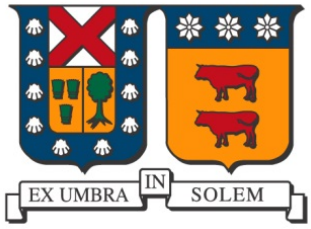
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“COMPARACIÓN DE ALGORITMOS BASADOS EN DEEP-LEARNING PARA CLASIFICACIÓN DE POTENCIALES EVOCADOS P300 EN EEG”

TOMÁS RODRIGO MORALES GARCÍA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Dra. Carolina Saavedra
Profesor Correferente: Dr. Julio Sotelo

Agosto - 2024

DEDICATORIA

A mis queridos padres, Carmen y Rodrigo, quienes siempre confiaron en mí, me formaron con amor y dedicación, y me brindaron su amor incondicional.

A mis hermanas, Daniela y Camila, por su compañía constante y su gran sentido del humor.

A mis abuelos, Enrique, Mari y Miguel, por entregarme su sabiduría, enseñarme el valor del trabajo duro y por alimentarme los domingos con comida casera.

Y a mi pareja, Gabriela, quien me ha acompañado desde el principio de la carrera, brindándome todo su apoyo, amor y alegría, siendo mi fuente de inspiración y motivación. Sin ti, nada de esto seria posible

A todos ustedes, gracias de todo corazón. Este logro es tanto suyo como mío.

RESUMEN

Resumen— Las interfaces cerebro-computador (BCI, en inglés) son sistemas de control y comunicación que buscan traducir actividad cerebral en comandos para controlar diversos dispositivos. La técnica más popular para registrar señales cerebrales es la electroencefalografía (EEG), que permite la detección de potenciales evocados. En los últimos años, se han comenzado a implementar modelos de Deep Learning para la detección de potenciales evocados. Sin embargo, debido a la novedad de estas soluciones, aún no se ha realizado una comparación entre ellas. Por lo tanto, se propone comparar con una metodología estándar, diferentes modelos de Deep Learning propuestos en los últimos años.

Palabras Clave— Interfaces cerebro-computador; P300 Speller; Aprendizaje profundo.

ABSTRACT

***Abstract*—** Brain-computer interfaces (BCI) are control and communication systems that seek to translate brain activity into commands to control various devices. The most popular technique for recording brain signals is electroencephalography (EEG), which allows the detection of evoked potentials. In recent years, Deep Learning models have begun to be implemented for the detection of evoked potentials. However, due to the novelty of these solutions, a comparison between them has not yet been made. Therefore, it is proposed to compare with a standard methodology, different Deep Learning models proposed in the last years.

***Keywords*—** Brain-computer Interface; P300 Speller, Deep Learning.

GLOSARIO

BCI: *Brain-computer Interface*

CNN: *Convolutional Neural Networks*

ELA: Esclerosis Lateral Amiotrófica

EEG: Electroencefalograma

ALSFRS-R Escala revisada de valoración de la Esclerosis Lateral Amiotrofica

ROC: *receiver operating characteristic curve*, curva característica del receptor operativo

AUC: Área bajo la curva

ÍNDICE DE CONTENIDOS

RESUMEN III

ABSTRACT III

GLOSARIO IV

ÍNDICE DE FIGURAS VII

ÍNDICE DE TABLAS VIII

INTRODUCCIÓN **1**

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA **2**
1.1 Contexto . . . 2
1.2 Objetivos . . . 4
1.2.1 Objetivo General . . . 4
1.2.2 Objetivos específicos . . . 4

CAPÍTULO 2: MARCO CONCEPTUAL **5**
2.1 Redes Neuronales Artificiales . . . 5
2.1.1 Neurona Artificial . . . 5
2.1.2 Función de activación . . . 6
2.1.3 Aprendizaje . . . 8
2.2 Redes Neuronales Convolucionales . . . 9
2.2.1 *Depthwise separable convolutions* . . . 11
2.2.2 *Inception* . . . 14
2.3 *Transformers* . . . 15
2.3.1 Autoatención . . . 15
2.4 Preprocesamiento de datos . . . 16
2.4.1 Auto Reject . . . 16
2.4.2 Segmentación y recombinación de señales . . . 18
2.5 Métricas de evaluación . . . 19
2.6 Función de pérdida . . . 21
2.7 Sistemas de posicionamiento de EEG . . . 22

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN **24**
3.1 Modelos . . . 24
3.2 Base de datos . . . 33
3.2.1 *Dataset* I . . . 33
3.2.2 *Dataset* II . . . 34
3.2.3 *Dataset* III . . . 34
3.3 Preprocesamiento . . . 36
3.4 División de datos . . . 37

3.5 Aumentado de datos . . . 38
3.6 Entrenamiento . . . 38
3.6.1 Parámetros de modelos . . . 39
3.6.2 Métricas . . . 40

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN **41**
4.1 Resultados para el Dataset I . . . 42
4.1.1 F1-Score . . . 42
4.1.2 ROC . . . 43
4.1.3 Matriz de confusión . . . 45
4.1.4 Tiempo de Entrenamiento . . . 47
4.2 Resultados para el Dataset II . . . 49
4.2.1 F1-Score . . . 49
4.2.2 ROC . . . 51
4.2.3 Matriz de confusión . . . 53
4.2.4 Tiempo de Entrenamiento . . . 54
4.3 Resultados para el Dataset III . . . 56
4.3.1 F1-Score . . . 56
4.3.2 ROC . . . 58
4.3.3 Matriz de Confusión . . . 60
4.3.4 Tiempo de Entrenamiento . . . 61
4.4 Experimentos Batch size 512 . . . 63
4.4.1 Dataset I . . . 63
4.4.2 Dataset II . . . 65
4.4.3 Dataset III . . . 67

CAPÍTULO 5: CONCLUSIONES **69**
5.1 Conclusiones generales . . . 69
5.2 Cumplimiento de objetivos . . . 70
5.3 Trabajo futuro . . . 71

ANEXOS **73**

6 *F1-score* por Dataset **73**

7 *ROC* por Dataset **74**

8 Diferencia porcentual de métricas **76**

REFERENCIAS BIBLIOGRÁFICAS **77**

ÍNDICE DE FIGURAS

1 Matriz de caracteres P300 *Speller* . . . 3
2 Modelo de neurona . . . 6
3 Gráfico comparando las funciones de activación Sigmoidal, Tanh y ReLU. . . . 8
4 Filtro 3x3 de una capa convolucional actuando sobre un dato. . . . 10
5 Capa de *Max Pooling* 2x2 con *stride* = 2. . . . 11
6 Esquema de red CNN con capas convolucionales MNIST . . . 12
7 *Depthwise convolution* . . . 12
8 *Pointwise convolution* . . . 13
9 Modulo *Inception* . . . 14
10 Módulos de atención y atención múltiple. . . . 16
11 Ejemplo Auto Reject . . . 18
12 Segmentación y recombinación de señales . . . 19
13 Matriz de confusión . . . 20
14 Sistema 10-20 . . . 22
15 Posición de electrodos en el sistema 10-10 . . . 23
16 Arquitectura *Shallow Net* para una tarea de imaginación motora. . . . 25
17 Arquitectura EEGNet. . . . 28
18 Arquitectura *Inception*. . . . 29
19 Arquitectura Conformer. . . . 31
20 *Accuracy* para EEG Conformer con distintas configuraciones . . . 32
21 Posición de electrodos en el sistema 10-20 para el *dataset* III. . . . 35
22 Interfaz gráfica de Brain Invaders . . . 35
23 *Autoreject* aplicado . . . 37
24 Flujo de datos simplificado. . . . 39
25 F1-score para *dataset* I . . . 43
26 Curva ROC para *dataset* I . . . 44
27 Matriz de confusión para el *dataset* I . . . 47
28 Tiempo de entrenamiento para *dataset* I . . . 48
29 F1-score para *dataset* II . . . 50
30 Curva ROC para *dataset* II . . . 52
31 Matriz de confusión para el *dataset* II . . . 54
32 Tiempo de entrenamiento para *dataset* II . . . 55
33 F1-score para *dataset* III . . . 57
34 Curva ROC para dataset III . . . 59
35 Matriz de confusión para el *dataset* III . . . 61
36 Tiempo de entrenamiento para *dataset* III . . . 62
37 F1-Score para *batch size* 512 en Dataset I . . . 64
38 AUC para *batch size* 512 en Dataset I . . . 65
39 F1-Score para *batch size* 512 en Dataset II . . . 66
40 AUC para *batch size* 512 en Dataset II . . . 66
41 F1-Score para *batch size* 512 en Dataset III . . . 67
42 AUC para *batch size* 512 en Dataset III . . . 68

43 F1-score para sujetos en dataset I . . . 73
44 F1-score para sujetos en dataset II . . . 73
45 F1-score para sujetos en dataset III . . . 74
46 AUC para sujetos en dataset I . . . 74
47 AUC para sujetos en dataset II . . . 75
48 AUC para sujetos en dataset III . . . 75

ÍNDICE DE TABLAS

1 Parámetros del modelo Shallow Net . . . 26
2 Parámetros entrenables ShallowNet y EEGNet . . . 27
3 Parámetros del modelo EEGNet . . . 27
4 Parámetros del modelo EEG-Inception . . . 30
5 Arquitectura del módulo de convolución . . . 30
6 Parámetros del modelo EEG Conformer . . . 33
7 Información demográfica de los participantes, incluyendo ALSFRS-R[1] . . . 34
8 Parámetros utilizados para *Autoreject* . . . 36
9 Datos de entrenamiento de cada *dataset*, junto a la comparación antes y después de aplicado el balanceo . . . 38
10 Descripción de *datasets* . . . 41
11 F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* I . . . 43
12 AUC promedio para diferentes modelos y metodologías de entrenamiento en el *Dataset* I . . . 45
13 Numero de parámetros entrenables por modelo . . . 48
14 Tiempo de ejecución por época, promediando los sujetos, para diferentes modelos y metodologías de entrenamiento (en segundos) . . . 49
15 Comparación de datos de entrenamiento normales versus limpios Dataset II . 51
16 F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* II . . . 51
17 AUC promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* II . . . 53
18 Diferencia porcentual en el tiempo de entrenamiento por época . . . 55
19 F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* III . . . 57
20 AUC promedio para diferentes modelos y metodologías de entrenamiento en el dataset III . . . 58
21 Tiempo de ejecución por época, promediando los sujetos, para diferentes modelos y metodologías de entrenamiento (en segundos) en el *dataset* III . . 62
22 Diferencia porcentual de *F1-Score* entre método de entrenamiento normal y el resto. Una diferencia positiva indica una mejora respecto al método normal 76
23 Diferencia porcentual de AUC entre método de entrenamiento normal y el resto. Una diferencia positiva indica una mejora respecto al método normal . 76

INTRODUCCIÓN

Las interfaces Cerebro-Computador (BCI) permiten la interacción y comunicación entre humanos y su entorno, utilizando señales cerebrales extraídas con un método no invasivo, llamado electroencefalograma (EEG). Este método de comunicación, al no utilizar nervios o músculos periféricos, es altamente beneficioso para personas con graves dificultades motoras, tales como personas con Esclerosis Lateral Amiotrófica (ELA).

La comunicación es una necesidad fundamental para los seres humanos, pero algunas personas, debido a condiciones médicas como la ELA o el síndrome de enclaustramiento, no pueden interactuar eficazmente con su entorno. El potencial evocado P300 es el más utilizado en BCI para traducir señales cerebrales en acciones, como escribir utilizando el P300 *Speller*, un sistema que permite seleccionar letras mediante la detección de un potencial eléctrico generado al prestar atención a estímulos visuales específicos. El problema que presenta la detección del potencial P300 es el bajo cociente de señal a ruido. Es por esto que se utilizan modelos de aprendizaje automático, y recientemente, aprendizaje profundo, para detectar dichos potenciales. A pesar de esto, no existe una comparación completa de algoritmos de aprendizaje profundo para la decodificación de señales en el contexto del P300 *Speller*.

En la presente memoria, se propone una metodología de comparación, la cual busca examinar de manera generalizada cuatro modelos de aprendizaje profundo para el problema de P300 *Speller*, con cuatro métodos de entrenamientos distintos (estándar, datos limpios, datos balanceados y datos limpios y balanceados), entregando así un completo análisis de cada modelo y sus respectivos resultados. Para aquello, se aplicó la metodología en 3 conjuntos de datos heterogéneos, con un total de 43 sujetos de distintas edades, sexo, estado de salud y conocimiento de sistemas BCI, para los cuales se analizaron los resultados obtenidos utilizando cada modelo con los cuatro métodos de entrenamiento disponibles.

Las 5 secciones de este trabajo se estructuran de la siguiente manera: En la sección 1, definición del problema, se introducen los objetivos del texto y el contexto del cual surge la problemática a resolver; La sección 2, marco conceptual define y explica todos los conceptos claves, necesarios para llevar a cabo la aplicación de la metodología que se propone, tales como las arquitecturas utilizadas por los modelos y técnicas de preprocesamiento; En la sección 3, propuesta de solución, se precisa la metodología utilizada para la comparación de modelos con sus respectivos métodos de entrenamiento; En la sección 4, validación de la solución, se presentan los resultados obtenidos luego de utilizar la metodología anteriormente descrita, junto con observaciones obtenidas en base a dichos resultados; Finalmente, en la sección 5, conclusión, se concluye y profundiza sobre los resultados obtenidos, y se presentan trabajos futuros para profundizar la investigación introducida en esta memoria.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Contexto

La comunicación es una necesidad fundamental para los seres humanos, ya que posibilita el intercambio de ideas, la resolución de conflictos, la expresión libre y una multitud de beneficios adicionales. Sin embargo, debido a circunstancias ajenas a su control, un grupo de personas carece de la capacidad de interactuar con su entorno de manera eficaz. Este grupo incluye a pacientes que padecen condiciones médicas debilitantes como la Esclerosis Lateral Amiotrófica (ELA), parálisis cerebral y otras discapacidades neuromusculares o físicas que derivan en el síndrome de enclaustramiento o *locked-in syndrome*.

Las interfaces cerebro-computador (*BCI*, por sus siglas en inglés) tienen la tarea de inferir acciones basadas en las señales cerebrales detectadas en intervalos específicos de tiempo. [Farwell y Donchin, 1988] sentaron las bases contemporáneas para el diseño de estas interfaces utilizando el paradigma *Oddball*. En este paradigma, el usuario es expuesto a estímulos visuales diseñados para evocar potenciales relacionados con un evento (*ERP, por sus siglas en inglés*), los cuales son detectados utilizando un electroencefalograma (EEG). El EEG es una prueba que evalúa la actividad eléctrica de las neuronas situadas en la corteza cerebral, mediante el uso de electrodos colocados en el cuero cabelludo del paciente. Desde su inicio, el EEG ha sido el método no invasivo más popular debido a su seguridad y bajo costo [Liu *et al.*, 2018]. Entre los ERPs obtenidos, el potencial evocado P300 ha demostrado ser el más consistente y estudiado, siendo utilizado en *BCIs* para traducir este potencial en acciones tangibles, como escribir una frase [Farwell y Donchin, 1988] o controlar una silla de ruedas [Yu *et al.*, 2017].

El potencial evocado P300 es una señal eléctrica captada mediante EEG que se produce como reacción a un estímulo inusual en una serie de estímulos repetitivos y predecibles. Dicho potencial se detecta aproximadamente 300 milisegundos después de la exposición al estímulo; sin embargo, esto puede depender de la tarea cognitiva a realizar, además de la variabilidad natural entre las personas. La amplitud de la señal también varía según la importancia o novedad del estímulo, lo que complica la detección de este potencial.

De las acciones que se pueden lograr con un *BCI*, nos centraremos en el P300 *Speller* [Farwell y Donchin, 1988], que consta de una matriz de 6x6 que contiene letras y otros caracteres, como se observa en la figura 1, cuyas filas y columnas se iluminan de manera aleatoria. Cuando el usuario del *BCI* presta atención a una letra del monitor y esta se ilumina, se detecta una diferencia de voltaje positivo en el cerebro aproximadamente 300 milisegundos después del estímulo iluminado, de ahí el nombre P300.

A pesar de que el potencial evocado P300 es la señal más utilizada en las soluciones actuales,



Figura 1: Matriz de caracteres P300 *Speller*
Fuente: [Farwell y Donchin, 1988].

no está exento de problemas, ya que presenta un bajo cociente señal-ruido. Esto requiere la repetición de los estímulos en múltiples ocasiones para reducir el ruido ambiental en las señales. Esta repetición de estímulos disminuye la velocidad de las acciones que el usuario del *BCI* intenta llevar a cabo, por lo que es imperativo buscar soluciones que logren resultados precisos con un menor número de repeticiones.

El uso de métodos clásicos de aprendizaje automático tales como *stepwise linear discriminate analysis*, *support vector machines* y *matched filtering* han sido ampliamente investigados y utilizados para la detección de señales P300, obteniendo resultados muy positivos [Krusienski *et al.*, 2006]. Un problema que presenta el uso de *machine learning* es el preprocesamiento de datos, que usualmente requiere conocimiento del dominio; esto eleva la complejidad de las soluciones propuestas. Por esta razón, durante la última década, se han implementado distintos algoritmos de detección de potenciales P300 usando *Deep Learning* como base. El beneficio de estos métodos es que son capaces de utilizar datos sin previo preprocesamiento de dominio, encontrando complejos patrones entre atributos sin mayor intervención de expertos del área [Roy *et al.*, 2019].

Si bien los resultados de algoritmos de *Deep Learning* aplicados a la detección de señales P300 son prometedores, no hay estudio que compare las soluciones propuestas durante los últimos años, y es precisamente eso lo que busca este trabajo, aplicar un marco de comparación justo que ayude al personal del área de la salud a cargo de sistemas *BCI*, a decidir qué modelo elegir en base a las necesidades que presentan los pacientes.

1.2. Objetivos

1.2.1. Objetivo General

Comparar bajo las mismas condiciones distintos algoritmos basados en *Deep Learning* para clasificación/detección de potenciales evocados P300 utilizados en una interfaz cerebro-computador (*BCI*) en el contexto del problema P300 *Speller*.

1.2.2. Objetivos específicos

- Estudiar el estado del arte para modelos de *Deep Learning* aplicados para la detección de potenciales evocados P300.
- Definir y diseñar una metodología estándar para la justa comparación entre los modelos estudiados.
- Evaluar cada modelo en base a la metodología y obtener conclusiones respecto a sus resultados.
- Identificar elementos relevantes que asocien la arquitectura de un modelo y sus resultados.

CAPÍTULO 2
MARCO CONCEPTUAL

2.1. Redes Neuronales Artificiales

Las redes neuronales artificiales son modelos matemáticos y computacionales que se inspiran fuertemente en el funcionamiento del sistema nervioso. Estos modelos son capaces de realizar tareas que, aunque son fácilmente ejecutadas por el cerebro humano, resultan difíciles de explicar de manera algorítmica, como la categorización de objetos y la toma de decisiones basadas en información visual. Este logro se materializa a través del aprendizaje automático, donde un modelo se entrena con un objetivo claro y una entrada de información definida. Durante el proceso de entrenamiento, el modelo toma la información de entrada, la procesa y genera una respuesta. Según el resultado obtenido, el modelo ajusta sus parámetros con el fin de acercarse a su punto óptimo, en donde minimiza el error de una función denominada *Loss*.

2.1.1. Neurona Artificial

La unidad fundamental de procesamiento en las redes neuronales es la neurona, cuyo modelo matemático fue originalmente definido por McCulloch y Pitts en su trabajo pionero de 1943 [McCulloch y Pitts, 1943]. En dicho documento, se propuso una neurona de tipo “todo o nada”, caracterizada por un sistema de activación binario que simplifica el comportamiento de una neurona real. A pesar de su aparente simpleza, se teorizaba en el documento que una red suficientemente compleja y densa de estas neuronas podría aproximarse a cualquier función computable.

El modelo de la neurona es el siguiente:

$$N(X, W, b) = \phi\left(\sum_{i=1}^{n} X_i * W_i - b\right)$$

- X_i: Vector de entrada.
- W_i: Peso de la entrada i, determina cuánta importancia va a tener la entrada sobre el resultado final.
- b: Umbral de activación.
- $\phi(\cdot)$: Función de activación.

Como se puede observar en la Figura 2, la ejecución de una neurona se puede resumir como:

1. Recibe un vector de datos.
2. Multiplica cada elemento por su respectivo peso y agrega el sesgo (b).
3. Suma cada elemento y entrega el resultado a la función de activación.
4. Toma el resultado y lo transmite a la siguiente neurona.

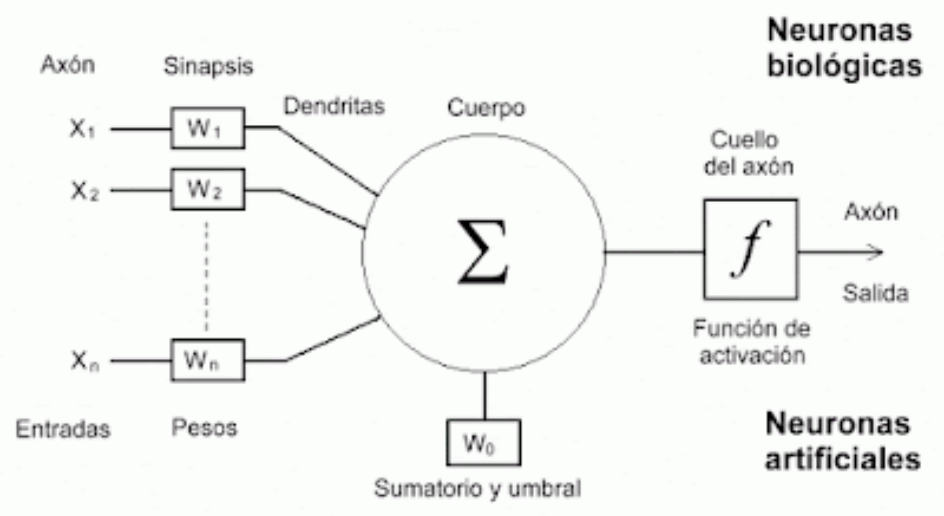


Figura 2: Modelo de neurona como compuerta binaria propuesto por [McCulloch y Pitts, 1943].

Fuente: www.internsofdata.com

2.1.2. Función de activación

La función de activación desempeña un papel fundamental en el proceso de aprendizaje de una red neuronal, permitiéndole capturar patrones complejos en los datos y determinar el valor final de una neurona al introducir no linealidades en la red. Estas funciones posibilitan la extracción de representaciones más ricas y abstractas, mejorando la capacidad de la red para comprender características complejas en los datos.

Estas funciones son vitales para prevenir problemas inherentes a las redes neuronales y a la matemática computacional, como el gradiente desvaneciente [Bishop y Nasrabadi, 2006]. Este fenómeno surge cuando los gradientes disminuyen de manera exponencial durante el entrenamiento, lo que puede dificultar el aprendizaje efectivo del modelo.

Las funciones más comúnmente usadas son las siguientes:

- **Sigmoidal**: Normaliza el valor de cada neurona, manteniéndolo entre 0 y 1. Se utiliza comúnmente para predecir probabilidades en modelos de clasificación.

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

  - **Desventajas:** Propenso al problema del gradiente desvaneciente. Cuando el valor a activar (x) es muy alto o muy bajo, la derivada calculada se vuelve muy pequeña, cercana a 0, dificultando el aprendizaje óptimo del modelo.

- **Tangente Hiperbólica**: Similar a la función sigmoidal, pero con un rango de valores entre -1 y 1, permitiendo que los valores negativos sean negativos y los cercanos a 0 se mantengan cercanos a 0.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

  - **Desventajas:** Aunque aborda la limitación de la función sigmoidal, aún presenta posibilidades de desvanecimiento de gradiente.

- **ReLU**: Actualmente, es la función de activación más utilizada [Rasamoelina *et al.*, 2020], con un rango entre 0 e infinito. Destaca por su manejo eficiente del problema del gradiente desvaneciente cuando el valor es positivo y por su alta velocidad de cómputo en comparación con otras funciones.

$$\phi(x) = \text{máx}(0, x)$$

  - **Desventajas:** Cuando el valor de una neurona es negativo para todas las entradas, se desactiva, lo que limita su capacidad para contribuir al aprendizaje del modelo.

- **ELU**: La función de activación *Exponential Linear Unit* (ELU) es una alternativa a ReLU que intenta mitigar algunos de sus problemas, especialmente la "muerte" de neuronas. A diferencia de ReLU, ELU puede generar valores negativos, lo que permite que las neuronas sigan contribuyendo al aprendizaje incluso cuando su entrada es negativa. ELU tiene un rango entre $-\alpha$ e infinito y una suave transición entre los valores negativos y positivos, lo que ayuda a acelerar el entrenamiento y a mejorar la robustez del modelo.

$$\phi(x) = \begin{cases} x, & \text{si } x > 0 \\ \alpha(e^{x} - 1), & \text{si } x \leq 0 \end{cases}$$

  - **Ventajas:** ELU introduce una pequeña pendiente negativa en lugar de anular completamente las entradas negativas, lo que ayuda a evitar la desactivación completa de las neuronas y mejora la capacidad de generalización del modelo.

Cada función de activación tiene su aplicación específica, siendo crucial encontrar aquella que se adapte mejor al modelo. Desde la introducción de ReLU en [Agarap, 2018], es frecuente encontrarse con esta función, o su variante ELU, en las capas ocultas. En contraste, funciones como la sigmoidal son comúnmente empleadas en la última capa para obtener resultados en problemas de clasificación binaria. En la Figura 3 se aprecia la comparación entre las tres funciones.

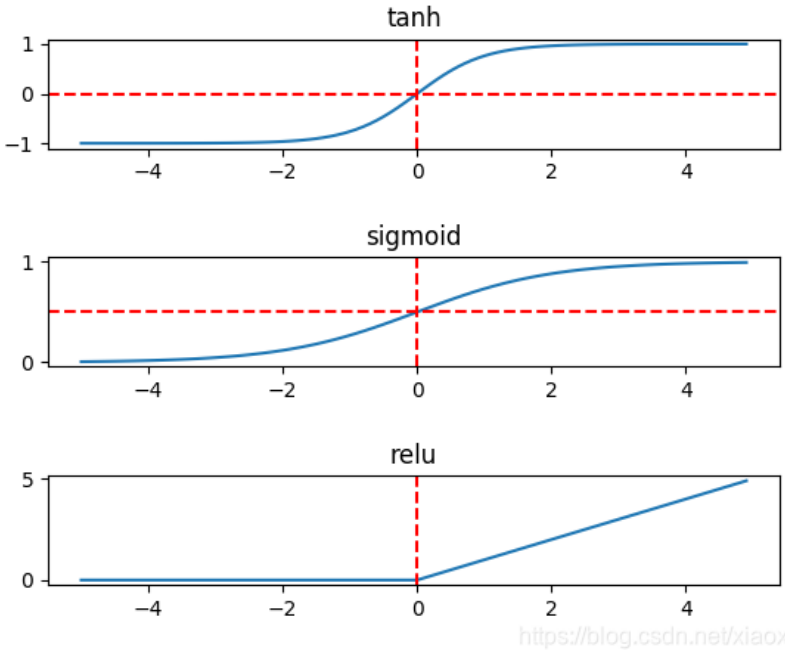


Figura 3: Gráfico comparando las funciones de activación Sigmoidal, Tanh y ReLU.
Fuente: www.ResearchGate.net

2.1.3. Aprendizaje

En este proceso, el objetivo es que el modelo ajuste sus parámetros, como pesos y sesgos, durante el entrenamiento para minimizar una función de pérdida seleccionada previamente. Este ajuste progresivo busca obtener un modelo capaz de resolver la tarea propuesta con un alto nivel de precisión.

Antes de iniciar el entrenamiento y aprendizaje del modelo, es crucial definir sus características clave, como la arquitectura a utilizar, el número de capas y neuronas, y las funciones de activación y pérdida, entre otros. Este paso, conocido como selección y optimización del modelo, se realiza de la mano con el aprendizaje. Durante el proceso de entrenamiento, se evalúan diversas configuraciones del modelo, eligiendo aquella que mejor se adapte a la situación según el rendimiento obtenido.

Estos son los pasos generales del algoritmo de aprendizaje de una red neuronal artificial:

1. **Definición de arquitectura**: Previo al entrenamiento, se debe definir la arquitectura de la red, junto con sus hiperparámetros y funciones de activación a utilizar.
2. **Inicialización**: En este paso, asignamos valores iniciales a los pesos y sesgos de cada neurona en la red. Distintos métodos existen para asignar dichos valores, tales como la inicialización aleatoria o de Xavier [Boulila *et al.*, 2024]. Una inicialización adecuada

facilita el aprendizaje óptimo y la convergencia en un tiempo adecuado. Por otro lado, una inicialización incorrecta puede llevar a problemas como la explosión o desvanecimiento del gradiente, lo que provoca que la red diverja o converja antes de llegar al mínimo, respectivamente [Thimm y Fiesler, 1995].

3. ***Forward propagation***: Se propagan los datos hacia adelante en la red, utilizando los pesos, sesgos y funciones de activación previamente definidos para calcular el resultado y enviarlo a la siguiente capa.

4. **Cálculo de función de pérdida**: Al llegar a la capa de salida, comparamos las etiquetas obtenidas de la predicción con las reales para calcular el resultado de la función de pérdida.

5. ***Backpropagation***: Introducido por [Rumelhart *et al.*, 1986], el *backpropagation* ajusta los valores de los pesos y sesgos para minimizar el valor de la función de costo, con un costo computacional relativamente bajo.

6. **Iteración**: Finalmente, repetimos el *Forward propagation*, el cálculo de la función de pérdida y el *backpropagation* según el número de épocas seleccionado.

2.2. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son poderosas herramientas para el análisis de datos de alta dimensionalidad, utilizadas en aplicaciones como visión de computadora, análisis de video y clasificación de imágenes. Al procesar datos complejos, como una imagen a color, la base convolucional del modelo aprende a captar dependencias espaciales y temporales cada vez más intrincadas, con el objetivo de comprender la estructura y significado de las características obtenidas, logrando así la óptima representación del dato para que el clasificador se acerque al resultado óptimo.

En la sección convolucional, los elementos fundamentales para capturar dependencias multidimensionales son los filtros o kernels. Estas matrices se deslizan sobre el dato aplicando transformaciones numéricas para detectar patrones y características locales como se observa en la Figura 4. Dependiendo del tamaño del filtro, *stride*[2] y la cantidad de filtros, también contribuyen a la reducción de dimensionalidad, condensando la información de la matriz a un solo valor.

[2]El *stride* (paso) en una operación de convolución se refiere al número de píxeles que el filtro se mueve en cada paso mientras se desliza a lo largo de la entrada. Un *stride* mayor reduce la resolución espacial de la salida.

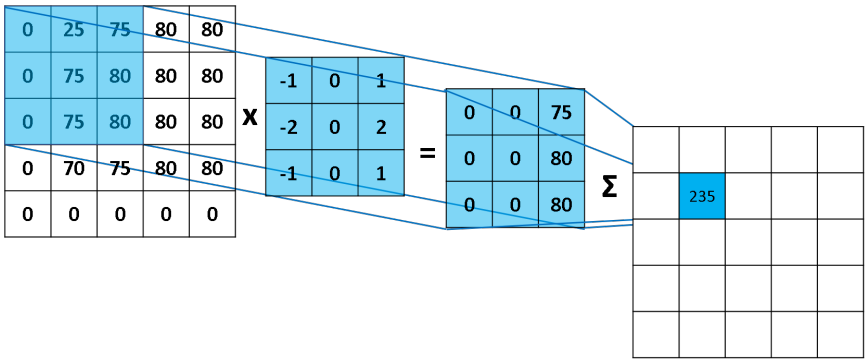


Figura 4: Filtro 3x3 de una capa convolucional actuando sobre un dato.
Fuente: mlnotebook.github.io/post/CNN1

En la estructura de una CNN, los bloques convolucionales desempeñan un papel fundamental. Estos bloques permiten la conexión secuencial de diversas capas, entre ellas la capa convolucional mencionada previamente, la cual aplica filtros a los datos de entrada. Además, estos bloques incluyen capas de regularización y normalización, las cuales serán explicadas a continuación:

1. ***Pooling***: Las capas de *Pooling* están específicamente diseñadas para reducir la dimensionalidad de los datos y disminuir el *sobreajuste*[3] y costo computacional del modelo [Zafar *et al.*, 2022]. Similar al filtro, esta capa recorre el dato y elige el máximo valor o el valor promedio, dependiendo de la modalidad de la capa de *Pooling*, conservando las características más importantes y reduciendo la cantidad de datos procesados. En la Figura 5 se observa la aplicación de *max pooling* 2x2 con paso de 2 unidades en una matriz de 4x4, obteniendo como resultado una matriz de 2x2 con los máximos valores de cada cuadrante.

2. ***Normalización por lotes***: La normalización por lotes (*Batch Normalization* en inglés) es una técnica diseñada para mejorar la estabilidad y eficiencia del entrenamiento en redes neuronales profundas. Consiste en normalizar la salida de cada capa oculta mediante el ajuste de sus activaciones para lograr una media cercana a cero y una varianza cercana a uno. Esto ayuda a mitigar el problema del cambio de covariable interna durante el entrenamiento, lo que puede generar que las capas posteriores sean más difíciles de entrenar, mejorando así la capacidad de generalización del modelo [Ioffe y Szegedy, 2015].

[3]El sobreajuste de modelos de inteligencia artificial ocurre cuando este aprende demasiado bien los datos de entrenamiento, al punto de tratar ruido y fluctuaciones aleatorias como patrones reales, lo que lleva a una pobre generalización.

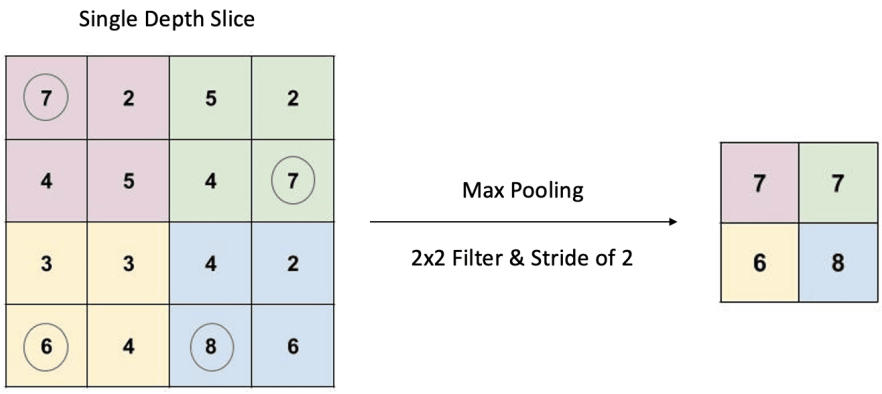


Figura 5: Capa de *Max Pooling* 2x2 con *stride* = 2.
Fuente: www.learnopencv.com

3. ***Batch Dropout***: La eliminación aleatoria por lotes (*Batch Dropout* en inglés) es una técnica utilizada para prevenir el sobreajuste y mejorar la generalización en modelos de redes neuronales. Consiste en apagar aleatoriamente un cierto porcentaje de neuronas durante el entrenamiento, lo que provoca que las neuronas restantes se vuelvan más robustas e independientes entre sí. Esto evita que las neuronas dependan en exceso unas de otras y promueve una representación más diversa y robusta de los datos en la red [Salehin y Kang, 2023].

Una vez que se extraen las características mediante los bloques convolucionales, estas se transforman en un vector unidimensional. Este paso es crucial, ya que permite que una red *Feed Forward* aprenda combinaciones no lineales de los patrones identificados, generando así la respuesta final. Este proceso se ilustra en la Figura 6, que muestra el flujo de datos desde la entrada de una imagen hasta la entrega de la respuesta, en este caso, la clasificación.

2.2.1. *Depthwise separable convolutions*

Originalmente introducidas en [Sifre y Mallat, 2014], las convoluciones por profundidad separables, o *Depthwise separable convolutions*, tienen como objetivo reducir el alto costo computacional de las convoluciones 2D. Además, permiten aprender de manera independiente representaciones espaciales y temporales, logrando así una representación más significativa de los datos y características.

Las convoluciones por profundidad dividen sus operaciones en dos secciones:

1. ***Depthwise convolution***: Se aplica un filtro por canal, a diferencia de las convoluciones 2D, en donde se elige un número N de filtros, que se aplican a todos los canales dis-

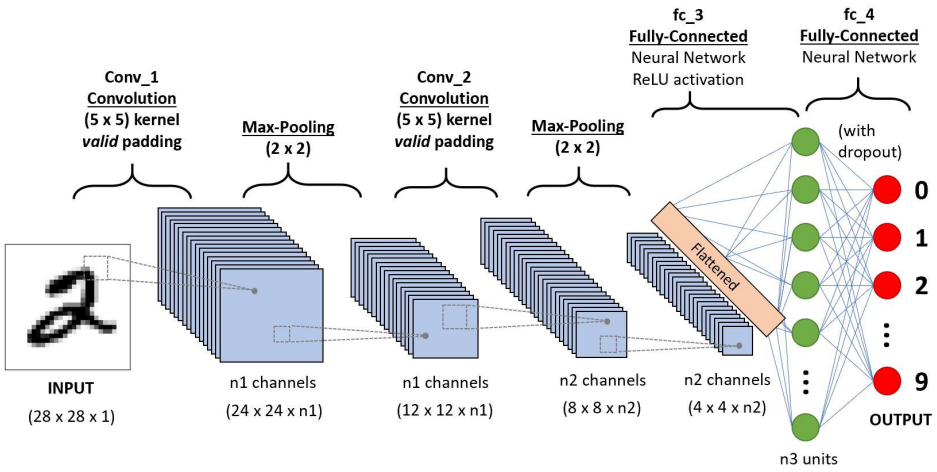


Figura 6: Esquema de red CNN con capas convolucionales, *max pooling* y red de clasificación para la base de datos MNIST, que consta de imágenes de dígitos de 0 a 9 escritos a mano.
Fuente: www.analyticsvidhya.com

ponibles. Como resultado, se obtiene un mapa de características intermedio por cada canal.

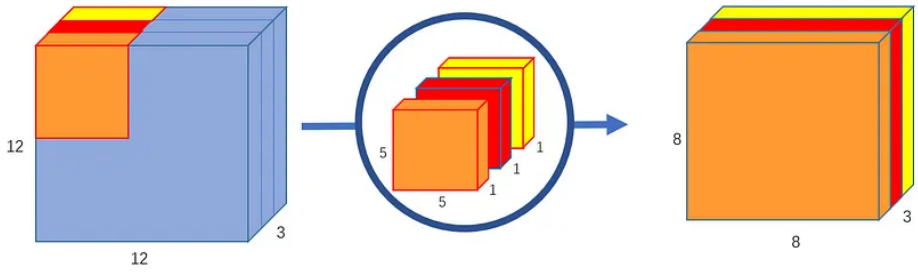


Figura 7: *Depthwise convolution* sobre una imagen de dimensiones 12×12 con 3 canales, en donde se aplica un filtro 5×5 a cada canal, resultando en un mapa de características intermedio de tamaño $8 \times 8 \times 3$.
Fuente: wwwtowardsdatascience.com

2. ***Pointwise convolution***: Una vez obtenidos los mapas de características de cada canal, se aplica una convolución 1x1 para combinar dichos mapas y obtener el output final de la convolución por profundidad.

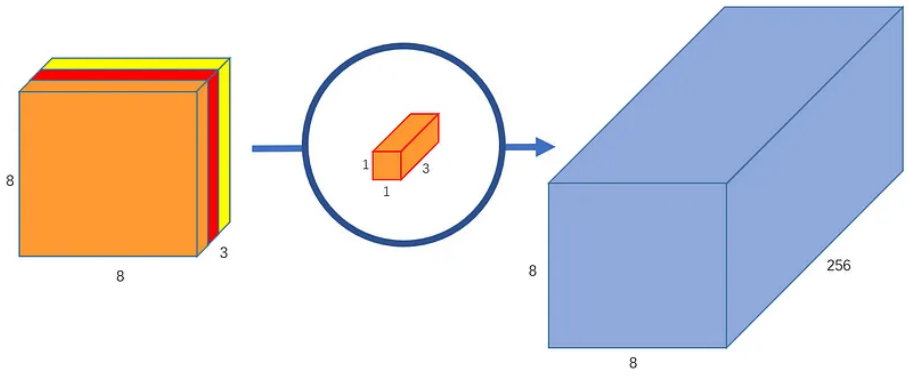


Figura 8: *Pointwise convolution* aplicada sobre un mapa de características intermedio, en donde 256 filtros de 1×1 convolucionan dichos mapas para crear los 256 mapas de características finales.

Fuente: wwwtowardsdatascience.com

Costo computacional

El número de operaciones de una capa de convolución 2D, asumiendo *stride* igual a 1, y *padding*[4] es de:

$$D_K \times D_K \times M \times N \times D_F \times D_F$$

Donde $D_K \times D_K$ es el tamaño del filtro, $D_F \times D_F$ es el tamaño del mapa de características, M es el número de canales de *input* y N es el número de canales de *output* [Howard *et al.*, 2017].

Tomando en cuenta los ejemplos propuestos en 7 y 8, con valores de $D_K = 5$, $M = 3$, $N = 256$ y $D_F = 8$, el costo de la operación es:

$$5 \times 5 \times 3 \times 8 \times 8 = 1228800 \quad (1)$$

Por otro lado, el número de operaciones de *Depthwise convolution*, asumiendo que hay un filtro por canal, es:

$$D_K \times D_K \times M \times D_F \times D_F$$

Este paso solo calcula el mapa de características para cada canal. Ahora debemos combinar dichos mapas para generar nuevas características, tarea que realiza la *Pointwise convolution*, cuyo número de operaciones es:

$$M \times N \times D_F \times D_F$$

[4]El *padding* (relleno) es una técnica que permite controlar el tamaño del mapa de características después de una convolución. Consiste en agregar píxeles alrededor de los bordes de la imagen para evitar pérdida de información

Resultando en el costo final de:

$$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F$$

Utilizando los valores anteriormente descritos, el costo es de:

$$5 \times 5 \times 3 \times 8 \times 8 + 3 \times 256 \times 8 \times 8 = 53952$$

Al expresar la convolución como un proceso de dos partes, primero aplicando filtros y luego combinando, obtenemos una reducción del número de operaciones de:

$$\frac{1}{N} + \frac{1}{D_K^2}$$

2.2.2. *Inception*

Las capas *Inception*, propuestas y popularizadas por [Szegedy *et al.*, 2015], buscan resolver el problema de la variabilidad de tamaño en los objetos y patrones a detectar en los datos. Este problema radica en la necesidad de determinar un tamaño fijo de filtro o *kernels* en una capa de convolución. Lo que se propone en [Szegedy *et al.*, 2015] es utilizar filtros de múltiples tamaños en la misma capa convolucional, para captar tanto patrones globales con los filtros de mayor tamaño como patrones locales con los filtros pequeños, como se aprecia en la Figura 9.

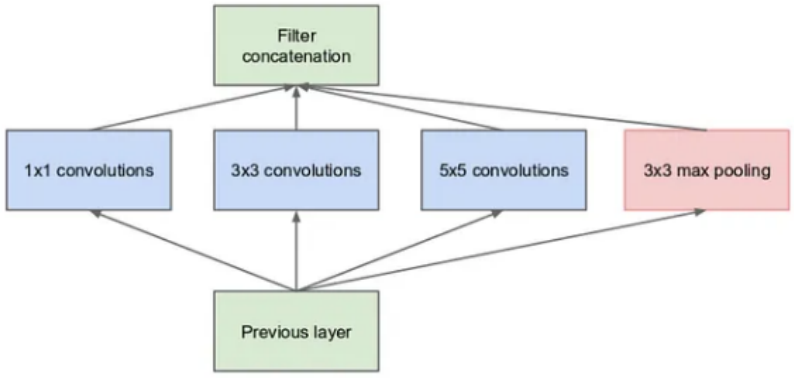


Figura 9: Módulo de *Inception* con 3 filtros convolucionales de distintos tamaños y una capa *max pooling*.

Fuente: www.towardsdatascience.com

2.3. *Transformers*

El *Transformer*, introducido en [Vaswani *et al.*, 2017], es una arquitectura de aprendizaje profundo enfocada en aprender contexto y significado de datos secuenciales tales como texto. La innovación de esta arquitectura está en el uso de módulos de autoatención, los cuales capturan dependencias globales en grandes secuencias de datos.

2.3.1. Autoatención

Este mecanismo permite al modelo entregar pesos a segmentos del mismo dato para determinar su importancia en el contexto actual. Cada segmento puede ponerle atención a todo otro segmento del dato, captando dependencias y relaciones a largo plazo.

Hay tres componentes principales de la autoatención:

1. *Query* (Consulta): Para cada elemento (token) en los datos de entrada, se calcula un vector de consulta (Q). Este vector indica qué partes de la secuencia son relevantes para ese elemento en particular. Esencialmente, representa las relaciones que el token debe considerar con otros elementos de la secuencia.
2. *Key* (Clave): Asimismo, se calculan vectores clave (K) para cada token en la secuencia. Estos vectores ayudan a identificar los elementos importantes o significativos dentro de la secuencia. Son esenciales para determinar la relevancia y la importancia relativa de cada token en relación con los demás.
3. *Value* (Valor): Los vectores de valor (V) complementan el proceso. Proporcionan información sobre qué características o aspectos específicos del token se deben tener en cuenta al evaluar su importancia en la secuencia.

Cada token calcula los vectores Q, K y V, que se utilizan para obtener el *attention score* entre todos los pares de tokens disponibles. Este *attention score* cuantifica la compatibilidad o relevancia entre los tokens. Finalmente, los pesos de atención se utilizan para obtener la representación contextualizada de los datos de entrada.

La incorporación de la atención múltiple permite al modelo dirigir su atención hacia múltiples fuentes de información simultáneamente, enriqueciendo su capacidad para capturar relaciones complejas y contextos más amplios en los datos, como se ve en la Figura 10.

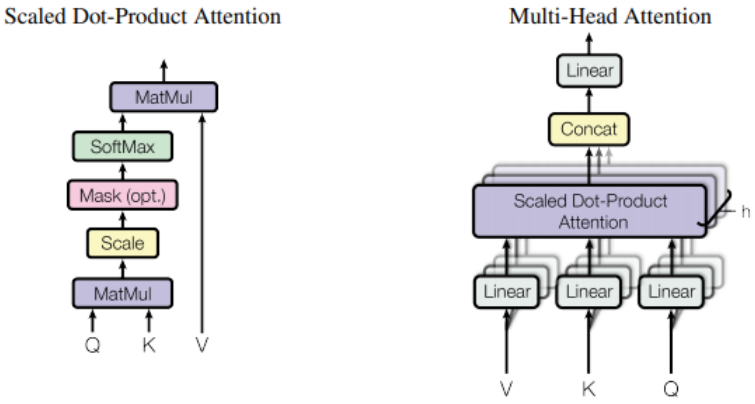


Figura 10: Módulos de atención y atención múltiple.
Fuente: www.vaclavkosar.com

2.4. Preprocesamiento de datos

2.4.1. Auto Reject

Los datos de EEG obtenidos de forma no invasiva, como en el caso de las bases de datos utilizadas, suelen estar afectados por artefactos causados por movimientos oculares, actividad cardíaca, actividad muscular, entre otros, tal como se describe en [Jiang *et al.*, 2019]. Antes de la disponibilidad de bases de datos públicas con datos de EEG, que solían tener un número muy limitado de pacientes, filtrar manualmente canales y épocas para los experimentos no representaba un gran problema. Sin embargo, en los últimos años, la necesidad de grandes volúmenes de datos para entrenar modelos de aprendizaje automático ha generado la necesidad de automatizar el proceso de filtrado y reparación de datos.

Auto Reject es un algoritmo introducido por [Jas *et al.*, 2017] diseñado para limpiar y gestionar artefactos en datos EEG. Su objetivo principal es eliminar o reparar canales o señales; esto se logra encontrando el umbral de amplitud óptimo ($\tau_j^\star$) para cada canal j, minimizando la diferencia entre los conjuntos de datos de entrenamiento y validación.

Para determinar este umbral τ_j, se emplea el método de validación cruzada con F segmentos, donde se establece una lista de umbrales candidatos automáticamente, los cuales son aplicados al conjunto de entrenamiento actual, canal por canal y época por época. Posteriormente, se calcula el valor promedio de las señales que caen bajo el umbral establecido, y se compara con la media de todas las señales del conjunto de validación. Esto garantiza la selección del umbral que minimice el error cuadrático medio (RMSE), como se observa en la ecuación 2.

$$\tau_j^\star = \arg \min_{\tau_j \in [\tau_{j_{\text{mín}}}, \tau_{j_{\text{máx}}}]} \frac{1}{F} \sum_{k=1}^{F} e_F(\tau_j) \tag{2}$$

Una vez obtenidos los umbrales óptimos para los canales, se utiliza la siguiente ecuación 3 para identificar los ejemplos defectuosos C.

$$C_{ij} = \begin{cases} 0 & \text{si } A_{ij} \leq \tau_j \\ 1 & \text{si } A_{ij} > \tau_j \end{cases} \tag{3}$$

En donde C_{ij} denota la cantidad de ejemplos defectuosos para el canal j y época i, y A_{ij} corresponde a la amplitud de señal en dicho canal y época.

Una innovación introducida en Auto Reject es su capacidad para interpolar y reparar señales que, de otro modo, se considerarían perdidas. Sin embargo, para que esto ocurra, deben cumplirse ciertas condiciones. En primer lugar, si un número significativo de canales está fuera de los umbrales, la interpolación no puede proporcionar un resultado confiable y la señal se elimina. Sin embargo, si definimos dos parámetros, κ, que representa el máximo número de canales corruptos, y ρ, que indica el máximo número de canales que pueden ser interpolados, podemos determinar si una señal es adecuada para la interpolación y puede proporcionar resultados aceptables. Esta determinación se realiza mediante la siguiente ecuación 4, donde $G_{\kappa,F}$ son las señales buenas para el subconjunto de datos F y T_f representa los datos de entrenamiento en el subconjunto de validación cruzada F. Q representan los canales EEG.

$$G_{\kappa,F} = \left\{ i \in T_F \mid \sum_{j=1}^{Q} C_{ij} < \kappa \right\} \tag{4}$$

En caso de tener un número de sensores defectuosos $\sum_{j=0}^{PQ} C_{ij}$ menor a ρ, se interpolan todos los canales. Mientras que si es mayor a ρ, se interpolarán los ρ canales más defectuosos, es decir, con mayor amplitud de señal. El resultado del filtrado e interpolado se puede ver en la Figura 11.

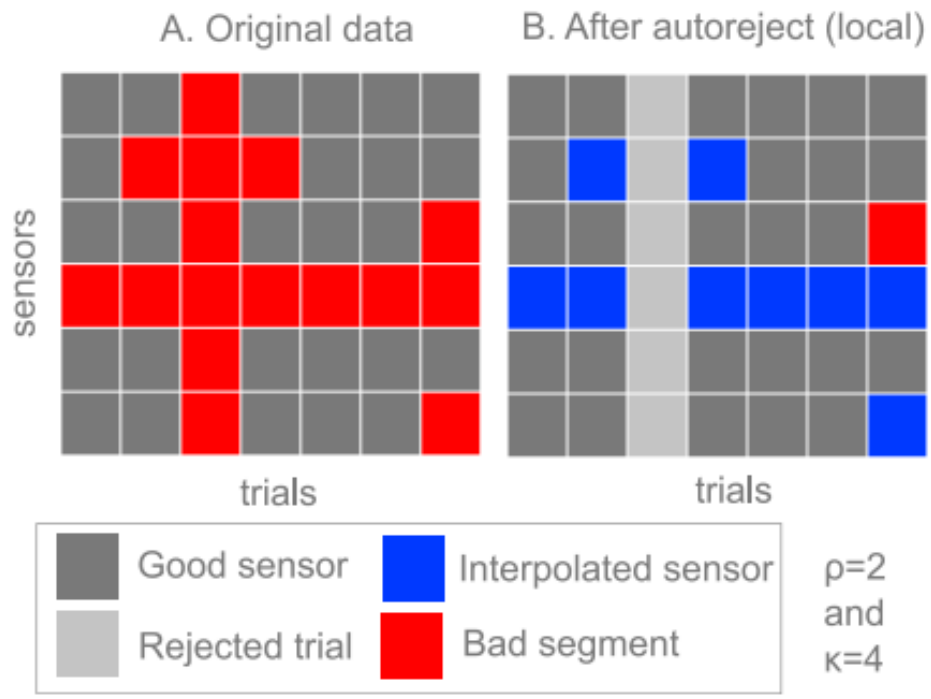


Figura 11: Diagrama explicativo sobre el funcionamiento de Auto Reject, mostrando el antes y después de los datos, incluyendo las señales interpoladas y eliminadas.
Fuente: [Jas *et al.*, 2017]

2.4.2. Segmentación y recombinación de señales

El método de segmentación y recombinación de señales (S&R) es una técnica de aumento de datos[5] aplicada principalmente a series de tiempo, como las señales EEG. Este enfoque se basa en la idea de que, al dividir una señal continua en segmentos más pequeños, se pueden crear nuevas combinaciones de estos segmentos que preserven las características temporales y secuenciales de la señal original.

La segmentación implica cortar la señal en múltiples fragmentos, que pueden ser de longitud fija o variable, dependiendo de la naturaleza de la señal y del objetivo del experimento. Posteriormente, estos segmentos se recombinan en un orden que mantiene la coherencia temporal, lo que permite generar nuevas secuencias de datos que, aunque artificiales, conservan las propiedades fundamentales de la señal original [Lotte, 2015]. Este proceso es especialmente útil cuando se trabaja con conjuntos de datos limitados, ya que aumenta efectivamente la cantidad de datos disponibles para el entrenamiento de modelos.

El uso de S&R puede ser particularmente beneficioso en contextos donde la variabilidad temporal es crítica, como en el análisis de señales biomédicas, donde la preservación del orden

[5]Generación artificial de datos basada en datos reales

temporal es esencial para mantener la validez clínica de los datos. A través de este método, es posible mejorar la capacidad de generalización de los modelos de aprendizaje automático, reduciendo al mismo tiempo el riesgo de sobreajuste. La Figura 12 muestra un ejemplo ilustrativo del proceso de segmentación y recombinación de señales, donde se respeta el orden temporal de los segmentos para asegurar la integridad de los datos generados.

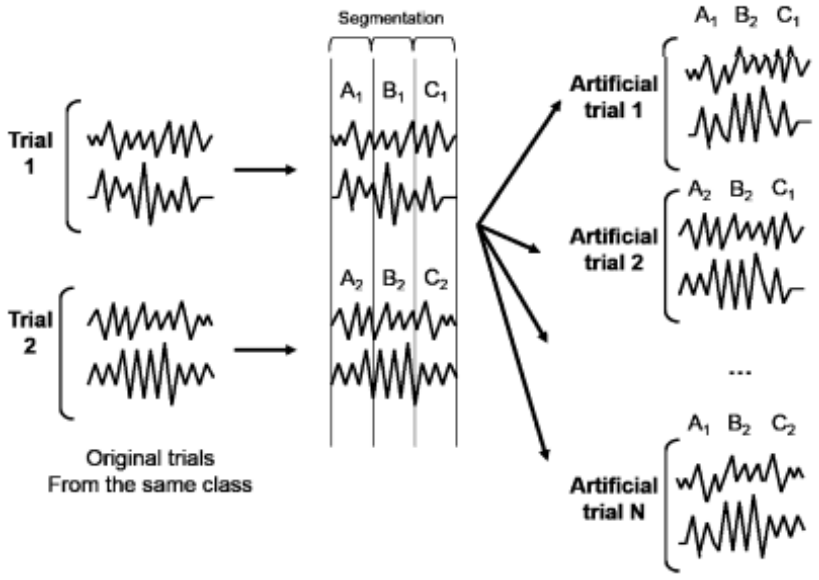


Figura 12: Ejemplo de segmentación y recombinación de señales, en donde se respeta el orden temporal de los datos.

Fuente: [Lotte, 2015]

2.5. Métricas de evaluación

Al abordar un problema de clasificación binaria como el P300 *Speller*, la elección adecuada de métricas de evaluación es fundamental para garantizar una comparación justa entre modelos, especialmente en presencia de clases desbalanceadas.

Para estandarizar las ecuaciones, se utilizará la matriz de confusión, visualizada en la Figura 13, en donde se visualizan los 4 posibles estados de una predicción:

- TP (Verdadero positivo, en español): Casos positivos correctamente predichos.
- TN (Verdadero negativo, en español): Casos negativos correctamente predichos.
- FP (Falso positivo, en español): Casos negativos incorrectamente predichos como positivos.

- FN (Falso negativo, en español): Casos positivos incorrectamente predichos como negativos.



Figura 13: Matriz de confusión, donde se muestran las predicciones correctas en verde y las incorrectas en rojo.
Fuente: www.towardsdatascience.com

A continuación, se presentan las métricas a utilizar:

- ***Precisión***: Proporción entre predicciones positivas correctas (TP) entre todas las predicciones positivas realizadas por el modelo.

$$\text{Precisión} = \frac{TP}{TP + FP} \tag{5}$$

  Esta métrica no considera los falsos negativos (FN), lo cual puede ser problemático en casos donde estos son críticos, como en el diagnóstico de enfermedades.

- ***Sensibilidad***: Proporción de verdaderos positivos (TP) entre todos los casos realmente positivos.

$$\text{Sensibilidad} = \frac{TP}{TP + FN} \tag{6}$$

Aunque útil, la Sensibilidad no considera los falsos positivos (FP), lo que significa que un modelo puede generar muchos falsos positivos y no ser detectado por esta métrica.

- ***F1-Score***: Media armónica entre precisión y Sensibilidad.

$$F1\text{-Score} = 2 \times \frac{\text{Precisión} \times \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \tag{7}$$

Esta métrica busca compensar las desventajas de precisión y Sensibilidad, ya que tiene en cuenta tanto los falsos positivos como los falsos negativos, siendo útil cuando ambos casos son importantes de detectar.

- ***ROC-AUC***: Área bajo la curva ROC (*Receiver Operating Characteristic*). La curva ROC es una representación gráfica que muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) para diferentes umbrales de clasificación.

$$\text{TPR} = \frac{TP}{TP + FN} \tag{8}$$

$$\text{FPR} = \frac{FP}{FP + TN} \tag{9}$$

Un valor de 1 indica un clasificador perfecto, mientras que un valor de 0.5 indica un clasificador sin capacidad de discriminación, similar a una clasificación aleatoria. El ROC-AUC es útil para comparar diferentes modelos, especialmente cuando se trabaja con conjuntos de datos desequilibrados, dado que la métrica utiliza múltiples umbrales de decisión para llegar a su resultado final.

2.6. Función de pérdida

Las funciones de pérdida se encargan de cuantificar el rendimiento del modelo durante el entrenamiento; esta función se debe minimizar para lograr los mejores resultados. Esto se logra utilizando distintas fórmulas, las cuales se deben elegir acorde al problema que se busca resolver. Dado que el P300 *Speller* es un problema de clasificación binaria, la función de pérdida utilizada por los cuatro modelos seleccionados es la siguiente:

Pérdida de entropía cruzada

Conocida como *cross entropy loss* en inglés, se define como la diferencia o distancia entre dos distribuciones de probabilidad: la distribución real de las clases y la distribución predicha por el modelo [Zhang y Sabuncu, 2018]. La ecuación para la versión binaria del algoritmo es la siguiente:

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(p_i) + (1-y_i)\log(1-p_i)\right]$$

Donde:

- N es el número de muestras.
- y_i es la etiqueta verdadera de la i-ésima muestra, cuyos valores son 0 o 1.
- p_i es la probabilidad predicha para la i-ésima muestra obtenida de la función Softmax.

2.7. Sistemas de posicionamiento de EEG

Los sistemas de posicionamiento nacen de la necesidad de estandarizar las metodologías de obtención de datos EEG, logrando definir colocaciones exactas de los electrodos en la corteza cerebral.

Sistema 10-20

El sistema 10-20 es un método estandarizado internacional para el posicionamiento de los electrodos sobre el cuero cabelludo. El nombre 10-20 se refiere a los intervalos de 10 % o 20 % de distancia entre los electrodos respecto a la longitud craneal.

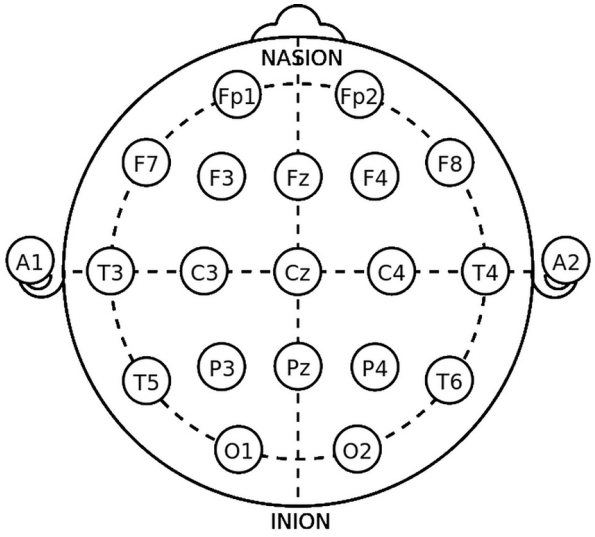


Figura 14: Posicionamiento de electrodos y sus respectivos nombres para el sistema 10-20. Fuente: [Rojas *et al.*, 2018]

Sistema 10-10

El sistema 10-10 es una extensión del sistema 10-20, que entrega una mayor densidad de electrodos y, por lo tanto, una mayor resolución espacial de la actividad cerebral. En este sistema, los electrodos se colocan en intervalos de 10 % respecto a la longitud craneal.

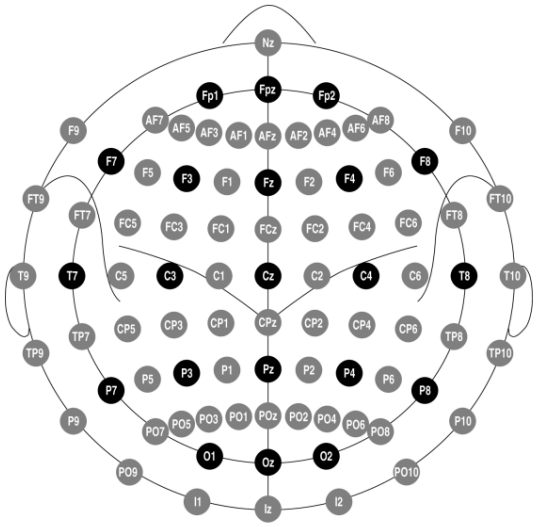


Figura 15: Posición de electrodos en el sistema 10-10. Los círculos negros indican la posición en el sistema 10-20 original y los grises los electrodos añadidos en la extensión 10-10.
Fuente: [Oostenveld y Praamstra, 2001]

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

En el campo del análisis de datos electroencefalográficos (EEG), el desarrollo y la implementación de modelos de *deep learning* han demostrado ser herramientas valiosas para diversas aplicaciones, incluida la detección del potencial evocado P300 en sistemas de comunicación basados en señales cerebrales, como el P300 *Speller*. No obstante, la diversidad de arquitecturas y enfoques en *deep learning* presenta un desafío significativo: la necesidad de una metodología robusta y equitativa para comparar el rendimiento de estos modelos. Esta propuesta de solución se centra en establecer una metodología integral que permita la comparación equitativa, generalizada y reproducible de modelos de *deep learning* aplicados a datos EEG, con especial atención en el P300 *Speller*.

La metodología propuesta se diseña para garantizar que ningún modelo obtenga ventajas injustas sobre otros, más allá de sus méritos técnicos inherentes. Para lograr esto, se consideran una serie de factores críticos en el diseño experimental y en los criterios de evaluación, asegurando una comparación equitativa, transparente y reproducible.

3.1. Modelos

Las arquitecturas utilizadas para abordar este problema deben adaptarse al formato específico de los datos EEG. Por esta razón, las redes convolucionales, combinadas con *transformers* y *feed forward*, son las más aplicadas tanto en la academia como en el ámbito clínico [Roy *et al.*, 2019]. En este estudio, vamos a comparar los modelos Shallow Net, EEGNet, EEG-Inception y EEG Conformer. Estos modelos han sido seleccionados debido a la diversidad de sus arquitecturas y a los resultados prometedores que han demostrado en estudios previos. Además, su implementación es facilitada por la librería *Braindecode* [Schirrmeister *et al.*, 2017a].

Shallow Net

Cronológicamente, la arquitectura introducida por [Schirrmeister *et al.*, 2017b] es la primera propuesta que se analizará. El modelo surge de la pregunta: ¿Cuál es el impacto del diseño de las redes convolucionales sobre la precisión de decodificación de señales EEG?

Se propusieron tres arquitecturas, y se seleccionó aquella que ofrecía un equilibrio óptimo entre eficiencia computacional y precisión para compararla con los demás modelos. *Shallow Net*, el modelo elegido, ilustrado en la Figura 16, consta únicamente de cuatro capas. En la primera capa convolucional, se aplican filtros de tamaño $[1 \times 25]$ a los datos temporales.

En la segunda capa convolucional, se emplea un filtro espacial de tamaño $[Ch \times 1]$, donde Ch representa el número de canales en el estudio, buscando identificar correlaciones o dependencias en un tiempo t entre diferentes canales. Una vez obtenidos los mapas de características, se realiza una normalización por lotes para mejorar la estabilidad y el rendimiento del modelo. Posteriormente, se aplica una capa de *pooling* promedio para mitigar el riesgo de *overfitting* en un modelo compacto, además de reducir el costo computacional. Luego de esta reducción de datos, se aplanan para ser procesados por la red *Feed Forward*, culminando en la etapa de clasificación.

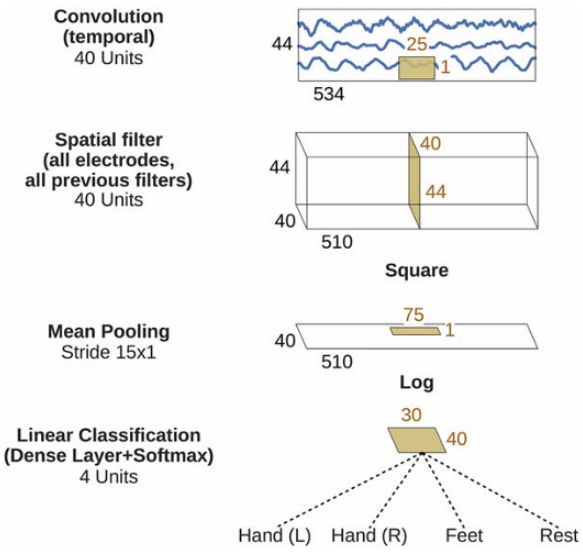


Figura 16: Arquitectura *Shallow Net* para una tarea de imaginación motora.
Fuente: [Schirrmeister *et al.*, 2017b]

Si bien estos modelos se crearon con la idea de decodificar señales EEG para una variedad de tareas, en el artículo original [Schirrmeister *et al.*, 2017b] solo se obtuvieron resultados para la tarea de imaginación motora, dejando de lado la experimentación con sistemas P300 *speller*. La imaginación motora implica detectar los patrones generados al imaginar o visualizar movimientos, como levantar el pie izquierdo, mover la mano derecha, entre otros. Para obtener los resultados, se utilizó la base de datos pública BCI Competition IV [Brunner *et al.*, 2008], que consta de 9 pacientes, 2 sesiones y 48 pruebas. Los resultados obtenidos para *Shallow Net* fueron de un 85.3 % de *accuracy*.

Los hiperparámetros del modelo *Shallow Net*, presentados en la Tabla 1, juegan un papel crucial en la determinación de su desempeño y eficiencia para esta comparación. El rango de valores para el número de canales EEG (`n_chans`) varía entre 8 y 16, lo que permite al modelo adaptarse a diferentes configuraciones de datos. El número de salidas (`n_outputs`) se establece en 2, reflejando la tarea de clasificación binaria del modelo. En cuanto a las muestras temporales de la ventana de entrada (`n_times`), se consideran dos opciones: 256 y 128, lo que impacta en la resolución temporal del análisis. Los filtros temporales (`n_filters_time`)

y espaciales (`n_filters_spat`) están configurados ambos en 40, permitiendo una captura detallada de las características tanto en el tiempo como en el espacio. La longitud del filtro temporal (`filter_time_length`) se fija en 25, y el filtro de pooling temporal está diseñado con una longitud de 75 (`pool_time_length`) y un paso de 15 (`pool_time_stride`), equilibrando la reducción de dimensionalidad con la preservación de información relevante. Cada uno de estos parámetros contribuye a optimizar el rendimiento del modelo, ajustando su capacidad para aprender y generalizar a partir de las señales EEG.

Parámetro	**Valor**	**Descripción**
n_chans	8-16	Número de canales EEG.
n_outputs	2	Número de salidas del modelo. Número de clases en clasificación.
n_times	256-128	Número de muestras temporales de la ventana de entrada.
n_filters_time	40	Número de filtros temporales.
filter_time_length	25	Longitud del filtro temporal.
n_filters_spat	40	Número de filtros espaciales.
pool_time_length	75	Longitud del filtro de pooling temporal.
pool_time_stride	15	Longitud del paso entre filtros de pooling temporal.

Tabla 1: Parámetros del modelo Shallow Net

EEGNet

El segundo modelo a analizar es EEGNet, introducido por [Lawhern *et al.*, 2018], el cual presenta una arquitectura versátil y minimalista que logra excelentes resultados para un gran rango de tareas de decodificación EEG.

Este enfoque introduce el uso de capas convolucionales profundas y separables (*depthwise separable convolutions*), como se observa en la Figura 17, las cuales desempeñan dos funciones principales: en primer lugar, reducen la cantidad de parámetros que deben ser entrenados, como se observa en la Tabla 2; y en segundo lugar, facilitan la separación del componente temporal y espacial de los datos. Esta capacidad de separación permite optimizar el aprendizaje de los mapas de características de manera individual y, además, optimizar la combinación de dichos mapas para crear nuevos mapas de características.

Los principales parámetros del modelo EEGNet, como se muestra en la Tabla 3, incluyen el número de canales de EEG utilizados (*n_chans*), que varía entre 8 y 16 dependiendo del conjunto de datos, y el número de salidas del modelo (*n_outputs*), fijado en 2 para las tareas de clasificación binaria. El modelo también se configura con 256 a 128 muestras temporales por ventana de entrada (*n_times*), lo que permite ajustar la resolución temporal según las características de la señal.

Además, EEGNet emplea un número de filtros en las capas convolucionales que varía entre 8 en la primera capa (*F1*) y 16 en la segunda capa (*F2*), con un factor de multiplicación (*D*)

	Tiempo de prueba (s)	ShallowConvNet	EEGNet
P300	1	104002	1066
ERN	1.25	91602	1082
MRCP	1.5	104722	1098
SMR	2	40644	796

Tabla 2: Cantidad de parámetros entrenables de modelos ShallowConvNet y EEGNet para cuatro paradigmas distintos: P300, ERN (Negatividad cerebral relacionada con el error), MRCP (Potencial cortical relacionado al movimiento) y SMR (ritmo sensori-motor).
Fuente: [Lawhern *et al.*, 2018]

de 2 para aumentar la complejidad del modelo en la segunda capa. La longitud del filtro en la primera capa (*kernel_length*) es de 64, mientras que en la tercera capa se utiliza un filtro de tamaño 8 (*third_kernel_size*). Finalmente, la probabilidad de *dropout* (*drop_prob*) se establece en 0.25, lo que ayuda a prevenir el sobreajuste durante el entrenamiento.

Parámetro	Valor	Descripción
n_chans	8-16	Número de canales EEG.
n_outputs	2	Número de salidas del modelo. Número de clases en clasificación.
n_times	256-128	Número de muestras temporales de la ventana de entrada.
F1	8	Número de filtros en la primera capa.
D	2	Factor de multiplicación para el número de filtros en la segunda capa.
F2	16	Número de filtros en la segunda capa.
kernel_length	64	Longitud del filtro en la primera capa.
third_kernel_size	8	Tamaño del filtro en la tercera capa.
drop_prob	0.25	Probabilidad de *dropout*.

Tabla 3: Parámetros del modelo EEGNet

Para el estudio de EEGNet, Lawhern realizó experimentos con tres bases de datos, entre las cuales se destaca el interés particular en la base de datos relacionada con la señal P300. Para dichos datos, el modelo obtuvo un AUC de 91 % en promedio, entrenando un modelo con 4-Fold por usuario.

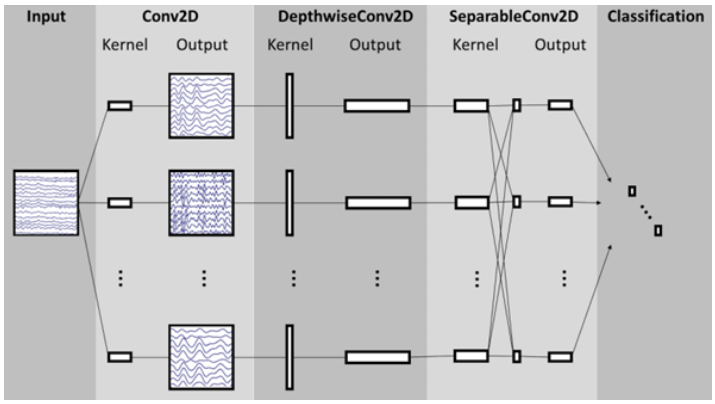


Figura 17: Arquitectura EEGNet.
Fuente: [Lawhern *et al.*, 2018]

EEG-Inception

El tercer modelo a comparar lo introduce [Santamaria-Vazquez *et al.*, 2020], llamado EEG-Inception, el cual es capaz de captar tanto patrones globales como locales gracias a las capas *inception* [Szegedy *et al.*, 2015].

Previo a la revisión de la arquitectura, el autor modifica el formato de los datos a [Tiempo $\times$ Canal], en vez de [Canal $\times$ Tiempo], que es el estándar utilizado; se respetará este formato en el marco conceptual.

La arquitectura propuesta se muestra en la Figura 18, la cual consta de dos módulos de *inception* descritos a continuación. En primer lugar, se tienen tres convoluciones 2D, cada una seguida de normalización por lotes, activación y *dropout*. Los tamaños de los filtros están determinados por la longitud de la secuencia temporal, que en los experimentos realizados en [Santamaria-Vazquez *et al.*, 2020] consta de 128 datos. En este caso, se emplea un filtro de 64×1, que abarca la mitad de los datos; otro filtro de 32×1; y finalmente, un filtro de 16×1. Esta primera convolución está diseñada para detectar patrones temporales en los datos, de manera independiente para cada canal. Luego, se aplica una convolución *depthwise* de tamaño 1×8, con el objetivo de identificar patrones espaciales entre los canales.

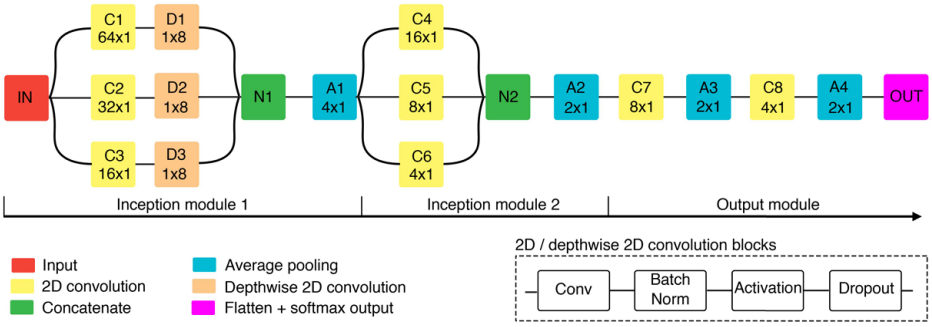


Figura 18: Arquitectura *Inception*.
Fuente: [Santamaria-Vazquez *et al.*, 2020]

El segundo módulo de *inception* consta de una única capa de convolución, donde se buscan patrones temporales complejos a nivel global y local. Posteriormente, se aplican dos bloques de convolución y *pooling* para aumentar la complejidad y condensar los mapas de características obtenidos. Finalmente, estos mapas de características se introducen en una red *feed forward* para su clasificación.

Para evaluar el modelo, se empleó un sistema BCI *Speller*, similar al P300 *Speller*, con la participación de usuarios con y sin condiciones degenerativas previas. Los resultados arrojaron un promedio de precisión del 84.6 %, lo que lo posicionó, al momento de su publicación, como el modelo con mejores resultados.

Los hiperparámetros del modelo *EEG-Inception*, detallados en la Tabla 4, son fundamentales para su capacidad de captar tanto patrones globales como locales en las señales EEG. El rango del número de canales EEG (`n_chans`) varía entre 8 y 16, adaptándose a diferentes configuraciones experimentales. El número de salidas del modelo (`n_outputs`) se fija en 2, adecuado para tareas de clasificación binaria. La entrada temporal del modelo (`n_times`) está configurada en 256 muestras, lo que permite un análisis detallado de la serie temporal. La tasa de *dropout* (`drop_prob`) se establece en 0.5, lo que ayuda a prevenir el sobreajuste durante el entrenamiento. Las escalas temporales (`scales_samples_s`) para las convoluciones en cada módulo Inception son (0.5, 0.25, 0.125) segundos, lo que permite al modelo capturar patrones en diferentes resoluciones temporales. El número inicial de filtros de convolución (`n_filters`) varía entre 8 y 16, influenciando la capacidad de extracción de características. Finalmente, los tamaños de *pooling* (`pooling_sizes`) para los bloques Inception son (4, 2, 2, 2), equilibrando la reducción de dimensionalidad con la preservación de información relevante. Cada uno de estos hiperparámetros contribuye a mejorar la capacidad del modelo para identificar y clasificar patrones complejos en las señales EEG.

Parámetro	Valor	Descripción
n_chans	8-16	Número de canales EEG.
n_outputs	2	Número de salidas del modelo. Número de clases en clasificación.
n_times	256	Tamaño de la entrada en número de muestras.
drop_prob	0.5	Tasa de *dropout* en toda la red.
scales_samples_s	(0.5, 0.25, 0.125)	Escalas temporales (segundos) de las convoluciones en cada módulo Inception.
n_filters	8-16	Número inicial de filtros de convolución.
pooling_sizes	(4, 2, 2, 2)	Tamaños de *pooling* para los bloques Inception.

Tabla 4: Parámetros del modelo EEG-Inception

EEG Conformer

Finalmente, el modelo Conformer, introducido por [Song *et al.*, 2022], combina capas convolucionales con capas de autoatención (*self attention*) para capturar dependencias y patrones locales, y al mismo tiempo, relacionarlos a través del tiempo. El modelo consta de tres partes, las cuales se presentarán a continuación:

1. **Convolución**: Similar a modelos como EEGNet, se dividen las convoluciones 2D en dos convoluciones 1D (espacial y temporal). En la primera capa, se utilizan k filtros temporales de tamaño (1×25), mientras que en la segunda capa se emplean filtros espaciales de tamaño $(Ch \times 1)$, donde Ch representa la cantidad de canales disponibles. Posteriormente, se aplica normalización por lotes a los mapas de características obtenidos anteriormente. Luego, se pasa a una capa de *average pooling* de tamaño (1×75) con pasos de (1×15), lo que reduce la dimensionalidad y ayuda a prevenir el sobreajuste. Finalmente, los mapas de características se redimensionan para pasar a la siguiente capa.

Capa	In	Out	Filtro	Paso
Conv Temporal	1	k	(1, 25)	(1, 1)
Conv Espacial	k	k	(Ch, 1)	(1, 1)
Avg Pooling	k	k	(1, 75)	(1, 15)

Tabla 5: Arquitectura del módulo de convolución, con k filtros
Fuente: [Song *et al.*, 2022]

2. **Atención**: El autor argumenta la utilidad de la capa de atención en base a que la actividad neuronal, en este caso las señales EEG, son actividades coherentes, lo cual da paso a la necesidad de aprender dependencias globales temporales.

Los *tokens* del módulo anterior se triplican, y se asignan a las variables Q (Consulta o *query*), K (llave o *Key*) y V (valor). Se aplica producto punto entre Q y K para evaluar la correlación entre los *tokens*. Este resultado se pasa por una función softmax para obtener la matriz de atención, la cual luego se multiplica sobre V para obtener la respectiva atención para el token actual. Este proceso puede formularse como:

$$\text{Atención(Q, K, V)} = \text{Softmax}\left(\frac{QK^T}{\sqrt{F}}\right)V$$

Donde F denota el largo de un token.

Para mejorar la representación de las características, se aplica la estrategia "multi-cabeza", en donde los *tokens* se distribuyen en s segmentos y se entregan independientemente a módulos de autoatención, para luego concatenar el resultado.

3. **Clasificación**: Finalmente, se utilizan dos capas completamente conexas, junto con una función *softmax* para obtener las probabilidades de cada clase disponible, eligiendo aquella con mayor probabilidad.

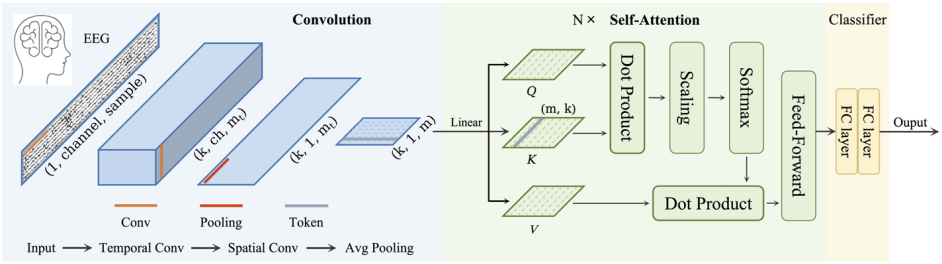


Figura 19: Arquitectura Conformer.
Fuente: [Song *et al.*, 2022]

Para tareas de imaginación motora, el modelo EEG-Conformer obtuvo un 84.63 % de *accuracy* promedio, consistentemente obteniendo mejores resultados cuando se aplica aumento de datos con segmentación y recombinación, además de contar con la capa de atención, como se observa en la Figura 20.

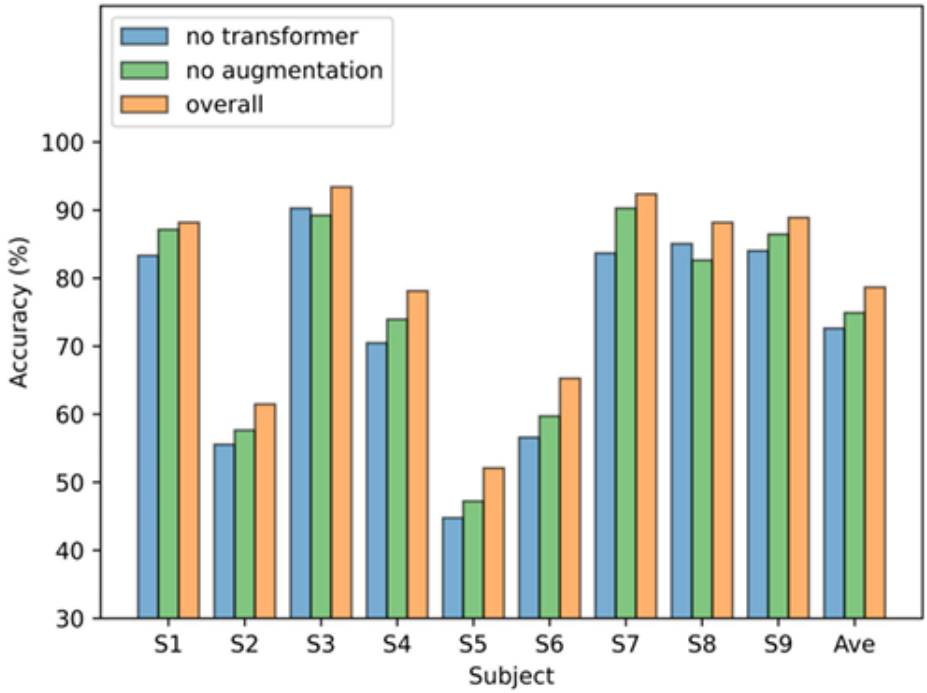


Figura 20: Gráfico comparativo de *accuracy* en EEG-Conformer para las distintas configuraciones experimentales [*overall*, no aumentado, no *transformer*].
Fuente: [Song *et al.*, 2022]

Los hiperparámetros del modelo *EEG Conformer*, especificados en la Tabla 6, son esenciales para la capacidad del modelo de combinar la convolución con autoatención para analizar señales EEG. El número de salidas (`n_outputs`) se establece en 2, adaptado para tareas de clasificación binaria. El número de canales EEG (`n_chans`) varía entre 8 y 16, permitiendo la adaptación a diferentes configuraciones de datos. Los filtros temporales (`n_filters_time`) están configurados en 40, lo que define también el tamaño del embedding del modelo. La longitud del filtro temporal (`filter_time_length`) se fija en 25, mientras que el filtro de *pooling* temporal tiene una longitud de 75 (`pool_time_length`) y un paso de 15 (`pool_time_stride`), optimizando la reducción de dimensionalidad y la preservación de características importantes. La tasa de *dropout* en la capa convolucional (`drop_prob`) se establece en 0.5 para evitar el sobreajuste. En la capa de autoatención, se utilizan 6 capas (`att_depth`) y 10 cabezas de atención (`att_heads`), con una tasa de *dropout* de 0.5 (`att_drop_prob`) para mejorar la representación de las características. Finalmente, la dimensión de la capa completamente conectada (`final_fc_length`) es de 2440, y el número de muestras temporales de la ventana de entrada (`n_times`) varía entre 256 y 128. Cada uno de estos hiperparámetros juega un rol crucial en el rendimiento del modelo, permitiendo una captura detallada y precisa de los patrones en las señales EEG.

Parámetro	Valor	Descripción
n_outputs	2	Número de salidas del modelo. Número de clases en clasificación.
n_chans	8-16	Número de canales EEG.
n_filters_time	40	Número de filtros temporales, define también el tamaño del embedding.
filter_time_length	25	Longitud del filtro temporal.
pool_time_length	75	Longitud del filtro de *pooling* temporal.
pool_time_stride	15	Longitud del paso entre filtros de *pooling* temporal.
drop_prob	0.5	Tasa de *dropout* de la capa convolucional.
att_depth	6	Número de capas de auto-atención.
att_heads	10	Número de cabezas de atención.
att_drop_prob	0.5	Tasa de *dropout* de la capa de auto-atención.
final_fc_length	2440	Dimensión de la capa completamente conectada.
n_times	256-128	Número de muestras temporales de la ventana de entrada.

Tabla 6: Parámetros del modelo EEG Conformer

3.2. Base de datos

Se eligieron tres conjuntos de datos para el entrenamiento, validación y evaluación de modelos, basándose en la calidad de los datos, la diversidad de características de los pacientes y la relevancia exclusiva del experimento P300 *Speller*. Todos los datos fueron obtenidos de la librería MOABB [Aristimunha *et al.*, 2023], que busca facilitar la reproducibilidad de experimentos BCI, ofreciendo datos y modelos implementados en academia.

3.2.1. *Dataset* I

Para el dataset I, los datos fueron extraídos de [Aricò *et al.*, 2014], donde participaron 10 usuarios con una edad media de 26.8 ± 5.6 años, sin evidencia de enfermedades degenerativas y con experiencia previa en sistemas BCI P300. Estos usuarios fueron parte de tres sesiones utilizando el sistema P300 *Speller*.

Durante el experimento, se grabaron las señales (g.USBamp, gTec, Austria) utilizando 16 canales, acorde al estándar 10-10 visto en la Figura 15. Las señales se digitalizaron a 256 Hz, con un *bandpass filter* de 0.1 a 20 Hz utilizado para eliminar artefactos de la señal.

Los participantes debieron copiar tres palabras predefinidas de 6 caracteres cada una, usando un P300 *Speller* [Farwell y Donchin, 1988]. En este caso, las filas y columnas fueron intensificadas por 125 ms, con un intervalo entre estímulos de 125 ms, dejando en total 250 ms entre dos estímulos. Por cada carácter a escribir, el sujeto debía realizar 8 secuencias de estímulos.

3.2.2. *Dataset* II

Para el dataset II, los datos fueron recopilados por el equipo de [Riccio *et al.*, 2013], en el cual 8 participantes, resumidos en la Tabla 7, diagnosticados con ELA y sin otras condiciones degenerativas que interfieran con el experimento, así como sin experiencia previa en sistemas BCI, participaron en el experimento del P300 *Speller*.

Durante el experimento, se grabaron las señales (g.MOBILAB, g.tec, Austria) utilizando 8 canales acorde al estándar 10-10 visto en la Figura 15[Chatrian *et al.*, 1985] (Fz, Cz, Pz, Oz, P3, P4, PO7 y PO8) usando electrodos activos (g.Ladybird, g.tec, Austria). Las señales EEG se digitalizaron a 256 Hz, con un *bandpass filter* entre 0.1 y 30 Hz.

Los participantes debieron copiar siete palabras predefinidas de 5 caracteres cada una, usando un P300 *Speller* [Farwell y Donchin, 1988]. En este caso, las filas y columnas fueron intensificadas por 125 ms, con un intervalo entre estímulos de 125 ms, dejando en total 250 ms entre dos estímulos. Se realizaron 10 secuencias completas, donde cada secuencia consiste en la iluminación de una fila y columna.

Participantes	Edad	Sexo	ALSFRS-R
1	56	M	13
2	59	M	37
3	43	M	33
4	75	F	38
5	60	F	34
6	40	M	31
7	61	M	28
8	72	F	41

Tabla 7: Información demográfica de los participantes, incluyendo ALSFRS-R[6]

3.2.3. *Dataset* III

El tercer y último conjunto de datos se obtuvo de [Van Veen *et al.*,], en donde 25 participantes, con una edad media de 24.4 años, participaron en un estudio de un sistema BCI P300 llamado “Brain Invaders”.

Las señales cerebrales se midieron con 16 electrodos de plata, posicionados según el estándar 10-20, como se muestra en la Figura 21, con una frecuencia de muestreo de 128 muestras por segundo. Durante el experimento, se le instruyó a los sujetos que limitaran pestañeos, movimientos faciales y contracciones musculares, las cuales pueden perturbar señales EEG.

[6] ALSFRS-R: La escala revisada de valoración de la Esclerosis Lateral Amiotrófica [Cedarbaum *et al.*, 1999], es una herramienta para evaluar la capacidad funcional de los pacientes, en donde un puntaje alto indica mejor función y menor severidad de la condición. El rango de puntaje va de 48 a 0.

Además, se les recomendó contar el número de iluminaciones del objetivo para mejorar la concentración.

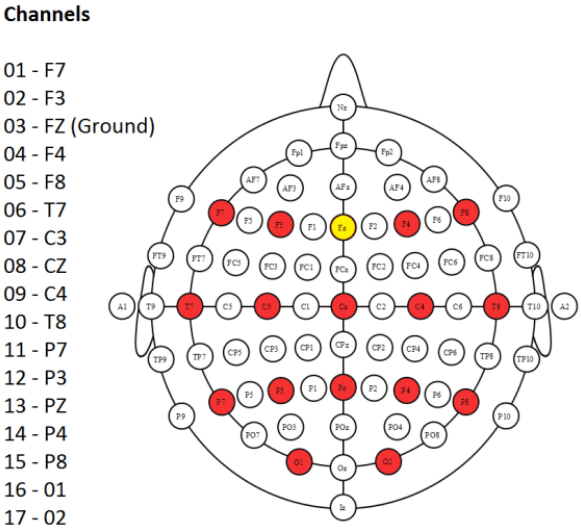


Figura 21: Posición de electrodos en el sistema 10-20 para el *dataset* III.
Fuente: [Oostenveld y Praamstra, 2001]

El experimento “Brain Invaders” es similar al P300 *Speller*, ya que también utiliza una matriz de 6x6, pero en este caso, se trata de aliens en lugar de letras y símbolos, como se observa en la Figura 22. El objetivo del experimento es eliminar a un alien objetivo. Para lograrlo, cada alien se ilumina dos veces de manera aleatoria, buscando generar un potencial P300 cada vez que se ilumina el objetivo. La proporción de iluminaciones objetivo y no objetivo sigue siendo de 1:5.

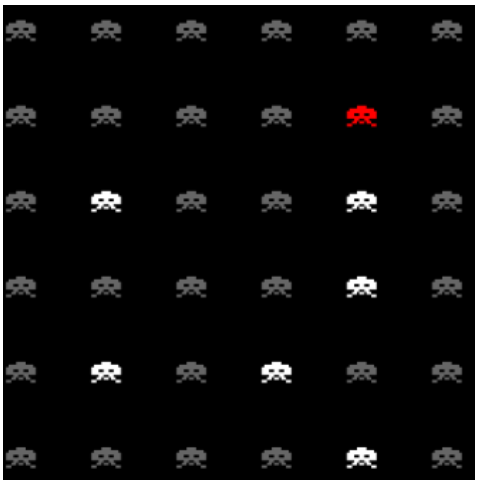


Figura 22: Interfaz gráfica de *Brain Invaders*, donde se aprecia la matriz 6x6, con 6 aliens iluminados, siendo uno de ellos, el objetivo, iluminado de rojo.
Fuente: [Oostenveld y Praamstra, 2001]

3.3. Preprocesamiento

Para preprocesar los datos, se utilizaron métodos de la librería de código abierto MNE, creada por [Gramfort *et al.*, 2013]. Esta librería estandariza el tratamiento y visualización de datos EEG.

Inicialmente se aplica un filtrado de amplitud entre 0.1 Hz y 20 Hz para estandarizar los conjuntos de datos, ya que no todos aplicaron filtros en este rango. Este filtro atenúa las frecuencias fuera de este rango, eliminando el ruido de baja y alta frecuencia, mejorando así la calidad de las señales EEG relevantes para el análisis.

La necesidad de atenuar señales fuera de dicho rango tiene como objetivo mejorar la proporción de señal a ruido, lo cual presenta un gran problema en la decodificación de señales EEG. Las fuentes de ruido nacen tanto de factores externos, como torres de alta tensión (entre 50 y 60 Hz), como internos, como el ruido electromiográfico [7] (mayor a 30 Hz) y el sudor, el cual genera artefactos con frecuencias menores a 1 Hz gracias a la interferencia de los electrolitos (sodio y potasio) con los electrodos [Luck, 2014].

Si bien el filtrado por amplitud cumple su labor de mejorar la proporción de señal a ruido, no es un proceso perfecto. Para complementar, se aplica *Autoreject*, que automatiza el proceso de limpieza de datos, identificando, corrigiendo y eliminando segmentos de datos contaminados o ruidosos. Como se observa en la Figura 23, el algoritmo calcula los umbrales óptimos de amplitud, en este caso para el sujeto N°1 del *dataset* I, para interpolar o eliminar señales ruidosas. Los parámetros utilizados para *Autoreject* se presentan en la Tabla 8:

Parámetro	**Valor**	**Descripción**
n_interpolate	[1, 4, 8]	Definido como ρ, es el máximo número de sensores a interpolar.
cv	10	Número de cortes utilizados para *cross_validation*.
picks	“eeg”	Canales a incluir, en este caso, se seleccionan solo aquellos que correspondan a electrodos EEG.

Tabla 8: Parámetros utilizados para *Autoreject*

[7] Estudio de señales generadas por movimientos o contracciones musculares

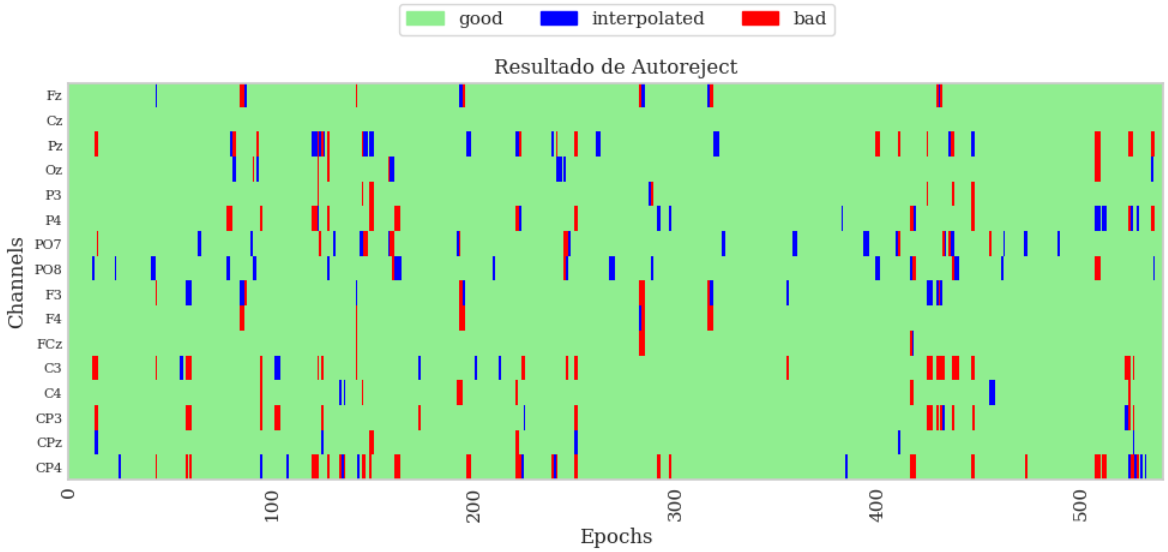


Figura 23: *Autoreject* aplicado al sujeto N°1 del *dataset* I, en donde se presentan los casos sin fallos en verde, los interpolados en azul, y los no corregidos en rojo.
Fuente: Elaboración propia

3.4. División de datos

Al trabajar con datos temporales, como los extraídos a partir de señales EEG, es crucial generar una correcta división de datos que permita simular el uso de los modelos en un ambiente realista. Por ello, se idearon dos métodos de división de datos, dependiendo de las características del conjunto de datos.

Para el *dataset* I, que posee tres sesiones, se eligieron las primeras dos como datos de entrenamiento, dejando la última sesión como datos de prueba. Esta separación permite simular el uso real de los modelos a comparar y, gracias a esto, generar modelos y comparaciones más robustas. Durante los experimentos realizados para obtener los datos, las señales EEG producidas por un sujeto expuesto al mismo estímulo van cambiando por distintas razones, tales como fatiga, cambios de ánimo y aprendizaje del sistema BCI *Speller*, lo que introduce ruido, artefactos y cambios en los datos utilizados para entrenar el modelo. Esto pone a prueba la capacidad de generalización de los modelos, ya que se entrenarán con datos que cambian constantemente y de manera impredecible [Roy *et al.*, 2019].

Para los Datasets II y III, que constan de una sola sesión, el método anterior no se puede aplicar. En este caso, simplemente se dividieron los datos utilizando la función “train_test_split” de la librería SKlearn, asignando el 80 % de los datos para entrenamiento y el 20 % restante para el conjunto de prueba. Esta separación se realiza de manera aleatoria, utilizando el parámetro “random_state” para asegurar la reproducibilidad de la división.

Al momento de entrenar el modelo, se crea otro conjunto de datos denominado validación, el cual corresponde al 20 % de los datos de entrenamiento, con el cual se asegurará que el modelo aprenda de manera óptima y generalizada.

3.5. Aumentado de datos

Una gran limitación de un sistema EEG, tal como el P300 *Speller*, es la obtención de datos, la cual es inherentemente lenta y laboriosa. Esta lentitud lleva a tener conjuntos de datos pequeños, lo que dificulta el entrenamiento de modelos de aprendizaje profundo, sobre todo aquellos que por su arquitectura necesitan más, como es el caso del EEG Conformer. Es por esto que en [Song *et al.*, 2022] se propone el uso de aumento de datos mediante segmentación y reconstrucción (S %R), un método computacionalmente barato y que demuestra significativas mejoras en métricas de evaluación como *accuracy* y AUC. Otro factor que influyó en la elección de un método de aumento de datos fue el desbalance de clases de 5 a 1 que existe en el sistema P300 *Speller*. Dicho desbalance puede generar modelos sesgados y no generalizados.

Para el entrenamiento de los modelos, el aumento de datos se aplica después del preprocesamiento y división en datos de entrenamiento y validación, ya que solamente queremos aplicar aumento a datos de entrenamiento. Se aplicó S&R sobre los datos de entrenamiento con el objetivo no solo de aumentar, sino que también de balancear las clases objetivo y no objetivo, ya que estas presentan una proporción de 1:5 respectivamente. Esto se logró aumentando cada clase por separado, en donde se agruparon datos en grupos de 20 eventos, los cuales se segmentaron en 25 secciones cada una, para luego combinar dichas secciones y crear nuevos datos. Para la clase objetivo se obtienen 100 datos nuevos por cada grupo de 20 eventos, y para la clase no objetivo, se obtienen 5 datos nuevos por cada grupo de 20 eventos. En la Tabla 9 se observan las nuevas representaciones del potencial P300 en los *datasets*, recordando que antes era de tan solo un 16 %.

dataset	**N° estímulos**	**N° estímulos balanceado**	**Representación P300**
Dataset I	1152	2292	47.64 %
Dataset II	3360	6855	49.05 %
Dataset III	611	1236	49.19 %

Tabla 9: Datos de entrenamiento de cada *dataset*, junto a la comparación antes y después de aplicado el balanceo

3.6. Entrenamiento

Para garantizar una comparación más global y comprensiva de los modelos, se emplearon cuatro métodos de entrenamiento diferentes. El primer método, denominado “Normal”, es

el entrenamiento estándar, que no incluye aumento de datos ni la limpieza de los mismos, ofreciendo una robusta base de comparación para los modelos en su forma más básica. El segundo método, denominado “Datos limpios”, aplica solo limpieza de datos, permitiendo evaluar la importancia del filtrado e interpolación de señales corruptas en el rendimiento del modelo. El tercer método, denominado “Datos balanceados”, aplica aumento de datos de tal manera que se logra un balance de las clases, como se explicó en la subsección anterior. Finalmente, el cuarto método, denominado “Datos limpios y balanceado”, incluye tanto aumento como limpieza de datos, proporcionando una perspectiva del rendimiento del modelo en su forma más compleja. Al comparar estos métodos, se obtiene una visión más completa de cómo cada técnica de preprocesamiento afecta el rendimiento y la capacidad de generalización de los modelos entrenados, asegurando así conclusiones más robustas y aplicables en distintos contextos de uso. Una simplificación del flujo de datos se aprecia en la Figura 24.

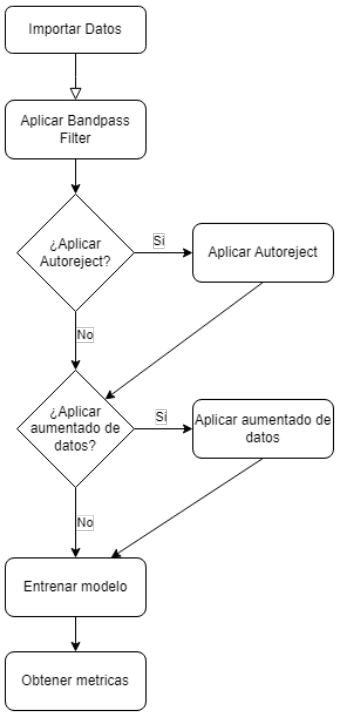


Figura 24: Flujo de datos simplificado.
Fuente: Elaboración propia

3.6.1. Parámetros de modelos

Al utilizar la metodología “*Within subject*”, que consiste en entrenar un modelo por cada sujeto estudiado, el costo computacional de obtener los hiperparámetros óptimos para los modelos con los cuatro métodos de entrenamiento distintos se vuelve prohibitivamente alto, dado que para cada uno de los 43 sujetos se deben entrenar 16 modelos en total. Por esta

razón, en lugar de recurrir a una optimización exhaustiva con métodos como *grid search*, se definió una heurística para obtener los hiperparámetros basada en la bibliografía original de los modelos utilizados, donde se estimaron valores óptimos utilizando los resultados presentados en esos estudios. Esta aproximación permite reducir significativamente el tiempo y los recursos necesarios para el entrenamiento, manteniendo una alta calidad en el ajuste de los modelos. Los parámetros seleccionados para los modelos se observan en las Tablas 1 para Shallow Net, 3 para EEGNet, 6 para EEG Conformer y 4 para EEG Inception.

3.6.2. Métricas

El paso siguiente a obtener los hiperparámetros es el entrenamiento de los modelos, durante el cual se monitorea la función de pérdida o (*loss*) llamada entropía cruzada definida anteriormente en la sección 2.6.

1. **Pérdida (*loss*)**: Evalúa qué tan bien está aprendiendo el modelo durante el entrenamiento. Esta métrica permite detectar el sobreajuste, ya que si, al evaluar la pérdida en el conjunto de entrenamiento, esta está descendiendo, pero la pérdida en el conjunto de validación está ascendiendo o se encuentra estancada, el modelo presenta problemas de sobreajuste, ya que no es capaz de generalizar el conocimiento de entrenamiento a validación. Esta métrica se calcula con el conjunto de validación.

Para dicho entrenamiento, se debe definir el modelo que se va a entrenar, el método a utilizar (Normal, Datos limpios, Datos balanceados o Datos limpios y balanceados) y el número de épocas para el entrenamiento. Sobre este último punto, se recomienda hacer un estudio sobre las épocas utilizadas para cada modelo, con el fin de evitar que el modelo se sobreajuste a los datos.

Una técnica que se empleó para manejar el número de épocas es el *early stopping*, que consiste en detener el entrenamiento una vez que el modelo en cuestión no mejore su métrica en un determinado número de épocas, denominado paciencia. En este caso, se utilizó *early stopping* con una paciencia de 15 épocas, monitorizando el valor de la función de pérdida para el conjunto de validación.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

Los experimentos fueron ejecutados en la plataforma Google Colab, donde se cuenta con una tarjeta gráfica T4 y 12.7 GB de RAM. Los modelos fueron entrenados en el mismo ambiente, reiniciando dicho ambiente cada vez que se entrena un modelo para asegurar las mismas condiciones y disponibilidad de hardware. Esta consistencia en el entorno de entrenamiento es crucial para garantizar que las diferencias en el rendimiento de los modelos sean atribuibles a sus arquitecturas y métodos de entrenamiento, y no a variaciones en los recursos computacionales.

En conjunto, se analizaron los resultados de 43 sujetos, resumidos en la Tabla 10. De estos, 10 tenían experiencia previa en el uso de sistemas BCI al momento de llevar a cabo el experimento, y 8 habían sido diagnosticados con Esclerosis Lateral Amiotrófica (ELA) en distintas etapas de la condición. La edad de los participantes varía entre 21 y 75 años. Esta diversidad en la muestra permite evaluar la robustez y generalización de los modelos en distintos escenarios de uso.

Tabla 10: Descripción de *datasets*

	Dataset I	Dataset II	Dataset III
N° Sujetos	10	8	25
N° Canales	16	8	16
N° Sesiones	3	1	1
Frecuencia [Hz]	256	256	128
Tiempo de prueba [s]	197	1359	361.4 ± 8.6
Estímulos totales	1728	4200	765 ± 2

A continuación, se presentan los resultados obtenidos aplicando la metodología descrita en el capítulo 3, utilizando los modelos Shallow Net, EEGNet, EEG Inception y EEG Conformer, con sus respectivos hiperparámetros. Cada modelo fue entrenado utilizando cuatro métodos distintos: normal, limpio, balanceado, y limpio y balanceado. Estos métodos se seleccionaron para investigar el impacto de diferentes técnicas de preprocesamiento y balanceo de datos en el rendimiento del modelo, particularmente en tareas de clasificación binaria para señales P300. Para asegurar un análisis más robusto y obtener resultados más generalizables y fieles, el entrenamiento de cada modelo se repitió 5 veces, y los resultados obtenidos fueron promediados, proporcionando así un valor medio representativo del comportamiento real del modelo.

Para cada dataset, se evaluó el rendimiento de los modelos utilizando métricas como la precisión, sensibilidad, F1-score y la curva ROC. Estas métricas proporcionan una visión completa

de las capacidades de clasificación de los modelos, permitiendo una comparación detallada entre ellos. Además, se generaron matrices de confusión para visualizar los errores de clasificación y entender mejor las fortalezas y debilidades de cada modelo.

Finalmente, se realizaron análisis estadísticos para determinar la significancia de las diferencias observadas entre los modelos y los métodos de entrenamiento. Estos análisis permiten validar que las mejoras observadas no son producto del azar, sino que reflejan diferencias significativas en el desempeño de los modelos.

4.1. Resultados para el Dataset I

El primer *dataset* cuenta con señales EEG extraídas de usuarios con experiencia previa en el uso de sistemas BCI, por lo que se esperan mejores resultados en comparación con los demás conjuntos de datos. Además, como se observa en la Tabla 10, se registran 256 muestras por segundo desde 16 electrodos posicionados según el estándar 10-10.

4.1.1. F1-Score

El F1-Score es una métrica crucial que combina precisión y sensibilidad, proporcionando una visión integral del rendimiento de los modelos en la detección de señales P300. Para el *dataset* I, como se ve en la Figura 25, se observó que el modelo EEGNet con el método balanceado alcanzó el mejor F1-Score, con un valor de 0.8894 ± 0.0211. Este resultado destaca la robustez de EEGNet, que no solo obtiene un rendimiento sobresaliente con el método balanceado, sino que también mantiene un alto nivel de desempeño con los demás métodos. Esta configuración balanceada y el modelo EEGNet son, por lo tanto, particularmente efectivos para manejar los desafíos de este *dataset*.

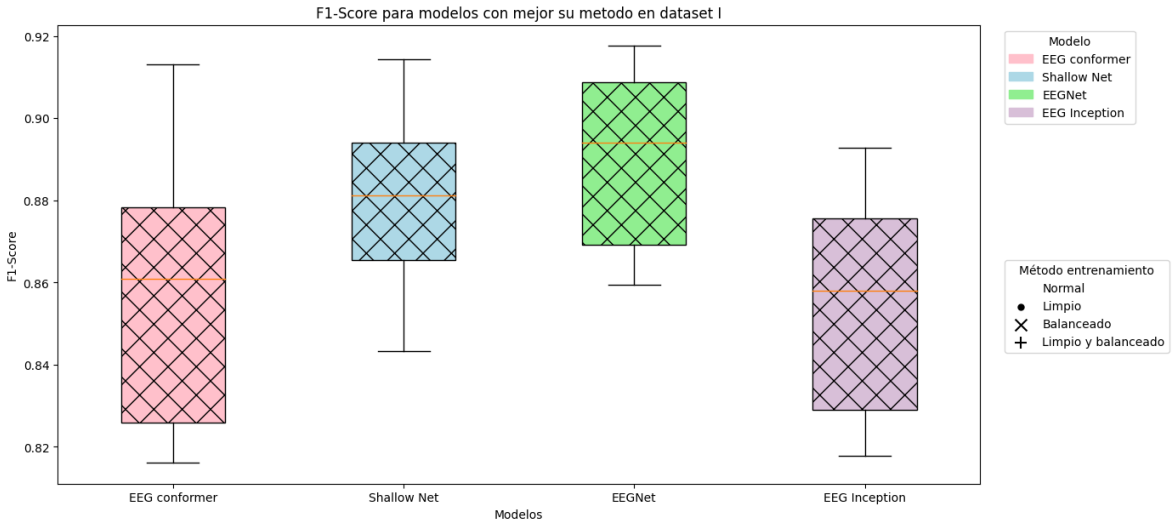


Figura 25: Gráfico boxplot en donde se muestra para cada modelo, el mejor método utilizado junto con su F1-Score. Para este gráfico se utilizan todos los sujetos disponibles en el conjunto de datos I, siendo aquellos que se alejan del promedio, también llamados *outliers*, visualizados como un punto. Cada modelo junto a su método fueron entrenados 5 veces.
Fuente: Elaboración propia

En comparación, los modelos Shallow Net y EEG Inception también demostraron buenos resultados, con F1-Scores de 0.8692 ± 0.0308 y 0.8419 ± 0.0331, respectivamente, como se aprecia en la Tabla 11. Estos resultados subrayan que, aunque EEGNet ofrece el mejor desempeño, otros modelos también presentan capacidades competitivas dependiendo del método de entrenamiento aplicado. Para complementar este resultado, se realizó un análisis por sujeto, el cual se muestra en la Figura 43, encontrada en el anexo.

Tabla 11: F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* I

Métodos	EEG Inception	EEG Net	Shallow Net	EEG Conformer
Normal	0.8419 ± 0.0331	0.8459 ± 0.019	0.8692 ± 0.0308	0.8243 ± 0.0541
Limpio	0.8315 ± 0.0311	0.8411 ± 0.032	0.8485 ± 0.0365	0.8061 ± 0.0547
Balanceado	0.8536 ± 0.0271	$\mathbf{0.8894 \pm 0.0211}$	0.8797 ± 0.020	0.8588 ± 0.0334
Limpio y Balanceado	0.8460 ± 0.0267	0.8841 ± 0.026	0.8720 ± 0.0266	0.8468 ± 0.0378

4.1.2. ROC

El ROC proporciona una visión general de la capacidad de un modelo para distinguir entre clases bajo diferentes umbrales de decisión. Para el *dataset* I, el modelo EEGNet con el método balanceado sobresale con un área bajo la curva (AUC, por sus siglas en inglés) de 0.9151

$\pm$ 0.0318. Este alto valor indica una excelente capacidad del modelo para identificar correctamente las señales P300 en comparación con otras configuraciones. La diferencia en AUC entre el método balanceado y el método normal (un aumento del 2.1 %) demuestra cómo el balanceo de datos puede mejorar significativamente la clasificación en entornos P300, permitiendo una distinción más clara entre las dos clases, especialmente en problemas de clases desbalanceadas.

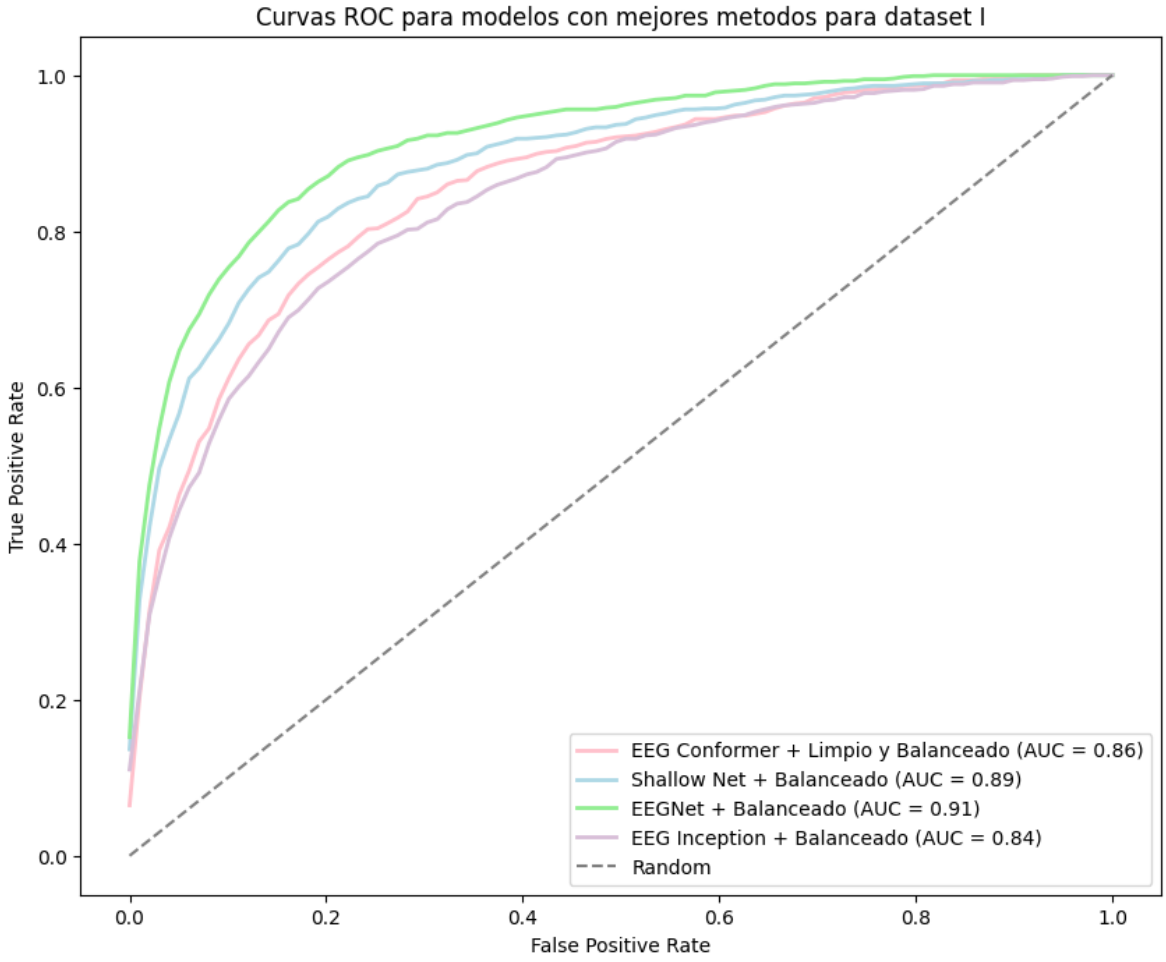


Figura 26: Curva ROC, graficando el modelo junto al método de entrenamiento que mayor AUC consigue para el *dataset* I.

Fuente: Elaboración propia

A pesar de los sobresalientes resultados de EEGNet, otras combinaciones de modelos y métodos también mostraron un rendimiento notable. Por ejemplo, Shallow Net con entrenamiento balanceado obtuvo resultados bastante cercanos, con un AUC de 0.8864 $\pm$ 0.0421, como se observa en la Figura 26. Este rendimiento cercano resalta la eficacia de Shallow Net en la clasificación de señales P300 cuando se aplica un balanceo adecuado de los datos.

Por otro lado, el modelo EEG Inception obtuvo los peores resultados entre todos los mode-

los evaluados, como se muestra en la Figura 26. Esto nos lleva a cuestionar la efectividad del módulo Inception, que aplica filtros de múltiples tamaños para encontrar patrones tanto locales como globales. Estos patrones que el modelo encuentra pueden no estar directamente relacionados con la detección de la señal P300, resultando en una capacidad de separación de clases deficiente.

El EEG Conformer tampoco logró resultados positivos, mostrando un rendimiento subóptimo en comparación con otros modelos. Al igual que EEG Inception, EEG Conformer podría estar detectando patrones no relevantes de manera global debido al módulo de atención empleado, lo que dificulta su capacidad para distinguir entre clases en este contexto específico.

La Tabla 12 proporciona un resumen detallado de los valores promedio de AUC para diferentes modelos y metodologías de entrenamiento en el *dataset* I. Se puede observar que, aunque EEGNet con datos balanceados proporciona los mejores resultados, otras configuraciones como Shallow Net balanceado también muestran un rendimiento competitivo. En contraste, EEG Inception y EEG Conformer presentan resultados menos satisfactorios en todas las condiciones evaluadas. Para profundizar estos resultados, se realizó un análisis por sujeto en la figura 46, encontrada en el anexo.

Tabla 12: AUC promedio para diferentes modelos y metodologías de entrenamiento en el *Dataset* I

Métodos	**EEG Inception**	**EEG Net**	**Shallow Conv**	**EEG Conformer**
Normal	0.8403 ± 0.0561	0.8962 ± 0.0440	0.8580 ± 0.0487	0.8129 ± 0.0713
Limpio	0.8112 ± 0.0630	0.8789 ± 0.0429	0.8468 ± 0.0499	0.8031 ± 0.0641
Balanceado	0.8436 ± 0.0588	$\mathbf{0.9151 \pm 0.0318}$	0.8864 ± 0.0421	0.8553 ± 0.0468
Limpio y Balanceado	0.8404 ± 0.0563	0.9100 ± 0.0329	0.8746 ± 0.0378	0.8569 ± 0.0492

4.1.3. Matriz de confusión

Para complementar los resultados obtenidos de las métricas anteriores, se presenta la matriz de confusión del dataset I, en donde los modelos están organizados en las columnas, mientras que el método de entrenamiento está en las filas.

A primera vista, observando la Figura 27, se nota la baja cantidad de señales P300 detectadas en general por los modelos. El error más común es el de falsos negativos, es decir, señales P300 que son predichas como no P300. Este resultado no es sorprendente debido al severo desbalance de clases presente en el problema, que, si no se aborda adecuadamente, puede llevar a los modelos a un óptimo local en el que solo se clasifica la clase mayoritaria.

Hay una clara diferencia en los resultados obtenidos según el método de entrenamiento utilizado. Usando el método de entrenamiento normal como punto de referencia, el modelo Shallow Net presenta los mejores resultados con una sensibilidad del 41 % y una precisión

del 93 %. Recordemos que la sensibilidad es la proporción de casos realmente P300 que son detectados correctamente.

Al utilizar el método limpio, todos los modelos empeoran tanto en sensibilidad como en precisión, poniendo en duda la efectividad del método *autoreject* para este conjunto de datos.

El método balanceado, por otra parte, muestra los mejores resultados, con un incremento de la sensibilidad hasta el 72 % para el modelo EEGNet balanceado, más que duplicando esta métrica. Aunque aumenta el error de falsos positivos en un 6 %, este incremento es insignificante en comparación con la mejora obtenida en la detección de señales P300.

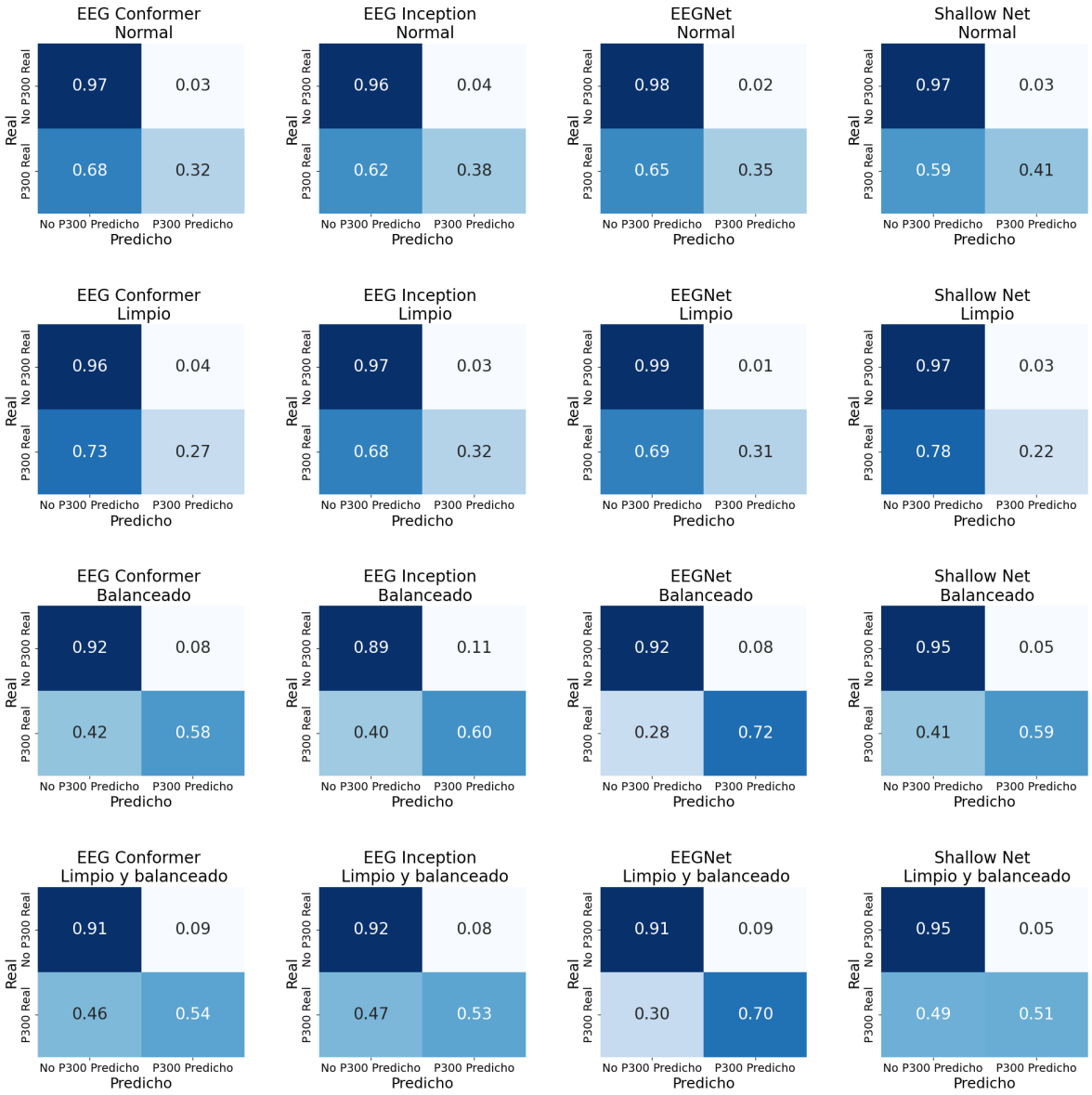


Figura 27: Matriz de confusión para el *dataset* I, en donde las columnas representan los modelos entrenados, y las filas los métodos utilizados.
Fuente: Elaboración propia.

4.1.4. Tiempo de Entrenamiento

El tiempo de entrenamiento es una consideración crucial para la viabilidad práctica de los modelos, ya que influye directamente en la escalabilidad de dichos modelos para su uso real en ambientes clínicos.

EEGNet destaca por su bajo tiempo de entrenamiento sin importar el método utilizado, lo-

grando completar el entrenamiento de 100 épocas en tan solo 4.19 segundos en promedio utilizando el método normal, aproximadamente el doble de rápido que EEG Inception, siete veces más rápido que EEG Conformer y ocho veces más rápido que Shallow Net, como se observa en la Tabla 14. Esto se comprueba con el bajo número de parámetros entrenables del modelo, con tan solo 1490 para este conjunto de datos, como se ve en la Tabla 13. La reducción de parámetros en comparación con modelos con convoluciones estándar, como Shallow Net, se debe al uso de convoluciones separables, las cuales dividen el cómputo de la convolución en dos secciones, logrando ahorrar una gran cantidad de parámetros entrenables.

Tabla 13: Numero de parámetros entrenables por modelo

Modelos	*Dataset* I	*Dataset* II	*Dataset* III
Shallow Net	14802	27602	26882
EEG Net	1490	1618	1490
EEG Inception	14974	15358	15310
EEG Conformer	255106	267906	175746

En contraste, Shallow Net es el modelo que más demora en entrenar, como se observa de manera clara en la Figura 28, sin importar el método utilizado. Esto se debe al alto costo de las convoluciones utilizadas en el modelo y a la cantidad de filtros empleados (40), ya que estas no son separables. Los tiempos de entrenamiento de EEG Conformer no están muy lejanos de Shallow Net, a pesar de implementar el módulo de atención.

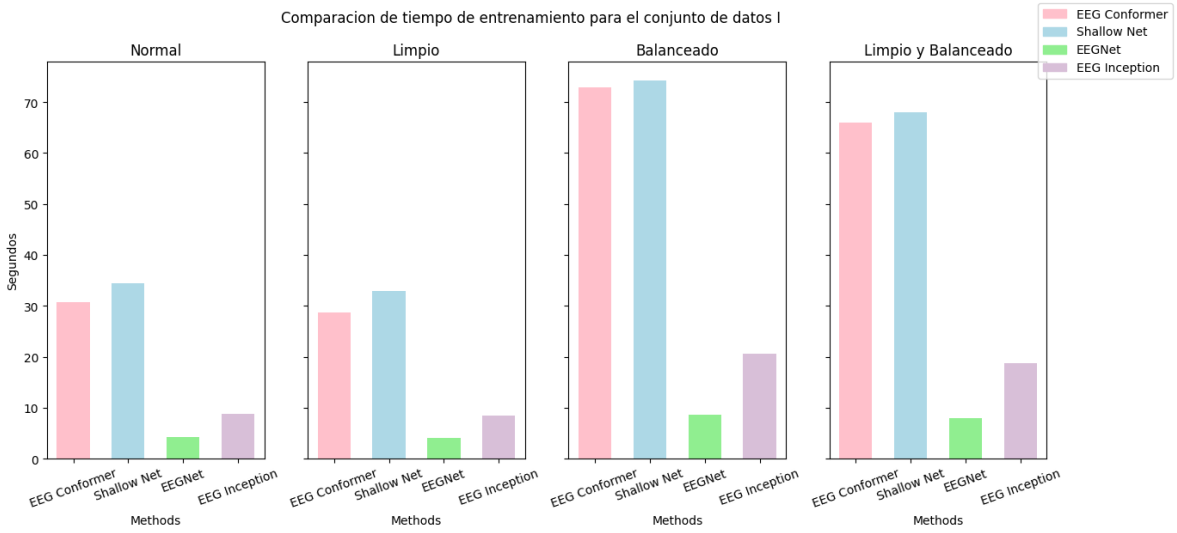


Figura 28: Tiempo de entrenamiento para *dataset* I, entrenando con 100 épocas.
Fuente: Elaboración propia

En resumen, para el *dataset* I, el modelo EEGNet con el método balanceado ofrece el mejor

rendimiento en términos de F1-Score y ROC. Otros modelos, como Shallow Net y EEG Inception, también muestran capacidades competitivas. La elección del método de entrenamiento y la configuración del modelo son factores clave para optimizar la detección de señales P300 en el contexto de este *dataset*.

Tabla 14: Tiempo de ejecución por época, promediando los sujetos, para diferentes modelos y metodologías de entrenamiento (en segundos)

Métodos	EEG Inception	EEG Net	Shallow Net	EEG Conformer
Normal	8.88 ± 0.09	$\mathbf{4.19 \pm 0.29}$	34.42 ± 0.15	30.66 ± 0.77
Limpio	8.54 ± 0.83	$\mathbf{4.09 \pm 0.38}$	32.90 ± 2.80	28.75 ± 2.81
Balanceado	20.55 ± 0.44	$\mathbf{8.59 \pm 0.29}$	74.22 ± 0.27	72.94 ± 0.63
Limpio y Balanceado	18.70 ± 1.73	$\mathbf{7.95 \pm 0.92}$	68.07 ± 6.48	65.94 ± 6.36

4.2. Resultados para el Dataset II

El *dataset* II cuenta con señales EEG extraídas de usuarios que padecen ELA en distintos niveles de avance, los cuales podrían beneficiarse directamente del uso de sistemas P300 speller para comunicarse con el exterior. Este *dataset* tiene 8 canales posicionados según el sistema 10-10, digitalizados a una frecuencia de 256 Hz. Como característica única, presenta 4200 estímulos totales, 2.43 veces más que el *dataset* I, lo que implica un tiempo de prueba más largo.

4.2.1. F1-Score

En el *dataset* II, se observó que, nuevamente, el modelo EEGNet con el método balanceado alcanzó el mejor F1-Score, con un valor de 0.8710 ± 0.0195, como se observa en la Figura 29. Recordemos que este es el modelo con menos parámetros, lo que, en principio, podría complicar la búsqueda de patrones complejos en un *dataset* con más datos. Sin embargo, esto parece ayudar al modelo a no sobreajustarse a los datos de entrenamiento.

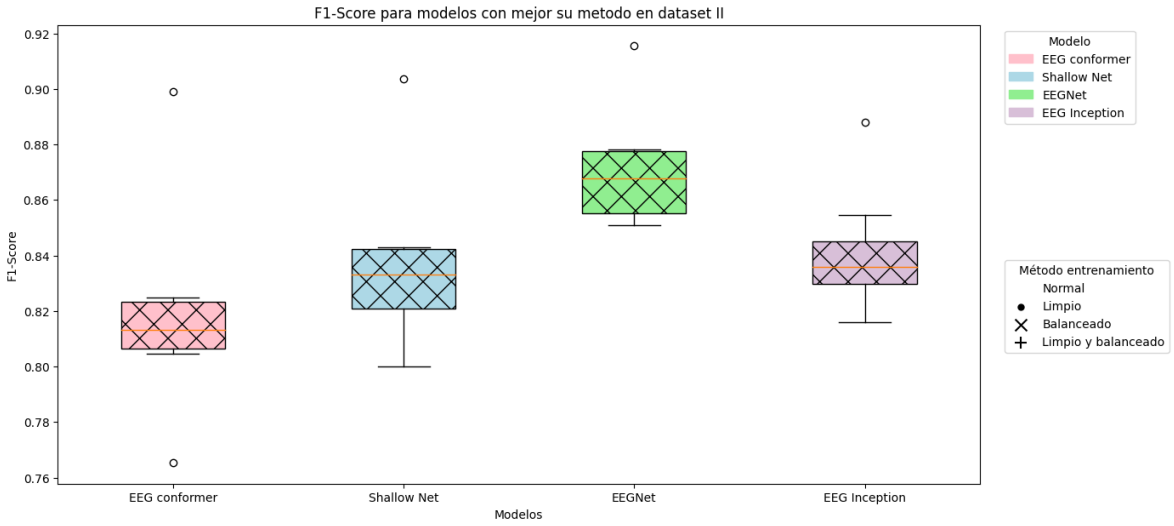


Figura 29: Gráfico boxplot en donde se muestra para cada modelo, el mejor método utilizado junto con su F1-Score. Para este gráfico se utilizan todos los sujetos disponibles en el conjunto de datos II, siendo aquellos que se alejan del promedio, también llamados *outliers*, visualizados como un punto. Cada modelo junto a su método fueron entrenados 5 veces.
Fuente: Elaboración propia

Este conjunto de datos presenta una alta discrepancia entre el F1-score obtenido por cada modelo, sobre todo para el mejor modelo, EEGNet y el segundo mejor, EEG Inception. En parte, esto se debe a la calidad de los datos, como se observa en la Tabla 16, los resultados con datos limpios disminuyeron el desempeño para todos los modelos. En la Tabla 15, se muestra la diferencia porcentual en la cantidad de datos antes y después de la limpieza, es decir, cuántos datos fueron eliminados durante el proceso de *autoreject*. En este caso, vemos un alto porcentaje de estímulos borrados al utilizar *autoreject*, eliminando un 12.89 % de datos en promedio por sujeto, llegando a borrar hasta un 70 % de datos al sujeto n°5. Esto podría deberse a un error en la adquisición de los datos, como un electrodo defectuoso. Sin embargo, al consultar con la fuente de los datos, no se encontró una razón específica para este comportamiento.

Tabla 15: Comparación de datos de entrenamiento normales versus limpios Dataset II

N° Sujeto	N° Eventos normales	N° Eventos limpio	Diferencia porcentual [%]
1	3360	3293	1.01
2	3360	3290	1.05
3	3360	3171	2.89
4	3360	3299	0.92
5	3360	588	70.21
6	3360	3013	5.44
7	3360	2457	15.52
8	3360	2973	6.11
Promedio			12.894 ± 22.132

A modo de comparación, los modelos EEG Inception y Shallow Net, vistos en la tabla 16, obtuvieron F1-Scores de 0.8405 ± 0.0213 y 0.8364 ± 0.0288, respectivamente. EEG Conformer obtuvo el peor F1-Score del conjunto de datos, con un máximo de 0.8187 ± 0.0349 usando el método balanceado. Para complementar estos resultados, se realizó un análisis del F1-Score por sujeto en la figura 44, encontrada en el anexo.

Tabla 16: F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* II

Métodos	EEG Inception	EEG Net	Shallow Net	EEG Conformer
Normal	0.8167 ± 0.0356	0.8597 ± 0.0293	0.8236 ± 0.0308	0.7874 ± 0.043
Limpio	0.8058 ± 0.0373	0.8379 ± 0.0391	0.810 ± 0.0384	0.7798 ± 0.0418
Balanceado	0.8405 ± 0.0213	$\mathbf{0.8710 \pm 0.0195}$	0.8364 ± 0.0288	0.8187 ± 0.0349
Limpio y Balanceado	0.8205 ± 0.375	0.8501 ± 0.032	0.8206 ± 0.0338	0.8089 ± 0.0442

4.2.2. ROC

Para el *dataset* II, el modelo EEGNet, utilizando el método normal, alcanza un AUC de 0.8574 ± 0.0400. Este resultado confirma la robustez de EEGNet en la clasificación de señales EEG a través de múltiples condiciones de preprocesamiento de datos. Este rendimiento superior sugiere que EEGNet es particularmente eficaz en la detección de patrones relevantes en los datos EEG, incluso cuando no se aplican técnicas adicionales de limpieza o balanceo de datos.

En este conjunto de datos, se observa una mayor brecha en los resultados obtenidos en comparación con otros conjuntos, como se muestra en la Figura 30. La diferencia porcentual entre EEGNet con el método normal y EEG Inception con datos balanceados, que son los dos mejores modelos para este conjunto de datos, es de 6.3 %. Esta diferencia significativa destaca la superioridad de EEGNet en la detección de potenciales en EEG bajo condiciones

normales de preprocesamiento, en comparación con EEG Inception que requiere un balanceo de datos para optimizar su rendimiento.

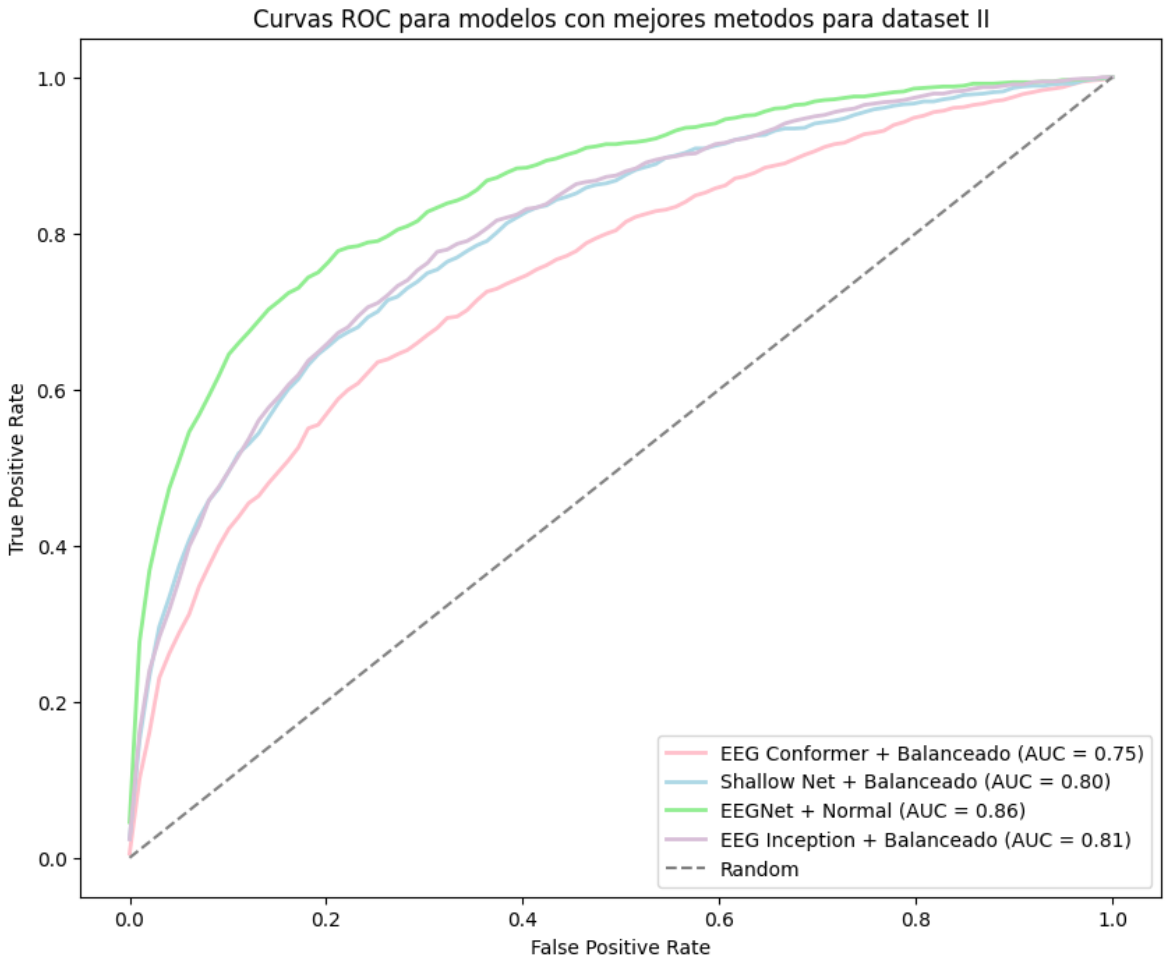


Figura 30: Curva ROC, graficando el modelo junto al método de entrenamiento que mayor AUC consigue para el *dataset* II.

Fuente: Elaboración propia

Por otro lado, el modelo EEG Conformer muestra una capacidad significativamente menor para distinguir entre las clases a clasificar, obteniendo el peor AUC del conjunto con una diferencia del 4.3 % respecto a Shallow Net. Esta marcada diferencia en el rendimiento de EEG Conformer sugiere que este modelo puede no ser adecuado para este tipo de datos EEG o que requiere ajustes adicionales en su arquitectura o en las técnicas de preprocesamiento de datos utilizadas.

La Tabla 17 proporciona un resumen detallado de los valores promedio de AUC para diferentes modelos y metodologías de entrenamiento en el dataset II. Se puede observar que, aunque el método normal proporciona los mejores resultados para EEGNet, los métodos

de balanceo de datos también mejoran significativamente el rendimiento de otros modelos como EEG Inception y Shallow Conv. En contraste, EEG Conformer sigue mostrando un rendimiento subóptimo en todas las condiciones evaluadas. Para complementar estos resultados, se realizó un análisis de la curva ROC por sujeto en la figura 47, encontrada en el anexo.

Tabla 17: AUC promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* II

Métodos	**EEG Inception**	**EEG Net**	**Shallow Conv**	**EEG Conformer**
Normal	0.7871 ± 0.0582	$\mathbf{0.8574 \pm 0.0400}$	0.7816 ± 0.0640	0.7230 ± 0.0917
Limpio	0.7638 ± 0.0676	0.8119 ± 0.0773	0.7419 ± 0.0909	0.7301 ± 0.0748
Balanceado	0.8053 ± 0.0490	0.8548 ± 0.0464	0.7998 ± 0.0548	0.7466 ± 0.0708
Limpio y Balanceado	0.7603 ± 0.0836	0.8327 ± 0.0499	0.7457 ± 0.1016	0.6984 ± 0.1016

4.2.3. Matriz de confusión

En la Figura 31 vemos que EEGNet es el modelo que mejor se comporta para todos los métodos de entrenamiento, obteniendo un 49 % de sensibilidad y una precisión del 92 % con el método de entrenamiento balanceado. Para el mismo modelo, es notable la poca diferencia que hay entre los resultados obtenidos entrenando de manera normal y balanceada, entregando un aumento menor que para otros modelos, como el EEG Conformer, el cual pasa de una sensibilidad del 11 % a un 29 %.

Para el resto de los modelos, se observa una mejora al usar el método balanceado, reafirmando su utilidad al tratar con señales P300. Mientras que para el método limpio, nuevamente este empeora los resultados obtenidos con el método normal, lo cual se replica al utilizar el método limpio y balanceado, donde el método balanceado obtiene mejores resultados.

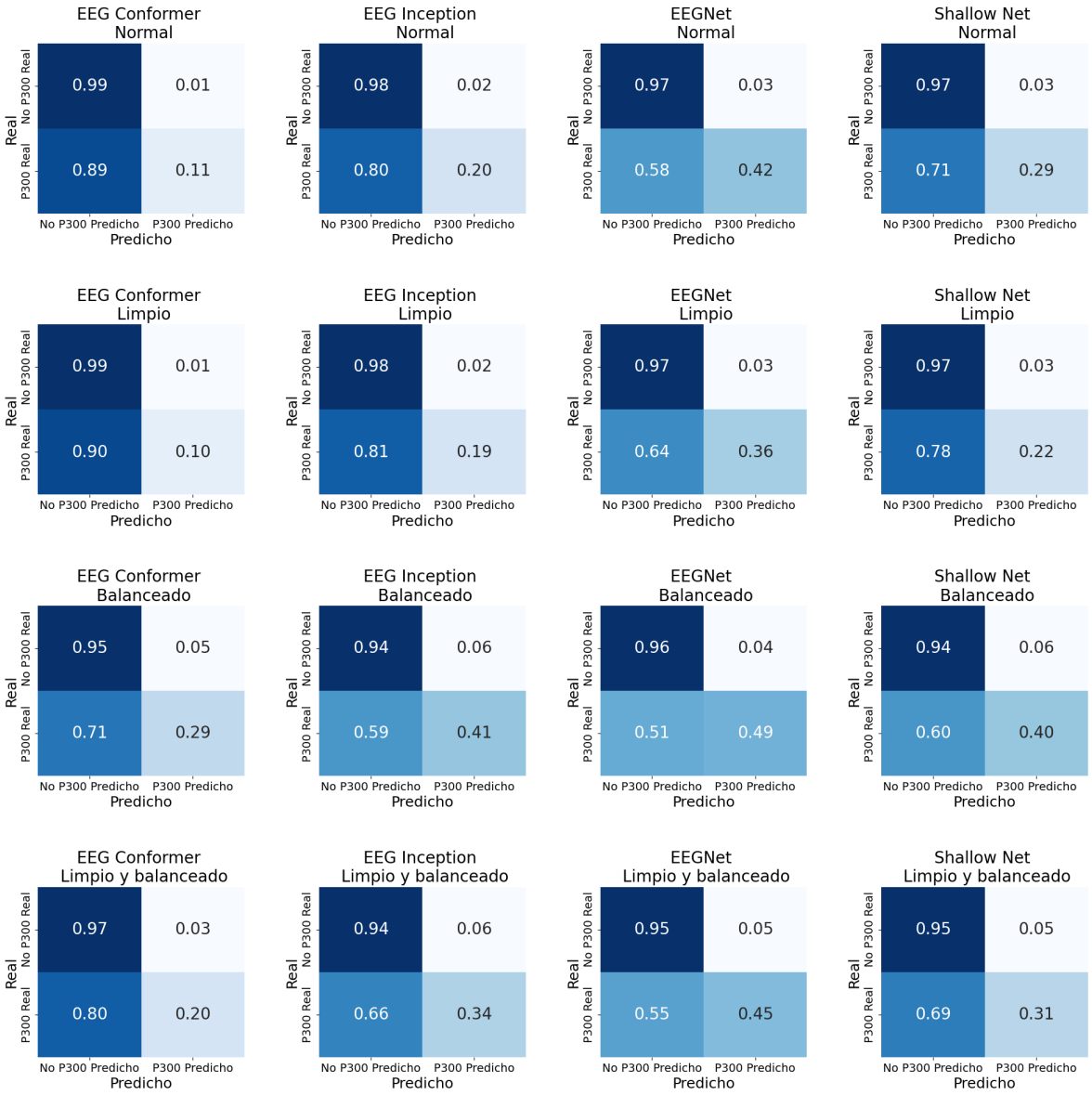


Figura 31: Matriz de confusión para el *dataset* II, en donde las columnas representan los modelos entrenados, y las filas los métodos utilizados.

Fuente: Elaboración propia.

4.2.4. Tiempo de Entrenamiento

Como se puede observar en la Figura 32, EEGNet obtiene los menores tiempos de entrenamiento, a pesar del aumento de estímulos totales y la disminución de canales utilizados. Ahora la diferencia entre los dos modelos más lentos, en este caso Shallow Net y EEG Conformer respectivamente, es mucho mayor. Mientras que en el primer conjunto, Shallow Net era 1.2 veces más lento que EEG Conformer para el entrenamiento normal, ahora es 3.2

veces más lento. Esto puede deberse al pobre escalamiento de la cantidad de operaciones necesarias para realizar las convoluciones de Shallow Net respecto a la cantidad de estímulos totales, que ahora son 2.4 veces más que en el dataset I. También puede deberse a la eficiencia del módulo de multiatención, que recordamos puede ser paralelizado para obtener tiempos de entrenamiento menores.

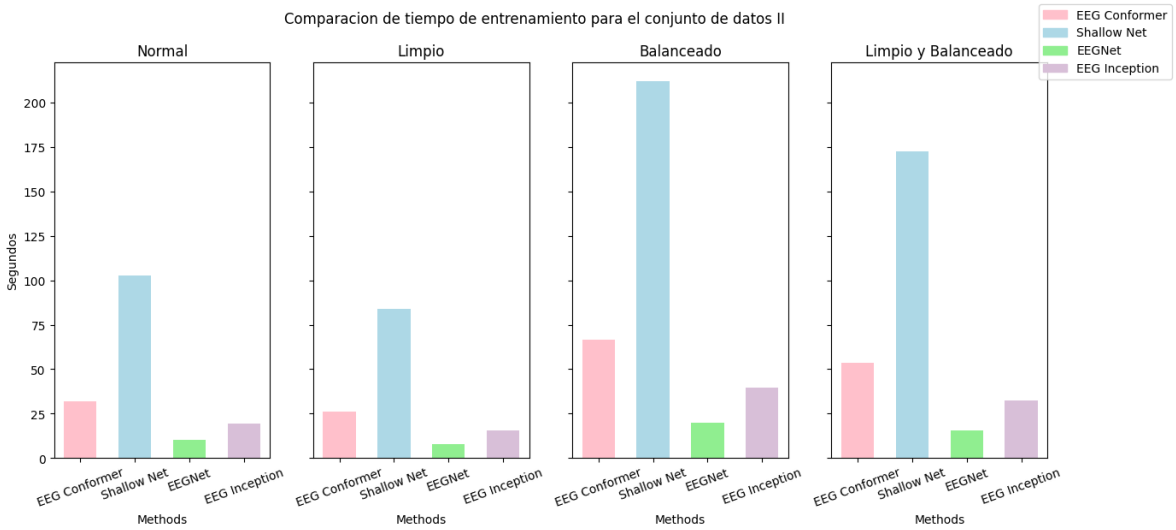


Figura 32: Tiempo de entrenamiento para *dataset* II, entrenando con 100 épocas.
Fuente: Elaboración propia

Los tiempos empleados por los cuatro métodos de entrenamiento son los esperados, ya que el balanceo de datos aplicado aumenta los datos aproximadamente al doble del tamaño original. Esto claramente indica un aumento en el tiempo de entrenamiento al utilizar el método balanceado, que en este caso, como se puede ver en la Tabla 18, es del 34.13 % para el dataset II. Por otro lado, el hecho de eliminar datos con *autoreject* disminuye el tiempo de entrenamiento al emplear una menor cantidad de datos para dicho proceso. La eliminación del 12.89 % de los datos se correlaciona con una disminución del 10.97 % en el tiempo de entrenamiento para este conjunto de datos.

Tabla 18: Diferencia porcentual en el tiempo de entrenamiento por época, en donde la diferencia negativa una disminución del tiempo de entrenamiento, y positiva, un aumento.

Metodos	Dataset I	Dataset II	Dataset III
Normal (Segundos)	0.195 ± 0.132	0.41 ± 0.364	0.031 ± 0.01
Limpio %	-2.165 ± 0.721	-10.975 ± 0.994	-0.923 ± 2.108
Balanceado %	37.88 ± 2.51	34.13 ± 1.307	39.88 ± 2.463
Limpio y Balanceado %	33.975 ± 2.208	24.235 ± 1.901	40.77 ± 1.831

Como resumen, para el *dataset* II, el modelo EEGNet con el método balanceado nuevamente ofrece el mejor rendimiento en términos de F1-Score y ROC, destacándose especialmente en la clasificación de señales EEG de usuarios con ELA. Sin embargo, en este caso, el método normal de EEGNet también muestra un rendimiento competitivo en términos de ROC. A pesar de las dificultades adicionales introducidas por la calidad variable de los datos y la eliminación de eventos durante el proceso de limpieza, EEGNet ha demostrado una robustez significativa. Otros modelos como Shallow Net y EEG Inception también muestran capacidades competitivas, aunque con una mayor variabilidad en su desempeño.

4.3. Resultados para el Dataset III

El *dataset* III se obtuvo leyendo señales de 25 sujetos utilizando el sistema Brain Invaders, el cual es similar al P300 speller, compartiendo una matriz de 6x6 en donde el sujeto debe concentrarse en una locación específica. Este conjunto de datos cuenta con la menor cantidad de estímulos totales, con tan solo 762 ± 2 por usuario, además de tener una frecuencia de 128 Hz, la mitad que los demás conjuntos.

4.3.1. F1-Score

En el *dataset* III, el modelo Shallow Net demuestra un rendimiento superior en comparación con otros modelos, alcanzando un F1-Score de 0.8122 ± 0.0537 cuando se utilizan datos limpios y balanceados. Este resultado resalta la eficacia del método de balanceo y limpieza en mejorar la capacidad del modelo para distinguir entre las clases, especialmente en un conjunto de datos con una cantidad limitada de ejemplos por sujeto.

El resto de los modelos, aunque también se benefician de los datos balanceados, no logran superar el rendimiento de Shallow Net. EEGNet, por ejemplo, alcanza un F1-Score máximo de 0.7889, lo que indica que, si bien es competitivo, todavía necesita una mayor cantidad de datos para optimizar su capacidad de clasificación. Esta observación sugiere que modelos como EEGNet y EEG Conformer son más sensibles a la cantidad de datos disponibles y pueden caer en el sobreajuste cuando los datos son limitados, es decir, tienden a predecir mayoritariamente la clase más representada (no P300) para obtener mejores resultados superficiales.

La Figura 33 presenta un gráfico boxplot que muestra el F1-Score para cada modelo utilizando el mejor método de entrenamiento. Se puede observar que, aunque todos los modelos muestran mejoras con datos limpios y balanceados, Shallow Net se destaca con una menor variabilidad en su rendimiento, lo cual es visible por la menor dispersión en el boxplot. Los puntos que se alejan del promedio, conocidos como *outliers*, también están representados en el gráfico.

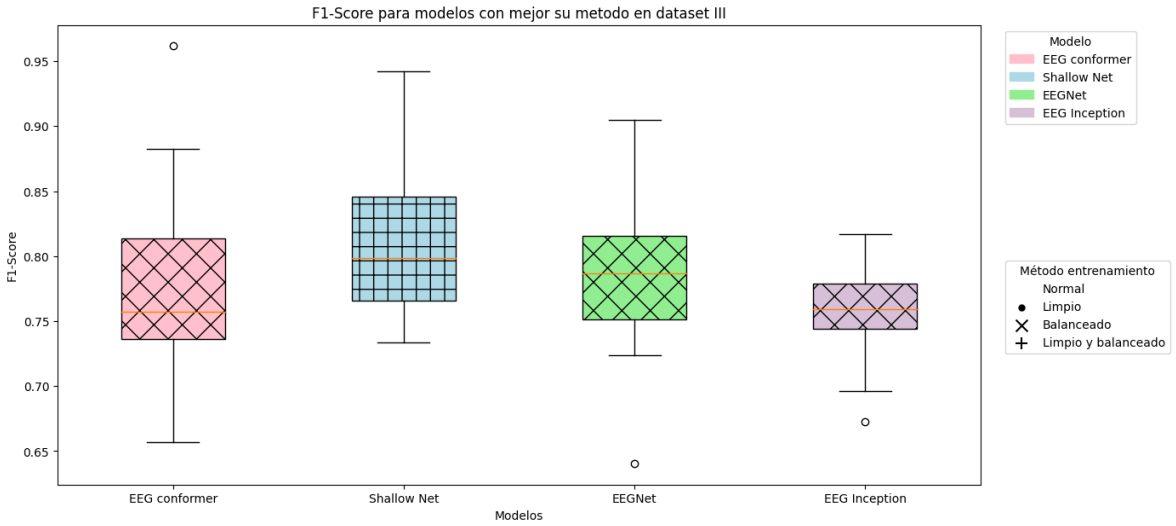


Figura 33: Gráfico boxplot en donde se muestra para cada modelo, el mejor método utilizado junto con su F1-Score. Para este gráfico se utilizan todos los sujetos disponibles en el conjunto de datos III, siendo aquellos que se alejan del promedio, también llamados *outliers*, visualizados como un punto. Cada modelo junto a su método fueron entrenados 5 veces.
Fuente: Elaboración propia

La Tabla 19 proporciona un resumen detallado de los F1-Scores promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* III. Se observa que los métodos de balanceo de datos consistentemente mejoran el rendimiento de los modelos. Para Shallow Net, el uso de datos limpios y balanceados resulta en el mejor desempeño, con un F1-Score de 0.8122 ± 0.0537, seguido de cerca por su rendimiento con datos balanceados solamente. EEGNet y EEG Conformer también muestran mejoras notables con estos métodos, aunque no alcanzan los niveles de Shallow Net. Para complementar estos resultados, se realizó un análisis del F1-Score por sujeto en la figura 45, encontrada en el anexo.

Tabla 19: F1-Score promedio para diferentes modelos y metodologías de entrenamiento en el *dataset* III

Métodos	**EEG Inception**	**EEG Net**	**Shallow Net**	**EEG Conformer**
Normal	0.7421 ± 0.0298	0.7432 ± 0.0288	0.7841 ± 0.0601	0.7407 ± 0.0284
Limpio	0.7484 ± 0.0329	0.7481 ± 0.0323	0.8050 ± 0.0541	0.7587 ± 0.0495
Balanceado	0.7580 ± 0.0328	0.7889 ± 0.0584	0.8102 ± 0.0567	0.7793 ± 0.0701
Limpio y Balanceado	0.7315 ± 0.1466	0.7867 ± 0.0439	$\mathbf{0.8122 \pm 0.0537}$	0.7746 ± 0.0736

4.3.2. ROC

Shallow Net con el método limpio y balanceado logra el mejor resultado, con un AUC de 0.7478 ± 0.1089, como se ve en la Figura 34. Este hallazgo es consistente con las observaciones de las otras métricas, donde Shallow Net muestra un desempeño competitivo en este *dataset*, a pesar de presentar la menor cantidad de datos por sujeto. En este conjunto de datos, se nota una marcada diferencia entre métodos de entrenamiento balanceados y no balanceados, especialmente para EEG Conformer y EEGNet, con una diferencia porcentual del 18.3 % y 15.3 % de AUC, respectivamente. Esto subraya la dependencia de un procesamiento de datos adecuado para el correcto funcionamiento del modelo, particularmente cuando los datos son escasos.

A continuación, se presenta la Tabla 20, que resume el rendimiento promedio de AUC para diferentes modelos y metodologías de entrenamiento en el *dataset* III. Se puede observar que los modelos balanceados consistentemente superan a sus contrapartes no balanceadas, lo que indica la importancia de manejar el desbalance de clases en este *dataset*. El modelo EEG Inception, aunque mejora ligeramente con el balanceo, aún muestra un rendimiento inferior en comparación con los otros modelos, sugiriendo que su arquitectura podría no ser la más adecuada para este conjunto de datos específico.

Tabla 20: AUC promedio para diferentes modelos y metodologías de entrenamiento en el dataset III

Métodos	**EEG Inception**	**EEG Net**	**Shallow Conv**	**EEG Conformer**
Normal	0.6039 ± 0.0692	0.6404 ± 0.0927	0.6759 ± 0.1431	0.5504 ± 0.0910
Limpio	0.6106 ± 0.0644	0.6506 ± 0.0772	0.7020 ± 0.1330	0.5679 ± 0.1128
Balanceado	0.6593 ± 0.0892	0.7466 ± 0.0908	0.7313 ± 0.1173	0.7366 ± 0.1000
Limpio y Balanceado	0.6570 ± 0.0765	0.7147 ± 0.1049	$\mathbf{0.7478 \pm 0.1089}$	0.7092 ± 0.1199

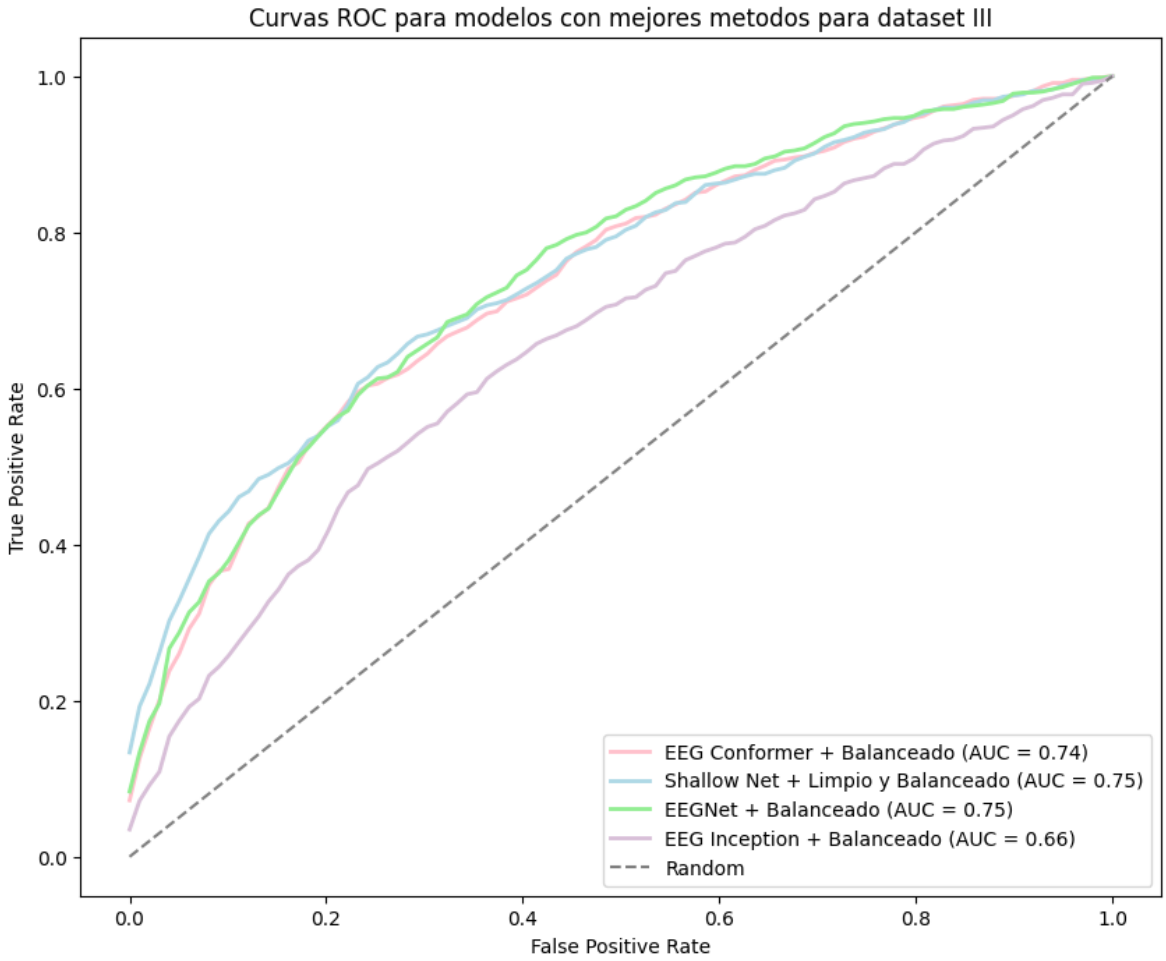


Figura 34: Curva ROC, graficando el modelo, junto al método de entrenamiento que mayor ROC consigue para el dataset III.

Fuente: Elaboración propia

La Figura 34 muestra la curva ROC del modelo con el mejor AUC en el *dataset* III, ilustrando visualmente el rendimiento de Shallow Net bajo el método limpio y balanceado. Esta representación gráfica refuerza los resultados numéricos, mostrando claramente cómo este método de entrenamiento proporciona una mejora significativa en la capacidad de discriminación del modelo. Para complementar estos resultados, se realizó un análisis de la curva ROC por sujeto en la figura 48, encontrada en el anexo.

En resumen, los resultados obtenidos resaltan la importancia de utilizar métodos de balanceo de datos en el *dataset* III, donde la cantidad limitada de datos por sujeto puede afectar negativamente el rendimiento de los modelos. La combinación de limpieza y balanceo de datos no solo mejora los resultados de Shallow Net, sino también los de EEGNet y EEG Conformer, aunque en menor medida.

4.3.3. Matriz de Confusión

Para las matrices de confusión presentadas en la imagen 35, se observa un claro problema en los métodos normal y limpio para los modelos EEG Conformer, EEG Inception y EEGNet, ya que la cantidad de predicciones hechas para la clase P300 es prácticamente nula. Este comportamiento, que lleva a los tres modelos a no superar una sensibilidad del 1 %, se puede atribuir a la baja cantidad de datos presentes. Los modelos fallan en aprender patrones significativos en los datos, llevándolos a un óptimo local en donde se predice solamente la clase no P300, que tiene una representación cinco veces mayor que la P300.

Con el método balanceado, el modelo EEG Conformer pasa de no detectar ninguna señal como P300 en el método normal a tener una sensibilidad del 48 %. Si bien no es un resultado óptimo, es una mejora considerable en comparación con los resultados anteriores. Siguiendo la misma línea, todos los modelos ven mejoras al utilizar el método de balanceo. A diferencia de los *datasets* anteriores, el método limpio y balanceado mejora los resultados para EEG Conformer y EEGNet, manteniendo los resultados para los demás modelos. Esto indica que para este conjunto de datos, la interpolación de canales y la limpieza de señales ruidosas con *autoreject* previa al balanceo de datos puede entregar resultados positivos para ciertos modelos.

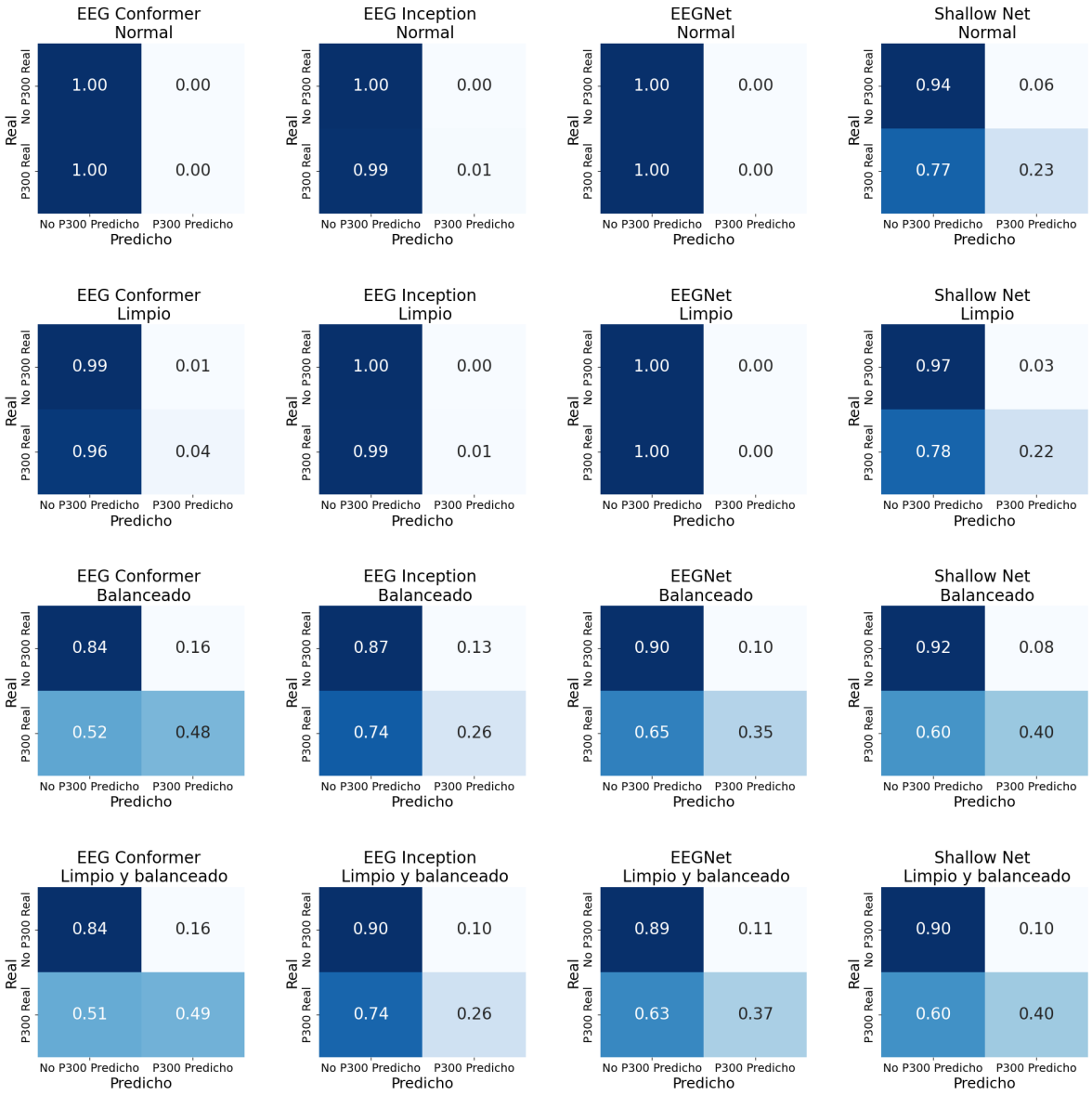


Figura 35: Matriz de confusión para el *dataset* III, en donde las columnas representan los modelos entrenados, y las filas los métodos utilizados.
Fuente: Elaboración propia.

4.3.4. Tiempo de Entrenamiento

En el conjunto de datos III, el modelo EEG Conformer resulta ser el más costoso en términos de tiempo de entrenamiento, aunque la diferencia con Shallow Net es mínima. EEG Conformer no solo aplica convoluciones profundas y separables, sino que también utiliza módulos de atención, lo que añade complejidad y tiempo de cómputo. Sin embargo, su tiempo de entrenamiento es comparable al de Shallow Net, un modelo que simplemente aplica capas

convolucionales 2D.

La Figura 36 ilustra el tiempo de entrenamiento para cada modelo en el conjunto de datos III, utilizando 100 épocas. Se observa que, aunque EEG Conformer tiene un tiempo de entrenamiento ligeramente superior, la diferencia no es significativa en comparación con Shallow Net. Esto sugiere que la implementación de módulos de atención en EEG Conformer no necesariamente incrementa drásticamente el tiempo de entrenamiento, lo que puede ser ventajoso en aplicaciones prácticas donde se busca un equilibrio entre complejidad del modelo y eficiencia temporal.

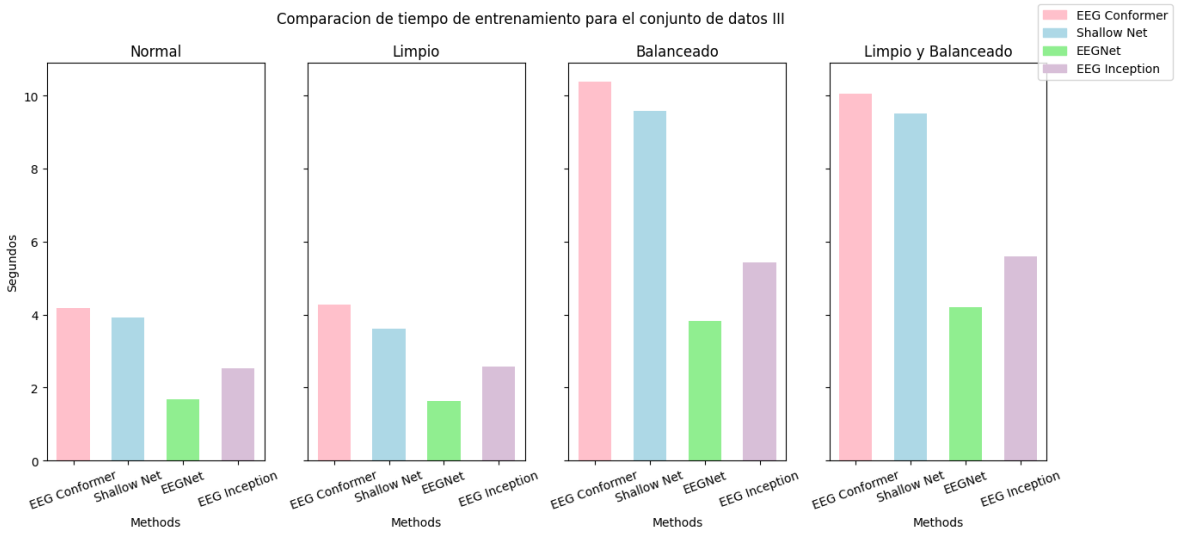


Figura 36: Tiempo de entrenamiento para *dataset* tres, entrenando con 100 épocas.
Fuente: Elaboración propia

Los datos presentados en la Tabla 21 muestran que el método balanceado incrementa el tiempo de entrenamiento para todos los modelos, especialmente para Shallow Net y EEG Conformer. Esto se debe al aumento en el tamaño del conjunto de datos efectivo tras el balanceo, lo cual requiere más recursos computacionales y tiempo para procesar. Sin embargo, este incremento en el tiempo de entrenamiento se justifica por las mejoras en las métricas de desempeño, como el ROC y el F1-Score, que se han observado en secciones anteriores.

Tabla 21: Tiempo de ejecución por época, promediando los sujetos, para diferentes modelos y metodologías de entrenamiento (en segundos) en el *dataset* III

Métodos	**EEG Inception**	**EEG Net**	**Shallow Net**	**EEG Conformer**
Normal	2.53 ± 0.20	$\mathbf{1.69 \pm 0.09}$	3.92 ± 0.06	4.18 ± 0.39
Limpio	2.57 ± 0.19	$\mathbf{1.64 \pm 0.08}$	3.61 ± 0.55	4.28 ± 0.67
Balanceado	5.42 ± 0.35	$\mathbf{3.82 \pm 0.30}$	9.57 ± 0.22	10.38 ± 0.75
Limpio y Balanceado	5.59 ± 0.63	$\mathbf{4.20 \pm 0.61}$	9.50 ± 0.64	10.05 ± 0.92

Como resumen para el *dataset* III, Shallow Net con el método limpio y balanceado se destaca como el modelo con el mejor rendimiento en términos de F1-Score y ROC, a pesar de la menor cantidad de datos por sujeto. Esto demuestra su capacidad para manejar datos limitados y balanceados de manera efectiva. EEG Conformer y EEGNet también muestran mejoras significativas con métodos balanceados, aunque en menor medida. En términos de tiempo de entrenamiento, EEGNet es el modelo más eficiente, mientras que EEG Conformer, a pesar de su complejidad, tiene tiempos de entrenamiento comparables a Shallow Net. Estos resultados subrayan la importancia de utilizar técnicas de balanceo de datos y considerar tanto el rendimiento como la eficiencia temporal al seleccionar modelos para aplicaciones con datos limitados y desbalanceados.

4.4. Experimentos Batch size 512

Con el fin de robustecer los experimentos realizados, se volvieron a entrenar todos los modelos con sus respectivos métodos de entrenamiento para los 43 sujetos estudiados, evaluando cómo varían los resultados al utilizar un tamaño de *batch* mayor. Específicamente, se compararon los tamaños de *batch* de 256 y 512. La elección de un tamaño de *batch* de 512 incrementa significativamente el requerimiento computacional, dado que se almacena el doble de datos en memoria RAM, lo cual podría limitar su uso en entornos con recursos de cómputo restringidos.

Para evaluar el impacto de este cambio en el rendimiento de los modelos, se generaron gráficos boxplot para las métricas F1-Score y ROC. Cada gráfico muestra la diferencia en el rendimiento de los modelos entrenados con *batch size* de 512 frente a aquellos entrenados con *batch size* de 256. Un valor positivo en los gráficos indica una mejoría en las métricas al utilizar un *batch size* de 512, mientras que un valor negativo sugiere que un *batch size* de 256 produjo mejores resultados. Esta comparación proporciona una visión detallada de cómo el tamaño del *batch* influye en el desempeño de los modelos, especialmente en el contexto de un problema de desbalanceo de clases.

4.4.1. Dataset I

F1-Score Para el *dataset* I, el análisis del F1-Score revela que la mayoría de los modelos muestran una tendencia a mejorar con un *batch size* de 256, aunque la variabilidad es notable. EEGNet es el único modelo que consistentemente se beneficia del *batch size* de 256, mientras que otros modelos, como EEG Conformer y EEG Inception, muestran una variabilidad que sugiere la necesidad de ajustes personalizados.

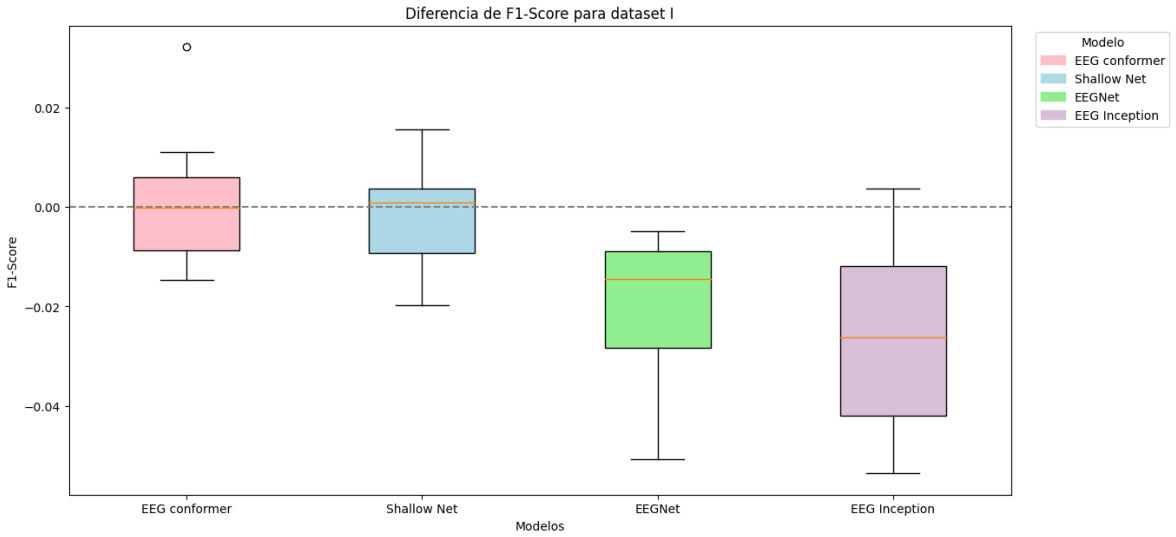


Figura 37: Gráfico boxplot de F1-Score para *batch sizes* 256 vs 512 en Dataset I. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

ROC En términos de ROC, los resultados muestran que EEGNet sigue favoreciendo el *batch size* de 256, mientras que otros modelos presentan variabilidad. EEG Inception, similar a F1-Score, también muestra una ligera preferencia por 256, aunque con ciertas excepciones. Esta consistencia en el rendimiento con *batch size* de 256 refuerza la conclusión de que este tamaño de *batch* puede ser óptimo en ciertos casos, pero también resalta la importancia de considerar ajustes específicos.

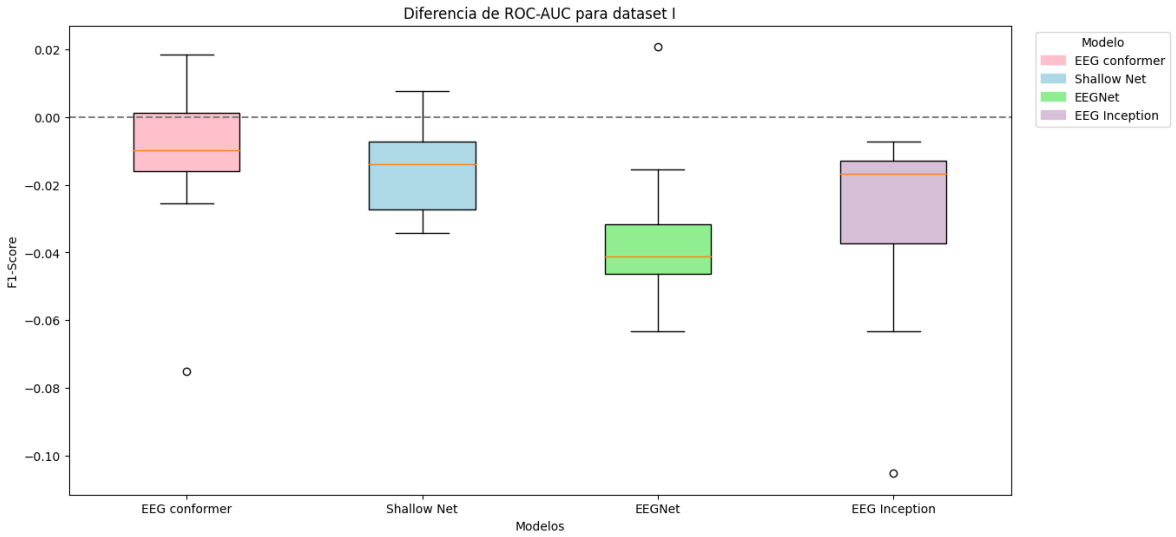


Figura 38: Gráfico boxplot de AUC para *batch sizes* 256 vs 512 en Dataset I. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

4.4.2. Dataset II

F1-Score Para el *dataset* II, los resultados del F1-Score muestran una ligera preferencia por el *batch size* de 512 en la mayoría de los modelos, aunque las diferencias no son significativas. La variabilidad en el desempeño indica que ambos *batch sizes* pueden ser válidos dependiendo de las características del *dataset*.

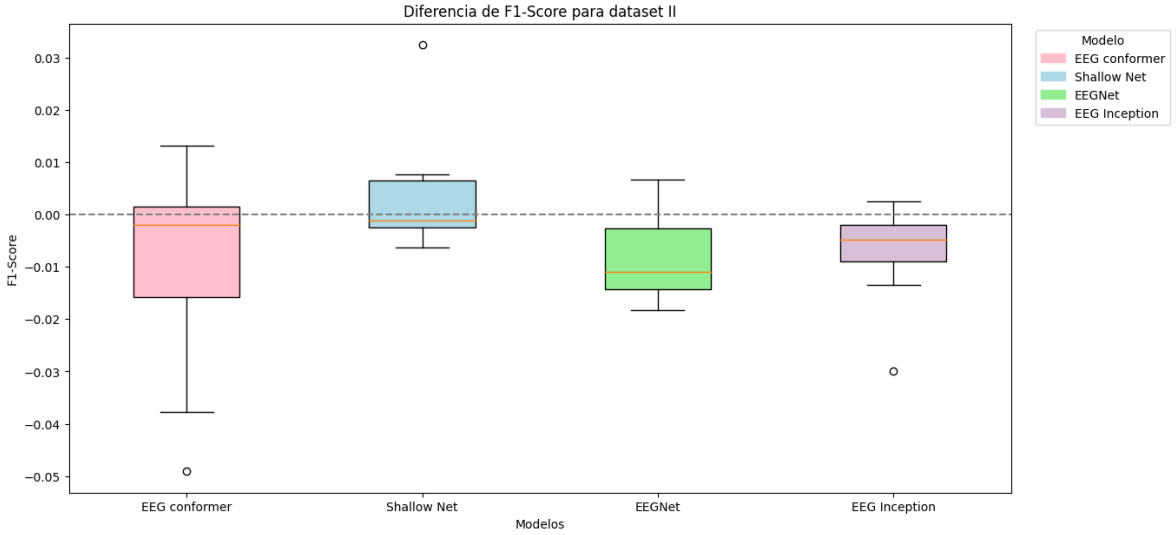


Figura 39: Gráfico boxplot de F1-Score para *batch sizes* 256 vs 512 en Dataset II. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

ROC El análisis de ROC para el *dataset* II muestra un patrón similar, con una ligera inclinación hacia el *batch size* de 512 para algunos modelos. Esta consistencia con F1-Score subraya la importancia de considerar tanto la precisión como la capacidad de separación del modelo al elegir el *batch size*.

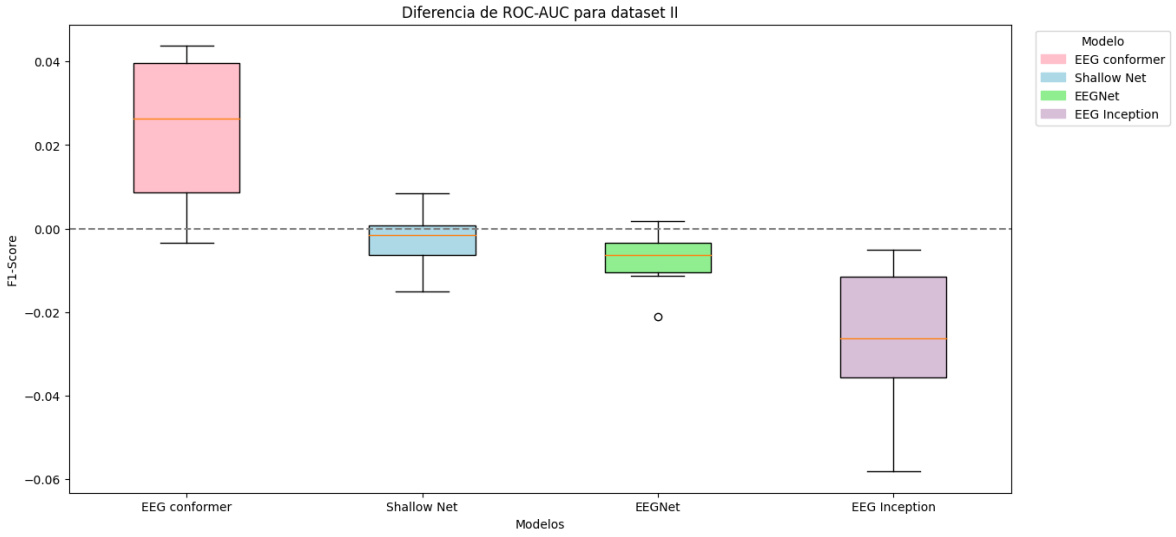


Figura 40: Gráfico boxplot de AUC para *batch sizes* 256 vs 512 en Dataset II. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

4.4.3. Dataset III

F1-Score En el *dataset* III, el F1-Score revela que el *batch size* de 256 es generalmente preferido por la mayoría de los modelos, aunque hay algunas excepciones. La variabilidad en el rendimiento muestra que, aunque 256 puede ser ventajoso en términos generales, las características del sujeto pueden influir significativamente en el desempeño.

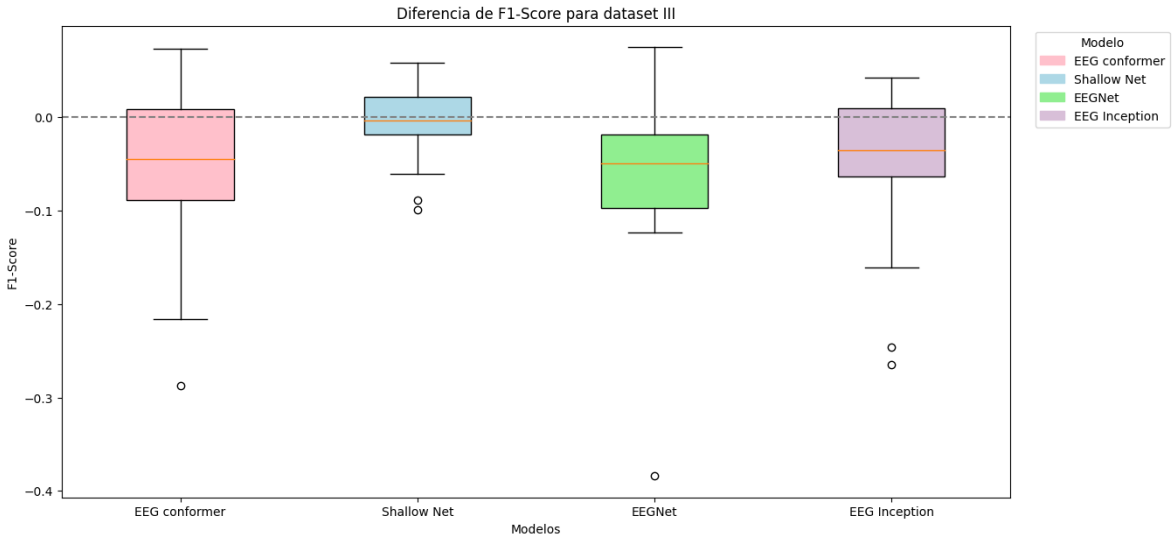


Figura 41: Gráfico boxplot de F1-Score para *batch sizes* 256 vs 512 en Dataset III. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

ROC Similar a F1-Score, el ROC también muestra una inclinación hacia el *batch size* de 256 en la mayoría de los modelos, aunque con algunas variaciones. Esta tendencia consistente refuerza la utilidad del *batch size* de 256, pero también indica que ajustes específicos pueden ser necesarios para maximizar el rendimiento.

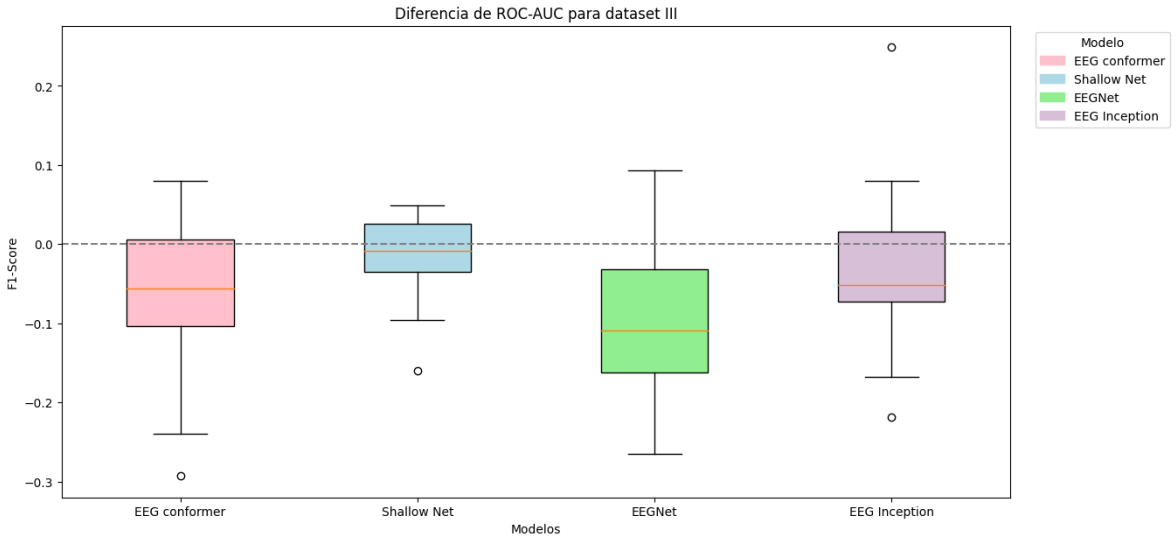


Figura 42: Gráfico boxplot de AUC para *batch sizes* 256 vs 512 en Dataset III. Cada modelo fue entrenados 5 veces.

Fuente: Elaboración propia.

En conclusión, la evaluación de diferentes tamaños de batch (256 y 512) mostró que, en general, un *batch size* de 256 proporciona un mejor equilibrio entre rendimiento y requerimientos computacionales para la mayoría de los modelos y datasets analizados. Sin embargo, algunas excepciones y variaciones específicas sugieren la necesidad de ajustes personalizados para optimizar el desempeño en contextos particulares.

CAPÍTULO 5
CONCLUSIONES

5.1. Conclusiones generales

En esta tesis, se realizó una comparación exhaustiva de cuatro modelos de aprendizaje profundo (Shallow ConvNet, EEGNet, EEG-Inception y EEG Conformer) aplicados a tres conjuntos de datos distintos, utilizando cuatro métodos de entrenamiento diferentes. Los experimentos se llevaron a cabo en *Google Colab*, asegurando condiciones homogéneas de hardware para cada modelo y conjunto de datos. Se analizaron los resultados de 43 sujetos, con una variabilidad significativa en términos de experiencia previa con sistemas BCI, diagnósticos de ELA, edad y sexo, obteniendo una muestra representativa de los posibles usuarios.

Rendimiento de los Modelos

Los resultados obtenidos de los 43 sujetos, que incluían tanto a aquellos con experiencia previa en BCI como a diagnosticados con ELA, indican que EEGNet destacó como el modelo más eficiente. Este modelo mostró un desempeño superior en términos de tiempo de entrenamiento, F1-Score y ROC-AUC en la mayoría de los sujetos analizados. En particular, EEGNet utilizando el método balanceado obtuvo un F1-Score y ROC-AUC superiores en comparación con otros modelos y métodos de entrenamiento. Estos hallazgos sugieren que la arquitectura EEGNet, con su reducido número de parámetros, es particularmente adecuada para la detección de potenciales P300 en diversas condiciones, especialmente cuando el conjunto de datos incluye una gran cantidad de estímulos.

Importancia del Balanceo de Datos

El método de balanceo de datos demostró ser esencial para mejorar el desempeño de todos los modelos estudiados, resaltando la importancia de abordar el desbalance de clases en la preparación de datos. En particular, para el modelo EEG Conformer, el beneficio del balanceo de datos es evidente en el conjunto de datos III. Utilizando el método normal, el modelo no predice ninguna señal como potencial P300, mientras que con el método limpio y balanceado, se obtiene una sensibilidad del 49 %. Esto convierte a esta combinación de modelo y método de entrenamiento en la que presenta las mejores métricas para este conjunto de datos. Estos resultados corroboran la propuesta de [Song *et al.*, 2022], que recomienda fuertemente el uso de algoritmos de aumento de datos como el S&R utilizado en estos experimentos.

Impacto de la Limpieza e Interpolación de Datos

Por otro lado, la limpieza e interpolación de datos utilizando *autoreject* no tuvo un impacto significativo en las métricas obtenidas. Aunque el tiempo de entrenamiento fue menor debido a la eliminación de datos, la duración de la ejecución del algoritmo de limpieza es lo suficientemente alta como para contrarrestar esta reducción en el tiempo de entrenamiento. Esto sugiere que, si bien la limpieza de datos puede mejorar la calidad de los mismos, su costo computacional debe ser considerado.

Batch Size

El tamaño de *batch size* de 256 entregó resultados mejores para la mayoría de los sujetos estudiados en comparación con un *batch size* de 512. Al proporcionar actualizaciones de gradientes más frecuentes, el modelo tiene una mejor capacidad para escapar de mínimos locales. Esto es especialmente evidente en el *dataset* III, el cual es el único que muestra mejoras significativas en ambas métricas estudiadas y que, coincidentemente, presenta el problema más grave de óptimo local y sobreajuste, como se observó en la matriz de confusión 35. El cambio en la cantidad de datos por *batch* parece ser crucial en este contexto.

Hay excepciones, como el modelo EEG Conformer para el *dataset* II, en el cual la mayoría de los sujetos obtienen mejores resultados con un *batch size* de 512. Esto indica la gran variabilidad entre sujetos al analizar los resultados de estos experimentos, donde las diferencias entre ondas cerebrales obtenidas de distintas personas generan complicaciones al comparar modelos bajo las mismas condiciones.

5.2. Cumplimiento de objetivos

La completitud de esta memoria se puede estimar en base al cumplimiento de cada uno de los objetivos específicos, los cuales, a su vez, ayudan al cumplimiento del objetivo general descrito en la sección 1.1. A continuación, se presentan todos los resultados obtenidos para cada uno de ellos:

- **Estudiar el estado del arte para modelos de *Deep Learning* aplicados para la detección de potenciales evocados P300.**

  En la sección 3.1, se presentan 4 modelos de aprendizaje profundo creados para decodificar señales cerebrales extraídas con EEG. Además, se investigaron técnicas de preprocesamiento de datos utilizadas en la academia para ver el comportamiento de los modelos al utilizarlas.

- **Definir y diseñar una metodología estándar para la justa comparación entre los modelos estudiados.**

En la sección 3.7, se presenta la metodología de comparación, la cual se basa en analizar los modelos en la mayor cantidad de escenarios posibles, variando el preprocesamiento de datos utilizado, para sujetos de prueba heterogéneos. De esta manera se obtienen resultados generalizados para los modelos a comparar. Dicha metodología puede ser modificada según las necesidades de estudios futuros, aplicando diferentes modelos y técnicas de preprocesamiento.

- **Evaluar cada modelo en base a la metodología y obtener conclusiones respecto a sus resultados.**

  En el capítulo 4, se ejecutó la metodología de comparación en condiciones equitativas para todos los modelos, obteniendo los resultados para métricas como tiempo de entrenamiento por época, *accuracy* y F1-Score, visualizándolas de manera intuitiva para el correcto entendimiento de las conclusiones obtenidas.

- **Identificar elementos relevantes que asocien la arquitectura de un modelo y sus resultados.**

  Tanto en el capítulo 4 como en el capítulo 5, se analiza el efecto de las arquitecturas sobre los resultados obtenidos, en donde se ve una clara ventaja de arquitecturas livianas, como EEGNet con sus convoluciones por profundidad, sobre arquitecturas más necesitadas de datos como EEG Conformer basada en *transformers*.

Dado que se cumplieron los objetivos específicos en su totalidad, se puede declarar el cumplimiento del objetivo general de esta investigación, el cual corresponde a: **Comparar bajo las mismas condiciones distintos algoritmos basados en Deep Learning para clasificación/detección de potenciales evocados P300 utilizados en una interfaz cerebro-computador (BCI) en el contexto del problema P300 *Speller*.**

5.3. Trabajo futuro

Futuras investigaciones podrían enfocarse en la comparación de un mayor número de técnicas de preprocesamiento y balanceo de datos para mejorar aún más el rendimiento de los modelos. La inclusión de métodos avanzados de limpieza de señales, técnicas de aumento de datos y algoritmos de selección de características podría proporcionar mejoras significativas en la precisión de las clasificaciones. También es fundamental explorar nuevas arquitecturas de redes neuronales que puedan ofrecer mejoras adicionales en términos de eficiencia y precisión, como las redes neuronales recurrentes (RNN), redes neuronales convolucionales de mayor profundidad (CNN) y modelos híbridos que combinan diferentes enfoques.

Una búsqueda exhaustiva de los hiperparámetros óptimos para cada modelo es igualmente importante. La optimización de hiperparámetros mediante técnicas como la búsqueda en cuadrícula, la búsqueda aleatoria y los algoritmos genéticos puede ayudar a encontrar configuraciones que maximicen el rendimiento de los modelos.

Además, sería beneficioso realizar estudios con un mayor número de sujetos y diversidad en las condiciones clínicas para validar la generalización de los resultados obtenidos en este trabajo. Incluir una población más diversa, tanto en términos de demografía como de condiciones clínicas, puede proporcionar una mejor comprensión de la robustez de los modelos y su aplicabilidad en escenarios del mundo real.

Otra línea de investigación interesante es la comparación del rendimiento de los modelos para predicciones *inter-subject*, donde modelos basados en *transformers*, como EEG Conformer, podrían beneficiarse de un preentrenamiento. El preentrenamiento en grandes conjuntos de datos y la transferencia de aprendizaje pueden mejorar la capacidad del modelo para generalizar entre diferentes sujetos, lo cual es crucial para aplicaciones prácticas en entornos clínicos.

Además, investigar la integración de técnicas de aprendizaje activo y aprendizaje por refuerzo podría ofrecer nuevas perspectivas para la mejora continua de los modelos. Estos enfoques pueden permitir que los modelos aprendan y se adapten en tiempo real a nuevas señales y condiciones, lo que es particularmente útil en aplicaciones de monitoreo y diagnóstico continuo.

Finalmente, la colaboración interdisciplinaria con expertos en neurociencia, ingeniería biomédica y profesionales de la salud puede proporcionar una visión más completa y multidimensional para abordar los desafíos en el análisis de señales EEG y mejorar la aplicabilidad clínica de los modelos desarrollados.

ANEXOS

F1-score por Dataset

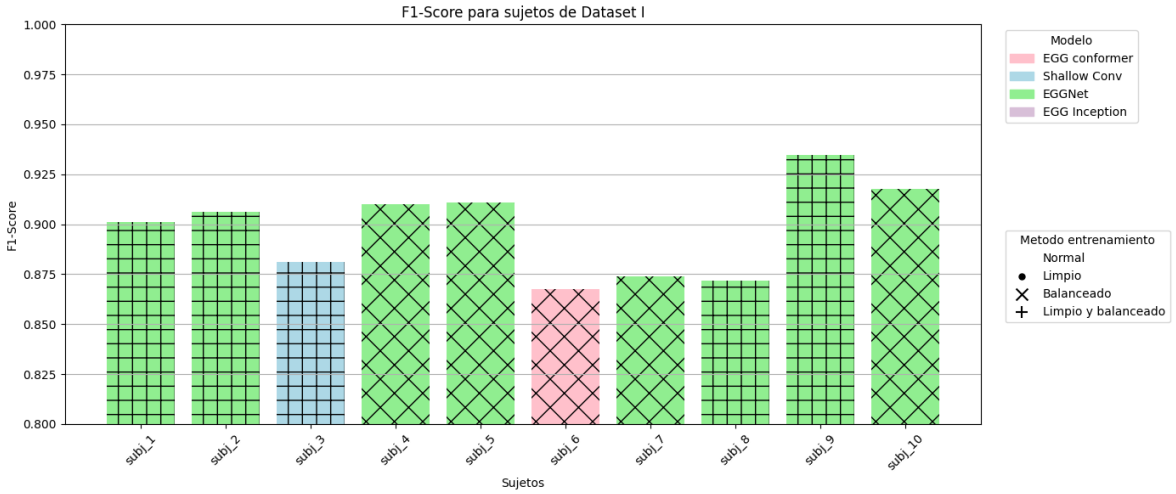


Figura 43: F1-score para cada sujetos en el dataset I. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

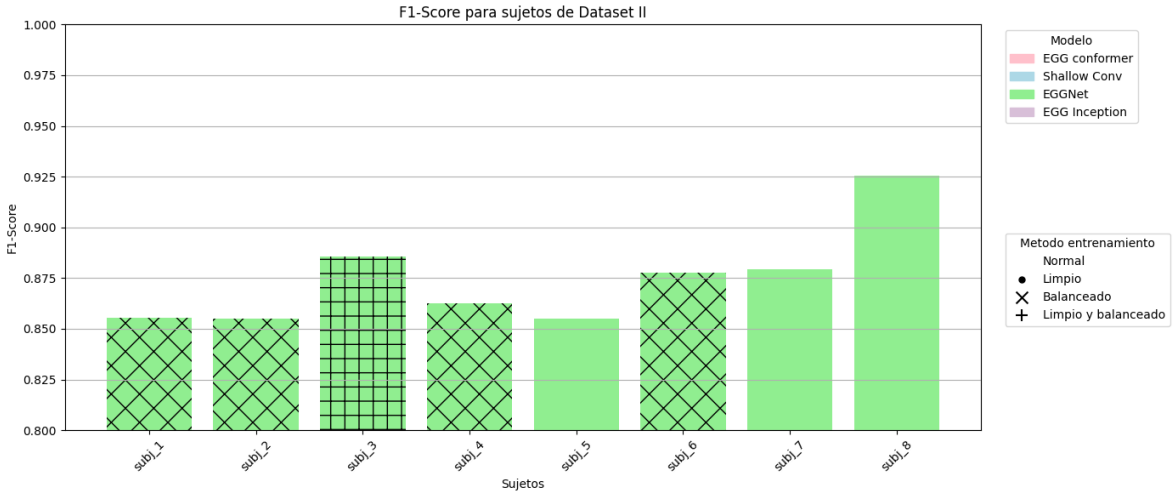


Figura 44: F1-score para cada sujetos en el dataset II. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

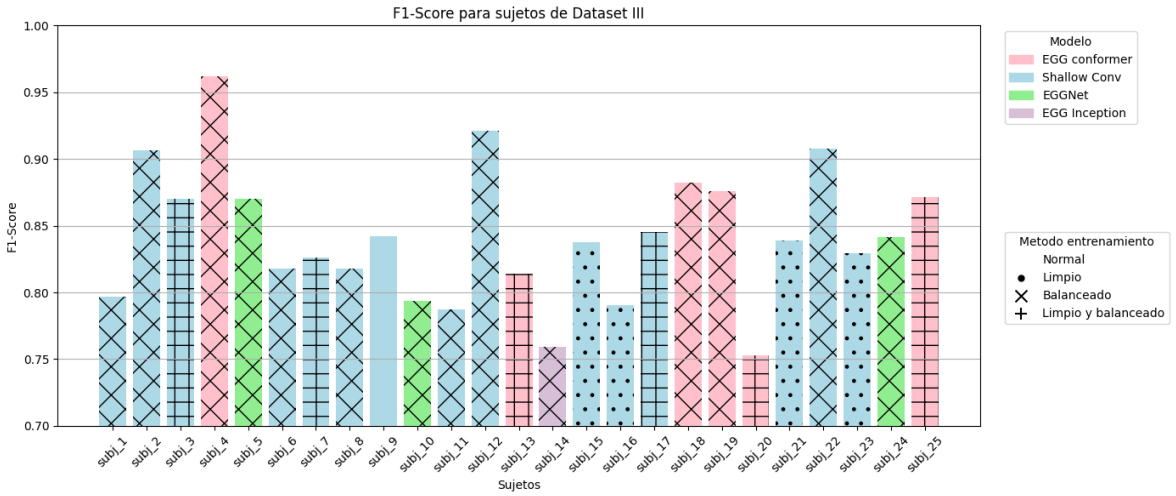


Figura 45: F1-score para cada sujetos en el dataset III. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

ROC por Dataset

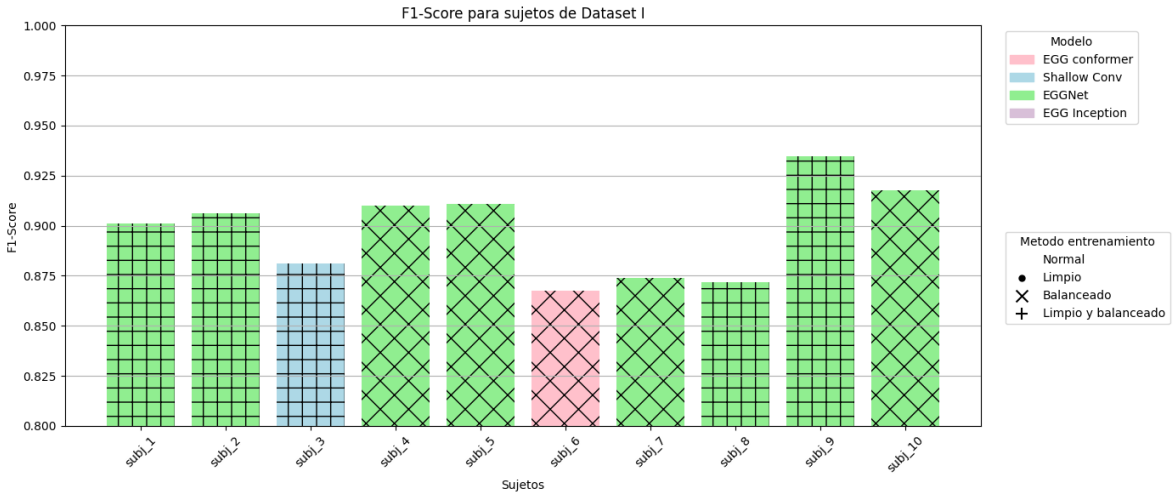


Figura 46: AUC para cada sujetos en el dataset I. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

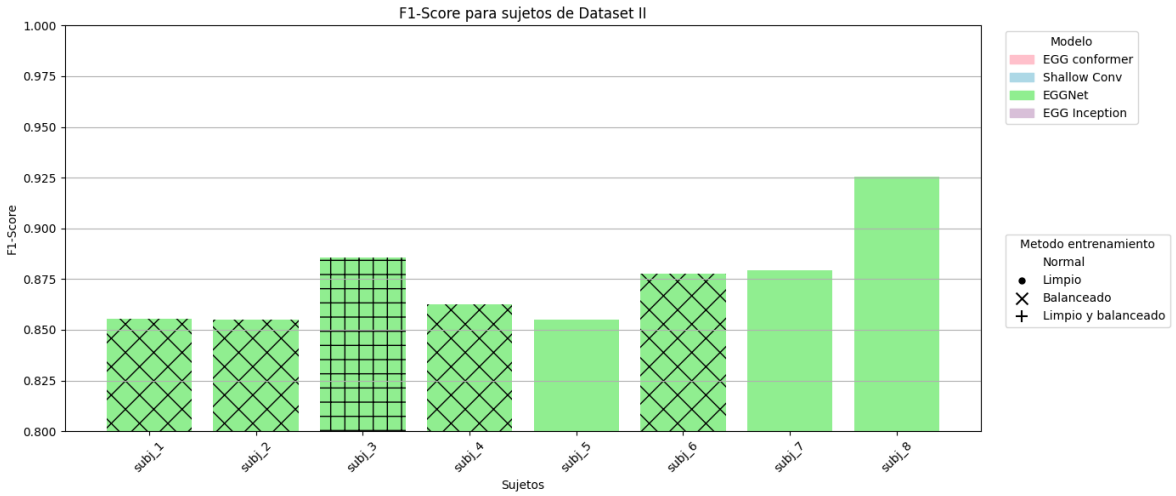


Figura 47: AUC para cada sujetos en el dataset II. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

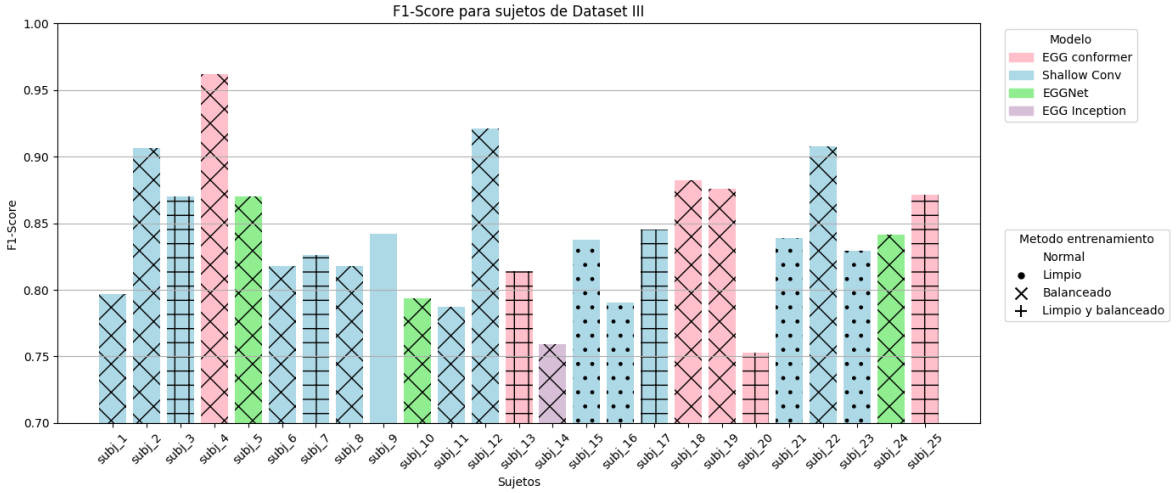


Figura 48: AUC para cada sujetos en el dataset III. Utilizando la mejor mezcla de modelo y método de entrenamiento para cada uno. Los métodos y modelos se pueden encontrar en las leyendas al lado derecho del gráfico.

Fuente: Elaboración propia

Diferencia porcentual de métricas

Tabla 22: Diferencia porcentual de *F1-Score* entre método de entrenamiento normal y el resto. Una diferencia positiva indica una mejora respecto al método normal

Dataset	Métodos	EEG Conformer	Shallow Net	EEG Net	EEG Inception
Dataset I	Normal	82.43	86.92	84.59	84.19
	Limpio	-2.21 %	-2.38 %	-0.57 %	-1.24 %
	Balanceado	4.19 %	1.21 %	5.14 %	1.39 %
	Limpio y Balanceado	2.73 %	0.32 %	4.52 %	0.49 %
Dataset II	Normal	78.74	82.36	85.97	81.67
	Limpio	-0.97 %	-1.65 %	-2.54 %	-1.33 %
	Balanceado	3.98 %	1.55 %	1.31 %	2.91 %
	Limpio y Balanceado	2.73 %	-0.36 %	-1.12 %	0.47 %
Dataset III	Normal	74.07	78.41	74.32	74.21
	Limpio	2.43 %	2.67 %	0.66 %	0.85 %
	Balanceado	5.21 %	3.33 %	6.15 %	2.14 %
	Limpio y Balanceado	4.58 %	3.58 %	5.85 %	-1.43 %

Tabla 23: Diferencia porcentual de AUC entre método de entrenamiento normal y el resto. Una diferencia positiva indica una mejora respecto al método normal

Dataset	Métodos	EEG Conformer	Shallow Net	EEG Net	EEG Inception
Dataset I	Normal	81.29	85.8	89.62	84.03
	Limpio	-1.21 %	-1.31 %	-1.93 %	-3.46 %
	Balanceado	**5.22 %**	3.31 %	2.11 %	0.39 %
	Limpio y Balanceado	5.41 %	1.93 %	1.54 %	0.01 %
Dataset II	Normal	72.3	78.16	85.74	78.71
	Limpio	0.98 %	-5.08 %	-5.31 %	-2.96 %
	Balanceado	**3.26 %**	2.33 %	-0.3 %	2.31 %
	Limpio y Balanceado	-3.4 %	-4.59 %	-2.88 %	-3.4 %
Dataset III	Normal	55.04	67.59	64.04	60.39
	Limpio	3.18 %	3.86 %	1.59 %	1.11 %
	Balanceado	**33.83 %**	8.2 %	16.58 %	9.17 %
	Limpio y Balanceado	28.85 %	10.64 %	11.6 %	8.79 %

REFERENCIAS BIBLIOGRÁFICAS

[Agarap, 2018] Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.

[Aricò *et al.*, 2014] Aricò, P., Aloise, F., Schettini, F., Salinari, S., Mattia, D., y Cincotti, F. (2014). Influence of p300 latency jitter on event related potential-based brain–computer interface performance. *Journal of neural engineering*, 11(3):035008.

[Aristimunha *et al.*, 2023] Aristimunha, B., Carrara, I., Guetschel, P., Sedlar, S., Rodrigues, P., Sosulski, J., Narayanan, D., Bjareholt, E., Quentin, B., Schirrmeister, R. T., Kalunga, E., Darmet, L., Gregoire, C., Abdul Hussain, A., Gatti, R., Goncharenko, V., Thielen, J., Moreau, T., Roy, Y., Jayaram, V., Barachant, A., y Chevallier, S. (2023). Mother of all BCI Benchmarks.

[Bishop y Nasrabadi, 2006] Bishop, C. M. y Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volumen 4. Springer.

[Boulila *et al.*, 2024] Boulila, W., Alshanqiti, E., Alzahem, A., Koubaa, A., y Mlaiki, N. (2024). An effective weight initialization method for deep learning: Application to satellite image classification. *Expert Systems with Applications*, p. 124344.

[Brunner *et al.*, 2008] Brunner, C., Leeb, R., Müller-Putz, G., Schlögl, A., y 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.

[Cedarbaum *et al.*, 1999] Cedarbaum, Jesse M and Stambler, Nancy and Malta, Errol and Fuller, Cynthia and Hilt, Dana and Thurmond, Barbara and Nakanishi, Arline and Bdnf Als Study Group and 1A complete listing of the BDNF Study Group and others (1999). The alsfrs-r: a revised als functional rating scale that incorporates assessments of respiratory function. *Journal of the neurological sciences*, 169(1-2):13–21.

[Chatrian *et al.*, 1985] Chatrian, G. E., Lettich, E., y Nelson, P. L. (1985). Ten percent electrode system for topographic studies of spontaneous and evoked eeg activities. *American Journal of EEG technology*, 25(2):83–92.

[Farwell y Donchin, 1988] Farwell, L. A. y Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70:510–523.

[Gramfort *et al.*, 2013] Gramfort, Alexandre and Luessi, Martin and Larson, Eric and Engemann, Denis A and Strohmeier, Daniel and Brodbeck, Christian and Goj, Roman and Jas, Mainak and Brooks, Teon and Parkkonen, Lauri and others (2013). Meg and eeg data analysis with mne-python. *Frontiers in neuroscience*, 7:70133.

[Howard *et al.*, 2017] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., y Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

[Ioffe y Szegedy, 2015] Ioffe, S. y Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. En *International conference on machine learning*, pp. 448–456. pmlr.

[Jas *et al.*, 2017] Jas, M., Engemann, D. A., Bekhti, Y., Raimondo, F., y Gramfort, A. (2017). Autoreject: Automated artifact rejection for meg and eeg data. *NeuroImage*, 159:417–429.

[Jiang *et al.*, 2019] Jiang, X., Bian, G.-B., y Tian, Z. (2019). Removal of artifacts from eeg signals: a review. *Sensors*, 19(5):987.

[Krusienski *et al.*, 2006] Krusienski, D. J., Sellers, E. W., Cabestaing, F., Bayoudh, S., McFarland, D. J., Vaughan, T. M., y Wolpaw, J. R. (2006). A comparison of classification techniques for the p300 speller. *Journal of neural engineering*, 3(4):299.

[Lawhern *et al.*, 2018] Lawhern, V. J., Solon, A. J., Waytowich, N. R., Gordon, S. M., Hung, C. P., y Lance, B. J. (2018). Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013.

[Liu *et al.*, 2018] Liu, M., Wu, W., Gu, Z., Yu, Z., Qi, F. F., y Li, Y. (2018). Deep learning based on batch normalization for p300 signal detection. *Neurocomputing*, 275:288–297.

[Lotte, 2015] Lotte, F. (2015). Signal processing approaches to minimize or suppress calibration time in oscillatory activity-based brain–computer interfaces. *Proceedings of the IEEE*, 103(6):871–890.

[Luck, 2014] Luck, S. J. (2014). *An introduction to the event-related potential technique*. MIT press.

[McCulloch y Pitts, 1943] McCulloch, W. S. y Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133.

[Oostenveld y Praamstra, 2001] Oostenveld, R. y Praamstra, P. (2001). The five percent electrode system for high-resolution eeg and erp measurements. *Clinical neurophysiology*, 112(4):713–719.

[Rasamoelina *et al.*, 2020] Rasamoelina, A. D., Adjailia, F., y Sinčák, P. (2020). A review of activation function for artificial neural network. En *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 281–286. IEEE.

[Riccio *et al.*, 2013] Riccio, A., Simione, L., Schettini, F., Pizzimenti, A., Inghilleri, M., Belardinelli, M. O., Mattia, D., y Cincotti, F. (2013). Attention and p300-based bci performance in people with amyotrophic lateral sclerosis. *Frontiers in human neuroscience*, 7:732.

[Rojas *et al.*, 2018] Rojas, G. M., Alvarez, C., Montoya, C. E., De la Iglesia-Vaya, M., Cisternas, J. E., y Gálvez, M. (2018). Study of resting-state functional connectivity networks using eeg electrodes position as seed. *Frontiers in neuroscience*, 12:301197.

[Roy *et al.*, 2019] Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T. H., y Faubert, J. (2019). Deep learning-based electroencephalography analysis: A systematic review.

[Rumelhart *et al.*, 1986] Rumelhart, D. E., Hinton, G. E., y Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

[Salehin y Kang, 2023] Salehin, I. y Kang, D.-K. (2023). A review on dropout regularization approaches for deep neural networks within the scholarly domain. *Electronics*, 12(14):3106.

[Santamaria-Vazquez *et al.*, 2020] Santamaria-Vazquez, E., Martinez-Cagigal, V., Vaquerizo-Villar, F., y Hornero, R. (2020). Eeg-inception: a novel deep convolutional neural network for assistive erp-based brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 28(12):2773–2782.

[Schirrmeister *et al.*, 2017a] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., y Ball, T. (2017a). Deep learning with convolutional neural networks for eeg decoding and visualization. *Human Brain Mapping*.

[Schirrmeister *et al.*, 2017b] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., Hutter, F., Burgard, W., y Ball, T. (2017b). Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*, 38(11):5391–5420.

[Sifre y Mallat, 2014] Sifre, L. y Mallat, S. (2014). Rigid-motion scattering for texture classification. *arXiv preprint arXiv:1403.1687*.

[Song *et al.*, 2022] Song, Y., Zheng, Q., Liu, B., y Gao, X. (2022). Eeg conformer: Convolutional transformer for eeg decoding and visualization. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 31:710–719.

[Szegedy *et al.*, 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., y Rabinovich, A. (2015). Going deeper with convolutions. En *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

[Thimm y Fiesler, 1995] Thimm, G. y Fiesler, E. (1995). Neural network initialization. En *From Natural to Artificial Neural Computation: International Workshop on Artificial Neural Networks Malaga-Torremolinos, Spain, June 7–9, 1995 Proceedings 3*, pp. 535–542. Springer.

[Van Veen *et al.*,] Van Veen, G., Barachant, A., Andreev, A., CATTAN G, C. R. P., y Congedo, M. Building brain invaders: Eeg data of an experimental validation [internet]. gipsa-lab; 2019 may [cited 2019 jul 2]. report no.: 1.

[Vaswani *et al.*, 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Yu *et al.*, 2017] Yu, Y., Zhou, Z., Liu, Y., Jiang, J., Yin, E., Zhang, N., Wang, Z., Liu, Y., Wu, X., y Hu, D. (2017). Self-paced operation of a wheelchair based on a hybrid brain-computer interface combining motor imagery and p300 potential. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25:2516–2526.

[Zafar *et al.*, 2022] Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A. K., y Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, 12(17):8643.

[Zhang y Sabuncu, 2018] Zhang, Z. y Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.