lv, D. M., Dang, W. D., Feng, J. H., & Gao, Z. K. (2024). Attention-based CNN model for motor imagery classification from nonlinear EEG signals. *Physica A: Statistical Mechanics and its Applications*, 130191.
- [161] Zhang, G., Davoodnia, V., Sepas-Moghaddam, A., Zhang, Y., & Etemad, A. (2019). Classification of hand movements from EEG using a deep attention-based LSTM network. *IEEE Sensors Journal*, 20(6), 3113-3122.
- [162] Zhang, D., Yao, L., Chen, K., Wang, S., Haghghi, P. D., & Sullivan, C. (2019). A graph-based hierarchical attention model for movement intention detection from EEG signals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(11), 2247-2253.
- [163] Dosovitskiy, A. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.
- [164] Ajali-Hernández, N. I., Ruiz-García, A., & Travieso-González, C. M. (2024). ANN based-model for estimating the boron permeability coefficient as boric acid in SWRO desalination plants using ensemble-based machine learning. *Desalination*, 573, 117180.
-

- [165] Ajali-Hernández, N. I., & Travieso-González, C. M. (2024). Novel cost-effective method for forecasting COVID-19 and hospital occupancy using deep learning. *Scientific Reports*, *14*(1), 25982.
- [166] Hernández-López, R., Travieso-González, C. M., & Ajali-Hernández, N. I. (2024). Sky Image Classification Based on Transfer Learning Approaches. *Sensors*, *24*(12), 3726.

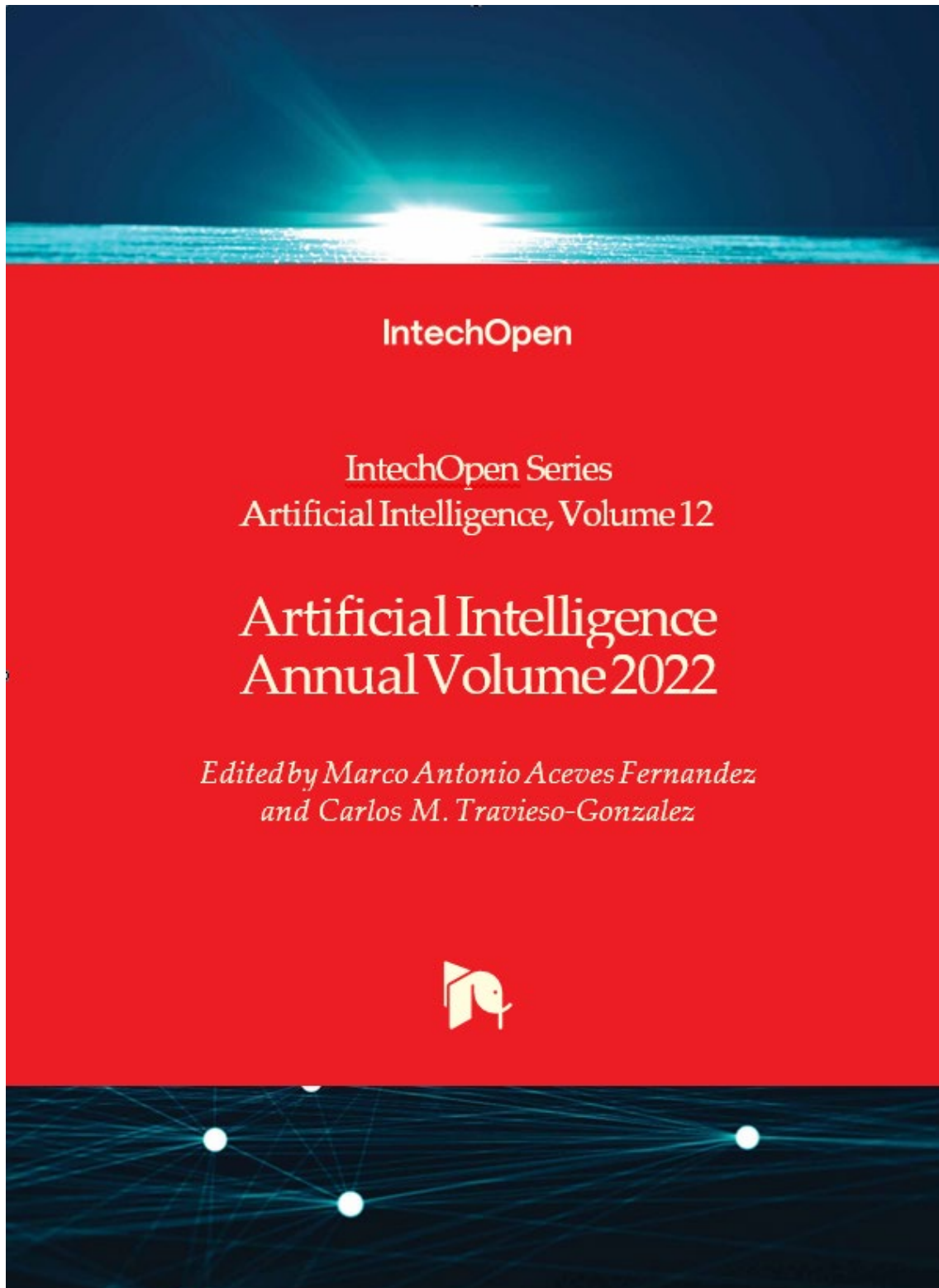
8. Anexos

En esta sección se recogen los diferentes trabajos publicados o en revisión que han sido fruto de esta investigación. Además, se añaden las diferentes apariciones en medios de difusión y por último las conclusiones y líneas futuras en español.

Se añaden en los anexos por orden; los dos capítulos de libro, seguidos de las publicaciones relacionadas con *BCI*. Posteriormente se añaden los artículos de *Desalination*, *COVID* y la colaboración en *Sensors*. Finalmente, se añaden los artículos bajo revisión de emociones y de firma seguido de los anexos de difusión realizados y de las conclusiones en español.

Los artículos bajo revisión no se añaden al completo por temas de exclusividad a la hora de publicarlos en las revistas, ya que si es publicado en una tesis no se pueden publicar como novedad en estas.

8.1. Anexo I. Publicación capítulo libro *IntechOpen (Springer)*



Artificial Intelligence Annual Volume 2022

<http://dx.doi.org/10.5772/intechopen.109246>

Edited by Marco Antonio Aceves Fernandez and Carlos M. Travieso-Gonzalez

Contributors

Jing-Sin Liu, Hc Liao, Han-Jung Chou, Iqbal Haq, Md. Ismail Hossain, Md. Moshur Rahman, Md. Inanul Haq Mithun, Ashis Talukder, Md. Jakaria Habib, Md. Sanwar Hossain, Ewa J. J. Klesczuk, Michele Bennett, Karim Hayes, Rajesh Mehta, Yalın Bastanlar, Semih Orhan, Nabil Ajali, Carlos M. Travieso-Gonzalez, Kishore Kumar Kamarajugadda, Pavani Morva, Suxia Cui, Soham Hamsi, Veeramani Sonai, Indira Bharathi

© The Editor(s) and the Author(s) 2022

The rights of the editor(s) and the author(s) have been asserted in accordance with the Copyright, Designs and Patents Act 1988. All rights to the book as a whole are reserved by INTECHOPEN LIMITED. The book as a whole (compilation) cannot be reproduced, distributed or used for commercial or non-commercial purposes without INTECHOPEN LIMITED's written permission. Enquiries concerning the use of the book should be directed to INTECHOPEN LIMITED rights and permissions department (permissions@intechopen.com).

Violations are liable to prosecution under the governing Copyright Law.



Individual chapters of this publication are distributed under the terms of the Creative Commons Attribution 3.0 Unported License which permits commercial use, distribution and reproduction of the individual chapters, provided the original author(s) and source publication are appropriately acknowledged. If so indicated, certain images may not be included under the Creative Commons license. In such cases users will need to obtain permission from the license holder to reproduce the material. More details and guidelines concerning content reuse and adaptation can be found at <http://www.intechopen.com/copyright-policy.html>.

Notice

Statements and opinions expressed in the chapters are those of the individual contributors and not necessarily those of the editors or publisher. No responsibility is accepted for the accuracy of information contained in the published chapters. The publisher assumes no responsibility for any damage or injury to persons or property arising out of the use of any materials, instructions, methods or ideas contained in the book.

First published in London, United Kingdom, 2022 by IntechOpen

IntechOpen is the global imprint of INTECHOPEN LIMITED, registered in England and Wales, registration number: 11086078, 5 Princes Gate Court, London, SW7 2QJ, United Kingdom
Printed in Croatia

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

Additional hard and PDF copies can be obtained from orders@intechopen.com

Artificial Intelligence Annual Volume 2022

Edited by Marco Antonio Aceves Fernandez and Carlos M. Travieso-Gonzalez
p. cm.

This title is part of the Artificial Intelligence Book Series, Volume 12

Series Editor: Andries Engelbrecht

Print ISBN 978-1-83768-946-0

Online ISBN 978-1-83768-947-7

eBook (PDF) ISBN 978-1-83768-948-4

ISSN 2633-1403

Chapter 7

Analysis of Brain Computer Interface Using Deep and Machine Learning

Nabil Ajali-Hernández and Carlos M. Travieso-Gonzalez

Abstract

Pattern recognition is becoming increasingly important topic in all sectors of society. From the optimization of processes in the industry to the detection and diagnosis of diseases in medicine. Brain-computer interfaces are introduced in this chapter. Systems capable of analyzing brain signal patterns, processing and interpreting them through machine and deep learning algorithms. In this chapter, a hybrid deep/machine learning ensemble system for brain pattern recognition is proposed. It is capable to recognize patterns and translate the decisions to BCI systems. For this, a public database (Physionet) with data of motor tasks and mental tasks is used. The development of this chapter consists of a brief summary of the state of the art, the presentation of the model together with some results and some promising conclusions.

Keywords: brain-computer interfaces, deep learning, machine learning, pattern recognition, artificial intelligence, neural network

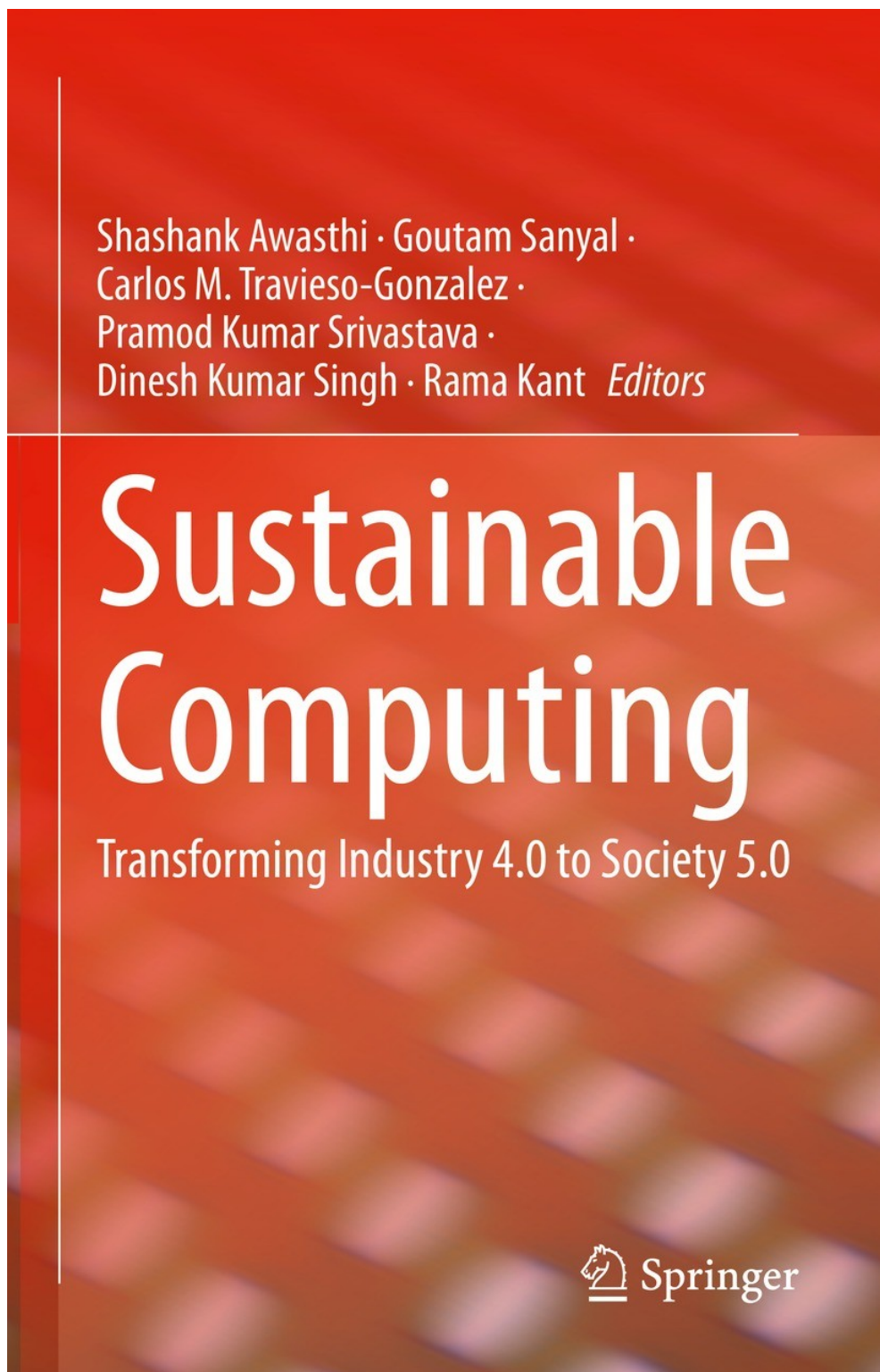
1. Introduction

The brain is the most important organ in the human body. It processes, integrates and coordinates the information it receives from the organs and the senses and makes decisions, sending them to the rest of the body, like a processor in a computer. The brain works through electrochemical impulses, called synapses, which allow the transmission of information between neurons [1–3].

These impulses could be classified by their frequency into different types of brain waves. Delta (1–3 Hz), theta (3–8 Hz), alpha (8–13 Hz), beta (13–30 Hz), and gamma (30–100 Hz) waves [4]. These brain waves are the reflection of electrical activity (in microvolts) and therefore of thoughts and motor intentions. They can be captured by the electroencephalogram (EEG) and their study can lead to the detection of pathologies related to the brain (Alzheimer's, Parkinson's, epilepsy) [5, 6].

Figure 1 shows the EEG of a normal person, where 64 channels have been placed throughout the head (10–20 system) and brain waves are monitored over time. It can be seen how there are an infinity of patterns that provide important information about what is happening.

8.2. Anexo II. Publicación capítulo libro *Sustainable Computing: Transforming Industry 4.0 to Society 5.0*, 31-47



“Analysis of brain signals to forecast motor intentions using artificial intelligence”

Nabil I. Ajali¹ and Carlos M. Travieso¹

¹ Signals and Communications department, Universidad de Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria (35001), Spain.
carlos.travieso@ulpgc.es

Abstract. In order to improve lifestyle of people with motor disabilities, Brain Computer Interfaces are a potential solution. BCI's seek to achieve control of a machine through the use of brain waves. In this work, A brief historical review of the state of the art in the field of BCIs is made in this work. How brain signal processing is carried out to finally obtain a BCI tool is showing. The feature extractions and the main machine learning classification methods for classifications are seen. Finally, a classification of a public EEG dataset (Physionet) is carried out to and compared with other systems as an example. Thus, showing the general aspects of the development of a BCI systems, which will be part of technologies in society 5.0.

Keywords: Brain Computer Interface, Brain signals, Machine Learning, Industry 4.0, Society 5.0

1 Introduction

In 21st century, more than 120 million people around the world have some kind of motor problems due to an accidents or mental illness.

As a results, amputations, atrophy or some loss of mobility in parts of the body is common. This hinders integration into society and worsens the quality of life [1], [2], [3], [4].

Brain Computer Systems (BCI's) use technology developed in the industry 4.0 to establish a direct communication pathway between the brain and an external device using brain electrical signals. BCIs are often focus at researching, mapping, assisting, augmenting, or repairing human cognitive or sensory-motor functions [5].

Hence, these devices are a potential solution for motor problems, when the brain is not involved directly. Signals could be analyzed and using the machine and deep learning algorithms, a human-computer interaction is established to interpret and allocate this signal in a robotic limb or in a exoskeleton to form complete BCI system.

In this work, a series of concepts necessary to understand the nature of the brain signal and the methods used to develop a BCI is shown. In addition, a classification of a public dataset is carried out with our system. This, allows to obtain conclusions and paradigms of this application for Industry 4.0. and the inclusion in society 5.0.

1.1 State of art

The brain is the most important organ in the human body. As the processors in computers, it controls most of the body's activities. It processes, integrates and coordinates the information received from the organs and the senses and send instructions to the rest of the body. Its basic units, neurons, work by electrochemical impulses called synapses [6]. The electrical nature of the brain was discovered by Richard Caton in 1875 and the first electroencephalogram was done in 1924 by Hans Berger [7], [8].

In the years after the second world war until 1970, advances in the field of technology promoted the society to industry 3.0. In 1970 machine-brain concept was developed by the professor Jacques Vidal. Reaching in 1988, the first non-invasive EEG control of an object, specifically a robot. The authors of this work were Stevo Bozinovski, Mihail Sestakov and Liljana Bozinovska, pioneers in the first robot control using the EEG [9], [10].

Consequently, followed by these works of BCI systems, different concepts of neurobiology such as brain neuroplasticity were discovered in this field. A range of possible applications were open up, e.g., the implantation of electrodes directly in the cerebral cortex to handle with the brain as natural sensors or effector channels [11]. After years of experimentation with animals, the first neuroprosthetic devices implanted in humans was emerged in the mid-1990s.

8.3. Anexo III. Publicación *IRBM*

IRBM 45 (2024) 100836



Contents lists available at ScienceDirect

IRBM

journal homepage: www.elsevier.com/locate/irbm

Elsevier Masson France

EM|consulte
www.em-consulte.com

Original Article

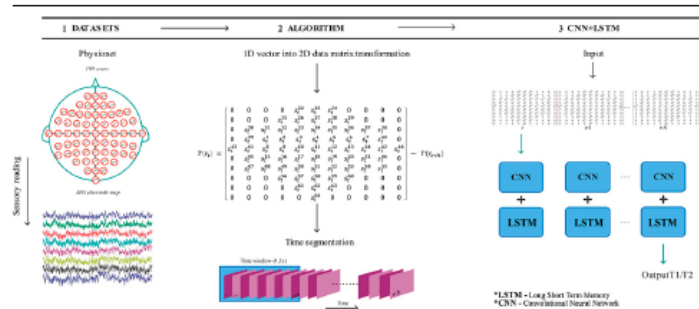
Study of an Optimization Tool Avoided Bias for Brain-Computer Interfaces Using a Hybrid Deep Learning Model

Nabil I. Ajali-Hernández^{a,*}, Carlos M. Travieso-González^a, Nayara Bermudo-Mora^b, Patricia Reino-Cacho^b, Sheila Rodríguez-Saucedo^b^a Signal and Communications department of the University of Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria, 35001, Spain^b School of Industrial and Civil Engineering, Universidad de Las Palmas de Gran Canaria, 35001, Spain

HIGHLIGHTS

- Hybrid model that tackles bias in user-specific Brain Computer Interfaces training.
- Combined 2D CNN and LSTM for EEG mental task classification.
- Classification rates of up to 90% with an average of 74.54% for EEG tasks.
- Set thresholds increase accuracy by 21.34%, remove low affinity.
- Methodology enhances BCI systems, improves accuracy and reliability avoiding bias.

GRAPHICAL ABSTRACT



ARTICLE INFO

Article history:

Received 11 May 2023

Received in revised form 13 January 2024

Accepted 16 April 2024

Available online 22 April 2024

Keywords:

Brain-computer interface
Human-machine interaction
Motor imagery
Deep learning
Brain tools

ABSTRACT

Objective: This study addresses the challenge of user-specific bias in Brain-Computer Interfaces (BCIs) by proposing a novel methodology. The primary objective is to employ a hybrid deep learning model, combining 2D Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) layers, to analyze EEG signals and classify imagined tasks. The overarching goal is to create a generalized model that is applicable to a broader population and mitigates user-specific biases.

Materials and Methods: EEG signals from imagined motor tasks in the public dataset Physionet form the basis of the study. This is due to the need to use other databases in addition to the BCI competition. A model of arrays emulating the electrode arrangement in the head is proposed to capture spatial information using CNN, and LSTM algorithms are used to capture temporal information, followed by signal classification.

Results: The hybrid model is implemented to achieve a high classification rate, reaching up to 90% for specific users and averaging 74.54%. Error detection thresholds are set to eliminate subjects with low task affinity, resulting in a significant improvement in classification accuracy of up to 21.34%.

Conclusion: The proposed methodology makes a significant contribution to the BCI field by providing a generalized system trained on diverse user data that effectively captures spatial and temporal EEG signal features. This study emphasizes the value of the hybrid model in advancing BCIs, highlighting its potential for improved reliability and accuracy in human-computer interaction. It also suggests the exploration of additional advanced layers, such as transformers, to further enhance the proposed methodology.

© 2024 AGBM. Published by Elsevier Masson SAS. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

* Corresponding author.

E-mail addresses: nabil.ajali101@alu.ulpgc.es (N.I. Ajali-Hernández), carlos.travieso@ulpgc.es (C.M. Travieso-González).<https://doi.org/10.1016/j.irbm.2024.100836>1959-0318/© 2024 AGBM. Published by Elsevier Masson SAS. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

8.4. Anexo IV. Publicación bajo revisión *BCI. Scientific Reports*

A Hybrid Deep Learning Approach with 3D Temporal Data for User-Independent Classification of Imagined Motor Tasks in Brain-Computer Interfaces

Nabil I. Ajali-Hernández^a, Carlos M. Travieso-González^a and Ruymán Hernández-López^a

^a *Signal and Communications department of the University of Las Palmas de Gran Canaria (ULPGC), Las Palmas de Gran Canaria, Spain, 35001, ES (e-mail: nabil.ajali101@alu.ulpgc.es; carlos.travieso@ulpgc.es; ruymán.hernandez@ulpgc.es;)*

Abstract

This paper introduces a novel deep learning framework for improved classification of imagined motor tasks in Brain-Computer Interfaces (BCIs). Our approach emphasizes a unique 3D data packaging technique that leverages spatial electrode information and carefully calibrated temporal windows. These windows are designed to mirror human reaction times, enhancing the model's ability to discern relevant EEG patterns. The large-scale Physionet dataset is employed, thereby minimizing the bias that is prevalent in studies that rely on smaller, user-specific datasets (such that 75% of the total number of works have used BCI competition datasets with less than 10 users).

The hybrid model integrates several Convolutional Neural Networks (CNNs) for spatial feature extraction with Long Short-Term Memory (LSTM) networks to capture temporal dependencies. It is of note that this approach yields classification binary accuracy exceeding 80% with remarkably low standard deviation values (<1.5%), which surpasses results in comparable generalized training scenarios. These findings highlight the efficacy of our 3D data packaging and processing for taking into account spatiotemporal information and building a robust BCI system. Obtaining a generalized system, trained without user specificity, with high performance, stability and with a wide potential to be merged with other systems such as transformers.

Keywords: Brain-computer interface; Human-machine interaction; Motor Imagery; Deep Learning; Brain devices

1. Introduction

Brain-computer interfaces (BCIs) are systems that provide a direct communication pathway between the brain and external devices through the use of brain signals. These signals are collected by an electroencephalogram (EEG) device located on the test subject's head. Milliseconds after the subject performs or imagines a task, the signal goes through the device, and deep learning algorithms process and recognize certain brain states or patterns. This pattern is then classified and used to control some type of device or perform a task. For example, to move a prosthesis, a mouse pointer, or write words on a screen. Thus, BCIs can be used for a variety of purposes, including rehabilitation for individuals with paralysis or other neurological conditions, control of prosthetic devices, and even enhancement of cognitive or physical abilities improving the quality of life of people [1], [2]. In the last decade notable studies and articles on BCIs have proposed BCI systems for communication and control, as seen in works such as [3] and [4]. Nicolas-Alonso and Gomez-Gil [5] have provided a comprehensive and detailed review of BCIs in their work. In the field of medicine, BCIs have been applied with promising results, such as the development of a model for neurorehabilitation proposed by Shih *et al.* [6].

8.5. Anexo V. Publicación *Desalination*

Desalination 573 (2024) 117180



Contents lists available at ScienceDirect

Desalination

journal homepage: www.elsevier.com/locate/desal

ANN based-model for estimating the boron permeability coefficient as boric acid in SWRO desalination plants using ensemble-based machine learning

Nabil I. Ajali-Hernández^{a,*}, A. Ruiz-García^b, Carlos M. Travieso-González^{a,c}^a Institute for Technological Development and Innovation in Communications (IDeTIC), University of Las Palmas de Gran Canaria, Campus Universitario de Tafra, Las Palmas de Gran Canaria 35017, Spain^b Department of Electronic Engineering and Automation, University of Las Palmas de Gran Canaria, Campus Universitario de Tafra, Las Palmas de Gran Canaria 35017, Spain^c Signals and Communications Department (DSC), University of Las Palmas de Gran Canaria, Campus Universitario de Tafra, Las Palmas de Gran Canaria 35017, Spain

HIGHLIGHTS

- Predicting the Boron Permeability Coefficient of Seawater in a Reverse Osmosis Desalination Plant Using Machine Learning
- Analysis of different input conditions for prediction in SWRO systems with multiple membrane elements varying conditions
- Ensemble-based machine learning model for prediction of boron rejection permeability coefficient
- Calculation of boron rejection concentration from boron permeability coefficient

ARTICLE INFO

Keywords:
Desalination
Reverse osmosis
Boron rejection
Neural networks
Machine learning

ABSTRACT

This study examines the relationship between input conditions and the prediction of boron rejection in full-scale seawater reverse osmosis (SWRO) desalination plants using ensemble-based machine learning. While reverse osmosis is the dominant desalination technology, limited research has focused on analyzing plant performance under actual operating conditions. To address this gap, we developed and implemented machine learning algorithms to forecast boron permeability coefficient values, which are indicative of boron rejection concentrations in the permeate. Our analysis utilizes data from a SWRO desalination plant in southeast Spain, examining various input variables and their influence on the prediction of these parameters. The results demonstrate that our ensemble-based machine learning approach can predict boron permeability coefficient values with a reasonable margin of error of 1 mgL^{-1} , as evidenced by mean average error (MAE) and mean absolute percentage error (MAPE) values of $7.93 \cdot 10^{-8}$ and 11.8% , respectively. In conclusion, an innovative application of artificial intelligence algorithms in the field of water purification under real operational conditions has been introduced, thus introducing valuable insights into the use of machine learning algorithms for forecasting boron rejection concentrations in full-scale SWRO desalination plants. The findings lay the foundation for future researches exploring automated and deep-learning methods in water purification.

1. Introduction

Water scarcity is one of the main challenges for human life and has stimulated the use of desalination technologies to produce water for different purposes. Among the available technologies, reverse osmosis (RO) is in the lead due to its reliability [1,2] and lower specific energy consumption in comparison with other technologies [3,4]. Unfortunately, RO has some weak points such as loss of performance due to the

fouling of RO membranes [5–7] and low rejection rates of some ions that can be toxic to humans such as boron, fluorine, etc. [8,9]. This has led to significant efforts being made to improve the selectivity of RO membranes, trying not to penalize the production of permeate per unit of membrane-active surface [10,11]. Usually, the determination of ion concentration in the permeate (C_{PB}) is carried out in a laboratory so it is not measured in real-time. This can be problematic because it is not known in real time how a change in operating conditions affects the ion permeability coefficients (B_i) and consequently the C_{PB} . It is important to

* Corresponding author.

E-mail address: nabil.ajali101@alu.ulpgc.es (N.I. Ajali-Hernández).<https://doi.org/10.1016/j.desal.2023.117180>

Received 5 September 2023; Received in revised form 7 November 2023; Accepted 19 November 2023

Available online 28 November 2023

0011-9164/© 2023 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

8.6. Anexo VI. Publicación *Scientific Reports* COVID

www.nature.com/scientificreports

scientific reports

Check for updates

OPEN

Novel cost-effective method for forecasting COVID-19 and hospital occupancy using deep learning

Nabil I. Ajali-Hernández[✉] & Carlos M. Travieso-González

The emergence of the COVID-19 pandemic in 2019 and its rapid global spread put healthcare systems around the world to the test. This crisis created an unprecedented level of stress in hospitals, exacerbating the already complex task of healthcare management. As a result, it led to a tragic increase in mortality rates and highlighted the urgent need for advanced predictive tools to support decision-making. To address these critical challenges, this research aims to develop and implement a predictive system capable of predicting pandemic evolution with accuracy (in terms of Mean Absolute Error (MAE), Root Mean Square Error (RMSE), R^2 , and Mean Absolute Percentage Error (MAPE)) and low computational and economic cost. It uses a set of interconnected Long Short Term-memory (LSTM) with double bidirectional LSTM (BiLSTM) layers together with a novel preprocessing based on future time windows. This model accurately predicts COVID-19 cases and hospital occupancy over long periods of time using only 40% of the set to train. This results in a long-term prediction where each day we can query the cases for the next three days with very little data. The data utilized in this analysis were obtained from the "Hospital Insular" in Gran Canaria, Spain. These data describe the spread of the coronavirus disease (COVID-19) from its initial emergence in 2020 until March 29, 2022. The results show an improvement in MAE (< 161), RMSE (< 405), and MAPE (> 0.20) compared to other studies with similar conditions. This would be a powerful tool for the healthcare system, providing valuable information to decision-makers, allowing them to anticipate and strategize for possible scenarios, ultimately improving public health outcomes and optimizing the allocation of healthcare and economic resources.

Keywords COVID-19, Covid prediction, Daily covid, Hospital occupancy, LSTM

The global pandemic of COVID-19 currently accumulates more than 770 million cases¹. Although it seems under control, Coronaviruses are susceptible to genetic mutation, which can cause a high number of cases in a short period of time, putting pressure on hospitals and healthcare systems.

Figure 1 shows the historical number of cases of COVID-19. It can be seen how different peaks of COVID-19 strains have rapidly reached high levels of weekly cases in very short periods. This situation could lead to a new global crisis like the one we experienced from 2020 to 2022.

Artificial Intelligence (AI) and all its branches, such as deep and machine learning algorithms, neural networks, and others, are being intensively explored for new healthcare applications in areas such as diagnostic imaging, lifestyle management, early detection of neurological and neurodegenerative diseases, risk analysis and monitoring, health information management, rehabilitation, and virtual healthcare. The expected benefits in these areas are broad and include optimization of diagnostics, monitoring, and control of disease progression, increased speed in imaging, greater understanding of predictive screening, and reduced costs and inefficiencies in healthcare².




In this work, we focus on achieving a new methodology based on combining artificial intelligence algorithms related to time-series and attention layers to predict different daily caseload ranges. In this way, we hope that hospitals around the world will have access to a low-cost technology capable of predicting each day's occupancy for the next three days. This gives them some leeway and reduces pressure and stress on the system.

Signals and Communications Department (DSC), University of Las Palmas de Gran Canaria, Campus Universitario de Tafira, 35017 Las Palmas de Gran Canaria, Spain. [✉]email: nabil.ajali101@alu.ulpgc.es

8.7. Anexo VII. Publicación colaboración en *Sensors*

Article

Sky Image Classification Based on Transfer Learning Approaches

Ruymán Hernández-López , Carlos M. Travieso-González * and Nabil I. Ajali-Hernández 

Signals and Communications Department (DSC), Institute for Technological Development and Innovation in Communications (IDeTIC), University of Las Palmas de Gran Canaria (ULPGC), 35017 Las Palmas de Gran Canaria, Spain; ruyman.hernandez@ulpgc.es (R.H.-L.); nabil.ajali101@alu.ulpgc.es (N.I.A.-H.)

* Correspondence: carlos.travieso@ulpgc.es

Abstract: Cloudy conditions at a local scale pose a significant challenge for forecasting renewable energy generation through photovoltaic panels. Consequently, having real-time knowledge of sky conditions becomes highly valuable. This information could inform decision-making processes in system operations, such as determining whether conditions are favorable for activating a standalone system requiring a minimum level of radiation or whether sky conditions might lead to higher energy consumption than generation during adverse cloudy conditions. This research leveraged convolutional neural networks (CNNs) and transfer learning (TL) classification techniques, testing various architectures from the *EfficientNet* family and two *ResNet* models for classifying sky images. Cross-validation methods were applied across different experiments, where the most favorable outcome was achieved with the *EfficientNetV2-B1* and *EfficientNetV2-B2* models boasting a mean *Accuracy* of 98.09%. This study underscores the efficacy of the architectures employed for sky image classification, while also highlighting the models yielding the best results.

Keywords: cloudiness classification; deep learning; transfer learning; convolutional neural networks; *EfficientNet* models; *ResNet* models; sky images; photovoltaic power; renewable energy



Citation: Hernández-López, R.; Travieso-González, C.M.; Ajali-Hernández, N.I. Sky Image Classification Based on Transfer Learning Approaches. *Sensors* **2024**, *24*, 3726. <https://doi.org/10.3390/s24123726>

Academic Editors: Stefano Berrett, Jean-Baptiste Thomas, Baptiste Magnier and Khizar Hayat

Received: 6 May 2024

Revised: 31 May 2024

Accepted: 6 June 2024

Published: 8 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The production of electric power from solar energy is influenced by variations caused by solar obstruction due to cloud cover. Detecting and understanding cloud cover have been investigated for estimating and forecasting solar irradiance, and in this way, predicting photovoltaic power generation [1]. The amount of electrical energy generated in a photovoltaic plant is not solely determined by infrastructure specifics like the number and **type of solar panels employed, it is also directly impacted by the radiation received by the panels**, hence, the prevailing cloud cover. Sky conditions are contingent upon the time of day, season, and geographical location. Nevertheless, forecasting local cloud conditions remains highly challenging.

Furthermore, identifying specific cloud cover conditions through automatic pattern recognition in sky images can prove invaluable across various applications. For instance, it could aid in automated decision-making processes governing the operation of systems in remote, off-grid locations where power consumption is critical. In such scenarios, it becomes crucial to ascertain whether conditions are conducive for activating a standalone system requiring a minimum radiation threshold or if cloudier conditions might result in higher consumption than energy generation, potentially leading to system failure.

1.1. Related Work

In contemporary times, a plethora of research endeavors are dedicated to gauging solar irradiation levels and assessing sky conditions via image processing. In recent years, several surveys have scrutinized cloud conditions utilizing various pattern recognition technologies and diverse approaches to image acquisition.

8.8. Anexo VIII. Publicación bajo revisión *Scientific Reports*, clasificación de emociones

Emotions for Everyone: A Low-Cost, High-Accuracy Method for Emotion Classification

Nabil I. Ajali-Hernández and Carlos Manuel Travieso-González

Signals and Communications Department (DSC), University of Las Palmas de Gran Canaria,
Campus Universitario de Tafira, Las Palmas de Gran Canaria 35017, Spain

Corresponding autor: Nabil I. Ajali-Hernández

Email: nabil.ajali101@alu.ulpgc.com

Abstract. The classification of emotions is of vital importance in health care, particularly in the context of early detection of cognitive disorders. Given the critical role of emotions as early indicators of cognitive health, this study addresses the need to develop effective and accessible classification methods. In this research, we present an innovative approach to emotion classification using a proprietary dataset and harnessing the power of deep learning. In particular, we use a specific, innovative combination of attentional layers and Long-Short Term Memory (LSTM) algorithms to achieve an emotion classification.

A key differentiator of our methodology is the use of a compact and low-cost array of biometric sensors. This approach provides a cost-effective alternative to traditional systems, which often rely on more complex and expensive sensor arrays, such as those using electroencephalography (EEG). Despite the affordability of our sensor configuration, our classification model achieves an outstanding accuracy rate of 93.75%. This performance not only demonstrates the effectiveness of our method, but also positions it at the forefront of emotion classification using these sensors. By significantly reducing cost while increasing classification accuracy, our method helps to push the boundaries of current state-of-the-art approaches and provides a novel and cost-effective solution for emotion classification and cognitive health monitoring.

Keywords: Emotion recognition, Cognitive Health, Emotional classification, Biometric sensors, Deep learning.

1. Introduction

In the ever-evolving landscape of human-machine interaction, emotion recognition has emerged as a critical frontier that transcends disciplinary boundaries and has profound implications across multiple domains. The historical trajectory of emotion recognition has been, and continues to be, deeply rooted in societal advances, technological innovations, and our evolving understanding of human cognition, since we are emotions [1]. From its earliest stages, when psychological theories laid the groundwork for understanding emotions, to the present era of unprecedented technological advances, the pursuit of deciphering human emotions has become a compelling imperative.

8.9. Anexo IX. Publicación bajo revisión en *IEEE*. Firma offline

Angular contour parametrization for the in-air signature-based identification

Elyoenai Guerra-Segura, David S. Delgado-Amador, Carlos M. Travieso-González

Abstract— This work presents a novel in-air signature identification system that significantly improves the accuracy and efficiency of biometric authentication. By leveraging 3D sensor data (X, Y, Z) and adapting robust offline signature identification techniques (hidden Markov model probabilistic kernel with support vector machines), we analyze signature contours with unprecedented accuracy. Our method demonstrates the value of three-dimensional analysis, with XZ and YZ projections complementing traditional techniques. The result is a system that achieves remarkable performance even with reduced training data. Experiments on a 100-user database show an average recognition rate of 99.8857% and an EER of 0.0286% using only 30% of the training samples. Using 50% of the training data, we maintain a 100% recognition rate with an even lower EER of 0.0068%. This system outperforms previous approaches and demonstrates the potential of combining offline techniques with in-air signature analysis, opening avenues for future refinements in biometric security.

Index Terms—Biometric systems, In-air signature, Offline signature identification, 3D sensor data

I. INTRODUCTION

AS it is well-known, biometrics refers to the technology field which focuses on identifying individuals from their biometric characteristics. These biometric characteristics are mainly divided into two groups: physical characteristics (fingerprint, iris, face, etc.) and behavioral characteristics (speech, walk, handwritten, signature, etc.).

been made, usually by scanning it. Secondly, online systems capture the data while the signature is being done - being able to gather information about how the signature was created during the process and not only on how the signature looks after the process - and usually show better results (Plamondon & Srihari, 2000b; Vargas et al., 2009). Moreover, offline systems are based on image processing and are usually more complex due to the lack of information.

A. In-air signature

Over the past few years, in-air signature has started to receive special attention due to all the information that can be obtained from the process. This type of signing is a combination of both physical and behavioral systems, since it depends not only on the way the user draws their signature in the air but also on some physical characteristics of the person (length of the arm, capability of turning the wrist, etc.) (Bailador et al., 2011; Behera et al., 2021; Farella et al., 2006; Lu et al., 2019; Rusydi, 2020).

The online information used is the velocity (Anand et al., 2010), the pressure (Haskell et al., 2006) and others (Anand et al., 2010; Hangai et al., 2000; Haskell et al., 2006; Lu et al., 2019; Rusydi, 2020; Sesa i Noguera, 2012; van Nguyen et al., 2017). This information is obtained by capturing gestures with different devices, such as sensor pens, cameras, accelerometers or intelligence surfaces (Bailador et al., 2011; Ferrer et al., 2023; Hangai et al., 2000; Jung et al., 2021; Katagiri & Sugimura, 2002; Liu et al., 2009; Lu et al., 2019; Mendels et

8.10. Anexo X. Otras acciones de difusión

Anexo X.1. Publicación RTVC

(<https://rtvc.es/ulpgc-algoritmo-para-predecir-el-movimiento/>)

ES NOTICIA → Migración | Planes en Canarias | Festivales Canarias 2025 | Tiempo en Canarias | Calendario UD Las Palmas 24/25 | Calendario CD Tenerife 24/25 | Co

rtvc RADIO TELEVISIÓN CANARIA

Noticias - Televisión - Radio -

INFORMACIÓN Acuerdo de la Administradora General del Ente Público Radiotelevisión Canaria (RTVC) por el que se h

Inicio » Noticias

Un brazo robótico para mejorar la vida de personas con movilidad reducida

Redacción RTVC • 24 septiembre 2024 11:31

La ULPGC trabaja en el desarrollo de un algoritmo de IA para predecir el movimiento humano solo con el pensamiento

Investigadores de la **Universidad de Las Palmas de Gran Canaria (ULPGC)** desarrollan un algoritmo de **inteligencia artificial** capaz de leer señales cerebrales y clasificar movimientos imaginados por las personas. El **objetivo es crear una herramienta que «adivine» si alguien piensa en mover el brazo y prediga la dirección del movimiento con un alto grado de precisión.**



La ULPGC desarrolla un algoritmo para predecir el movimiento humano con el pensamiento | ULPGC

El proyecto está dirigido por el estudiante de doctorado Nabil Isaac Ajali Hernández y su tutor, Carlos Travieso, del **Departamento de Señales y Comunicaciones**. También participan las estudiantes Nayara Bermudo, Patricia Reino y Sheila Rodríguez. La investigación cuenta con el apoyo del Gobierno de Canarias, a través de la Agencia Canaria de Investigación, Innovación y Sociedad de la Información.

El objetivo es mejorar la vida de personas con movilidad reducida

Este avance representa un paso previo hacia un objetivo más ambicioso: **trasladar las señales cerebrales a un brazo robótico, permitiendo que este ejecute el movimiento pensado por la persona.** Las simulaciones realizadas hasta ahora muestran un éxito superior al 80% y los investigadores ya han probado un prototipo en condiciones controladas.

La tecnología podría **mejorar significativamente la calidad de vida de personas con movilidad reducida, lesiones medulares, enfermedades neurodegenerativas** como el Parkinson o la ELA. También a aquellas que han perdido un miembro y no pueden usar prótesis biónicas convencionales debido a la falta de respuesta en los nervios.

Además, esta innovación tiene **aplicaciones potenciales** en sectores como el entretenimiento, el desarrollo de exoesqueletos y la docencia para el aprendizaje motor.

La IA en la ingeniería biomédica

El avance de la IA en la ingeniería biomédica es evidente, con ejemplos como el robot quirúrgico «Da Vinci», que realiza operaciones con gran precisión. En el caso del estudio de la ULPGC, el reconocimiento de patrones y la predicción de señales cerebrales convierten este desarrollo en una herramienta esencial, al **gestionar relaciones complejas que otros métodos no pueden manejar.**

Los resultados de esta investigación han sido publicados en la **revista científica IRBM**, una de las más influyentes en ingeniería biomédica.

Anexo X.2. Difusión en la web y redes sociales de la ULPGC

<https://www.ulpgc.es/noticia/2024/09/24/ulpgc-trabaja-desarrollo-algoritmo-predecir-movimiento-humano-solo-pensamiento>

La ULPGC trabaja en el desarrollo de un algoritmo para predecir un movimiento humano sólo con el pensamiento

24 SEP 2024

Compartir en las redes:   

El objetivo es aplicar esta herramienta de IA a un brazo robótico.

Un equipo de investigadores de la ULPGC trabaja en el desarrollo de un **algoritmo de Inteligencia Artificial** especializado en leer señales completas en el cerebro, a través del cual se pueda realizar una clasificación de movimientos imaginados por las personas. Se pretende así disponer de una herramienta capaz de "adivinar" si un sujeto está pensando en mover el brazo, y tras este pensamiento, realizar una clasificación que pueda predecir, con cierto grado de éxito, hacia dónde se va a ejecutar ese movimiento.



Firman este trabajo el estudiante de doctorado [Nabil Isaac Ajali Hernández](#), y su director de tesis [Carlos Travieso](#), adscritos al Departamento de Señales y Comunicaciones, así como las estudiantes [Nayara Bermudo](#), [Patricia Reino](#) y [Sheila Rodríguez](#). El estudio cuenta con financiación del Gobierno de Canarias a través de la **Agencia Canaria de Investigación, Innovación y Sociedad de la Información**.

Esta investigación se plantea como un paso previo para un objetivo mayor, que sería trasladar la lectura de esa señal a un **brazo robótico**, de forma que éste pueda efectuar el movimiento que ha sido pensado por la

persona. Actualmente, las simulaciones hechas por los investigadores arrojan resultados de más de un **80% de éxito** y existe un prototipo que se ha hecho funcionar de manera controlada

Esta tecnología podría suponer una mejora significativa en la calidad de vida de personas con movilidad reducida, lesiones medulares, enfermedades neurodegenerativas como Parkinson o ELA, o en personas que han perdido un miembro y cuyos nervios de la extremidad no responden a la intención motora, haciendo que no funcionen los brazos biónicos convencionales.

Además, esta tecnología tiene potencial para aplicarse en otros sectores, como el entretenimiento, la creación de exoesqueletos o la docencia, como herramienta de aprendizaje motor.

El avance de la Inteligencia Artificial en el campo de la ingeniería biomédica está siendo notable; ya existen robots, como "Da Vinci", capaces de realizar cirugías precisas gracias al uso de los algoritmos de IA. En el caso de la investigación desarrollada por la ULPGC, el reconocimiento de patrones, la clasificación de las señales y las predicciones realizadas hacen de esta herramienta un factor esencial en el día a día, dada su competencia para captar relaciones no lineales que otros métodos simplemente no son capaces de gestionar.

Los resultados de este trabajo han sido publicados en la **revista científica IRBM**, una de las revistas más relevantes en el campo de la ingeniería biomédica.



8.11. Anexo IX. Conclusiones y líneas futuras en español

Conclusiones

La presente investigación confirma y valida la hipótesis central que aquí ha sido planteada: *“La información extraída de la señal cerebral mediante el EEG contiene información valiosa relacionada con la intención motora. El correcto análisis y tratamiento de esta permite clasificar los pensamientos y llevar a cabo una acción motora a partir de una intención”*.

Esta validación se fundamenta en el desarrollo de un sistema de clasificación preciso y robusto para intenciones motoras basado en el procesamiento avanzado de señales EEG y en técnicas de *Machine* y *Deep Learning*. Se destaca que las señales EEG, adecuadamente preprocesadas y descompuestas en sus componentes relevantes, aportan patrones neuronales que permiten identificar de manera efectiva la intención motora, confirmando el valor de esta información para aplicaciones BCI.

Se ha ido mejorado gradualmente el éxito en la clasificación de las señales cerebrales y de la intención motora a lo largo del desarrollo de la investigación. Esto ha sido gracias a utilizar un amplio abanico de posibilidades e ir ajustando mediante diferentes ensayos las técnicas y arquitecturas. Nutriendo el estudio con todo tipo de preprocesamientos, técnicas y arquitecturas simples, complejas e híbridas de *Machine* y *Deep Learning*.

En cuanto a los objetivos planteados, la investigación los ha cumplido en su totalidad:

- 6) Estudiar el campo de las señales cerebrales y de las *Brain-Computer Interfaces (BCI)*:** Se profundizó en la comprensión de las señales EEG, su naturaleza, su obtención y procesamiento, lo cual fue fundamental para entender el tipo de señal a tratar y a su vez para diseñar un sistema de clasificación efectivo.
- 7) Profundizar en los algoritmos de *Machine* y *Deep Learning* aplicados a este paradigma:** Se exploraron y adaptaron distintas arquitecturas avanzadas, incluyendo modelos de *Machine Learning*, modelos híbridos y modelos de *Deep Learning* como *CNN-LSTM* con capas de atención o *Vision Transformers*, para

mejorar la precisión en la clasificación de las intenciones motoras. Con este objetivo se logra diseñar la columna vertebral de los experimentos para poder validar la hipótesis.

- 8) Analizar del estado del arte en lo relacionado con la intención motora y su correcta clasificación:** Se realizó una revisión exhaustiva de estudios previos en *BCI*, lo cual permitió identificar áreas de mejora y definir un enfoque innovador que combina procesamiento de señales y arquitecturas híbridas para clasificación. Gracias a este objetivo se tiene un punto de partida de la tesis y un objetivo final que ha sido cumplido.
- 9) Proponer un sistema que supere el estado del arte:** El sistema desarrollado con *CNN-LSTM* logra una precisión del 80,2%, mejorando los resultados de modelos previos en *BCI* y siendo validado por la comunidad científica. Por otra parte, la combinación de *CNN-LSTM* y capas de atención para el análisis de EEG aporta un avance significativo en la clasificación de intenciones motoras, obteniendo un valor superior al 85% de éxito en la clasificación, con un enfoque generalizado y una robustez que pretende ser plasmada en un artículo que se encuentra bajo desarrollo. Por lo tanto, este objetivo ha sido alcanzado con amplitud.
- 10) Aportar al tejido investigador mediante publicaciones en revistas indexadas:** Este trabajo ha generado múltiples publicaciones en revistas Q1 y Q2, así como capítulos de libro, destacando la aceptación de los resultados en la comunidad científica y subrayando su relevancia en el ámbito de *BCI*. Estos se muestran en la tabla 6.1. y se recogen al completo en los Anexos. **Se muestran los artículos publicados en revistas Q1 en verde, en azul los publicadas en Q2. Los capítulos de libro en amarillo y por último se muestra en naranja los artículos bajo revisión**

En síntesis, esta tesis no solo cumple con sus objetivos, sino que también establece un marco innovador y sólido para el avance de la clasificación de intenciones motoras mediante técnicas de aprendizaje profundo y procesamiento de señales, aportando una herramienta precisa, robusta y validada para futuras aplicaciones en sistemas *BCI*.

Tabla 8.1. Publicaciones en el desarrollo de la tesis doctoral.

Título	Revista/libro	Índice de impacto (2023)	Ámbito	Estado
<i>Study of an Optimization Tool Avoided Bias for Brain-Computer Interfaces Using a Hybrid Deep Learning Model. [19]</i>	<i>IRBM</i>	5,6 (Q1 en «Engineering, Biomedical» 22/123)	<i>BCI</i>	Publicado
<i>ANN based-model for estimating the boron permeability coefficient as boric acid in SWRO desalination plants using ensemble-based machine learning. [164]</i>	<i>Desalination</i>	8,4 (Q1 en «Water Resources» 3/128 y Q1 en «Engineering, Chemical» 13/170)	Series temporales en desalinización	Publicado
<i>Novel cost-effective method for forecasting COVID-19 and hospital occupancy using deep learning. [165]</i>	<i>Scientific Reports</i>	3,8 (Q1 en «Multidisciplinary Science» 25/134)	Series temporales y COVID	Publicado
<i>Sky Image Classification Based on Transfer Learning Approaches [166]</i>	<i>Sensors</i>	3,4 (Q2 en «Chemistry, Analytical» 34/106 y Q2 en «Engineering, Electrical & Electronic» 122/353)	Clasificación de imágenes	Publicado
<i>Analysis of Brain Computer Interface Using Deep and Machine Learning [140]</i>	<i>IntechOpen (capítulo)</i>	-	<i>BCI</i>	Publicado
<i>Analysis of Brain Signals to Forecast Motor Intentions Using Artificial Intelligence [152]</i>	<i>Sustainable Computing (capítulo Springer)</i>	-	<i>BCI</i>	Publicado
<i>A Hybrid Deep Learning Approach with 3D Temporal Data for User-Independent Classification of Imagined Motor Tasks in Brain-Computer Interfaces</i>	<i>Scientific Reports</i>	3,8 (Q1 en «Multidisciplinary Science» 25/134)	<i>BCI</i>	Bajo revisión
<i>Emotions for Everyone: A Low-Cost, High-Accuracy Method for Emotion Classification</i>	<i>Scientific Reports</i>	3,8 (Q1 en «Multidisciplinary Science» 25/134)	<i>Clasificación de emociones</i>	Bajo revisión
<i>Angular contour parametrization for the in-air signature-based identification</i>	<i>IEEE Transactions on Information Forensics and Security</i>	6,3 (Q1 en «Computer Science, Theory & Methods» 13/144 y Q1 en «Engineering, Electrical & Electronic» 42/353)	Reconocimiento de patrones en aire	Bajo revisión

Líneas futuras

Las interfaces cerebro-ordenador presentan un potencial significativo en diversos campos. Los algoritmos de *Deep Learning* son cada vez más avanzados y el *hardware* necesario para acompañar esta revolución está siendo cada vez más tenido en cuenta. Algunas de las líneas futuras en este campo son:

- Mejorar e implementar estos algoritmos para permitir el control de dispositivos externos mediante señales cerebrales. Este campo abre diversas líneas futuras de investigación tanto a nivel algorítmico como en aplicaciones prácticas.

Desde la perspectiva algorítmica, una dirección prometedora es el desarrollo de modelos basados en transformadores híbridos, algoritmos de autoatención o incluso el uso de IA generativa, que pueda autoaprender a partir de las diferentes lecturas que se vayan haciendo de los datos.

La implementación de los *ViT* y modelos similares ya utilizados, podría mejorarse mediante técnicas de optimización de datos y preprocesamiento más detallado, haciendo posible el análisis en tiempo real de las señales EEG con mayor precisión y velocidad. Esto llevaría a la mejora de la calidad de vida de las personas con discapacidades, algún tipo de problema neurodegenerativo o incluso a aplicaciones para rehabilitación, juegos o enseñanza.

- Otra línea de interés es la creación de modelos adaptativos que ajusten su desempeño en función de la actividad cerebral de cada individuo, abordando la variabilidad fisiológica que se observa en la respuesta EEG entre usuarios. Esto implica el desarrollo de sistemas *BCI* personalizados, los cuales podrían mejorar la efectividad y accesibilidad de estas tecnologías en personas con necesidades específicas. Este tipo de modelos personalizados ofrecerían una mayor precisión y reducirían la variabilidad en los resultados de clasificación, siendo especialmente relevantes en aplicaciones de neurorrehabilitación y control de prótesis.
- A nivel macroscópico, las *BCI* ofrecen un amplio abanico de aplicaciones más allá del control de dispositivos, como en el tratamiento de trastornos neurológicos, la

mejora de habilidades cognitivas mediante *neurofeedback* o la interacción avanzada en aplicaciones de realidad virtual.

- Las investigaciones futuras también deberán enfocarse en la seguridad y ética del manejo de datos cerebrales, asegurando la privacidad y el uso responsable de estas tecnologías en aplicaciones comerciales y médicas. Las *BCI* tienen el potencial de ser una herramienta fundamental en el campo de la neurotecnología, y su desarrollo podría revolucionar tanto el campo clínico como la interacción humano-computadora en la sociedad

