UNIVERSIDAD
DE GRANADA

E.T.S. Ingeniería Informática y de Telecomunicación
Departamento de Arquitectura y Tecnología de Computadores
Programa de Doctorado en Tecnologías de la Información y la Comunicación

Tesis Doctoral

Optimización multi-objetivo de arquitecturas de aprendizaje profundo para el procesamiento de señales EEG en plataformas de cómputo heterogéneas

Diego Ariel Aquino Brítez

Directores Dr. Andrés Ortiz García
Dr. Juan José Escobar Pérez

22 de Abril de 2022

Editor: Universidad de Granada. Tesis Doctorales
Autor: Diego Ariel Aquino Brítez
ISBN: 978-84-1117-355-1
URI: http://hdl.handle.net/10481/75440

Diego Ariel Aquino Brítez

Optimización multi-objetivo de arquitecturas de aprendizaje profundo para el procesamiento de señales EEG en plataformas de cómputo heterogéneas

Directores:
Dr. Andrés Ortiz García
Dr. Juan José Escobar Pérez

Universidad de Granada
E.T.S. Ingeniería Informática y de Telecomunicación
Departamento de Arquitectura y Tecnología de Computadores
Programa de Doctorado en Tecnologías de la Información y la Comunicación
Calle Periodista Daniel Saucedo Aranda
18014, Granada
España

Abstract

In recent years, models based on deep learning have revolutionized several areas of science allowing the resolution of a wide variety of complex problems, many of which were considered unsolvable. Compared to alternatives for the classification of samples that require a priori knowledge of the descriptors with greater representation or discriminant power, deep neural networks have the advantage of extracting descriptors tailored to a specific problem through a learning process. In this way, feature extraction and classification processes are integrated in the same architecture. Although some of these architectures (such as convolutional networks) were presented in the 70s and 80s, their training applied to real problems involved a computational load that could not be assumed using computers of that time. Nowadays, with the development of heterogeneous computing devices, these networks have gained popularity, making it possible to efficiently perform the computations required by deep neural networks (in some cases even inference is performed at real-time) and to develop effective, specific optimization algorithms.

However, the design of deep network architectures is a complex task that usually depends on the final application and there are no general design rules beyond the experience of the designer. Therefore, although the designer plays an important role, having an architecture that provides the required accuracy usually involves a trial-and-error process based on partial results or through the analysis of the evolution of the network parameters during training. On the other hand, the efficiency and performance of a given architecture depends to a great extent on the selection of hyperparameters, and in this case, it is again necessary to resort to a trial-and-error process. Therefore, the design and tuning of deep architectures is a complex and computationally time-consuming task even for experienced designers. Moreover, the development of specific tools to optimize the models is essential to take advantage of the capabilities of deep learning models.

With this in mind, this thesis proposes the development of a fully configurable optimization framework for deep learning architectures based on evolutionary computation. The framework not only performs the optimization of hyperparameters but also the network architecture, including regularization parameters to reduce the generalization error. In addition, it is possible to perform the optimization of an architecture based on multiple objectives that may be in conflict during

the optimization process. The results (solutions) provided by the framework will correspond to a set of non-dominated solutions that provide a trade-off between the proposed objectives, being possible to select the most appropriate solution within the Pareto front. On the other hand, it is worth to note that both CPUs and GPUs are used to speed-up the execution of the optimization procedure by taking advantage of the parallelism and heterogeneity present in the current computing nodes.

The developed tool has been applied to the optimization of deep learning architectures for electroencephalography (EEG) signal processing. The classification of EEG signals is a complex task that usually uses a priori known statistical descriptors. However, there is no guarantee that the chosen descriptors provide the best classification rate. On the other hand, previously proposed deep learning networks for EEG classification using the signal in the time domain contain a large number of hyperparameters. In the experiments performed, it is shown how to use the framework to optimize architectures based on one-dimensional convolutional networks for extracting descriptors and classification. These experiments are not only performed to optimize hyperparameters, but the framework is configured so that the optimization process can perform structural changes in the network or introduce regularization layers to improve the generalization capability.

The experimental results corroborate that the deep architectures optimized by the method proposed in this thesis improve the baseline networks and produce computationally efficient models, not only from the point of view of classification rate but also in terms of computational efficiency and, therefore, in energy efficiency.

Resumen

En los últimos años, los modelos basados en aprendizaje profundo han revolucionado diversas áreas de la ciencia permitiendo la resolución de una gran variedad de problemas complejos, muchos de los cuales se consideraban sin solución. Frente a las alternativas para la clasificación de muestras que requieren conocer a priori los descriptores con mayor poder de representación o discriminante, las redes neuronales profundas tienen la ventaja de extraer descriptores a medida para un problema concreto mediante un proceso de aprendizaje. De esta forma, se integran en una misma arquitectura los procesos de extracción de características y clasificación. Algunas de estas arquitecturas (como las redes convolucionales), aunque fueron presentadas en los años 70-80, su entrenamiento aplicado a problemas reales suponía una carga computacional no asumible para los computadores de la época. Actualmente, con el desarrollo de dispositivos de cómputo heterogéneo, estas redes han ganado popularidad, permitido realizar de manera eficiente los cálculos requeridos por la redes neuronales profundas (en mucho casos la inferencia se realiza incluso en tiempo real) y desarrollar algoritmos efectivos de optimización para las mismas.

No obstante, el diseño de arquitecturas de redes profundas es una tarea compleja que normalmente depende de la aplicación final y para la que no existen reglas de diseño. Por tanto, aunque la experiencia del diseñador juega un papel importante, disponer de una arquitectura que proporcione la precisión requerida conlleva normalmente un proceso de ensayo y error basado en resultados parciales o a través del análisis de la evolución de los parámetros de la red durante el entrenamiento. Por otro lado, la eficiencia y prestaciones de una determinada arquitectura depende en gran medida de la selección que se haga de los hiperparámetros, y en este caso, de nuevo es necesario recurrir a un proceso de ensayo y error. Por tanto, el diseño y ajuste de arquitecturas profundas es una tarea compleja que requiere mucho tiempo de cómputo incluso para diseñadores experimentados. Además, desarrollar herramientas específicas para optimizar los modelos resulta esencial de cara a aprovechar las capacidades de los modelos basados en aprendizaje profundo.

Con todo esto en mente, en esta tesis se propone el desarrollo de un framework de optimización para arquitecturas de aprendizaje profundo totalmente configurable basado en computación evolutiva. El framework no sólo contempla la optimización de hiperparámetros sino también la propia arquitectura, incluyendo parámetros de

regularización para reducir el error de generalización. Además, es posible realizar la optimización de una arquitectura en base a múltiples objetivos que pueden estar en conflicto durante el proceso de optimización. Los resultados (soluciones) proporcionados por el framework corresponderán a un conjunto de soluciones no dominadas que permiten el balance entre los objetivos propuestos, siendo posible seleccionar la solución más adecuada dentro del frente de Pareto. Por otra parte, conviene destacar que se utilizan tanto CPUs como GPUs para acelerar la ejecución del procedimiento de optimización, aprovechando el paralelismo y la heterogeneidad presente en los nodos de cómputo actuales.

La herramienta desarrollada se ha aplicado a la optimización de arquitecturas de aprendizaje profundo para el procesamiento de señales de electroencefalografía (EEG). La clasificación de señales EEG es una tarea compleja que suele utilizar descriptores de las señales conocidos a priori. Sin embargo, no se tiene garantía de que los descriptores elegidos proporcionen la mejor tasa de clasificación. Por otro lado, las propuestas de redes de aprendizaje profundo para la clasificación de EEG utilizando la señal en el tiempo contienen un gran número de hiperparámetros. En los experimentos realizados se muestra cómo usar el framework para optimizar arquitecturas basadas en redes convolucionales unidimensionales para extraer descriptores y clasificación. Dichos experimentos no sólo se realizan para optimizar hiperparámetros, sino que se configura el framework para que el proceso de optimización pueda realizar cambios estructurales en la red o introducir capas de regularización que mejoren la capacidad de generalización.

Los resultados experimentales obtenidos corroboran que las arquitecturas profundas optimizadas por el método propuesto en esta tesis mejoran las redes de partida y dan lugar a modelos computacionalmente eficientes, no sólo desde el punto de vista de la tasa de clasificación sino también en cuanto a eficiencia computacional y, por tanto, en eficiencia energética.

Agradecimientos

Este trabajo de tesis es el fruto de varios años de trabajo y dedicación. Su culminación ha sido posible gracias a la ayuda y oportunos consejos por parte de diferentes personas así como por el apoyo brindado por diferentes instituciones, a las cuales extiendo mi más afectuoso agradecimiento.

En primer lugar, a mi esposa Cinthia por su compresión y acompañamiento continuo, motivando mi dedicación y esfuerzo para afrontar con éxito este desafío académico. También a mi hijo Dieguito por ser una fuente inagotable de inspiración para mi vida.

A mis padres Herminia y Sergio, por haberme dado la vida y por brindarme su total confianza y apoyo en todos mis emprendimientos.

A mis hermanos, Sergio por todo el apoyo brindado y los consejos claves en los momentos oportunos, al igual que a Liz por su apoyo incondicional. A mi querida tía Nélida Picagua, por acompañarme y estar siempre pendiente de mí.

A Julio Ortega, por confiar en mi persona, abrirme las puertas del grupo de investigación y hacerme sentir parte de él desde el primer momento. Porque más allá de sus directrices académicas, con su ejemplo de vida me ha enseñado que ser una buena persona prevalece sobre cualquier otra virtud y que eso constituye el verdadero legado.

A mis directores de Tesis, Andrés Ortiz por su incansable y constante apoyo, por la motivación brindada a lo largo de estos años, la oportunidad de trabajar, aprender y desarrollarme en este área de investigación académica, y por su valioso apoyo para la culminación exitosa de este trabajo. A Juan José Escobar por tomar la posta en la dirección de esta tesis, su guía, trabajo y consejos para la conclusión exitosa de esta tesis.

A Rosa y Alfredo, por su grata compañía en largas conversaciones y sus oportunos consejos.

Al Programa de Becas de Postgrado “Carlos Antonio López” del Gobierno de la República del Paraguay por la oportunidad brindada para desarrollarme personal y académicamente.

Asímismo, agradecer a los proyectos de investigación del Ministerio de Ciencia, Innovación y Universidades (PGC2018-098813-B-C31 y PGC2018-098813-B-C32). También al proyecto UMA20-FEDERJA-086 de la Consejería de Economía y Conocimiento de la Junta de Andalucía. Por último, gracias a los Fondos Europeos de Desarrollo Regional (FEDER) por cofinanciar los proyectos anteriores, lo cual ha hecho posible disponer de los recursos y medios materiales necesarios para el desarrollo de esta tesis doctoral.

A todos, muchas gracias.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

Hoy día, las tecnologías de la información y la comunicación se encuentran omnipresentes en nuestras vidas ya que su notable evolución durante las últimas décadas han significado una enorme expansión e influencia transversal a prácticamente todos los ámbitos cotidianos. Sus crecientes avances determinan un punto de inflexión y marcan el punto de partida de una nueva era [1]: la denominada “Era Digital”. Este nuevo escenario trae consigo nuevas interrogantes, oportunidades y desafíos que como sociedad se deben enfrentar para brindar respuestas adecuadas a este contexto.

En dicha era digital se constituye la sociedad de la información [2] como producto de la interconexión global donde se generan datos en todo momento y desde cualquier sitio, lo cual constituye una verdadera revolución con un creciente volumen de datos a ser distribuido, almacenado y procesado para obtener información útil al servicio de la sociedad. De acuerdo a las estadísticas acerca del tráfico mundial de internet [3], en 2021 se han alcanzado los 235,7 EB mensuales y un tráfico medio por habitante de 35,5 GB al mes (45,5 GB en el caso de España [4]).

A lo largo de las últimas décadas se han logrado considerables avances tecnológicos apoyados en gran medida por el incremento capacidad de los computadores, permitiendo la resolución de diversos tipos de problemas provenientes de diversas áreas de las ciencias que durante mucho tiempo fueron considerados irresolubles. Además, en un mundo que cuenta con recursos limitados, al resolver una tarea computacional cada vez se tiene más en cuenta maximizar el uso de dichos recursos. Sin embargo, lograr ambos objetivos constituye un desafío complejo e importante, por lo que cada vez es más necesario recurrir a los avances tecnológicos para la búsqueda de soluciones óptimas.

Todos los individuos se enfrentan diariamente a situaciones donde deben decidir la opción más adecuada para resolver un determinado problema [5]. En algunos casos, estas decisiones están tomadas en base a la intuición, al sentido común, a la experiencia, el azar o como una combinación de todas ellas. Sin embargo, problemas más complejos requieren de modelos matemáticos e incluso de programación computacional para para poder tomar las decisiones óptimas. Este proceso de toma de decisiones [6] puede definirse a través de una serie de pasos sucesivos. En primer

lugar, requiere la definición de los objetivos del problema y las alternativas para llevarlo a cabo. Seguidamente, se busca determinar el impacto en los objetivos al aplicar alguna de las alternativas, con lo cual se obtiene un conjunto de posibles soluciones. A continuación, se evalúa individualmente a todo el conjunto de soluciones con el fin de determinar el valor o índice de rendimiento de cada una de las soluciones y, finalmente, se toma la decisión más satisfactoria para el problema en cuestión.

El trabajo de investigación llevado a cabo en las últimas décadas para la resolución de problemas de optimización ha sido impulsado por los computadores debido a la capacidad que disponen para realizar cálculos avanzados y simulación de eventos. Esto los ha convertido en parte fundamental del complejo proceso para la toma de decisiones. Para este fin, desde mediados de la década de 1940 han sido concebidos diversos paradigmas computacionales entre los que se encuentran los modelos de computación bioinspirados [7], donde la naturaleza se convierte en fuente de inspiración para el desarrollo de sistemas mediante la replicación básica y simplificada del comportamiento de organismos. En este sentido, los algoritmos bioinspirados actualmente cuentan con abundante literatura científica y gozan de gran popularidad, habiendo proporcionado enfoques alternativos para la solución de problemas que no habrían sido resueltos mediante la aplicación de técnicas clásicas tales como la programación lineal, no lineal y dinámica. Los enfoques de algoritmos bioinspirados más conocidos corresponden a las redes neuronales artificiales (ANNs), los algoritmos evolutivos (EAs), la inteligencia de enjambre y los sistemas inmunitarios artificiales.

En el caso de los algoritmos evolutivos, éstos han sido concebidos a partir de las nociones aprendidas en el campo de la biología evolutiva, las cuales se utilizan para el desarrollo de técnicas que resuelven complejos problemas mediantes técnicas de búsqueda y optimización. Básicamente, este procedimiento establece una población inicial con individuos que tienen capacidad de reproducirse entre sí y cuyos descendientes pueden ser sometidos a ciertas variaciones genéticas. De esta forma, se crean nuevas generaciones de individuos que pueden adaptarse de forma dinámica a un entorno determinado. Sin embargo, los problemas de optimización suelen ser multi-objetivos (MOO) [8], es decir, que habitualmente tratan con varios objetivos que han de ser optimizados de forma simultánea, por lo que puede haber conflictos entre ellos. Además, en ocasiones estos problemas se encuentran sujetos a restricciones que hay que tener en cuenta en la selección de soluciones.

Los métodos de optimización multi-objetivo se clasifican principalmente en dos grupos [9]. El primer grupo involucra a los métodos exactos, los cuales proponen

encontrar una única solución óptima del problema en cuestión. Su principal inconveniente radica en el elevado coste computacional que supone su ejecución, lo que puede suponer la inviabilidad en la aplicación de este método. El segundo grupo corresponde a los métodos heurísticos que son procedimientos que permiten la resolución de problemas específicos. Se utilizan cuando los problemas implican un alto grado de complejidad debido a la presencia de múltiples objetivos, complejas funciones de coste, o la imposición de restricciones que resulten complicadas de satisfacer. Dentro de este último grupo se encuentran los algoritmos evolutivos.

Con respecto a los sistemas basados en aprendizaje, desde que en 1943 McCulloch y Pitts propusieron un modelo inspirado en el funcionamiento de una neurona artificial [10], los sistema de aprendizaje automático basados en redes neuronales han evolucionado considerablemente, siendo utilizados en una amplia gama de tareas. Además, en los últimos años, el desarrollo de soluciones basadas en las redes neuronales artificiales de aprendizaje profundo (DNNs) han permitido brindar soluciones a diversos tipos de problemas. Este tipo de arquitecturas han superado los resultados obtenidos por las técnicas clásicas [11] para tareas tales como la traducción automática, el reconocimiento de imágenes y del habla o el aprendizaje por refuerzo. Sin embargo, la implementación de estas redes también han traído aparejados otros desafíos como el diseño de redes óptimas para poder obtener el mejor desempeño para las tareas propuestas.

La principal fortaleza de las redes neuronales radica en su capacidad de jerarquizar la información en distintas capas de abstracción. Sin embargo, la resolución de problemas complejos hace que estas arquitecturas sean cada vez más complejas, contando con un mayor número de capas y, por tanto, con un número cada vez mayor de pesos e hiperparámetros a determinar. Los pesos son parámetros de la red que se adaptan mediante un proceso de aprendizaje para minimizar el error con respecto a la salida deseada. Por el contrario, los hiperparámetros no se calculan durante todo el proceso de aprendizaje, sino que tienen que ser seleccionados previamente. Por otro lado, las prestaciones de los modelos neuronales de aprendizaje profundo dependen de forma crucial tanto de los hiperparámetros como de la estructura del modelo. Los modelos con una complejidad excesiva no sólo consumen más recursos sino que pueden comprometer la calidad del modelo, especialmente en su capacidad de generalización. Por tanto, tiene sentido desarrollar procedimientos de optimización de hiperparámetros para generar clasificadores de aprendizaje profundo eficientes en términos de tasas de clasificación, tiempo de cómputo y consumo energético.

Tanto las técnicas de aprendizaje máquina como aquellas basadas en DNNs se utilizan en la actualidad en un extenso rango de aplicaciones de gran relevancia e

interés social. No obstante, alcanzar la correcta configuración de sus hiperparámetros es un desafío debido a que no existe una metodología explícita para este fin. Por otra parte, los modelos de aprendizaje profundo habitualmente se enfrentan a conjuntos de datos de gran dimensionalidad, los cuales incrementan la complejidad del problema en cuestión y propician la llamada *maldición de la dimensionalidad*. Esta circunstancia aparece cuando la cantidad de patrones es mucho menor que la cantidad de características del patrón, perjudicando la capacidad de generalización de los modelos.

Un ejemplo típico de datos de gran dimensionalidad son las señales biomédicas. El registro de estas señales corresponde a eventos biológicos producidos por la actividad eléctrica, química o mecánica de estructuras vivas que son registradas de forma espacial, temporal o ambas de forma simultánea. La información contenida en las señales biomédicas es de gran utilidad ya que permite comprender los mecanismos fisiológicos que intervienen en un evento biológico en particular. Esto por ejemplo puede favorecer el determinar con mayor precisión el diagnostico médico o explicar el origen de una determinada patología. De hecho, el uso de técnicas de aprendizaje máquina se ha extendido también al diagnóstico asistido por computador (CAD) [12], asumiendo un rol de vital importancia dentro del contexto de la biomedicina, sus diversas aplicaciones y proporcionando herramientas de gran utilidad para los médicos especialistas. Algunos ejemplos de su aprovechamiento en el ámbito clínico se pueden encontrar en el procesamiento de imágenes, como la mamografía [13], fondo de ojo [14], la radiografía [15], la tomografía axial computarizada (TAC), la resonancia magnética nuclear (MRN), la ecografía diagnóstica o la electroencefalografía [16].

Concretamente, las señales de electroencefalografía (EEG) corresponden al registro de los impulsos eléctricos producidos en el cerebro. Específicamente, son valores de potencia eléctrica capturados y registrados durante un periodo de tiempo. Cabe destacar, que su utilización es considerada de gran relevancia para el diagnóstico de trastornos neurológicos, así como también en las investigaciones científicas relacionadas con el estudio de las funciones cerebrales y los procesos involucrados. En definitiva, mediante el estudio de las señales EEG se ha alcanzado un mayor grado de comprensión acerca del funcionamiento del cerebro humano. Además, las señales EEG son componentes fundamentales de las interfaces cerebro-computador (BCI), pues mediante la captura e interpretación de estas señales se puede establecer un sistema de comunicación directo entre el cerebro y un dispositivo electrónico externo que reacciona ante al impulso emitido por el cerebro [17].

No obstante, las señales EEG presentan ciertas características que dificultan su clasificación. Entre sus principales inconvenientes se encuentra su baja relación señal-ruido (SNR), lo que implica que presentan un alto nivel de ruido proveniente de fuentes ajenas a la actividad cerebral de interés. Además, el registro de estas señales genera una ingente cantidad de datos de alta dimensionalidad por la existencia de múltiples canales, entre otras razones. En consecuencia, resulta necesario establecer un conjunto de procedimientos que permitan el óptimo procesamiento de las señales EEG con el fin de alcanzar la mayor precisión posible en tareas de clasificación automatizada.

Para el procesamiento y clasificación de señales o datos puede definirse un esquema estándar clásico. A continuación, se definen brevemente cada uno de los pasos [18] involucrados en el sistema:

1. **Adquisición de las señales**. Procedimiento que registra las señales biomédicas.
2. **Preprocesamiento de señales**. Procedimiento que se realiza para eliminar el ruido y artefactos habitualmente presentes en este tipo de señales. Se trata de minimizar la contaminación proveniente de fuentes ajenas a la señal que será objeto de estudio, mejorando su calidad y evitando la pérdida de información que podría ser relevante. El preprocesamiento de las señales también suele incluir un filtrado que elimina componentes de la señal fuera de la banda de frecuencias de interés y la normalización de los datos para que estos estén en un rango o escala determinados.
3. **Extracción de características (FE)**. Consiste en el cálculo de descriptores con capacidad representativa o discriminante que puedan ser utilizados para el procesamiento posterior. De esta forma, las señales originales quedan representadas por un número limitado de descriptores que condensan la información relevante y por tanto se reduce la dimensionalidad del conjunto de datos original.
4. **Selección de características (FS)**. Procedimiento que permite reducir la dimensionalidad de los datos mediante la selección de un subconjunto de características proveniente del conjunto original o de otro subconjunto extraído previamente.
5. **Clasificación de señales**. Procedimiento para identificar y categorizar a los patrones presentes en las señales.
6. **Módulo de aplicación**. Encargado de proporcionar una respuesta al usuario considerando la clase a la que corresponde el patrón de actividad cerebral.

Dentro del esquema de procesamiento de señales biomédicas, la precisión alcanzada en la fase de clasificación depende de forma crucial de los pasos llevados a cabo previamente (preprocesamiento, extracción y selección de características). Es importante indicar que dependiendo del tipo de señal a procesar y el objetivo de dicho procesamiento, el algoritmo o método más adecuado para cada paso podría variar. Esto es necesario tenerlo en cuenta si se desea alcanzar los resultados de clasificación esperados. Además, la tasa de clasificación alcanzada depende en gran medida de los descriptores utilizados. En el caso de señales EEG, existen diferentes métodos de cálculo de descriptores, los cuales se pueden agrupar [19] en aquellos que extraen información del dominio temporal, frecuencial, un híbrido entre ambos y los que descomponen la señal en diferentes componentes. Por ejemplo, en [20, 21] se utilizan descriptores en el espectro del tiempo para la detección precoz de episodios relacionados con la epilepsia. En [22], se utilizan para clasificar señales EEG de imaginación motora en el contexto de las aplicaciones BCI. Aquí, se calculan diferentes descriptores estadísticos de la señal en el dominio del tiempo como el valor absoluto medio, valor máximo del pico, integral cuadrada simple, amplitud de Willison [23-25] y longitud de la forma de onda. Con estos trabajos se ha demostrado que la fase de extracción de características es de fundamental importancia para alcanzar una óptima clasificación de las señales.

Una vez que se ha extraído un conjunto de descriptores, puede realizarse una selección que realce su capacidad representativa o discriminante para un problema concreto. Dentro del dominio de las señales biomédicas esta es una técnica de gran utilidad que permite la reducción de la dimensionalidad y además brinda soporte para la compresión de las causas de enfermedades [26]. Los métodos de selección de características pueden ser clasificados en tres grupos: filtros, métodos embebidos y de envoltura. Aunque actualmente existe una gran cantidad de métodos desarrollados para la selección de características, la correcta elección del método de selección de características para el problema en cuestión es una tarea compleja y cuyo coste computacional depende en gran medida del espacio de búsqueda. Por este motivo, en [27, 28] se propone la selección de características mediante algoritmos genéticos dado que son especialmente útiles en espacios de búsqueda muy grandes que no pueden ser explorados exhaustivamente.

En cuanto a la clasificación de señales biomédicas, existe una gran cantidad de métodos que han sido aplicados en diferentes problemas, tanto basados en aprendizaje estadístico [29] como basados en redes neuronales. En particular, en el ámbito de las señales de EEG se han implementado diversas técnicas de clasificación [30]: clasificadores lineales, bayesianos no lineales, de vecinos más cercanos, combinaciones de clasificadores y redes neuronales. La clasificación de señales EEG, no

obstante, es una tarea compleja dadas sus características tal y como se ha comentado anteriormente, por lo que requiere implementar técnicas elaboradas y a menudo hibridar diferentes métodos. En este sentido, las DNNs son una alternativa válida que ya han sido aplicadas con éxito en tareas de clasificación de señales EEG. No obstante, es necesario utilizar arquitecturas específicamente diseñadas y optimizadas para este fin. Habitualmente, se requiere de expertos con vasta experiencia en este ámbito donde, a partir de un diseño inicial, realizan un proceso de ensayo y error para definir la arquitectura de red que permite alcanzar los mejores resultados. No obstante, esto hace muy complicado tener en cuenta varios objetivos de diseño que deban ser atendidos de forma simultánea, como por ejemplo el tiempo de cómputo y la eficiencia energética. Consecuentemente, tiene sentido desarrollar procedimientos que permitan automatizar el diseño y la configuración de estas redes con el objetivo de optimizar múltiples objetivos.

Teniendo en cuenta todo lo anteriormente comentado, esta tesis propone establecer procedimientos que permitan diseñar y optimizar de forma automática las DNNs teniendo en cuenta múltiples objetivos que pueden entrar en conflicto. De esta forma, el procedimiento de optimización de arquitectura de redes neuronales puede centrarse en cumplir de forma simultánea los siguientes requisitos: mejorar el desempeño en tareas de clasificación, reducir la complejidad computacional y disminuir el consumo energético. En la literatura científica del tema abordado en esta tesis se muestran diversos enfoques para resolver eficientemente este problema de optimización y tienen como principal propósito [31] la toma de decisiones para alcanzar una solución óptima. En la Sección 1.1 se detallan algunos trabajos relacionados con esta tesis. De esta forma, es posible constatar la gran cantidad de esfuerzos de investigación que han sido realizados para proponer mecanismos de optimización automática y búsqueda de redes neuronales eficientes para la realización de las tareas propuestas.

1.1. Estado del arte

Las redes neuronales artificiales han demostrado su gran eficiencia y adaptabilidad para la resolución de diversos tipos de problemas provenientes de diferentes dominios. Sin embargo, esta gran capacidad se encuentra asociada a otros desafíos que deben ser abordados para alcanzar las más óptimas prestaciones de los modelos de redes neuronales. En este sentido, uno de los principales problemas está dado por la inmensa cantidad de decisiones que deben tomarse previamente durante el diseño de las arquitecturas de red así como también la selección de los hiperparámetros

asociados. Se debe tener en cuenta que el conjunto de parámetros seleccionados es lo que finalmente determinará la capacidad de la red neuronal para realizar tareas de gran complejidad de forma eficiente.

Dentro del conjunto de técnicas desarrolladas para la búsqueda de arquitecturas neuronales óptimas (NAS) [32], éstas se agrupan en dos categorías: NAS con aprendizaje por refuerzo (RL) [33-36] y NAS con algoritmos de optimización bio-inspirados [37-40]. Por tanto, con el fin de alcanzar el objetivo de encontrar arquitecturas óptimas, es necesario considerar aquellos elementos que determinan la topología de una red neuronal. En este sentido, algunos de los elementos a tomar en cuenta son los tipos de capas, la cantidad de ellas, tipos de operaciones e interconexiones entre capas de una red neuronal y las funciones de activación. Como contrapartida, también se cuentan con aquellos métodos que tratan de automatizar la selección de los hiperparámetros óptimos de una red neuronal [41]. Además, es importante mencionar que los hiperparámetros se definen con anterioridad al entrenamiento de la red neuronal y se mantienen inmutables durante todo el proceso de aprendizaje.

En cuanto a los hiperparámetros, podemos diferenciar dos tipos: por un lado, los que tienen una relación directa con las operaciones, como por ejemplo las características asociadas a las operaciones de convolución, entre los que se encuentran el tamaño del filtro o de zancadas, entre otros. Por otro lado, también se cuenta con los hiperparámetros que inciden de forma global en el proceso de aprendizaje de la red neuronal, como la tasa de aprendizaje, el tamaño de los lotes y los parámetros de regularización que incluyen las tasas de abandono o decaimiento de peso. La resolución del problema de optimización ha sido expuesta y abordada en la literatura científica, en donde es posible encontrar diversos enfoques [31] que abordan la toma de decisiones para encontrar una solución óptima y eficiente al problema a tratar.

Con respecto a la optimización de arquitecturas neuronales, la utilización de algoritmos bio-inspirados para resolver diversos tipos de problemas de optimización [42] se encuentra bastante extendida. Estos algoritmos de optimización pueden definirse como métodos heurísticos no deterministas que se adaptan a su entorno. Además, han demostrado ser de gran utilidad para la resolución de problemas en los que es particularmente difícil encontrar una solución óptima o satisfactoria. Por otra parte, debe indicarse que estos métodos imitan de una forma muy básica el comportamiento de los sistemas naturales, generando modelos relativamente aproximados de dichos fenómenos. En este contexto, es posible encontrar métodos basados en adaptaciones sociales [43], que imitan la inteligencia del comportamiento social

de los integrantes de un grupo para llevar a cabo una tarea. También algoritmos basados en computación evolutiva [44], inspirados en la teoría de la evolución [45] de Charles Darwin. En el ámbito de los métodos basados en las adaptaciones sociales se puede citar al algoritmo de optimización basado en enjambre de partículas (PSO) [46], que toma su inspiración en el comportamiento de las bandadas de aves o peces. Por otro lado, también se cuenta con el algoritmo de optimización basado en colonias de hormigas (ACO) que trata de imitar el comportamiento de éstas cuando buscan el mejor camino para la recolección de su comida.

En [47-49] pueden encontrarse algunos trabajos que proponen la utilización del algoritmo de optimización por enjambre de partículas para la búsqueda de arquitecturas de redes neuronales artificiales prealimentadas (FFNN). Los resultados alcanzados en tareas de clasificación corroboran la eficiencia de las propuestas. Por otro lado, Carvalho et al. en [48] plantea la mejora de la capacidad de generalización mediante el compromiso entre la reducción de la complejidad de la red neuronal y la reducción del error cometido durante el entrenamiento.

En [47] se propone un método para que la arquitectura y los pesos de las ANNs vayan ajustándose gradualmente de forma iterativa, tomando como base la calidad de la ANN. Con este fin, se implementa una estrategia evolutiva que permite añadir nodos a la red y además utiliza un algoritmo de entrenamiento parcial, con lo cual se trata de mantener el nexo de comportamiento entre padres y sus descendientes. Por otro lado, Yu et al. propone en [49] la utilización de una versión mejorada del algoritmo PSO que incorpora un método para la automatización del ajuste de sus parámetros, el reajuste de la velocidad, así como también del cruce y las mutaciones. De igual manera, en [50-55] se utiliza PSO para la exploración y búsqueda de arquitecturas de redes neuronales profundas. Específicamente, se busca permitir la optimización de redes neuronales convolucionales (CNN) con la idea de incrementar el desempeño en tareas de clasificación de imágenes, al igual que en [56]. Para comprobar la eficiencia de la propuesta utilizan conjuntos de datos tales como CIFAR-IO y de vegetación del borde de la carretera (RSVD).

En [57] plantean una nueva estrategia evolutiva basada en comportamiento cuántico [58] con codificación binaria aplicada a PSO. Con esto pretenden encontrar los parámetros de las CNNs que sean más adecuados para alcanzar los mejores índices de clasificación. Además, en el trabajo indican que con esta propuesta no se requiere de conocimientos previos para la configuración de CNNs óptimas. Los experimentos fueron llevados a cabo utilizando el conjunto de datos MNIST [59] y sus variantes [60].

Las propuestas de optimización anteriores han sido desarrolladas con diferentes fines, fundamentalmente dentro del área de la biomedicina, utilizando para su evaluación conjuntos de datos de diferente naturaleza. Así, en [47-49] se utilizan datos médicos de diferentes áreas tales como el cáncer, diabetes y datos cardíacos, disponibles en [61, 62].

En la propuesta [52], desarrollada para la búsqueda automática de CNNs, se introducen tres mejoras al PSO original. La primera es una estrategia de codificación con base a las redes informáticas. En la segunda, se incluyen arquitecturas CNNs de longitud variable mediante la inclusión de capas que inicialmente se encuentran deshabilitadas. En la última propuesta, si el conjunto de datos es grande se procede a dividir aleatoriamente dicho conjunto con el fin de incrementar la velocidad. Fernandes y Yen propone en [53] un método automático de búsqueda de arquitecturas CNNs óptimas para clasificación de imágenes mediante la utilización del PSO incorporando mejoras que permiten alcanzar una rápida convergencia del algoritmo de optimización. El resultado de los experimentos realizados con el conjunto de datos MNIST demostró que la calidad de las soluciones alcanzadas tienen un rendimiento similar al del estado del arte. También, se ha propuesto en [54] la utilización de redes neuronales convolucionales para la detección de cáncer de piel utilizando dos conjuntos de datos que corresponden al repositorio de aprendizaje automático UCI y ALL-IDB2. Entre las mejoras introducidas, destacan el uso de coeficientes de aceleración adaptativos y de mecanismos de reinicialización para evitar el estancamiento en mínimos locales. Por otra parte, el segundo modelo utiliza coeficientes con base a funciones no lineales, buscando aumentar la diversidad de soluciones.

En [55], Huang et al. desarrolla una variación del PSO que permite flexibilidad en la variabilidad de la longitud del algoritmo para el diseño eficiente de las CNNs. Con este fin, desarrollan un esquema de codificación que evita la restricción de longitud del PSO. Además, se busca que el nuevo esquema propuesto acelere la velocidad del algoritmo de optimización. Para corroborar la eficiencia del método propuesto se utilizaron los conjuntos de datos MNIST y CIFAR-10.

En el contexto de los algoritmos bio-inspirados de optimización también se encuentra la computación evolutiva [44], que se refiere a la simulación del proceso natural de evolución de las especies. Dada una población inicial, ésta va evolucionando mediante un proceso de cruce, mutación y selección, permitiendo de esta forma la generación de nuevos individuos con mejores características de adaptabilidad que sus antecesores. Los métodos basados en computación evolutiva tienen especial aplicación en problemas donde el espacio de búsqueda es muy amplio, como por ejemplo la optimización de hiperparámetros de una red neuronal.

León et. al propone en [63] algoritmos genéticos para optimizar los hiperparámetros de redes CNN, FFNN y RNN. Para determinar la calidad de las soluciones, realizan comparaciones entre la tasa de clasificación alcanzada y el tiempo de cómputo. El conjunto de datos utilizados para llevar a cabo sus experimentos corresponde a señales EEG utilizadas en aplicaciones BCI. Los resultados alcanzados por el método propuesto mejoran las tareas de clasificación en los diferentes tipos de redes analizadas.

En [64], se utilizan algoritmos genéticos para encontrar arquitecturas de redes neuronales que sean óptimas en el procesamiento de señales EEG. La arquitectura de cada una de las redes neuronales es variable: cantidad máxima de capas, tipo de capas, cantidad de filtros o el tamaño del *kernel*, entre otros. La función de evaluación viene dada por la precisión, área bajo la curva (AUC) y el índice kappa. Los conjuntos de datos utilizados para llevar a cabo los experimentos corresponden son: BCI Competition IV 2a Dataset, Kaggle Inria BCI Challenge y High Gamma Dataset.

Baldeon et al. [65] plantea el perfeccionamiento de la técnica de segmentación de imágenes médicas basada en redes neuronales convolucionales mediante la optimización de los hiperparámetros de la U-net [66]. La propuesta utiliza un algoritmo de optimización multi-objetivo que tiene por objetivo aumentar la precisión de la segmentación a la vez que se reduce el número de parámetros de entrenamiento. Para la realización de los experimentos utilizan dos conjuntos de datos de imágenes médicas: imágenes de próstata en MRI del desafío PROMISE12, y el conjunto de imágenes cardiacas en MRI. De acuerdo a los resultado alcanzados lograron mejorar la segmentación y reducir un 30 % la cantidad de parámetros de entrenamiento.

En [67], los autores indican la importancia del diagnóstico asistido por computadoras mediante la clasificación de imágenes de rayos X, y proponen llevar a cabo la clasificación de este tipo de imágenes mediante la utilización de CNNs. Para la configuración óptima de la arquitectura de redes neuronales proponen optimizar la arquitectura de redes mediante la utilización de algoritmos evolutivos.

Por otro lado, Martín et al. [68] propone la utilización de redes neuronales profundas (DNN) para el problema de clasificación de malwares, tarea que habitualmente es realizada por otras técnicas de clasificación donde su configuración resulta más sencilla en comparación con las redes neuronales profundas. Por lo tanto, para superar el inconveniente de establecer una correcta configuración de la DNNs proponen el uso de algoritmos genéticos. El objetivo es maximizar la precisión de la clasificación de malwares y minimizar la complejidad del modelo alcanzado. El resultado: alcanzar una precisión del 91 % en la detección de malware. Por lo tanto,

se puede corroborar la eficiencia y validez de utilizar algoritmos genéticos para buscar la configuración mas adecuada de las redes neuronales profundas.

Fan et al. propone en [69] aumentar la precisión en la segmentación de los vasos de la retina (RVS) mediante la utilización de redes neuronales cuyos parámetros de configuración se encuentran optimizados por medio de la implementación de algoritmos evolutivos. En este caso, se utiliza una red de tipo autoencoder para la tarea de segmentación de los vasos. Los conjuntos de datos utilizados para la experimentación son el DRIVE, STARE y CHASE DB1.

En los trabajos realizados en [70-74] se propone la implementación de procedimientos multi-objetivo para selección de características de señales EEG para aplicaciones BCI, teniendo como objetivos el incremento del desempeño en tareas de clasificación, la reducción del tamaño del subconjunto de características y la reducción del tiempo de cómputo. Además, en [70] se propone un enfoque para la selección de características de señales EEG basado en un modelo maestro-trabajador con dos algoritmos evolutivos diferentes. El primero paraleliza la ejecución de la función de evaluación para cada uno de los individuos, mientras que el segundo método permite la ejecución en paralelo de procedimientos evolutivos multi-objetivo sobre subconjuntos de poblaciones. Los resultados alcanzados en este trabajo indican una reducción en el tiempo de ejecución y mejora en la calidad de soluciones alcanzadas.

Martin et al. [73] propone utilizar un método de filtro supervisado mediante la implementación de procedimientos evolutivos para la selección del subconjunto de características más representativas de las señales EEG. El clasificador implementado corresponde al análisis lineal discriminante (LDA) que utiliza las propiedades del análisis multiresolución (MRA) de las señales EEG en los dominios temporal y espectral. Los resultados publicados indican mejoras en los tiempos de cómputo, mayor precisión en la clasificación e incremento en la capacidad de generalización.

En [74] se propone llevar a cabo la selección de características de señales EEG mediante la utilización de un método de tipo (*wrapper*) basado en un algoritmo evolutivo multi-objetivo. Los objetivos en este caso son: (i) minimizar el tamaño del subconjunto de características y (ii) maximizar la capacidad de generalización. Los resultados experimentales demuestran que con los pequeños subconjuntos de características seleccionadas se alcanza una gran precisión de clasificación.

1.2. Objetivos

Esta tesis tiene como objetivo desarrollar y evaluar procedimientos que permitan la optimización de redes profundas teniendo en cuenta mejorar la tasa de clasificación, reducir el número de parámetros o el consumo energético del modelo. Por otro lado, se aborda la clasificación de señales EEG/BCI en el dominio del tiempo sin ningún tipo de preprocesamiento ni extracción de características conocidas a priori, de forma que sea el propio método de optimización el que genere una arquitectura capaz de extraer automáticamente descriptores discriminantes mediante aprendizaje automático. A continuación, se detallan los principales objetivos considerados en esta tesis:

1. Optimizar automáticamente los hiperparámetros correspondientes a las Arquitecturas de Redes Neuronales que hayan sido elegidas previamente para la realización de ciertas tareas específicas. Es decir, que de forma automática sea posible encontrar la configuración óptima de hiperparámetros de una red neuronal específica, maximizando la tasa de clasificación conseguida.

2. Generar de forma automática nuevas Arquitecturas de Redes Neuronales capaces de adaptarse dinámicamente a su entorno para permitir la resolución de los problemas planteados. Esto implica que las nuevas arquitecturas podrían ser distintas a las ya existentes y que además, tanto la arquitectura de la red como sus respectivos hiperparámetros serán los óptimos para llevar a cabo las tareas propuestas.

3. Parametrizar las restricciones que sean consideradas necesarias para ser incorporadas a los procedimientos automáticos de optimización y búsqueda de hiperparámetros, así como de las arquitecturas de redes neuronales óptimas con el propósito de delimitar el espacio de búsqueda de soluciones. Es decir, debe ser posible establecer limitaciones a variables como el número de capas, su tipo, y funciones de activación, entre otros.

4. Registrar de forma detallada el proceso de optimización mediante la implementación de bases de datos que permitirán almacenar: (i) el conjunto de soluciones obtenidas; (ii) los resultados alcanzados; (iii) los parámetros de configuración; (iv) restricciones aplicadas; (v) tiempos de ejecución, y (vi) consumo energético. De esta forma se podrán realizar evaluaciones de las soluciones subóptimas alcanzadas y que podrían resultar de gran utilidad para su posterior análisis.

5. Proporcionar un framework de optimización de arquitecturas neuronales totalmente parametrizable y configurable que incluya las características citadas anteriormente.

1.3. Contribuciones

- *Optimization of Deep Architectures for EEG Signal Classification: An AutoML Approach Using Evolutionary Algorithms*. Sensors. (Q1). 2021 [75].

  En este artículo se presenta el procedimiento de optimización desarrollado en esta Tesis, el cual incluye un framework completo que permite no sólo la optimización de hiperparámetros sino también la optimización de una arquitectura neuronal mediante computación evolutiva. El procedimiento desarrollado, centrado en arquitecturas profundas, permite el establecimiento de restricciones para limitar el proceso de búsqueda así como la definición de múltiples objetivos. Permite también definir la métrica a utilizar para la evaluación de la calidad de las soluciones. Además, con el fin de poder seguir el proceso de optimización, se utiliza una base de datos donde quedan registradas las soluciones propuestas de cada subpoblación a lo largo de las diferentes generaciones del algoritmo evolutivo. Por otro lado, con el fin de acelerar este proceso, que es muy costoso computacionalmente, se ha ejecutado aprovechando las ventajas de los núcleos CPU y GPU de un sistema de cómputo heterogéneo. Finalmente, se realiza una evaluación de la propuesta a través de la optimización de arquitecturas basadas en capas convolucionales para la clasificación de señales EEG y se analiza el consumo energético de las diferentes soluciones propuestas por el procedimiento.

- *Temporal phase synchrony disruption in Dyslexia: anomaly patterns in auditory processing*. 9th Work-Conference on the Interplay between. Natural and Artificial Computation (IWINAC 2022) [76].

  En este artículo se muestra la extracción de características de señales EEG y permite ver patrones temporales de sincronización entre canales que sean diferenciales para sujetos controles y disléxicos.

- *An Anomaly Detection Approach for Dyslexia Diagnosis Using EEG Signals*. 8th Work-Conference on the Interplay between Natural and Artificial Computation (IWINAC 2019) [77].

En este artículo se aborda la clasificación de señales EEG a través de la extracción de descriptores temporales y en frecuencia con un enfoque de detección de anomalías. Uno de los objetivos de este trabajo es poner de manifiesto la utilidad de los sistemas de detección de anomalías en problemas de clasificación de EEG, especialmente cuando la varianza intraclase es alta.

1.4. Estructura de la tesis

El Capítulo 1 ha expuesto la principal motivación de este trabajo de investigación así como el trabajo relacionado con la tarea. En el Capítulo 2 se detallan las bases biológicas de las señales EEG y se describen sus comportamientos temporales y espectrales, que han sido, tradicionalmente (sobre todo las características espectrales). Tras esto, los capítulos posteriores tratan los diferentes algoritmos y métodos en los que se basa el desarrollo realizado en esta tesis. Así, en el Capítulo 3 se detallan los métodos basados en aprendizaje profundo (*Deep Learning*) para el procesamiento de series temporales. Especialmente, aquellas redes con capas convolucionales que son de especial utilidad para el procesamiento de señales EEG. Además, en dicho capítulo se muestra la necesidad de optimizar dichas estructuras debido al elevado número de hiperparámetros y por las diferentes combinaciones de capas, funciones de activación que pueden determinar la capacidad discriminante del modelo.

Dado que en esta Tesis se utilizan métodos de optimización basados en computación evolutiva, el Capítulo 4 trata las bases de dicha metodología: optimización, operadores genéticos, algoritmos más comunes, ventajas e inconvenientes. Tras esto, el Capítulo 5 describe con detalle la principal contribución de esta Tesis: una metodología de optimización para arquitecturas Deep Learning que se ha materializado en una herramienta de optimización totalmente configurable, diseñada no sólo para la optimización de hiperparámetros de redes neuronales sino también dentro del paradigma de *Auto Machine Learning*. Por otro lado, en el Capítulo 6 se detallan los resultados obtenidos utilizando descriptores espectrales conocidos a priori, para la clasificación de sujetos con dislexia. Además, se utiliza la herramienta desarrollada en esta Tesis para optimizar redes basadas en capas convolucionales unidimensionales para la clasificación multi-clase de señales EEG/BCI, comparando el resultado obtenido por el modelo optimizado con la línea base que suponen los modelos originales. Finalmente, el Capítulo 7 expone las conclusiones derivadas de este trabajo así como sus implicaciones en futuros desarrollos y el impacto en el ámbito del *Auto Machine Learning*.

2 Interfaces cerebro-computador (BCI)

El desarrollo de aplicaciones BCI tiene como premisa principal la mejora de la calidad de vida de las personas, y en especial, de aquellas que precisan de asistencia adicional para permitirles una adecuada interactividad con el mundo que les rodea. Diversos grupos de investigación están dedicados a estudiar y desarrollar aplicaciones BCI que permitan superar las deficiencias motoras de los sujetos. No obstante, en los últimos años también se está incrementando el número de investigaciones destinadas a permitir la utilización de las aplicaciones BCI en el ámbito del entretenimiento [78].

Este capítulo aborda temas relacionados con las interfaces cerebro-computador, definiendo los conceptos teóricos relacionados y los pasos necesarios para permitir su puesta en marcha. Específicamente, se realiza un breve repaso sobre la historia, su uso y los principios fisiológicos de la Electroencefalografía, incluyendo detalles acerca del origen de la actividad eléctrica cerebral y los diferentes tipos de procedimientos que permiten su captura mediante electrodos. También se detallan los sistemas estándar de posicionamiento de electrodos, las características de la actividad cerebral y los procedimientos utilizados para la extracción de dichas características (temporales, espectrales, análisis multiresolución, etc.).

2.1. Electroencefalografía

La Electroencefalografía es la ciencia que se encarga de la detección, registro y estudio de la actividad eléctrica cerebral mediante la exploración neurofisiológica, la cual tiene una singular importancia en la práctica de la neurología clínica. Los potenciales eléctricos generados por el cerebro son capturados por medio de electrodos colocados sobre el cuero cabelludo o debajo del mismo, dependiendo del tipo de electrodos utilizado.

El electroencefalograma (EEG) consiste en el registro de la actividad eléctrica generada por el cerebro. Es importante señalar la gran complejidad y variabilidad

de dichas señales, pues pueden diferir dependiendo del sujeto o tipo de electrodos utilizado. Además, el registro de EEGs en un mismo sujeto puede verse alterado dependiendo de las condiciones en las que se realice, así como de factores internos y externos del sujeto.

Entre las principales ventajas de las señales EEG, destaca su carácter no invasivo y fundamentalmente el tipo de información que proporciona esta técnica, ya que cuenta con una alta resolución temporal. Mientras que el EEG permite obtener información correspondiente a la actividad eléctrica cerebral a lo largo de un periodo de tiempo, las técnicas de imágenes utilizadas para analizar la actividad cerebral tienen menor resolución temporal y por tanto centran su atención en la resolución espacial. Como desventaja del EEG, está el alto coste de adquisición de dichas señales, pues los equipos de adquisición suelen ser costosos. Aun así, son menos costosos en comparación con otros métodos de adquisición de información funcional del cerebro.

El EEG se utiliza en la práctica clínica con diferentes aplicaciones. Esta herramienta permite el diagnóstico de enfermedades neurológicas, la evaluación de trastornos del sueño, el diagnóstico del origen de cefaleas, la evaluación de estados de coma y el control de la actividad cerebral durante el desarrollo de intervenciones quirúrgicas. Además, es de gran utilidad para la diferenciación de las enfermedades psiquiátricas sobre las orgánicas. Por otra parte, las señales EEG también son utilizadas en las interfaces cerebro-computador (BCI), que aplican procesos computacionales para permitir su interpretación con el objetivo de que el dispositivo externo que ha sido dispuesto para este fin realice la acción solicitada por el cerebro y genere una interacción con su entorno.

2.1.1. Reseña histórica

Los inicios de la Electroencefalografía se remontan al año 1875, cuando Richard Caton informa acerca de los resultados obtenidos por los experimentos realizados con animales tales como monos y conejos, con el fin de corroborar la presencia de la actividad eléctrica cerebral. Estos experimentos fueron realizados utilizando un dispositivo denominado galvanómetro de Thompson. Sin embargo, esta tecnología no permitía registrar esta actividad. Entre sus trabajos, se encuentra el que se considera como la primera descripción de un EEG.

En el año 1930, Hans Berger se convierte en la primera persona en registrar señales EEG provenientes de la actividad cerebral humana mediante un proceso de adquisición que utiliza un método no invasivo con electrodos dispuestos sobre el cuero

cabelludo [79]. Estos se encargan de registrar los cambios de potencial eléctrico procedentes de la actividad de las neuronas tras un proceso de amplificación de la señal. Berger es considerado el padre de la Electroencefalografía moderna como consecuencia de sus innovadores trabajos en esta área.

Así, a lo largo de los años han sido propuestos diferentes procedimientos de adquisición de señales eléctricas cerebrales, los cuales se clasifican en invasivos y no invasivos según su naturaleza. Los procedimientos invasivos implican la realización de un procedimiento quirúrgico que suele tener sus riesgos. La Electrocorticografía (ECoG) es un ejemplo de este tipo procedimiento, donde los electrodos se disponen en contacto directo con la corteza cerebral. Este tipo de procedimiento fue realizado por primera vez en 1935 [80]. En la Figura 2.1 es posible observar el mapa de estimulación cerebral elaborado por Otfrid Förster [81].

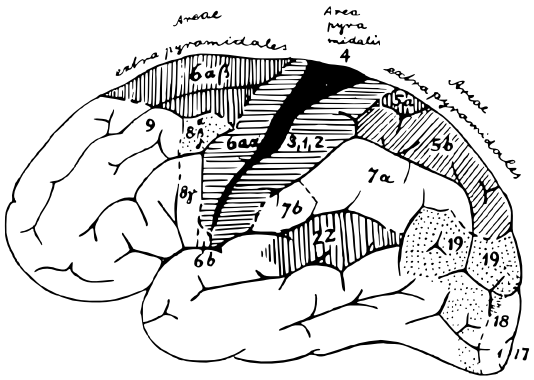


Figura 2.1.: Mapa de estimulación cerebral. Gráfico elaborado por Otfrid Förster que incluye respuestas sensitivas incluyendo aquellas que se encuentran fuera del área central. Imagen obtenida de [80].

Por otro lado, los procedimientos no invasivos son aquellos que no requieren de cirugía para la adquisición de las señales eléctricas cerebrales. El EEG es un ejemplo de este tipo de procedimiento. En un primer momento, el EEG registraba señales provenientes de una única área cerebral (un solo canal). Sin embargo, a partir de 1935 se empiezan a construir dispositivos que registran señales de al menos 3 canales de forma simultánea. Los actuales avances tecnológicos en dispositivos que permiten la captura de este tipo de señales han posibilitado registros de gran calidad al contar con mayor número de canales. También los amplificadores de bajo ruido, pues permiten una mayor precisión y fidelidad en la medición de esta señales. Además, en el ámbito de la visualización también se han producido importantes desarrollos tecnológicos que resultan en una mayor calidad de presentación de las señales EEG mediante la utilización de monitores especializados (originalmente estas señales eran registradas en papel).

En la actualidad, existen diversas herramientas que son utilizadas comúnmente en la práctica de la neurología clínica para obtener información del cerebro, tanto estructural (tomografía computarizada (TC), resonancia magnética (RM)), como funcional (resonancia magnética funcional (fMRI), tomografía por emisión de protones (PET), tomografía por emisión de un único fotón (SPECT) y EEG). Aunque el EEG sólo permite obtener información de áreas corticales, es una de las herramientas más utilizadas.

2.1.2. Electrogénesis cerebral

Las señales eléctricas cerebrales tienen su origen en las neuronas, que son las células principales del cerebro y de todo el sistema nervioso. Son responsables de la recepción, el procesamiento y la transmisión de información mediante señales que se producen a través de procesos electroquímicos. El potencial eléctrico generado por una única neurona es bastante tenue, lo cual dificulta su detección. No obstante, el registro de este tipo de señales es posible cuando una población de neuronas sincroniza su actividad mediante la sinapsis, capturando la actividad de conjuntos de neuronas.

En la Figura 2.2 es posible observar el esquema básico de una neurona, que incluye el núcleo, las dendritas, soma, axón y los nodos de Ranvier [82]. En esta figura también se indica la dirección del impulso nervioso que fluye a través de la neurona pasando finalmente por el axón terminal, donde se produce la trasmisión química a otra neurona a través de la sinapsis.

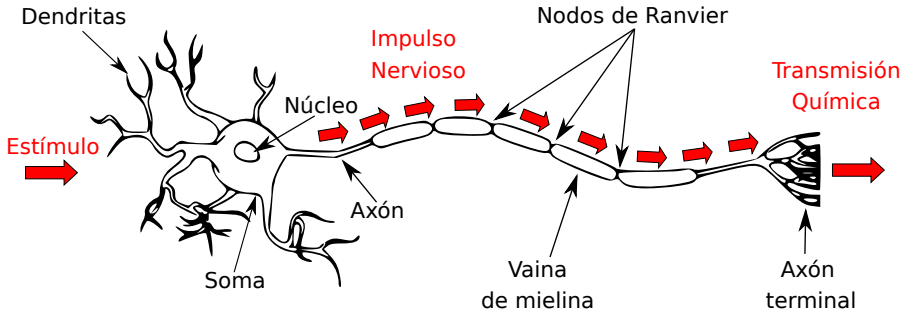


Figura 2.2.: Diagrama de una neurona biológica. Propagación del impulso nervioso a través de la neurona. Imagen adaptada de [83].

La sinapsis eléctrica es la conexión entre neuronas, o entre una neurona y una célula o viceversa. Este proceso se produce mediante la transmisión del impulso nervioso entre el emisor y el receptor debido al carácter electromagnético de la transmisión del impulso. Además, la sinapsis eléctrica concede a las neuronas la capacidad de

controlar una gran cantidad de procesos químicos y físicos del cuerpo, permitiendo la sincronización (activación e inhibición según voluntad). Las posibles fuentes del potencial eléctrico de una neurona son:

- **Potenciales de acción**. Impulso nervioso de muy corta duración (aproximadamente 10 ms) que recorre el axón permitiendo que se liberen neurotransmisores o iones, con lo que se genera un campo de acción muy limitado.
- **Potenciales postsinápticos (PPS)**. Encargados de inducir corrientes hacia el interior de las neuronas. Existen dos tipos: 1) excitatorios (PPSE), que permiten la entrada de iones positivos y 2) inhibitorios (PPSI), que permiten la entrada de iones negativos. Ambos potenciales son considerados como los principales responsables de la generación de la actividad eléctrica registrada y cuya duración se encuentra en un rango de entre 50 y 200 ms.
- **Células gliales o glías**. Son células muy abundantes en el cerebro cuya función principal es la de servir de soporte estructural y metabólico a las neuronas e intervienen de forma activa en la transmisión de información cerebral. Se encargan de controlar el microambiente celular en cuanto a la composición iónica y niveles de los neurotransmisores. No obstante, es bastante modesto el aporte que brindan al potencial eléctrico.

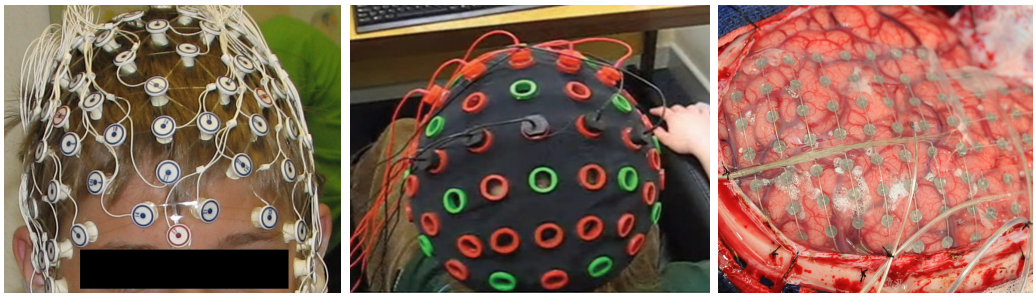
2.1.3. Electrodos para la adquisición de EEG

Los electrodos son pequeños sensores conectados que se encargan de detectar y transmitir el registro del valor de potencial eléctrico que proviene de la actividad cerebral. Además, se debe considerar que el potencial detectado es bastante reducido, por lo que es fundamental que los electrodos cuenten con propiedades que permitan minimizar el ruido en la señal, evitar la polarización, aumentar la capacidad específica, y ser capaces de minimizar la impedancia de contacto. Actualmente, es posible encontrar diversos tipos de electrodos:

- **Adheridos**. Son pequeños discos metálicos de aproximadamente 5 mm de diámetro que se adhieren al cuero cabelludo mediante un gel conductor. Cuando son aplicados correctamente proporcionan impedancias de contacto muy bajas (de 1 a 2 KΩ). Habitualmente, la cara activa cuenta con micro porosidades, con lo cual se incrementa el área de contacto y la capacidad especifica del electrodo.

- **De contacto**. Pequeños tubos fabricados en plata clorurada, los cuales constan de soportes de plástico. En el lugar de contacto se utilizan almohadillas humedecidas mediante una solución conductora. Son sujetados al cráneo con bandas elásticas que se conectan utilizando pinzas. Su colocación es relativamente sencilla, pero son muy incómodos para el sujeto bajo prueba, razón por la cual los registros son de corta duración.
- **En gorro**. Electrodos que se encuentran adheridos a un gorro de goma elástica. Es posible encontrarlos en diferentes tamaños con el fin de ajustarlos a la talla del paciente. Entre sus características más relevantes es posible identificar su comodidad para registros de larga duración así como también la facilidad y precisión en su colocación.
- **De aguja**. De uso bastante limitado ya que es posible utilizarlos únicamente en recién nacidos. Este tipo de electrodos pueden ser desechables o de uso múltiple. En el caso de reutilización, es importante extremar cuidados durante su manipulación y esterilización.
- **Quirúrgicos**. Utilizados únicamente por el neurocirujano durante el procedimiento quirúrgico. Estos electrodos pueden ser del tipo dural, cortical o intracerebral.

En la Figura 2.3 es posible observar los diferentes tipos de procedimientos para la colocación de electrodos.



(a) Electrodos superficiales. **(b)** Gorro de electrodos. **(c)** Electrodos intracraneales.

Figura 2.3.: Procedimientos de colocación de electrodos. Superficiales, intracraneales y gorro de electrodos. Imágenes adaptadas de [84-86].

Los electrodos extracraneales (superficiales) son los más utilizados en las pruebas de EEG y se disponen superficialmente sobre el cuero cabelludo según la actividad que se quiera monitorizar (corteza auditiva, corteza visual, corteza motora, etc.).

Por otro lado, en la sección 2.1.2 se ha mencionado la necesidad de utilizar amplificadores para la captura de señales EEG. Esto se debe a la muy baja amplitud de la señal

original, la cual se sitúa en el orden de los microvoltios (μV). Los amplificadores diferenciales permiten obtener señales de gran calidad puesto que se encargan de amplificar la diferencia entre dos señales de entrada correspondientes a una pareja de electrodos. Esto se basa en el hecho de que el ruido afecta a ambos sensores de forma equivalente, con lo cual se espera que el ruido se vea atenuado. Finalmente, el canal EEG de la salida corresponde a la pareja de electrodos. No obstante, existen equipos de adquisición de última generación con electrodos activos que incorporan el amplificador. De esta forma, se traslada el amplificador más cerca de la fuente, mejorando considerablemente la relación señal ruido.

La ubicación de los electrodos es un factor determinante para para poder monitorizar la actividad de una región concreta del cerebro. A mediados de la década de 1940, se estableció el primer estándar para este procedimiento y que fue denominado como sistema EEG 10/20, el cual determina un sistema de coordenadas para los electrodos relacionado con su posición neuroanatómica (indicado en el propio gorro EEG). De esta forma, los montajes son totalmente reproducibles y en consecuencia pueden realizarse experimentos en las mismas condiciones.

2.1.4. Sistema de colocación de electrodos superficiales

En la actualidad existen diversos sistemas de colocación de electrodos como Illinois, Montreal, Aird, Cohn, Lennox, Merlis, Oastaut, Schwab, Marshall, etc. No obstante, el sistema internacional 10/20 sigue siendo el más utilizado. Este sistema establece la cantidad mínima necesaria de electrodos para la realización del estudio neurofisiológico de un cerebro adulto. Además, especifica el posicionamiento y etiquetado de los electrodos. De esta forma, la nomenclatura utilizada para etiquetar a cada uno de los electrodos se encuentra compuesta por:

- **Letras**. Corresponden al lóbulo cerebral sobre el que se encuentra el electrodo. En la Figura 2.4 es posible observar la distribución de los diferentes lóbulos cerebrales.
- **Números**. Corresponden al hemisferio cerebral (derecho o izquierdo) sobre el que se encuentra el electrodo. Los números pares se utilizan para el hemisferio derecho del cerebro, mientras que los impares son utilizados para el hemisferio izquierdo. Además, la numeración se inicia partiendo del centro hacia el exterior del cráneo. Los electrodos situados en la linea media del cerebro utilizan la letra “z” por corresponder a la primera consonante de la palabra inglesa “zero”.

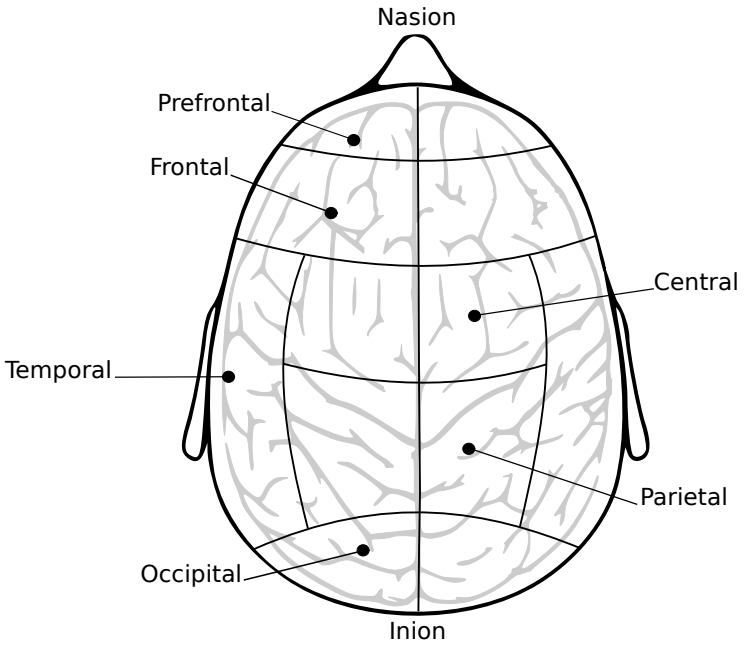


Figura 2.4.: Lóbulos cerebrales. Plano superior del cráneo que representa las zonas correspondientes a los lóbulos cerebrales. Imagen adaptada de [87].

Es importante indicar que el sistema internacional 10/20 utiliza cuatro puntos de referencia que son claves para la colocación de los electrodos:

- **Nasión**. Corresponde a la unión del hueso frontal del cráneo y el hueso de la nariz en la línea media.
- **Inión**. Corresponde a la protuberancia externa del hueso occipital, siendo la parte posterior mas baja del cráneo.
- **Punto preauricular**. Corresponde a la zona anterior del pabellón auricular (derecho e izquierdo).

A continuación se detalla el procedimiento para la colocación de electrodos basado en el sistema 10/20:

1. Se mide la distancia existente entre el nasión y el inión. El punto Fp (Frente polar) se señala al 10 % de esta distancia, considerando el nasión como punto inicial de referencia. Sin embargo, el punto O (Occipital) se señala al 10 % de esta distancia, considerando el inión como punto inicial de referencia.
2. Se toma como referencia los puntos FP y O y posteriormente se señalan entre ambos a tres puntos denominados Fz (Frontal), Cz (Central o Vertex) y el Pz (Parietal), todos ellos distribuidos en distancias equitativas.

3. Se mide la distancia existente entre los puntos preauriculares. Acto seguido se señalan los puntos temporales mediales. El punto T3 (izquierdo) se señala al 10 % de esta distancia considerando el punto preauricular izquierdo como punto inicial de referencia. Sin embargo, el punto T4 (derecho) se señala al 10 % de esta distancia considerando el punto preauricular derecho como punto inicial de referencia.

4. Se toma como referencia los puntos temporales mediales (derecho e izquierdo) y se colocan los electrodos a un 20 % por encima de ambos puntos de referencia, siendo C3 el del lado izquierdo y C4 el del lado derecho.

5. Se colocan los electrodos F3 (izquierdo) y F4 (derecho) a la misma distancia del punto frontal medio (Fz) y la línea de electrodos temporales.

6. Se colocan los electrodos P3 (izquierdo) y P4 (derecho) a la misma distancia del punto P medio y la línea de los electrodos temporales.

7. Se mide la distancia existente entre los puntos medio Fp y O pasando por T3. Los electrodos FP1 y FP2 se colocan al 10 % de esta distancia a través de Fp, mientras que los electrodos O1 y O2 se colocan a un 10 % a través de O.

8. Se colocan los electrodos F7 (izquierdo) y F8 (derecho) de forma equidistante con respecto a los puntos FP1 y T3 y FP2 y T4, respectivamente. Luego se colocan los electrodos T5 (izquierdo) y T6 (derecho) sobra la linea media entre los puntos T3 y O1 o T4 y O2 respectivamente.

9. Se colocan los electrodos los electrodos auriculares Al y A2, en un 10 % de la distancia tomando como puntos de referencia a los temporales T3 y T4 respectivamente.

En la Figura 2.5 se puede observar el esquema completo del sistema 10/20 considerando la disposición de los electrodos y los puntos establecidos para la medición. Además, en la Tabla 2.1 se indican las etiquetas correspondientes a cada uno de los electrodos utilizados para el procedimiento de colocación superficial.

Tabla 2.1.: Nomenclatura correspondiente a la posición de los electrodos.

Área Central	Hemisferio Izquierdo	Línea Media	Hemisferio Derecho
FrontoPolar	FP1		FP2
Frontal	F3	FZ	F2
Fronto Temporal	F7, C3	CZ	F8, C4
Temporal Medio y Parietal	T3, P3	PZ	T4, P4
Temporal Posterior y Occipital	T5, O1		T6, O2

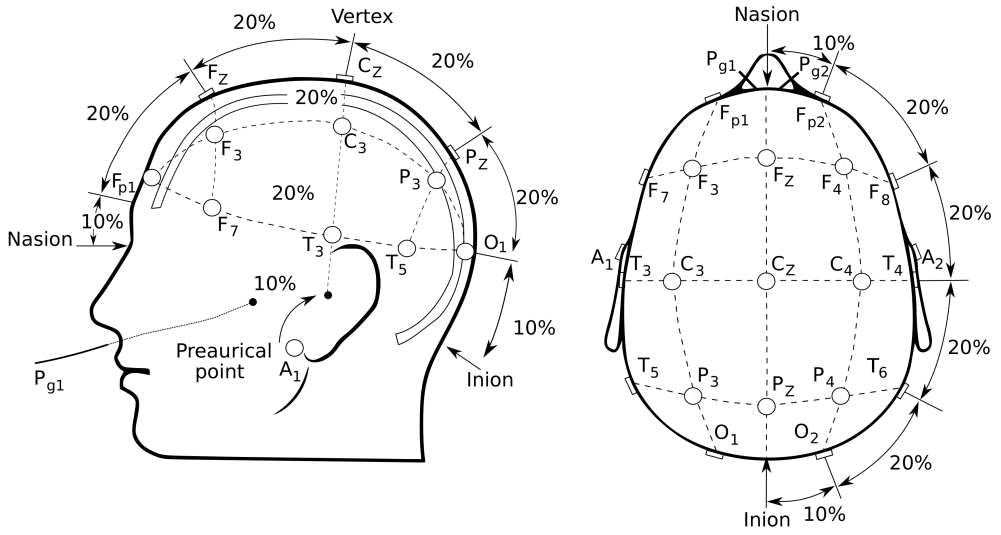


Figura 2.5.: Sistema internacional 10/20. Posición de los electrodos dispuestos sobre el cuero cabelludo. Imagen adaptada de [88].

2.1.5. Derivación y montaje de EEG

Los electrodos posicionados sobre el cuero cabelludo son los responsables del registro de las señales EEG. Éstos son colocados de acuerdo a protocolos establecidos que especifican de forma precisa el procedimiento a seguir, como en el caso del sistema 10/20. De forma individual, cada electrodo corresponde a un punto de registro de la señal eléctrica cerebral. No obstante, para realizar este registro es necesario contar con dos terminales pues uno de ellos actuará como referencia. Esto se debe a que cada canal del EEG corresponde a la diferencia de potencial entre dos electrodos. A esta diferencia se le denomina derivación y puede ser de dos tipos:

- **Derivación bipolar**. Ambos electrodos están ubicados sobre áreas en las que se genera actividad cerebral.
- **Derivación unipolar con electrodo de referencia**. Uno de los electrodos se utiliza como referencia común. Resulta habitual que éste se coloque en el vertex (Cz) o en una zona inactiva o neutra, como por ejemplo el lóbulo de la oreja (A1 ó A2) o el mastoides.

2.1.6. Características de la actividad cerebral

La naturaleza de las señales EEG hacen que éstas no sean periódicas ni estacionarias. Es decir, que no es posible determinar una única frecuencia de oscilación ni su su

amplitud porque varían a lo largo del tiempo. No obstante, se diferencian distintas bandas (también conocidas como ritmos cerebrales u ondas) de frecuencia asociadas con una determinada actividad cerebral. Las diferencias existentes entre las diversas bandas pueden agruparse de acuerdo a los siguientes parámetros:

- **Frecuencia**. Corresponde a la medición realizada a una onda cerebral considerando la cantidad de ciclos que se dan por segundo. Esto determina el ritmo cerebral, cuya unidad de medida es expresada en Hertz (Hz). De acuerdo a la frecuencia de las ondas cerebrales, éstas se dividen en los siguientes grupos: alfa, beta, gamma, delta y theta.
- **Distribución topográfica**. Corresponde a la región cerebral donde se produce la detección del impulso eléctrico, normalmente tomando como referencia el sistema de coordenadas 10/20.
- **Forma, amplitud y duración**. La forma de una onda aislada puede ser: regular, irregular, aguda, compleja, bifásica, y trifásica, entre otras. La amplitud se mide en microvoltios y es habitual que fluctúe entre 20 y 40 μV. La duración de una onda se expresa en milisegundos (ms).
- **Reactividad**. Es la capacidad de modificación de un ritmo ante estímulos como la apertura y el cierre de ojos, estimulación eléctrica, proceso mental, estado de alerta, etc.

A continuación se detallan las características de los diferentes ritmos cerebrales. En general, se distinguen cinco tipos de ondas que se diferencian en base a sus rangos de frecuencia, las cuales se muestran en la Figura 2.6:

- **Ondas delta (δ)**. Su rango de frecuencia está entre 0,1 y 3 Hz, mientras que su amplitud se sitúa entre 50 y 200 μV. En este sentido, son las ondas de mayor amplitud y menor frecuencia. Por otra parte, su origen preciso es indeterminado. No obstante, aparece en toda la corteza cerebral y considerando el tipo de actividad realizada su presencia adquiere mayor preponderancia en determinadas áreas. Normalmente, su presencia es típica en niños menores de 3 meses mientras que en los adultos su presencia se asocia con el sueño profundo, los trastornos cerebrales graves y el estado de vigilia.
- **Ondas theta (θ)**. Su rango de frecuencia está entre 3 y 8 Hz, mientras que su amplitud se sitúa entre 20 y 100 μV. Además, es posible localizarlas en las áreas frontal, central y temporal. Normalmente, su presencia es típica en niños menores de 15 años, aunque en los adultos su presciencia se caracteriza en la fase de sueño ligero o somnolencia. Además, guarda relación con el estrés

emocional debido a la frustración o la decepción, la inspiración creativa, la meditación profunda, incluso en estado de hiperventilación y la fatiga.

- **Ondas alpha (α).** Su rango de frecuencia está entre 8 y 13 Hz, mientras que su amplitud se sitúa entre 50 y 60 μV en niños y entre 15 y 45 μV en adultos. En cuanto a su localización se presentan habitualmente en la región occipital. En adultos, es posible observar la presencia de estas ondas cuando se encuentran en estado de relajación e inactividad (con los ojos cerrados).

- **Ondas beta (β).** Su rango de frecuencia está entre 13 y 30 Hz y dividido en 3 bandas: baja (13-18 Hz), media (18-25 Hz) y alta cuando es superior a los 25 Hz. Su amplitud varía entre 2 y 20 μV pudiendo en ocasiones alcanzar hasta los 50 μV. En cuanto a su localización se pueden encontrar en las regiones parietal y frontal. Suelen estar asociadas a la atención activa, situaciones de concentración y actividad mental para la resolución de problemas concretos.

- **Ondas gamma (γ).** Cuentan con un rango de frecuencia de entre 30 y 80 Hz, mientras que su amplitud varía entre 2 y 10 μV. En este sentido, son las ondas de mayor frecuencia y menor amplitud. En cuanto a su localización, se pueden encontrar en las región frontal y el área central. Estas ondas se asocian a estados de gran actividad mental, excitación, extrema atención, pensamientos abstractos y elevado nivel de procesamiento de información.

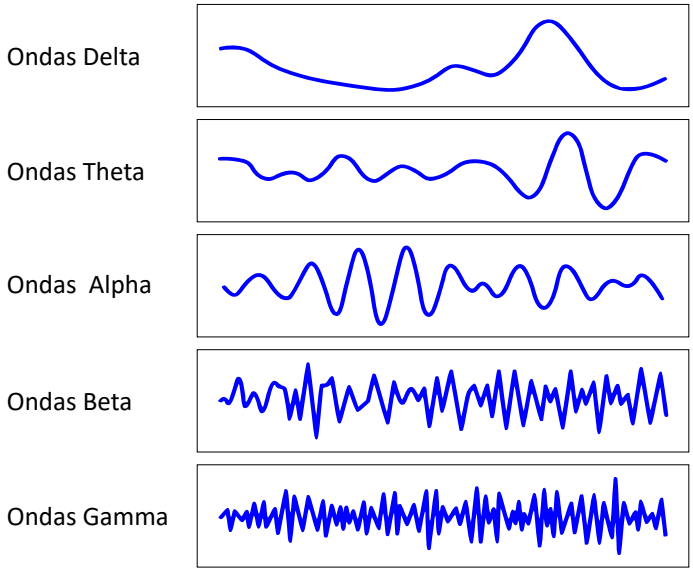


Figura 2.6.: Principales ondas cerebrales. Cinco tipos de ondas cerebrales diferentes ordenadas de menor a mayor por su rango de frecuencia. Imagen adaptada de [89].

Detectar y comprender las ondas cerebrales a partir del análisis visual del EEG entraña un alto grado de complejidad e implica un importante desafío incluso para los ojos más experimentados. Por lo tanto, hay que aplicar técnicas avanzadas de procesamiento de señales para el análisis y discriminación de los componentes que componen a la señal EEG. Este proceso requiere del cálculo de descriptores que caractericen la señal en el dominio del tiempo, de la frecuencia o ambos simultáneamente (descriptores tiempo-frecuencia). La determinación de los descriptores más adecuados, desde el punto de vista del poder discriminante de los mismos, es una tarea compleja y cuyo resultado es imposible conocer a priori. En aplicaciones BCI es habitualmente utilizar descriptores estadísticos de la señal en el dominio de la frecuencia con el fin de encontrar patrones en la potencia de las diferentes bandas EEG que permitan diferenciar los movimientos imaginados. No obstante, otros descriptores estadísticos comúnmente utilizados en procesamiento de señales también podrían aportar información discriminante o complementar la proporcionada por los descriptores espectrales.

2.1.7. Descriptores temporales de las señales EEG

Los descriptores temporales pueden definirse como estadísticos calculados para cualquier tipo de señal en el dominio del tiempo que proporcionan información condensada de las características de la señal. Concretamente, los descriptores estadísticos de las señales EEG han sido utilizadas con éxito para la detección de patologías neurológicas [90]. A continuación, se detalla de forma matemática el cálculo realizado para obtener dichas características.

Dada una señal x_i, correspondiente a un canal EEG en el instante de tiempo i, $i = 1, \ldots, N$, se define:

- **Amplitud media**. Medida en un periodo de tiempo determinado o a lo largo de un número de muestras determinado.

$$\mu_t = \frac{1}{N} \cdot \sum_{i=1}^{N} x_i \tag{2.1}$$

- **Varianza de amplitud**. Mide la dispersión de las muestras en torno al valor medio.

$$\sigma_t = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} (x_i - \mu_t)^2} \tag{2.2}$$

- **Asimetría temporal**. Indica el grado de simetría de una señal ante una transformación de inversión del eje del tiempo.

$$\beta_t = \frac{1}{N} \cdot \sum_{i=1}^{N} \left(\frac{x_i - \mu_t}{\sigma_t} \right)^3 \tag{2.3}$$

2.1.8. Descriptores espectrales de las señales EEG

Los descriptores espectrales pueden obtenerse a partir de señales en el dominio del tiempo mediante la aplicación de diferentes métodos matemáticos y de procesamiento de la señal para estimar la distribución de la potencia en el ancho de banda de la señal. En los trabajos [91, 92], se han utilizado descriptores espectrales para dar soporte a la detección de crisis epilépticas considerando los cambios que se producen en ciertas bandas de frecuencia cuando estas tienen lugar.

El cálculo de las características en el dominio de la frecuencia se realiza a partir de la densidad espectral de potencia (PSD), mediante la cual es posible observar la distribución de la potencia o la energía en las distintas frecuencias del espectro de la señal [93]. Sin embargo, realizar de forma directa el cálculo de la PSD mediante la aplicación de la transformada de Fourier a señales EEG no es lo más adecuado [94] ya que estas señales no son estacionarias ni periódicas y tienen ruido superpuesto. Además, el cálculo de la PSD basado en análisis de Fourier reduce la precisión de la estimación de la PSD. Esto se debe a que el espectro es ruidoso por la alta varianza en su estimación y por el sesgo que se genera como consecuencia de la fuga de energía a través de las frecuencias. Por tanto, con el fin de evitar las limitaciones de la transformada de *Fourier* para el cálculo de la PSD en el procesamiento de señales de EEG, se impone la utilización de técnicas de mayor solidez como la transformada de *Wavelet* [95] y el periodograma de *Welch* [96, 97].

Welch [98] propone un método para la estimación del periodograma que incorpora algunas mejoras sobre el estimador estándar con el fin de reducir el ruido en la señal. Sin embargo, el principal inconveniente de esta propuesta es la reducción de la resolución espectral de la señal. No obstante, se han desarrollado otras propuestas que combinan las ventajas de los estimadores estándar y el periodograma de Welch al mismo tiempo que evitan sus limitaciones e inconvenientes. En este contexto,

Thomson [99] propone la utilización de ventanas o *tapers* para señales en el dominio del tiempo con el objetivo de reducir la fuga producida por los diferentes lóbulos laterales de una ventana en el dominio de la frecuencia. Otra forma de conseguir esto es utilizar los *tapers* con baja potencia espectral en los lóbulos laterales. De esta forma, el cálculo de la PSD se realiza de la siguiente forma:

$$\text{PSD}\left(\omega\right) = \left| \sum_{i=0}^{N-1} x_i \cdot a_i \cdot e^{-j\omega i} \right|^2 \tag{2.4}$$

donde ω es la frecuencia, x_i, $i = 1, \ldots, N$, es la i-muestra de la serie temporal correspondiente a la señal y a_i es la ventana o *taper* en el dominio del tiempo. Con el fin de mantener invariable la potencia total se procede a normalizar la energía total de los *tapers*. Mediante la aplicación de este método para múltiples *tapers*, es posible reducir la estimación de la varianza en cada frecuencia.

Al método propuesto por Thomson se le denomina método *multi-taper*, y considera la utilización de K *tapers* ortogonales para obtener K muestras ortogonales de los datos x_i, con lo cual se consiguen K estimaciones espectrales $PSD_K(\omega)$ que pueden promediarse para disminuir la varianza. La propuesta de Thomson establece seleccionar mediante un proceso de optimización los *tapers* que minimizan la fuga y maximizan la energía dentro de un ancho de banda específico. El cálculo de la PSD puede utilizarse para caracterizar los siguientes descriptores:

- **Centroide espectral**. Obtenido a partir del cálculo de la media ponderada de las frecuencias de la señal analizada. Puede considerarse el centro de masas del espectro de potencia. Es decir, representa la frecuencia en donde se encuentra concentrada la mayor parte de la energía.

$$\text{SC} = \frac{\sum_{k=0}^{N} \omega_k \cdot \text{PSD}\left(\omega_k\right)}{\sum_{i=0}^{N} \text{PSD}\left(\omega_k\right)} \tag{2.5}$$

  donde $\text{PSD}(\omega_k)$ es la PSD del k-ésimo intervalo de frecuencia en el espectro, y ω_k es la frecuencia central de dicho intervalo.

- **Dispersión espectral**. Es una medida de la dispersión del espectro tomando como punto de referencia al centroide espectral.

$${\sigma_s}^2 = \frac{\sum_{k=0}^{N} \left(\omega_k - \text{SC}\right)^2 \cdot \text{PSD}\left(\omega_k\right)}{\sum_{i=0}^{N} \text{PSD}\left(\omega_i\right)} \tag{2.6}$$

- **Asimetría espectral**. Mide el grado de simetría de una distribución.

$$\beta_s = \frac{\sum_{k=0}^{N} \left(\frac{\omega_k - \mathrm{SC}}{\sigma_s}\right)^3 \cdot \mathrm{PSD}\left(\omega_k\right)}{\sum_{k=0}^{N} \mathrm{PSD}\left(\omega\_k\right)} \tag{2.7}$$

- **Planitud espectral**. Mide la forma en que la potencia se propaga a través del espectro.

$$\tau_s = \frac{\left(\prod_{k=0}^{N} \mathrm{PSD}\left(\omega_k\right)\right)^{1/N}}{(1/N)\sum_{k=0}^{N} \mathrm{PSD}\left(\omega_k\right)} \tag{2.8}$$

  donde N es el número de intervalos de frecuencia.

2.1.9. Análisis multiresolución de señales EEG

El análisis de señales basado en la transformada de Fourier para extraer características espectrales tiene dos limitaciones principales: por un lado, los cambios en la distribución del espectro con el tiempo son difíciles de visualizar, y por otro lado, las señales deben ser periódicas para no violar el supuesto de estacionariedad del análisis de Fourier. Dado que las señales EEG no son estacionarias ni periódicas, resulta más apropiado recurrir a métodos de análisis que no partan de los supuestos anteriores. En este sentido, los métodos de análisis tiempo-frecuencia permiten aprovechar las ventajas del análisis en tiempo y el de frecuencia pero sacrificando resolución en un dominio en pro de obtener una mayor resolución en el otro.

Existen diferentes métodos de análisis tiempo-frecuencia que permiten evitar las suposiciones de estacionariedad y periodicidad como la transformada corta de Fourier (STFT) [100]. Dicha transformada establece que una señal no periódica y no estacionaria sí pueda serlo en un intervalo de tiempo lo suficientemente corto para disponer de la suficiente resolución en el dominio de la frecuencia. Otro de los métodos de análisis tiempo-frecuencia más utilizados se basa en la transformada de *Wavelet* (WT) [101], que además, resulta apropiada para señales no estacionarias. En lugar de suponer una señal estacionaria para toda la duración de la señal, la WT supone estacionariedad sólo durante un corto periodo de tiempo en el que la función de *Wavelet* se asemeja a una señal sinusoidal. Esta nueva suposición resulta apropiada para señales biomédicas como las señales EEG, de electrocardiografía (ECG) o electromiografía (EMG) [101], pues se utiliza con frecuencia en el análisis y extracción de características de estas señales [102-104].

Transformada de *Wavelet* (WT)

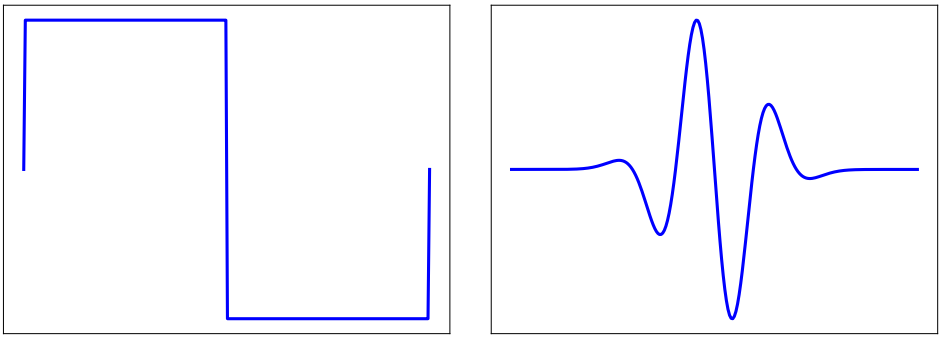
El análisis mediante la Transformada de *Wavelet* (WT) se basa en la introducción de una función base adecuada que permita caracterizar la señal a través de la distribución de la amplitud en dicha función. Dicho de otra forma, se trata de estimar la similitud local de una señal con una determinada función base que llamaremos *wavelet*. La WT constituye una poderosa herramienta para el análisis de señales que permite extraer información espectral y temporal local simultáneamente (lo que llamamos análisis tiempo-frecuencia). Además, ha tenido un gran impacto en diversos campos como el análisis de imágenes, la visión por computador y la codificación de señales. Así, la WT proporciona información en el plano tiempo-frecuencia teniendo en cuenta diferentes escalas en todos los rangos de frecuencia de la señal (entre 0 y la frecuencia de Nyquist [105]). Esto permite encontrar patrones de variación temporal de la frecuencia de una serie temporal.

Las llamadas *wavelets* $\psi(t)$ son funciones oscilantes (sinusoidales) que se desvanecen a lo largo de un tiempo t y están bien localizadas temporalmente (es decir, sus posiciones en el eje del tiempo). A partir de una función madre se define una familia de *wavelets*, $\psi_a^b(t)$, mediante operaciones de escalado y traslación de la *wavelet* madre $\psi(t)$. Por tanto, la forma de las *wavelets* que componen una familia la determinan el factor de escala (que define la dilatación o contracción de la *wavelet* madre y que determina la frecuencia de la función), y el factor de traslación, que determina la traslación de la *wavelet* en el tiempo (en el caso unidimensional). Una familia de *wavelet* puede definirse como:

$$\psi_a^b(t) = \frac{1}{\sqrt{a}} \cdot \psi\left(\frac{t-b}{a}\right) \tag{2.9}$$

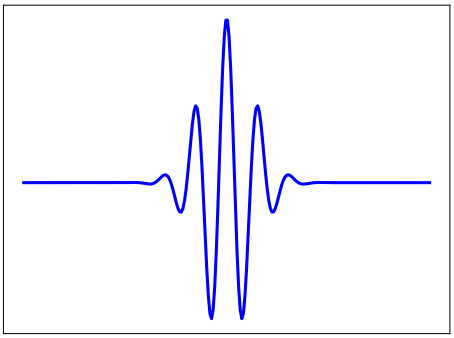
donde b es el factor de escala y a el factor de translación. Una familia de funciones *wavelet* $\psi_a^b(t)$ constituye una base ortonormal del espacio de funciones de cuadrado integrable $L^2(\mathbb{R})$.

Existen diferentes funciones *wavelet* madre que pueden utilizarse dependiendo de la información que se quiera extraer. En la Figura 2.7 se muestran algunas de las más comunes. No obstante, no todas las familias de *wavelet* madre resultan útiles para la descomposición tiempo-frecuencia. Concretamente, en el análisis de señales EEG suele utilizarse la familia *Morlet*, que surge a partir del enventanado Gaussiano de una señal sinusoidal. Dicha familia de *wavelets* proporciona buena resolución temporal para localizar información en el dominio de la frecuencia.

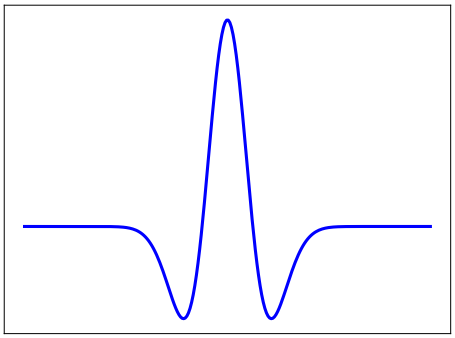


(a) Haar

(b) Gaussiano



(c) Morlet



(d) Sombrero mexicano

Figura 2.7.: ***Wavelets* madre.** Algunas de las funciones madre más utilizadas.

Una vez definida la función *wavelet* madre a utilizar, la transformada de *wavelet* se obtiene realizando la operación de convolución de la señal a transformar con las funciones del conjunto de *wavelet* que definen la familia $\psi_a^b(t)$. Existen dos tipos de WT: en primer lugar, la transformada continua de *wavelet* (CWT) [106], que utiliza el conjunto total de posibles *wavelets* en un rango infinito de factores de escala y translación. Dicha transformada $T(a,b)$ se define como:

$$T(a,b) = \frac{1}{\sqrt{|a|}} \cdot \int_{-\infty}^{\infty} x\left(t\right) \cdot \psi^* \left(\frac{t-b}{a}\right) dt \qquad a,b \in \mathbb{R}, a \neq 0. \tag{2.10}$$

donde ψ^* es el conjugado de la *wavelet* madre que será escalada y trasladada punto a punto con el fin de establecer los diferentes niveles de comparación respecto a la señal $x(t)$. Por otro lado, también existe la transformada discreta de *wavelet* (DWT) de una función $x(t)$, que utiliza un conjunto discreto de *wavelets* en un rango finito de factores de escala y traslación definidos por los parámetros m y n, respectivamente:

$$T_{m,n} = \int_{-\infty}^{\infty} x(t) \cdot \psi_{m,n}(t) dt. \tag{2.11}$$

De esta forma, una señal x(t) puede expresarse como

$$x(t) = \sum_{n=-\infty}^{n=\infty} c_n \phi(t-n) + \sum_{m=-\infty}^{m=\infty} \sum_{n=-\infty}^{n=\infty} d_{m,n} \psi(2^m t - n). \tag{2.12}$$

donde $\phi(\cdot)$ es la función de escalado, $\psi(t)$ la función *wavelet* madre, $d_{j,k}$ son los coeficientes de *wavelet* y c_n los coeficientes de escalado.

La CWT y DWT dan como resultado los coeficientes que representan las coordenadas de la función a descomponer en la base ortonormal de funciones de *wavelet* en cada instante de tiempo. Por tanto, la WT permite expresar una serie temporal en el espacio generado por una base ortonormal compuesta por un conjunto de funciones de *wavelet* de la familia que se adapte a la señal a descomponer. Esta es una de las ventajas de la WT con respecto a la transformada de Fourier, cuya base está únicamente compuesta por funciones seno y coseno.

Análisis multiresolución (MRA)

Como se ha indicado antes, los coeficientes obtenidos contienen información que permite estimar de forma directa la energía de la señal en diferentes frecuencias. La descomposición realizada puede organizarse según un esquema jerárquico de subespacios anidados. A esta descomposición se le llama análisis multiresolución (MRA) [107], donde el espacio de funciones de cuadrado integrable $L^2(\mathbb{R})$ se descompone en subespacios que son aproximaciones multiresolución de $L^2(\mathbb{R})$.

El algoritmo para el cálculo de la DWT propuesto por Mallat [108] descompone la señal en diferentes bandas de frecuencia haciéndola pasar consecutivamente por dos filtros de respuesta impulsiva infinita (IIR) en cuadratura, un paso alto (g) y un paso bajo (h). En esta implementación, g está relacionado con la función de escalado mientras que h depende de la función *wavelet* madre. Así, para una señal discreta $x(k)$, en [108] se define $g(h)$ como:

$$g(h) = (-1)^n \cdot h(1-n) \tag{2.13}$$

y la salida de los filtros paso bajo (H_L) y paso alto (G_L) como:

$$H_L = \sum_{k=0}^{N} h(k-2L) \cdot x(k) \tag{2.14}$$

$$G_L = \sum_{k=0}^{N} g(k-2L) \cdot x(k) \tag{2.15}$$

De esta forma, la convolución de la señal $x(k)$ con $h(k-2L)$ proporciona para el nivel L la componente paso-bajo, también llamada componente de baja resolución o coeficientes aproximados. Sin embargo, la convolución de $x(k)$ con $g(k-2L)$ proporciona la componente paso-alto, denominada también como componente de alta resolución o coeficientes detallados. Así, la señal original se descompone en dos subbandas: $[0, F_{N/2}]$ y $[F_{N/2}, F_N]$, donde F_N es la componente máxima de frecuencia de la señal (frecuencia de Nyquist/2) en el nivel anterior de descomposición.

La implementación de MRA consiste en una serie de etapas compuestas por los dos filtros digitales mencionados anteriormente y dos decimadores $1/2$. La salida submuestreada del primer filtro paso-alto proporciona los coeficientes aproximados (A_1), mientras que la del primer filtro paso-bajo proporciona los coeficientes detallados (D_1). En el siguiente paso, las salidas A_1 y D_1 se vuelven a descomponer, obteniendo un segundo nivel de descomposición, A_2 y D_2. Este proceso continúa iterativamente hasta el nivel máximo de descomposición, el cual vendrá determinado por la frecuencia de Nyquist de la señal original. El árbol de descomposición resultante para seis niveles puede verse en la Figura 2.8:

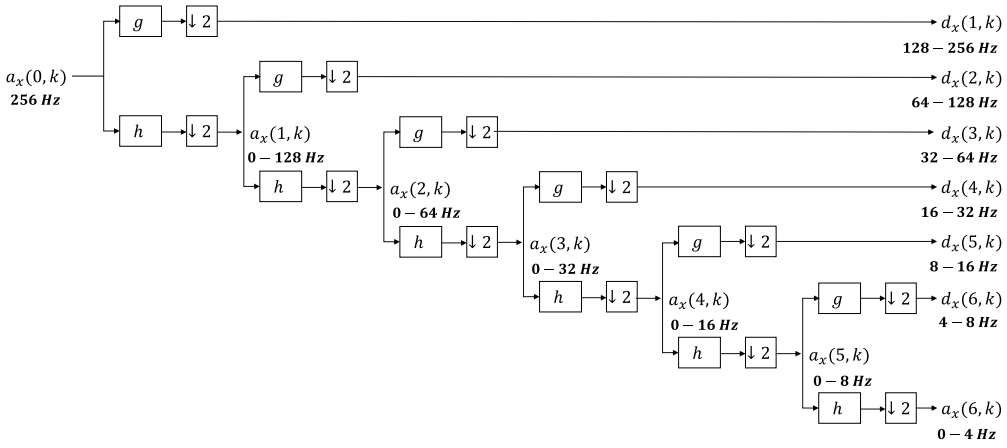


Figura 2.8.: Árbol de descomposición de *wavelet* de seis niveles.

El análisis multiresolución (MRA) proporciona alta resolución temporal y baja resolución espectral para altas frecuencias así como baja resolución temporal y alta

resolución espectral para bajas frecuencias. Esto resulta de especial interés dado que permite descomponer una señal en diferentes bandas y evaluar cada una de ellas con una resolución apropiada [107]. La teoría del análisis multiresolución desarrollada por Mallat [107] permite el estudio de la información contenida en diferentes componentes espectrales de la señal, es decir, analizar señales diferentes bandas de frecuencia. MRA utiliza la WT para descomponer la señal en distintos niveles de resolución, expresándola mediante la utilización de bases ortonormales de funciones *wavelets*.

Energía de la señal en cada banda de frecuencias

Como se ha visto, el análisis MRA proporciona una descomposición de la señal en sub-bandas. Un descriptor que suele resultar de interés en el reconocimiento de patrones y la clasificación de señales EEG es la energía total de las bandas de frecuencia Delta, Theta, Alpha, Beta y Gamma. Este descriptor puede calcularse a partir de los coeficientes aproximados y detallados del nivel l mediante la siguiente expresión:

$$E_l = \sum_{k=0}^{n} |d(l,k)|^2 \tag{2.16}$$

De esta forma, es posible calcular la energía la energía media de cada banda teniendo en cuenta los coeficientes correspondientes. La distribución de la energía en las diferentes bandas proporciona información de dónde se pueden extraer patrones relacionados con la actividad que el sujeto esté desarrollando [109].

2.2. Sistemas BCI basados en señales EEG

Actualmente existe un gran impulso en el desarrollo de aplicaciones BCI basadas en señales EEG como resultado de los importantes avances en las investigaciones realizadas en esta área. A pesar de que existen otros tipos de señales que pueden ser utilizadas para las aplicaciones BCI, habitualmente se prefieren los registros de la señal cerebral obtenidos mediante el procedimiento de la Electroencefalografía. Las ventajas del EEG son: la sencillez de la técnica, su carácter no invasivo (ningún tipo de riesgo), su portabilidad y el bajo coste de realización [110].

Las aplicaciones BCI tienen utilidades en diferentes campos como: (i) diagnóstico clínico (accidentes cerebrovasculares, epilepsia, autismo, control de atención, trastornos emocionales y dislexia); (ii) desarrollo de dispositivos para mejorar la calidad de vida de las personas que cuenten con discapacidades (neuroprótesis, sillas de ruedas o sistemas de comunicación); (iii) medición de factores humanos relacionados con su entorno (estrés o sueño), y (iv) creación de aplicaciones recreativas.

En general, los procesos de implementación de sistemas BCI basados en señales EEG requieren previamente definir el paradigma y protocolo que será utilizado a lo largo de las diferentes etapas del experimento. En este sentido, no existe un protocolo estándar para los sistemas BCI, por lo que los grupos de investigación diseñan diferentes tecnologías adaptadas a sus requisitos. Es por ello que no resulta sencillo establecer comparaciones entre los diferentes sistemas implementados.

El proceso se inicia definiendo la tarea (imaginaria o visual) que el sujeto deberá realizar hasta dominar su actividad cerebral durante el registro de la señal EEG. Posteriormente, la señal EEG registrada se utiliza para generar el decodificador neural para el reconocimiento de los patrones presentes en dicha señal. A continuación, se solicita al sujeto que realice nuevamente la tarea de tal modo que el decodificador neural interprete la señal y se encargue del control del dispositivo BCI. En la Figura 2.9 es posible observar el esquema básico de los sistemas BCI.

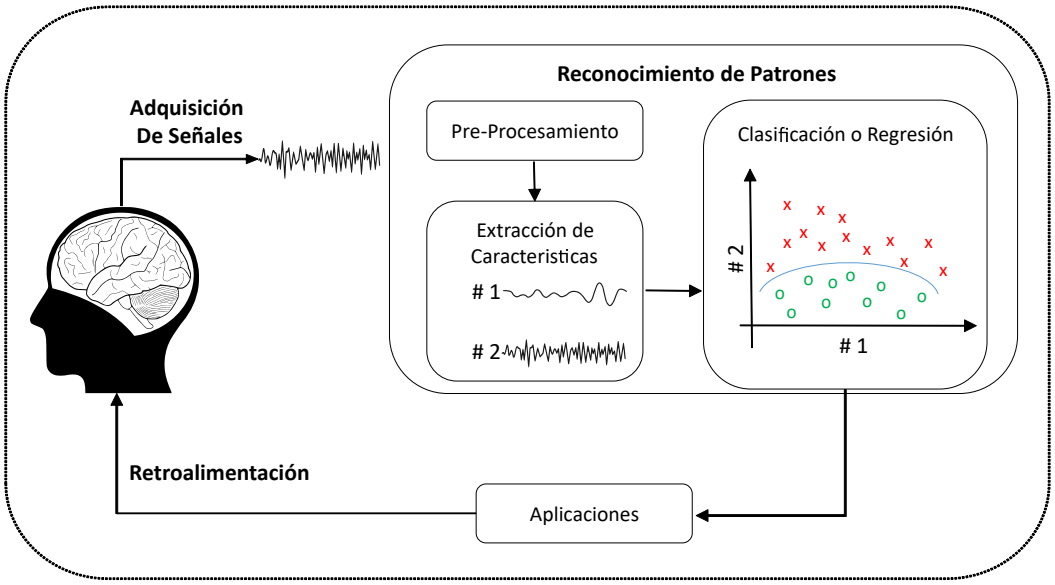


Figura 2.9.: Esquema básico de la interfaz cerebro-computador. Arquitectura general de un sistema BCI. Imagen adaptada de [111].

El rendimiento alcanzado por los sistemas BCI se encuentra supeditado en gran medida a las posibilidades que tiene el sujeto para dominar sus patrones cerebra-

les [109]. Habitualmente, es necesario que el sujeto se someta a un proceso de entrenamiento para adquirir estas capacidades de control. Para conocer la evolución del sujeto se requiere de una retroalimentación que indique los respectivos avances del entrenamiento.

Sistemas BCI según el procedimiento de adquisición de la señal

La clasificación de los sistemas BCI puede realizarse considerando diferentes características del sistema como son: (i) el procedimiento de adquisición de la señal de entrada (invasivo o no invasivo); (ii) si requiere de estimulación externa o es voluntario; (iii) el tipo de característica de la señal a utilizar, y (iv) si precisa o no de entrenamiento previo para realizar la tarea mental (imaginación motora). De entre los métodos de extracción de EEGs se encuentran:

- **Métodos invasivos**. Utiliza electrodos dispuestos directamente sobre la corteza cerebral para el registro de la actividad eléctrica. El Electrocorticograma (EcoG) es un ejemplo de método invasivo que tiene como principal inconveniente requerir de la intervención quirúrgica, lo cual agrega la complejidad y los riesgos propios de este tipo de procedimiento. Además, otro factor negativo es que estos dispositivos pueden ser rechazados por el sujeto portador del implante. También la vida útil de estos dispositivos es limitada. No obstante, las señales registradas tienen una mejor relación señal-ruido en comparación con los métodos no invasivos, por lo que es posible concentrar el estudio de las señales a grupos específicos de neuronas. Es decir, permite realizar el filtrado espacial de la señal de forma más eficiente y enfocarse exclusivamente en el grupo neuronal de interés [112]. Por otra parte, entre los factores positivos del EcoG destacan el ahorro de tiempo en cada sesión de BCI puesto que los electrodos ya se encuentran fijos de forma permanente.
- **Métodos no invasivos**. Es el método más utilizado en sistemas BCI y utiliza electrodos dispuestos en un gorro posicionados normalmente para monitorizar la corteza motora. Existen diferentes técnicas, como la MEG [113], que mide los campos magnéticos que producen las corrientes eléctricas cerebrales con el objetivo de mapear la actividad cerebral con gran resolución espacio-temporal. Una utilidad práctica consiste en conocer la ubicación precisa del origen de los ataques epilépticos y estudiar las funciones cognitivas. Por otro lado lado, la técnica fMRI [114] se utiliza para generar imágenes internas del cerebro a través de campos magnéticos de gran intensidad. Mediante la utilización de este procedimiento es posible detectar la actividad del cerebro en base a la

medición del flujo de sangre hacia las distintas zonas del mismo. Su principal ventaja consiste en ofrecer una gran resolución espacial, con lo cual es posible identificar con gran precisión la actividad de zonas específicas del cerebro en base al nivel de oxigeno en sangre. Como desventajas, su baja resolución temporal y la no portabilidad del sistema de adquisición.

Otra técnica no invasiva bastante novedosa es la fNIRS [115], la cual está basada principalmente en la emisión de haces de luz infrarroja. Estos haces se miden con medios ópticos para detectar el consumo de oxígeno dentro del cerebro a través de las células sanguíneas. Dado que la actividad cerebral requiere de oxígeno, detectar los niveles de concentración de hemoglobina en la sangre dentro de determinadas zonas del cerebro permiten conocer la fuente y el nivel de activación de la corteza cerebral.

Sistemas BCI según el tipo de estimulación

Otra forma de clasificar a los sistemas BCI es según el tipo de estimulación requerido para registrar el EEG. En este sentido, es posible dividir a los sistemas BCI en endógenos y exógenos:

- **BCI endógenos**. También son conocidos como paradigmas de imaginación motora. La señal de entrada corresponde a los ritmos de la señal EEG. No requieren de estímulos externos y son controlados según la voluntad del sujeto [116].
- **BCI exógenos**. También son conocidos como paradigmas de estimulación externa. La señal de entrada corresponde a los potenciales evocados producidos por el cambio de potencial de la señal EEG como resultado de la aparición de estímulos externos [117].

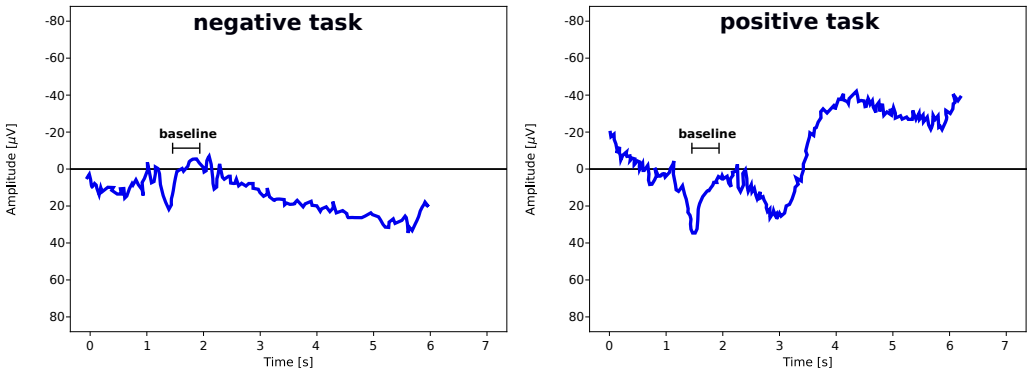
Sistemas BCI según las características de la señal EEG

La investigación en sistemas BCI para EEGs es la que tiene mayor difusión debido a todas las ventajas que ofrece este tipo de señal. No obstante, las señales EEG tienen diversas características que permiten su clasificación en distintos grupos:

- **Basados en potenciales corticales lentos**. Los potenciales corticales lentos (SCP) tienen su base en la teoría de la regulación voluntaria del sujeto, quien se encarga de regular la atención y excitabilidad del córtex cerebral [118]. Los

SCP son cambios lentos de voltaje que se generan en la corteza del cerebro y que tienen una duración aproximada de entre 0,5 y 10 segundos. Por otra parte, sus componentes espectrales varían entre 0,1 a 1 Hz.

En este sistema BCI se le solicita al sujeto que regule de forma voluntaria su actividad cerebral mediante la utilización de un paradigma de dos fases (binario). La primera fase corresponde a la etapa de preparación o atención, mientras que en la segunda se le solicita al sujeto que efectúe una actividad mental en particular. Tras concluir la primera fase se procede a tomar la medida del nivel del EEG ya que luego será utilizada como límite o umbral de comparación para la toma de decisiones al terminar la segunda fase. En el caso de los SCP negativos, éstos están habitualmente asociados al movimiento y otras funciones que impliquen activación cortical. Por otro lado, los SCP positivos se asocian con la reducción de la actividad cortical.



(a) SCP Negativo. **(b)** SCP Positivo.

Figura 2.10.: Potenciales corticales lentos. SCP en el EEG. Imágenes adaptadas de [119].

- **Basados en movimiento motor imaginario o ritmos sensoriomotores**. Los sistemas BCI centrados en el movimiento motor imaginario [120] imaginan el movimiento de alguna parte del cuerpo en vez de ejecutar el movimiento de forma física. En [121] se explica la responsabilidad de determinadas áreas del cerebro para la generación del movimiento físico. Existe una gran cantidad de paradigmas que corresponden a este grupo. No obstante, las investigaciones [122] están centradas principalmente en los ritmos sensoriomotores (SMR) y los basados en la cinemática corporal imaginada (IBK).

  Específicamente, el BCI basado en SMR utiliza diferentes clases de imágenes motoras como el movimiento de la manos (derecha o izquierda), de la lengua, de los pies, entre otros. También se incluyen tareas mentales como la rotación de cubos, operaciones aritméticas, etc. Las tareas mentales realizadas para BCI del tipo SMR pueden ser detectadas y registradas en la zona de la corteza

sensoriomotora [123]. Estos registros muestran cambios de amplitud en los SMR que corresponden a las ondas α (8–12 Hz) y β (13–28 Hz). Las utilidades implementadas mediante el uso del SMR han permitido a sus usuarios tener mayor control sobre dispositivos externos con el fin de contar con mayor autonomía física dentro de su medio.

- **Basados en potenciales evocados.** Los potenciales evocados (PE) son señales eléctricas que se producen en el cerebro ante diferentes estímulos aplicados sobre los sentidos de la vista, el oído o el tacto,permitiendo obtener el tiempo de respuesta del cerebro ante los diferentes estímulos sensoriales. La clasificación de los PE se determina en base al tipo de estímulo aplicado así como también de acuerdo al tipo de procesamiento cerebral que se realiza. Así pues, los PE se dividen en dos grandes grupos: exógenos y endógenos. Los PE exógenos se generan debido a estímulos externos aplicados sobre alguno de los sentidos del sujeto. Estos estímulos puede ser de tipo visual, auditivo, somatosensorial, gustativo y vestibular. Por otra parte, el grupo de los PE endógenos está relacionado con los procesos cognitivos. En general, los PE son importantes herramientas de diagnóstico que permiten determinar el estado de las vías sensitivas estimuladas. No obstante, para poder evaluar y medir correctamente la integridad de dichas vías es necesario realizar cientos de veces el estímulo correspondiente y finalmente promediar los resultados obtenidos.

2.3. Clasificación de señales EEG en aplicaciones BCI

La clasificación de las señales EEG resulta esencial para alcanzar el objetivo principal de los sistemas BCI. La precisión de los clasificadores depende en gran medida de las etapas previas. Por ejemplo, resulta crucial elegir el método adecuado para extraer y seleccionar las características que describan mejor a la señal. Si las características seleccionadas de la señal EEG original no son las más representativas, el algoritmo de clasificación no alcanzará el mejor rendimiento ni permitirá identificar de forma precisa a los patrones presentes en dichas características [124].

En las últimas décadas han sido propuestos e implementados diversos clasificadores para señales EEG [125]. Actualmente, la gran mayoría de los sistemas BCI que utilizan señales EEG implementan clasificadores no lineales basados en algoritmos de aprendizaje automático [30]. No obstante, los algoritmos de clasificación lineales son menos costosos computacionalmente. El criterio fundamental de este grupo

de algoritmos está basado en que los datos pueden ser separados de forma lineal mediante un proceso que permite dividir el espacio utilizando hiperplanos de separación. Esto genera distintas regiones que corresponden a las diferentes clases o grupos de datos. En contrapartida, los algoritmos no lineales son más costosos computacionalmente debido a la gran cantidad de parámetros que deben ser manejados. Así, en [126] se indica que los métodos lineales de clasificación alcanzan buenos resultados cuando se manejan pocos datos. Por el contrario, los métodos no lineales tienen mejor rendimiento cuando hay que procesar grandes cantidades de datos. Algunos ejemplos de clasificadores utilizados para el reconocimiento de los patrones presentes en la señales EEG son las máquinas de soporte vectorial (SVM) [127], el análisis discriminante lineal (LDA) [128] y las redes neuronales artificiales [129]. De hecho, muchos trabajos utilizan SVM para clasificación en sistemas BCI [111, 130-134]. Del mismo modo, el análisis discrimimante lineal también ha sido utilizado en diferentes trabajos [71-73, 135-137]. Finalmente, las ANN también han jugado su papel como clasificadores de señales EEG mejorando en algunos casos los resultados obtenidos por otros clasificadores. Esto se pone de manifiesto en trabajos de investigación como los desarrollados en [63, 138-141].

Aprendizaje profundo

3

En este capítulo se presentan los conceptos fundamentales de una de las ramas más innovadoras del aprendizaje máquina (ML): el denominado aprendizaje profundo (DL) [142]. Las redes neuronales profundas (DNNs) están basadas en las redes neuronales artificiales clásicas (ANNs) pero incluyen una gran cantidad de capas. El aprendizaje profundo intenta de reproducir de forma básica los procesos cerebrales con el objetivo de poder realizar complejas tareas que anteriormente eran competencia exclusiva de los seres humanos.

Las neuronas artificiales representan la unidad básica de procesamiento e intentan simular el comportamiento de una neurona cerebral de forma análoga a como lo hace el cerebro. En las DNNs, grandes cantidades de neuronas artificiales se interconectan entre sí, construyendo modelos que son capaces de realizar tareas complejas como la clasificación de imágenes, la traducción automática de voz a texto, la detección y reconocimiento de objetos, la detección de anomalías, reconocimiento de emociones a partir de registros de audio o vídeo, etc. La capacidad de aprendizaje de una red neuronal tiene relación directa con el aumento del número de capas ocultas que manejan estas arquitecturas y, de esta forma, han permitido encarar y resolver complejos problemas de detección de patrones dentro de enormes conjuntos de datos. No obstante, la capacidad de generalización de una red depende en gran medida de su arquitectura. La definición inicial y su optimización es una tarea compleja, especialmente en aquellas redes con capas que contienen un número elevado de hiperparámetros. Teniendo esto en cuenta, en este capítulo se explicarán las redes neuronales convolucionales (CNNs), pues son ampliamente utilizadas en el procesamiento de imágenes y series temporales como los EEG. Además, se detallarán algunas implementaciones de CNNs que han sido utilizadas para el procesamiento de señales EEG con el fin de encontrar patrones que permitan una clasificación eficiente de dichas señales.

3.1. Reseña histórica

El origen de las redes neuronales artificiales se remonta al año 1943 con el trabajo desarrollado [10] por Warren McCulloch y Walter Pitts en el que detallan su teoría

del cálculo lógico de las redes neuronales y esbozan una primera descripción formal de una neurona básica. Además, intentan demostrar que la máquina de Turing podría ser implementada mediante una red finita de neuronas.

Posteriormente, en el año 1958, Frank Rosenblatt toma como base el trabajo de McCulloch y Pitts y concibe formalmente el concepto de perceptrón o neurona artificial [143]. Además, construye un dispositivo electrónico basado en el perceptrón capaz de resolver algunos problemas básicos siguiendo principios neurológicos. Luego, concibe la idea de construir estructuras que utilicen varias capas de neuronas que posteriormente serían conocidas como perceptrón multi-capa (MLP). No obstante, debido a limitaciones tecnológicas de su tiempo su idea no pudo implementarse.

Marvin y Papert publican en el año 1969 el libro titulado *Perceptrons* [144], en el cual demuestran matemáticamente las limitaciones del perceptrón simple. Entre ellas, se encuentra su imposibilidad para resolver problemas que involucran a la función *XOR* (OR-Exclusiva) ya que sólo pueden realizar separaciones lineales. Este inconveniente fue resuelto posteriormente mediante la utilización de funciones de activación no lineales y la adición de capas ocultas (perceptrón multi-capa). Así, mientras que la combinación de varias rectas de separación permite separar más de dos clases, las funciones de activación no lineales permiten separar clases que no son linealmente separables. No obstante, el principal problema del MLP era encontrar la forma de dotarlo de capacidad de aprendizaje autónomo. Paul Werbos [145] en el año 1975 publicó la idea básica del algoritmo de retropropagación (*Backpropagation*), el cual permitiría que los MLP aprendan ajustando los pesos sinápticos en base al error cometido por la red durante el proceso de inferencia. En el año 1986, el aporte de Rumelhart, Hinton y Williams [146] fue sustancial para permitir que las redes neuronales adquieran capacidad de aprendizaje con la implementación del algoritmo de retropropagación que tiene por objetivo propagar los errores de las neuronas de salida hacia las anteriores capas de neuronas.

En el año 1989 se produce otro de los grande hitos del aprendizaje automático al definirse las CNNs en el trabajo de Yann LeCun [147]. Estas redes neuronales incluyen capas que realizan la operación de convolución entre la entrada y un *kernel* que se ajuste durante el proceso de aprendizaje. La aplicación de las CNNs ha tenido un profundo impacto para el procesamiento de imágenes en aplicaciones de visión por computador, mejorando considerablemente la capacidad de clasificación que proporcionaban sistemas anteriores basados en clasificadores estadísticos. Además, éstas han ido evolucionando para adaptarse a aplicaciones de diferentes áreas de interés, lo que ha permitido mejorar considerablemente los resultados proporcionados por los clasificadores basados en aprendizaje estadístico.

Entre los hitos importantes del aprendizaje profundo para el procesamiento y clasificación de series temporales está el trabajo de Hochreiter y Schmidhuber [148]. Este trabajo propone una nueva arquitectura denominada LSTM que permite a las ANNs almacenar valores a corto y largo plazo. Además, las LSTMs tienen la posibilidad de realizar conexiones entre las capas anteriores. La nueva arquitectura propuesta permitió alcanzar buenos resultados en el procesamiento de datos basados en series temporales.

Por otro lado, el problema del entrenamiento de redes profundas con un número de capas ocultas muy elevado fue abordado en 2006 por Hinton et al. [149]. En él, se presentan las redes de creencia profunda (DBNs) y un método para entrenar correctamente redes que cuenten con cientos o miles de capas. No obstante, la principal desventaja de este tipo de redes radica en que no consideran la estructura dimensional de los datos de entrada. Específicamente, esta situación afecta negativamente a problemas relacionados con tareas de visión por computador, lo cual dificulta la aplicación de las DBNs a este tipo de problemas.

Otro hito importante en el desarrollo de los métodos de DL son los modelos orientados a la generación de datos (modelos generativos). En el año 2014, Ian Goodfellow publica su trabajo *Redes Generativas Adversarias* (GANs) [150], en el que propone entrenar de forma simultánea dos ANNs que se encuentren en competencia, es decir, enfrentadas en un constante juego de suma cero. La primera ANN, llamada *generador*, se encarga de generar nuevas muestras de datos a partir de un conjunto real de datos de entrenamiento. La segunda ANN, denominada *discriminador*, recibe las muestras producidas por el generador para analizar y distinguir si dicha muestra es real o falsa.

En la Figura 3.1 se observa la línea temporal que muestra la evolución de las ANNs a lo largo de los años y los principales hitos alcanzados hasta que se materializa el aprendizaje profundo.

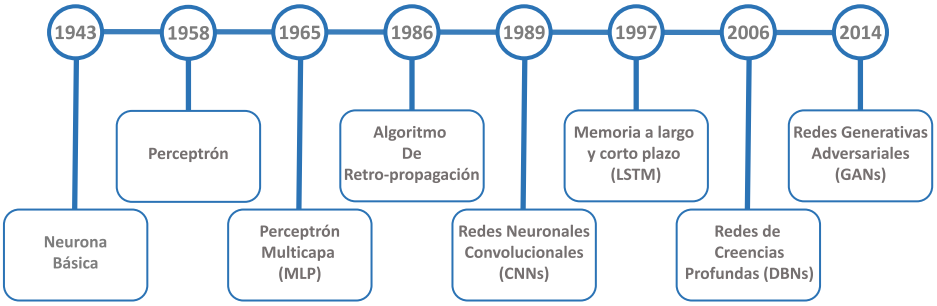


Figura 3.1.: Aprendizaje profundo. Línea temporal en la que se detallan algunos de los principales hitos que se han producido a lo largo de los años.

3.2. Neurona artificial

La neurona artificial está compuesta por entradas, pesos sinápticos, sesgo (*bias*) y una función de agregación que calcula el valor de salida. En la Ecuación 3.1 se detalla la función lineal matemática que expresa la salida z de la neurona:

$$z = \sum_{i=1}^{N} w_i \cdot x_i + b_i \tag{3.1}$$

donde x_i corresponde a los valores de entrada, w_i a los pesos relativos de los valores de entrada, b_i al sesgo y N al número de entradas. En la Figura 3.2 se observa la estructura de una neurona artificial y la función de activación, la cual es un elemento fundamental para la salida de la neurona tal y como se verá en la siguiente sección:

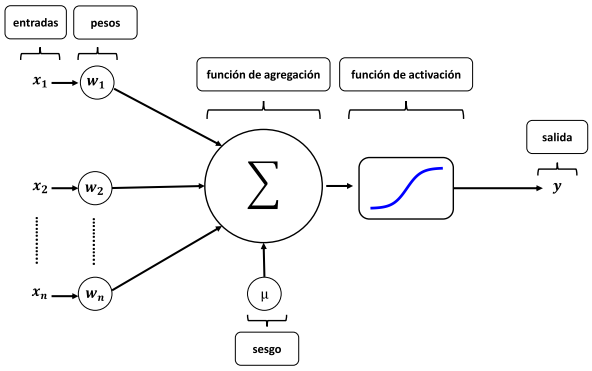


Figura 3.2.: Neurona artificial. Esquema de sus principales componentes.

3.3. Funciones de activación

Las funciones de activación son funciones generalmente no lineales que se aplican a las entradas de una neurona:

$$y = f(z) \tag{3.2}$$

Estas funciones juegan un papel muy importante en el proceso de aprendizaje puesto que determinan y limitan el rango de la salida. Las más utilizadas son:

- **Tangente hiperbólica (tanh)**. Actúa en el rango $(-1, 1)$, lo que permite un eficiente entrenamiento. Sin embargo, en el proceso de propagación hacia atrás tiene el inconveniente del desvanecimiento del gradiente, por lo que limita el ajuste del valor de los pesos. La función se define como:

$$f(z) = \frac{2}{1 + e^{-2z}} - 1 \tag{3.3}$$

- **Unidad rectificadora lineal (ReLU)**. Es una de las más utilizadas para mejorar el proceso de convergencia en redes profundas [151]. Si $z \leq 0$, entonces $z = 0$. En caso contrario, su valor no cambia. La función se define como:

$$f(z) = max(0, z) \tag{3.4}$$

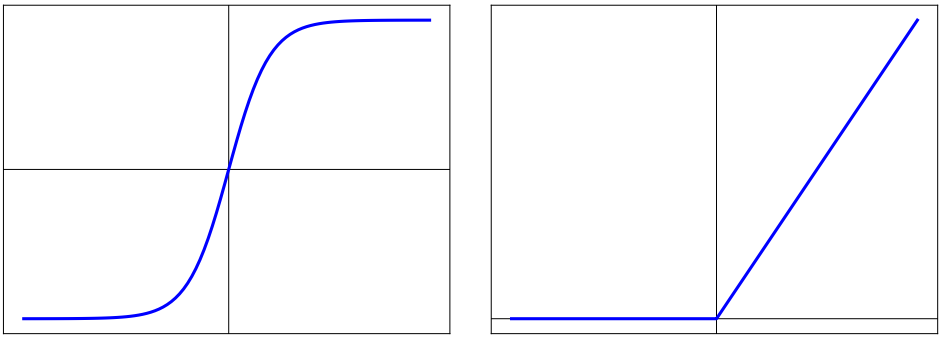
- **Leaky ReLU (LReLU)**. Similar a la unidad rectificadora lineal (ReLU). Sin embargo, se diferencian en que si $z \leq 0$ entonces z se multiplica por un coeficiente α que determina la pendiente para valores negativos de z. La función se define de la siguiente forma:

$$f(z) = \begin{cases} z & \text{si } z > 0 \\ \alpha \cdot z & \text{si } z \leq 0 \end{cases} \tag{3.5}$$

- **Unidad exponencial lineal (ELU)**. Tiene como predecesores a la ReLU y LReLU. La ventaja de esta función es que permite disminuir el problema del efecto del desvanecimiento del gradiente utilizado la operación exponencial e^z. Si el valor de entrada z es menor o igual a 0 entonces $(e^z - 1)$ se multiplica por el coeficiente α. La función se define como:

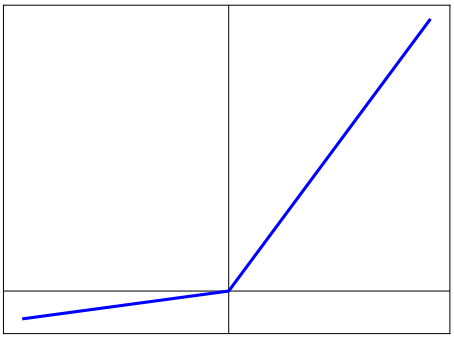
$$f(z) = \begin{cases} z & \text{si } z > 0 \\ \alpha \cdot (e^z - 1) & \text{si } z \leq 0 \end{cases} \tag{3.6}$$

Es importante resaltar que las funciones de activación con tramos lineales como ReLU, Leaky ReLU o ELU son de especial utilidad en redes con múltiples capas para evitar el problema del desvanecimiento de gradientes. Este es un efecto que se produce con funciones de activación clásicas como la tangente hiperbólica o la sigmoide, que están acotadas y pueden provocar el bloqueo del aprendizaje de la red si los gradientes toman valores muy pequeños [152]. En la Figura 3.3 se ilustran las funciones de activación mencionadas anteriormente:

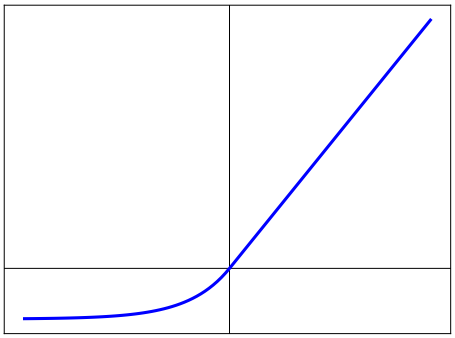


(a) Tanh

(b) ReLU



(c) Leaky ReLU



(d) ELU

Figura 3.3.: Funciones de activación más utilizadas.

3.4. Redes neuronales artificiales

Las neuronas son el elemento básico sobre el cual se construyen las ANNs. Su función es la de procesar las entradas y transmitir el resultado obtenido a una o más neuronas. La arquitectura de las ANNs está inspirada en el funcionamiento del cerebro humano, aunque su implementación difiere de los procesos biológicos llevados a cabo en él. La principal virtud de las ANNs radica en su capacidad de aprendizaje autónomo, lo que les permite realizar tareas especificas. Este proceso de aprendizaje se conoce como entrenamiento de la ANN y sirve para ajustar los pesos sinápticos. Con ello, se puede extraer información y reconocer patrones que en principio se encuentran ocultos dentro del conjunto de datos de entrada. En términos más formales, el entrenamiento de la red neuronal puede describirse como un proceso iterativo a través del cual se van ajustando los pesos de las neuronas hasta cumplir con los criterios previamente establecidos.

La combinación de las entradas a una neurona puede expresarse en notación vectorial a través de los correspondientes pesos sinápticos W tal y como se muestra en Ecuación (3.7):

$$f(x) = \sigma(W \cdot x + b) \tag{3.7}$$

donde σ es la función de activación para la capa en cuestión, W la matriz de pesos y b el vector de sesgos. Las ANNs se pueden definir como modelos matemáticos relativamente sencillos que describen una función continua $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ como resultado de la composición de funciones (capas). m representa el tamaño de entrada y n corresponde a la cantidad de neuronas de la capa. La ANN f tiene relación de dependencia con los parámetros w y b, los cuales son usualmente definidos como θ. Además, las capas sucesivas que componen la red son definidas por f^L, donde el número de capa se encuentra denotado por el superíndice L tal y como se indica en la siguiente ecuación.

$$f^L(a^{L-1}, \theta^L) = f^L(a^{L-1}, w^L, b^L) = \sigma(W^L \cdot a^{L-1} + b^L) \tag{3.8}$$

donde a^{L-1} es el vector de salidas de la capa anterior, W^L la matriz de pesos en la capa L, y b^L el vector de sesgos también en la capa L.

El objetivo de las ANNs consiste en determinar las relaciones existentes entre los valores de entrada y salida de la red mediante un proceso iterativo que finaliza cuando se cumplen los criterios establecidos previamente. En primer lugar se lleva a cabo la propagación hacia delante mediante la cual se obtienen los resultados de la red. A continuación, se ajustan estos resultados mediante el algoritmo de propagación hacia atrás teniendo en cuenta el error de la red.

A lo largo de los años, se han propuesto diferentes tipos de topologías para las redes neuronales. La principal diferencia entre ellas está en las operaciones llevadas a cabo por las capas, la arquitectura de interconexión entre éstas y el número de capas ocultas. Se pueden distinguir tres tipos básicos de arquitecturas en función de la profundidad de la red:

- **Redes mono-capa**. Es la forma más básica de una ANN pues se encuentra compuesta por una única capa (Figura 3.4). Las neuronas de la capa de entrada reciben la señal y la transfiere a la capa de salida. Cada una de las neuronas perteneciente a dicha capa obtiene la salida correspondiente realizando la suma de todas sus entradas y multiplicando por el peso asignado.
- **Redes multi-capa**. Existen dos capas visibles desde el exterior: la capa de entrada y la de salida (Figura 3.5). Sin embargo, se incorpora una capa intermedia (oculta) que no es visible desde el exterior. Esta red neuronal

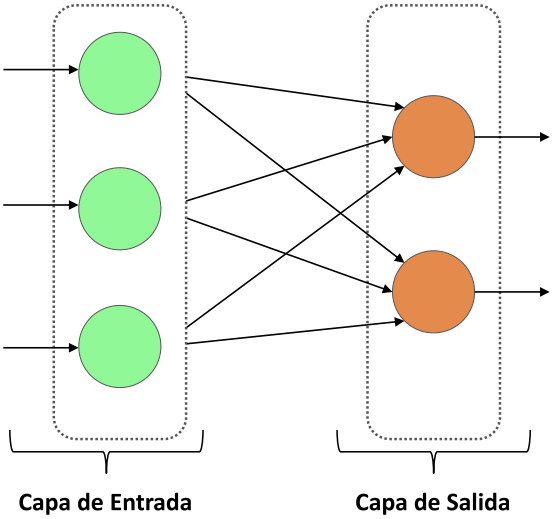


Figura 3.4.: Red mono-capa. Arquitectura de red formada por una sola capa.

requiere de eficientes algoritmos como el de retropropagación para ajustar los parámetros internos. Formalmente, con la inclusión de la capa oculta la red neuronal se convierte en un aproximador universal para cualquier función continua y diferenciable. En otras palabras, confiere a las ANNs la capacidad de aprendizaje.

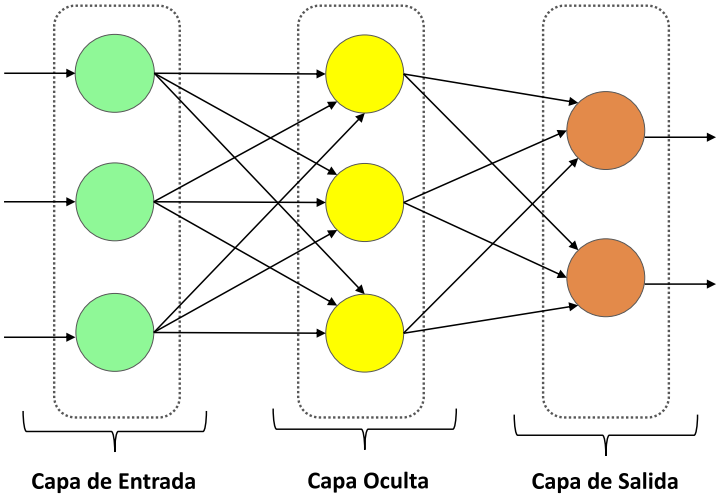


Figura 3.5.: Red multi-capa. Arquitectura de red formada por una capa de entrada, dos capas ocultas y una capa de salida.

- **Redes profundas**. Son redes multi-capa con la particularidad de tener una gran cantidad de capas ocultas (Figura 3.6). Con ello se pueden construir modelos complejos que permiten resolver diversos tipos de problemas ya que

durante el proceso de aprendizaje se establecen relaciones no lineales de gran complejidad.

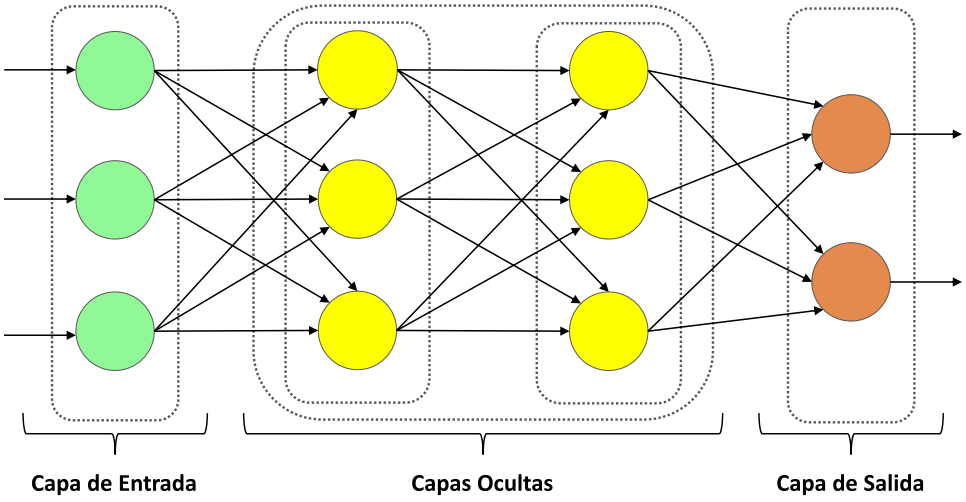


Figura 3.6.: Red profunda. Arquitectura de red formada por una capa de entrada, múltiples capas ocultas y una de salida.

Existen varios paradigmas de aprendizaje para ajustar los pesos de la red diferenciados por el uso o no de información a priori de los datos utilizados durante el entrenamiento:

- **Aprendizaje supervisado**. Se implementa un proceso que permite ajustar iterativamente los pesos de las neuronas hasta que la salida de la ANN coincida con la salida esperada. Si la salida esperada es un valor categórico entonces éste será un modelo de clasificación. Por el contrario, si la salida esperada es un valor continuo entonces se considera un modelo de regresión.
- **Aprendizaje no supervisado**. En este caso no se utiliza la salida esperada durante el proceso de entrenamiento, sino que la ANN trata de adquirir conocimiento acerca de la estructura de los datos para poder agruparlos de acuerdo a la similitud que exista entre ellos.
- **Aprendizaje por refuerzo**. Aquí, en lugar de utilizar una etiqueta o salida esperada como en el caso del aprendizaje supervisado, debe ser la red la que aprenda por sí misma cuál es la salida correcta ante una determinada entrada. Para ello, se utiliza un esquema de recompensa proporcionado por un sistema externo. De esta forma, el aprendizaje de la red tratará de maximizar la recompensa obtenida para cada entrada.

Diseñar una ANN eficiente continúa representando un gran desafío en la actualidad ya que requiere tomar una serie de decisiones como el tipo de capas a utilizar, el número de ellas, la cantidad de neuronas en cada una de las capas, el tipo de conexión entre capas o el algoritmo de aprendizaje, entre otros.

3.5. Aprendizaje supervisado en redes neuronales

En la literatura científica relacionada con el aprendizaje supervisado el uso del algoritmo de propagación hacia atrás (*backpropagation*) se encuentra bastante extendido hasta tal punto que se le considera como el enfoque estándar para este fin. Básicamente, este algoritmo utiliza la técnica del descenso del gradiente, que tiene por objetivo localizar el mínimo global de una función mediante un proceso iterativo de movimiento en dirección contraria al gradiente (pendiente) de la función en el punto actual.

El principal objetivo del proceso de aprendizaje de las ANNs es minimizar una función de coste cuya evaluación proporciona una métrica del error. El proceso de optimización, por tanto, deberá minimizar la suma de los errores en todo el conjunto de datos de entrenamiento. El proceso iterativo de retropropagación permite calcular el gradiente de la función de coste en relación al peso y el sesgo de cada una de las neuronas y, posteriormente, ajustar los sesgos y pesos en dirección contraria al gradiente con el fin de encontrar el mínimo local. El algoritmo de propagación hacia atrás puede expresare matemáticamente de la siguiente forma. Siendo $E(\omega) = \frac{1}{2} \cdot |y - f(x, \omega)|$ el error cometido por la red durante el proceso de inferencia, los pesos sinápticos ω' se actualizan de la forma:

$$\omega' = \alpha_0 \cdot w + \eta \cdot (-\nabla E(w)) \tag{3.9}$$

donde $\nabla E(\omega)$ es el gradiente del error (función de coste con respecto al peso ω).

3.5.1. Funciones de coste

Durante el proceso de entrenamiento de las ANNs es necesario cuantificar el desempeño que alcanza la red en cada iteración, es decir, realizar una medición del ajuste del modelo al conjunto de datos de entrenamiento. La función encargada de realizar

esta medición tiene varias denominaciones: función de coste, función objetivo o función de perdida. No obstante, la tarea de asignada a esta función es única y consiste en comprobar la diferencia existente entre la salida calculada y la esperada, con lo cual es posible obtener una medida acerca del ajuste de los parámetros del modelo en relación al conjunto de datos de entrenamiento. Existen diversos tipos de funciones de coste que pueden ser utilizadas. Sin embargo, la elección de la función de coste a utilizar por la ANN depende en gran medida del problema de interés que deba ser resuelto. A continuación se definen algunas de las funciones de coste más utilizadas en tareas de regresión:

- **Error cuadrático medio (MSE)**. Calcula la suma de las diferencias al cuadrado existente entre los valores esperados y los valores calculados por la red:

$$MSE = \frac{1}{M} \cdot \sum_{i=1}^{M} (y_i - \hat{y}_i)^2 \tag{3.10}$$

  donde $\hat{y}_i$ representa al valor de salida esperado para una entrada especifica, y_i es el valor calculado por la red para la misma entrada y M corresponde a la cantidad de entradas.

- **Error absoluto medio (MAE)**. Esta función calcula la suma de las diferencias absolutas entre los valores esperados y los valores calculados por la red. El resultado es la magnitud promedio de errores de los valores esperados sin considerar sus direcciones:

$$MAE = \frac{1}{M} \cdot \sum_{i=1}^{M} |y_i - \hat{y}_i| \tag{3.11}$$

- **Entropía cruzada (CE)**. Habitualmente se utiliza para medir el desempeño del modelo en problemas de clasificación. El resultado de esta función varía entre 0 y 1 y puede interpretarse como una medida de similitud entre la distribución de la probabilidad a posteriori $P(C|x_i)$ y la distribución real de los datos (*ground truth*). La distribución P denota la probabilidad de que la muestra de entrada x_i sea clasificada como perteneciente a la clase C. Por tanto, un valor bajo indica que ambas distribuciones son similares, mientras que un valor cercano a 1 indica que son diferentes. Existen dos variantes de la función de entropía cruzada. Por un lado, la entropía cruzada binaria (BCE):

$$BCE = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)) \tag{3.12}$$

siendo p la probabilidad para el valor calculado e y el indicador binario. Por otro lado, existe la entropía cruzada categórica (CCE), que normalmente se utiliza en problemas de clasificación multi-clase:

$$CCE = -\sum_{C=1}^{M} y_{o,C} \cdot \log(p_{o,C}) \tag{3.13}$$

siendo $p_{o,C}$ la probabilidad del valor de salida del modelo para la clase C y la entrada o. Si la clase C del valor de salida es igual al valor esperado o, entonces $y_{0,C} = 1$.

No obstante, existen variantes de estas funciones de coste que pueden ser convenientes de utilizar en determinados problemas dependiendo de la naturaleza, normalización y escalado de los datos de entrada [153].

3.5.2. Algoritmos de optimización

Los algoritmos de optimización se encargan de ajustar dinámicamente los pesos y sesgos de las neuronas que la componen para minimizar la función de coste con respecto a la salida esperada. En este contexto, los algoritmos de optimización tienen una vital importancia ya que se encargan de reducir las pérdidas durante el proceso de entrenamiento de cara a alcanzar precisión en los resultados. A continuación se definen algunos de los algoritmos de optimización más populares:

- **Descenso del gradiente (GD)**. La finalidad de este algoritmo es encontrar los parámetros (pesos y sesgos) de una ANN que minimicen el error calculado por la función de coste $f(x)$. El proceso del descenso se inicia en un punto aleatorio de la función para luego ir desplazándose en la dirección negativa del gradiente $-\nabla f(x)$. Es decir, siguiendo la dirección donde el declive es mayor hasta llegar al valor mínimo de la función. En cada iteración, los parámetros se actualizan haciendo que el valor de la función de coste decrezca progresivamente. El algoritmo de descenso del gradiente se define formalmente como:

$$\arg\min_{x} f(x) \tag{3.14}$$

donde $f : D \subset \mathbb{R}^n \to \mathbb{R}$ es una función diferenciable en D y $x_0 \in D$. Por definición, el gradiente $\nabla f(x)$ define la dirección de máximo crecimiento de la función $f(x)$. De forma contraria, el gradiente negativo $-\nabla f(x)$ define la

dirección hacia donde la función $f(x)$ decrece. Por lo tanto, para algún $\varepsilon > 0$ se tiene que:

$$f(x - \nabla f(x)) < f(x) \tag{3.15}$$

A continuación, el algoritmo selecciona de forma aleatoria el punto inicial x_0 que se encuentra dentro del dominio de f. Posteriormente, en cada iteración se seleccionará otro punto x_{n+1} tal que:

$$x_{n+1} = x_n - \gamma_n \nabla f(x_n) \tag{3.16}$$

donde γ_n es la tasa de aprendizaje que representa al avance que se produce en cada iteración. Si x_n converge lo hará en un mínimo local. Sin embargo, si $f(x)$ es una función convexa y $\nabla f(x)$ es una función (con derivada acotada), entonces converge a un mínimo global (ver Figura 3.7).

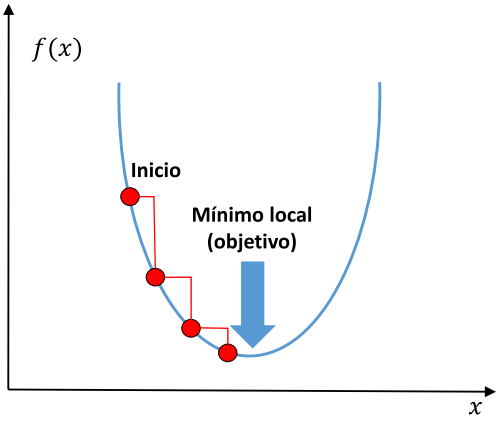


Figura 3.7.: Algoritmo de descenso de gradiente. Convergencia a un mínimo local.

- **Descenso estocástico del gradiente (SGD)**. El termino estocástico se debe a la forma aleatoria en que se selecciona cada ejemplo del conjunto o se mezcla el conjunto antes de que se inicie el entrenamiento. En líneas generales resulta recomendable la utilización del SGD cuando la función de coste que se trata de minimizar pueda ser descompuesta como el promedio de los valores de coste más pequeños. El algoritmo de descenso estocástico del gradiente se define de la siguiente forma:

$$\arg\min_x f(x) = \arg\min_x \left(\frac{1}{n} \cdot \sum_{i=1}^{n} f_i(x) \right) \tag{3.17}$$

el gradiente de f esta definido por:

$$\nabla f(x) = \frac{1}{n} \cdot \sum_{i=1}^{n} \nabla f_i(x) \tag{3.18}$$

Puesto que el gradiente corresponde a un promedio, es posible reemplazar el proceso de cálculo de todos los n costos por la utilización de un subconjunto $x < n$ del conjunto total. Este subconjunto se selecciona por muestreo simple estocástico (de k-elementos de los n posible costos que mediante la ley de los grandes números se obtiene:

$$\frac{1}{k} \cdot \sum_{i=1}^{k} \nabla f_i(x) \approx \sum_{i=1}^{n} \nabla f_i(x) = \mathbb{E}(\nabla f_i(x)) \tag{3.19}$$

De forma análoga al GD se parte del punto inicial aleatorio x_0. Posteriormente, en cada iteración se selecciona otro punto punto x_{n+1} tal que:

$$x_{n+1} = x_n - \gamma_n \cdot \frac{1}{k} \cdot \sum_{i=1}^{n} \nabla f_j(x_n) \tag{3.20}$$

A continuación se definen otros algoritmos de optimización basados en el SGD y que permiten acelerar la convergencia de la red mediante el ajuste dinámico de la tasa de aprendizaje:

- **Adagrad**. El algoritmo adaptativo del gradiente es una variación del SGD que utiliza una tasa de aprendizaje adaptativo de los parámetros de la red teniendo como referencia el gradiente acumulado en cada uno de ellos [154, 155]. El procedimiento indica que los parámetros frecuentes se actualizan con valores de menor magnitud, mientras que los menos frecuentes lo harán con valores de mayor magnitud. El principal inconveniente del algoritmo se produce durante el entrenamiento cuando la tasa de aprendizaje se reduce de forma muy acelerada. Esto se debe a la múltiple ocurrencia acumulada de valores de gran magnitud del gradiente cuando se inicia el proceso de entrenamiento de la red.

- **RMSProp**. El algoritmo de propagación de la raíz cuadrada es una variación del Adagrad que permite evitar el inconveniente de acumulación del gradiente durante las etapas iniciales del entrenamiento [152]. El funcionamiento del algoritmo consiste en calcular de forma recursiva la media móvil exponencial de la suma de todos los gradientes para normalizar el gradiente. De esta forma,

la tasa de aprendizaje puede variar con el tiempo evitando que se reduzca abruptamente.

- **Adam**. El algoritmo de estimación del momento adaptativo [156] es ampliamente utilizado en la actualidad debido a los importantes beneficios que conlleva su uso. La principal fortaleza radica en que integra los beneficios de los algoritmos Adagrad y RMSprop pues conserva la velocidad de aprendizaje cuando los gradientes se encuentran dispersos y ajusta los pesos en base a la media de los valores del tamaño de los gradientes. Además, es computacionalmente eficiente y bastante robusto tanto para conjuntos de datos de gran tamaño como para estructuras de redes con gran cantidad de parámetros.

3.5.3. Algoritmo de retropropagación

Como se ha comentado anteriormente, el algoritmo de *backpropagation* ajusta los pesos de las neuronas de las capas ocultas con el fin de minimizar el error cometido sobre el conjunto de datos de entrenamiento. La forma de cuantificar el error en cada iteración se hace definiendo una función de pérdida. En definitiva, el proceso de entrenamiento de las ANNs consta de dos fases bien definidas: la etapa de propagación hacia delante *(feedforward)*, donde el recorrido de los datos va desde las entradas hacia las salidas, y la etapa de retropropagación *backward*, donde el procedimiento de ajuste de pesos y sesgos consiste básicamente en propagar hacia atrás el gradiente de la función de pérdida. Este proceso se realiza de forma recursiva, es decir, partiendo de la última capa hacia la primera donde en cada capa se corrigen los pesos y sesgos correspondientes a las neuronas que la componen. El funcionamiento del algoritmo de *backpropagation* se puede explicar en tres etapas si se tiene en cuenta la inicialización de los pesos:

- **Etapa inicial**. Durante el inicio del proceso de entrenamiento. En este momento no se tienen conocimientos previos acerca de la red que será entrenada y los parámetros de la red (pesos y sesgos) se inicializan de forma aleatoria.
- **Etapa de propagación hacia adelante**. Conocida como *feedforward*. Es necesario utilizar una parte de los datos de la época n, es decir, los valores de entrada al vector representado por $x(n)$. Esta etapa se puede definir de la siguiente forma:

$$v_j^L(n) = \sum_{i=0}^{m_0} w_{j,i}^L(n) \cdot y_i^{L-1}(n) \tag{3.21}$$

donde $v_j^L(n)$ es el valor de la neurona j en la capa L, y_i^{L-1} es el valor de la neurona anterior i en la capa $(L-1)$ durante la época n. Finalmente, $w_{j,i}^L$ es el peso de la conexión existente entre la neurona j y la neurona i.

Por otro lado, para calcular el error se utiliza la siguiente función de coste:

$$error_i(n) = e_i(n) - p_j(n) \tag{3.22}$$

siendo $e_i(n)$ el i-ésimo valor esperado para la salida $e(n)$ y $p_j(n)$ el valor de salida de la neurona j.

- **Etapa de propagación hacia atrás**. Aquí, los errores de la salida de la n-ésima capa proporcionados por la función de gradiente del error se utilizan para ajustar los pesos de las neuronas de la capa procesada:

$$w_{j,i}^L(n+1) = w_{j,i}^L(n) + \alpha \cdot [w_{j,i}^L(n-1)] + \gamma \cdot \nabla_j^L(n) \cdot y_i^{L-1} \tag{3.23}$$

siendo γ la tasa de aprendizaje, $\nabla_j^L(n)$ el gradiente calculado con anterioridad y α la constante utilizada para reducir las oscilaciones causadas por la variación de los pesos de la red.

El número de iteraciones para la propagación hacia delante y hacia atrás es variable según si se ha alcanzado el mínimo o no. En cualquiera de los casos, la cantidad máxima de iteraciones se establece previamente.

3.5.4. Capacidad de generalización de los modelos

El proceso de entrenamiento de las ANNs permite extraer características y reconocer patrones presentes en un conjunto de datos. La idea es generar modelos que tengan capacidad de generalización, es decir, que sean capaces de inferir la salida correcta para datos de entrada que no han sido utilizados para la generación del modelo. Esta capacidad dota a las computadoras de autonomía para realizar de forma precisa tareas de interés. No obstante, existen varios factores que pueden interferir a la hora de alcanzar la capacidad de generalización esperada durante el proceso de entrenamiento. Un modelo generado mediante un proceso de aprendizaje puede presentar dos problemas que afectan a su capacidad de generalización:

- **Error por sesgo alto**. Es el error cometido por supuestos erróneos al generar el modelo. Un sesgo alto puede hacer que se pierdan las relaciones entre las características y las salidas objetivo. Este efecto de subajuste (*underfitting*) normalmente está provocado por tener un modelo demasiado simple (con muy pocos parámetros). Como consecuencia, no es capaz de retener la información necesaria para reconocer todos los patrones presentes en el conjunto de datos y se obtiene un error alto tanto con las muestras de entrenamiento como con las de test (aquellas que no se han utilizado para generar el modelo). También, reducir el sesgo normalmente implica la necesidad de disponer de un conjunto de entrenamiento más representativo de la población que se está modelando, cambiar el modelo o aumentar el número de parámetros del mismo.
- **Error por varianza alta**. Es el error producido por la sensibilidad a pequeñas fluctuaciones en el conjunto de entrenamiento. Una varianza alta puede hacer que el algoritmo modele el ruido aleatorio de los datos. Es decir, el modelo está sobreajustado (*overfitting*) [157]. El error por sobreajuste se produce cuando el modelo generado se ajusta excesivamente a los datos de entrenamiento, perdiendo capacidad de generalización. En otras palabras, mientras que el error por sesgo se produce cuando el modelo es demasiado simple, el error por alta varianza se produce cuando el modelo es demasiado complejo (demasiados parámetros).

Las redes neuronales profundas son especialmente propensas al sobreajuste por la gran cantidad de parámetros que contienen. No obstante, el sobreajuste de la ANN puede deberse al hecho de contar con pequeña cantidad de ejemplos en el conjunto de datos de entrenamiento en proporción a la dimensionalidad de los datos de entrada [158-160] (maldición de la dimensionalidad). Conseguir un balance sesgo-varianza y prevenir en la medida de lo posible el sobreajuste resulta de vital importancia para optimizar la arquitectura de las redes profundas. En la actualidad, existen diversos tipos de propuestas para mitigar el posible sobreajuste que pueden sufrir las redes. Algunos de los más utilizados son:

- **Regularización**. Consiste en penalizar los pesos de la red durante el entrenamiento al mismo tiempo que el algoritmo de retropropagación presiona a la red para generar la salida correcta. Existen varios métodos de regularización, entre los cuales se encuentra *Dropout* [161]. Este método trata de dificultar el aprendizaje durante el entrenamiento desactivando de forma aleatoria una cierta cantidad de neuronas con el propósito de evitar la sobre-adaptación de las neuronas. La técnica de regularización también puede ser aplicada sobre

los filtros introduciendo los términos de regularización ℓ_1 y ℓ_2 para penalizar los pesos grandes o activaciones:

$$\ell_1 = error(y - \hat{y}) + \lambda \cdot \sum_{i=1}^{N} |w_i| \tag{3.24}$$

$$\ell_2 = error(y - \hat{y}) + \lambda \cdot \sum_{i=1}^{N} \|w_i\|^2 \tag{3.25}$$

donde y es el dato esperado, $\hat{y}$ el valor calculado por el modelo, λ el parámetro de regularización y w_i el i-ésimo peso.

- **Parada anticipada**. Este método supone que la red alcanza un ajuste óptimo en alguna etapa del entrenamiento [162]. Sin embargo, si el algoritmo de entrenamiento prosigue con las iteraciones producirá el sobreajuste. Por lo tanto, es necesario que durante el proceso de entrenamiento los resultados sean validados con un conjunto de datos diferente al conjunto de entrenamiento (conjunto de validación) que deberá contener una distribución muestral similar a la usada para el entrenamiento. De esta forma, el algoritmo se detendrá cuando la capacidad de generalización comience a deteriorarse.
- **Aumento de datos (*data augmentation*)**. Se trata de generar muestras adicionales de forma artificial modificando de alguna forma las muestras originales. Este método es muy utilizado en las redes con capas convolucionales donde los procesos de aumento de datos suelen utilizar operaciones de escalado, rotación, emborronado o adición de ruido. No obstante, mientras que los métodos de aumento de datos son relativamente sencillos cuando los datos de entrada son imágenes, no son tan evidentes con otro tipo de datos como las series temporales.

3.5.5. Métricas de evaluación de rendimiento

Los modelos generados por las ANNs son representaciones simplificadas de la realidad que con cierto grado de precisión tienen la capacidad de realizar las tareas para las cuales han sido diseñadas. Por tanto, resulta imprescindible la utilización de métricas que permitan evaluar la eficiencia de los modelos. Las tareas de clasificación pueden ser binarias (tratan de predecir entre dos clases), o multi-clase cuando el conjunto de datos contiene muestras pertenecientes a tres o más categorías.

La matriz de confusión es una herramienta ampliamente utilizada para la evaluación de clasificadores. Dicha matriz establece la relación entre las predicciones realizadas por el modelo y los resultados esperados. Es una matriz de $m \times n$, con m filas y n columnas donde las filas corresponden a la cantidad de instancias de cada clase y las columnas corresponden a la cantidad de predicciones de cada clase. Cada posición i, j indica la cantidad de ocurrencias entre la clase predicha y la clase real (*ground truth*). Para la construcción de la matriz de confusión se tiene en cuenta que las predicciones realizadas por un clasificador pueden ser positivas (P) o negativas (N). Serán positivas cuando el clasificador establezca que la predicción realizada pertenece a una clase determinada y serán negativas cuando el clasificador establezca que la predicción realizada no pertenece esa clase. En función de la salida de un clasificador con respecto a la salida esperada existen cuatro tipos de predicciones:

- **Verdaderos positivos (TP)**. Corresponde al número de predicciones correctas que ha realizado el modelo y que pertenecen a la clase positiva.
- **Verdaderos negativos (TN)**. Corresponde al número de predicciones correctas que ha realizado el modelo y que pertenecen a la clase negativa.
- **Falsos positivos (FP)**. Corresponde al número de predicciones incorrectas que el modelo sugiere que pertenecen a la clase positiva pero que en realidad deberían pertenecer a la clase negativa. En estadística, la presencia de falsos positivos se denomina error de tipo I, indicando que se ha encontrado una diferencia inexistente con la realidad [163].
- **Falsos negativos (FN)**. Corresponde al número de predicciones incorrectas que el modelo sugiere que pertenecen a la clase negativa pero que en realidad deberían pertenecer a la clase positiva. La presencia de falsos negativos se denomina error de tipo II, indicando que no ha sido posible encontrar una diferencia existente con la realidad [163].

En la Tabla 3.1 se observa una matriz de confusión binaria que muestra los aciertos y errores en la tarea de predicción de un clasificador. A continuación, se detallan algunas de los principales métricas basadas en la matriz de confusión y son utilizadas habitualmente en tareas de clasificación:

- **Exactitud (*accuracy*)**. Representa el porcentaje total de predicciones que el modelo ha realizado correctamente. En términos formales se define como el cociente entre los casos que han sido correctamente clasificados, es decir, los verdaderos positivos y negativos sobre el total de casos. Esta métrica es

Tabla 3.1.: Matriz de confusión.

		Predicción	
		Negativo	Positivo
Clase Real	Positivo	Verdadero Positivo (TP)	Falso Positivo (FP)
	Negativo	Falso Negativo (FN)	Verdadero Positivo (TP)

significante cuando las clases se encuentran balanceadas. Sin embargo, cuando existe un desbalanceo de clases los resultados que provee esta métrica podrían no ser significantes dado que un valor alto puede deberse a la detección de positivos o negativos únicamente. La métrica de exactitud se define como:

$$exactitud = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.26}$$

- **Precisión**. Representa el porcentaje de predicciones positivas que el modelo ha realizado correctamente. En términos formales se define como el cociente entre las predicciones positivas verdaderas sobre el total de los positivos (verdaderos y negativos):

$$precision = \frac{TP}{TP + FP} \tag{3.27}$$

- **Sensibilidad (*recall*)**. También conocida como tasa de verdaderos positivos (TPR). Representa el porcentaje de predicciones positivas correctas con respecto al total de positivos del conjunto de datos. En términos formales se define como el cociente entre las predicciones positivas correctas sobre el total real de casos positivos del conjunto de datos:

$$TPR = \frac{TP}{TP + FN} \tag{3.28}$$

- **Especificidad**. También conocida como tasa negativa real de positivos (TNR). Representa el porcentaje de predicciones negativas verdaderas con respecto al total de negativos del conjunto de datos. En términos formales, se define como el cociente entre las predicciones negativas correctas sobre el total real de casos negativos del conjunto de datos:

$$TNR = \frac{TN}{TN + FP} \tag{3.29}$$

- **Tasa de falsos positivos (FPR)**. Porcentaje de casos negativos que fueron equivocadamente clasificados como positivos con respecto al total de negativos del conjunto de datos. En términos formales se define como el cociente entre las predicciones positivas incorrectas sobre el total real de casos negativos del conjunto de datos:

$$FPR = \frac{FP}{TN + FP} \tag{3.30}$$

- **Curva ROC** (*Receiver Operating Characteristic*). Representa la sensibilidad a la especificidad de un clasificador binario para diferentes valores del umbral de discriminación. Son especialmente útiles para mostrar el balance entre los verdaderos positivos y los falsos negativos y permiten ajustar el punto de operación (umbral de discriminación) para que el clasificador proporcione los niveles de sensibilidad y especificidad requeridos en una tarea concreta. Asociado a esta curva se define el área bajo la curva (AUC), que mide el grado de solapamiento entre las distribuciones de verdaderos negativos y verdaderos positivos. Un ejemplo de curva ROC puede verse en la Figura 3.8.

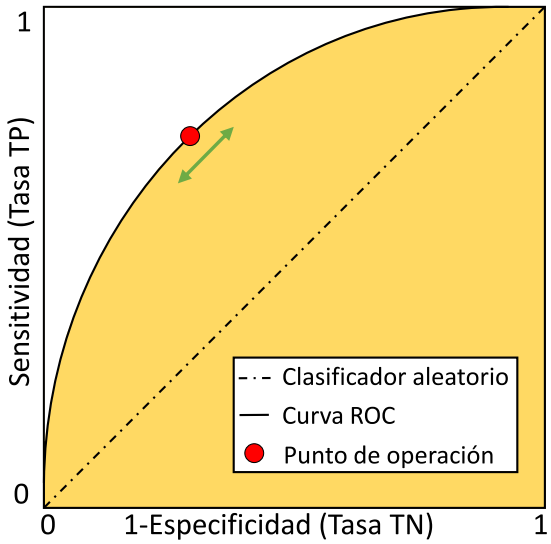


Figura 3.8.: Curva ROC. Ejemplo e interpretación en términos de los valores de sensibilidad y especificidad obtenidos al variar el punto de operación.

Aunque la matriz de confusión permite la evaluación de clasificadores multi-clase, las métricas anteriores se suelen utilizar para la evaluación de clasificadores binarios. En cambio, para la evaluación de clasificadores multi-clase suelen utilizarse las siguientes métricas:

- **Valor F1 *score***. Esta métrica tiene en cuenta las predicciones positivas incorrectas (falsos positivos), al igual que las predicciones negativas incorrecta (falsos negativos). En términos formales, se define como el promedio armónico entre las métricas de precisión y sensibilidad:

$$F1 = 2 \cdot \frac{sensibilidad \cdot precision}{sensibilidad + precision} \tag{3.31}$$

- **Índice Kappa**. Este índice permite calcular la medida de acuerdo o desacuerdo que existe entre dos evaluadores en la tarea clasificación que llevan a cabo [164]. Por otra parte, se considera la probabilidad de que los criterios de los examinadores coincidan por casualidad dando como resultado el valor 0. Cuando el acuerdo es perfecto el valor resultante es 1:

$$k = \frac{p_0 - p_c}{1 - p_c} \tag{3.32}$$

  siendo p_0 el acuerdo observado y p_c el acuerdo esperado. El valor del índice Kappa es siempre menor o igual a 1.

3.6. Redes neuronales convolucionales (CNN)

Son arquitecturas de aprendizaje profundo que surgen como alternativa a los métodos basados en aprendizaje estadístico para el procesamiento y clasificación de imágenes, donde han supuesto una verdadera revolución. No obstante, tienen también aplicación en el procesamiento de series temporales. La arquitectura de una CNN [165] está formada por múltiples capas de filtros convolucionales de una o más dimensiones. Estas capas serán las encargadas de extraer características relevantes de los datos de entrada a través de un proceso de aprendizaje. Tras ellas, se incluye una red de tipo perceptrón multi-capa que se encarga de la tarea de clasificación de los patrones (ver Figura 3.9).

La gran popularidad de las CNNs profundas se debe principalmente a su notable capacidad para clasificación de imágenes [166-168] superando a otros métodos que han sido utilizados tradicionalmente en este ámbito [169]. La principal ventaja que tienen las CNNs con respecto a los demás métodos es que aprenden a extraer características espaciales relevantes a partir de los datos de entrada [170] que permiten alcanzar un mejor desempeño en su posterior clasificación. Es decir, la

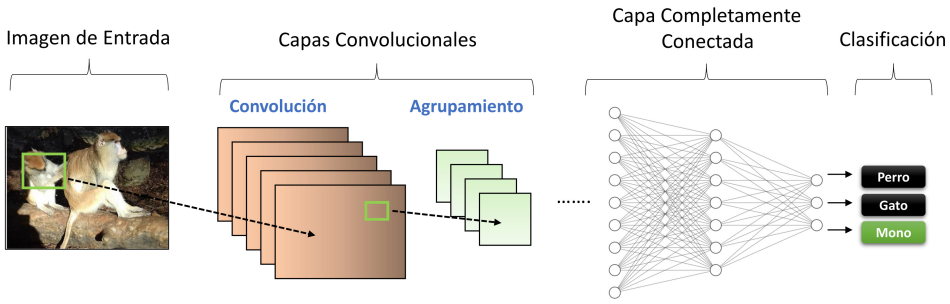


Figura 3.9.: Red neuronal convolucional. Arquitectura CNN con capas convolucionales.

arquitectura de las CNNs contempla tanto la fase de la extracción de características como la de clasificación. En contrapartida, los métodos tradicionales de clasificación requieren que los datos de entrada sean procesados previamente mediante otros métodos.

3.6.1. Redes neuronales convolucionales 2D

Las CNNs 2D son capaces de reconocer objetos y complejos patrones presentes en base de datos de visuales como las imágenes y vídeos. En la actualidad, las CNNs 2D son una herramienta ampliamente utilizada para el procesamiento de señales 2D. Algunas de sus ventajas son [171]:

- Son arquitecturas que permiten combinar en un único diseño los procesos de extracción y clasificación de características. Pueden aprender a optimizar las características seleccionadas a partir del conjunto de datos en bruto.
- La topología permite el procesamiento eficiente de grandes cantidades de datos de entrada como resultado de la baja densidad de conexiones entre las neuronas que componen las CNNs.
- Fácil adaptación a ligeras transformaciones realizadas sobre los datos de entrada. Estas transformaciones pueden incluir traslación, escalado, inclinación y distorsión de los datos.
- Facilidad para adaptarse a diferentes tamaños de entrada.

Para explicar algunos conceptos de las CNNs se considerará que la imagen desde la que se obtienen los datos corresponde a una imagen con tres canales (rojo, verde y azul). La imagen es vista por la computadora como una única matriz de píxeles de la forma alto $\times$ ancho $\times$ profundidad:

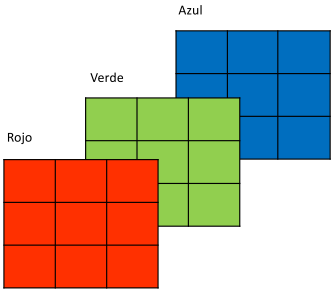


Figura 3.10.: Imagen RGB. Composición de una imagen de tres canales.

Capas convolucionales

Las capas convolucionales se encargan de extraer información sobre la relación existente entre los valores de píxeles vecinos utilizando pequeños bloques de la imagen. Este proceso se realiza mediante una operación de convolución que puede expresarse matemáticamente como una función que toma dos entradas: la imagen y el filtro (ver Figura 3.11).

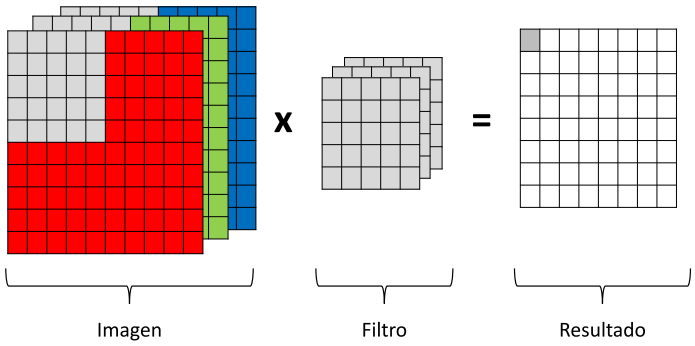


Figura 3.11.: Convolución. Estructura de una convolución realizada sobre una imagen RGB. Se puede observar la multiplicación entre la imagen y el filtro.

La operación de convolución se denota como $(*)$ y se aplica entre las funciones de entrada $f(x)$ y filtro $g(x)$, produciendo la función de salida $s(x)$:

$$s(x) = (f * g)[x] = \sum_{i=1}^{n} f(i) \cdot g[x - i] \tag{3.33}$$

donde x es una variable discreta y n el tamaño del filtro. En la Ecuación (3.34) se muestra la operación de convolución 2D sobre los datos de entrada. Esta operación se realiza desplazando una ventana del tamaño del *kernel* con un solapamiento

determinado por el parámetro de zancada o *stride*. Para que esta operación sea posible independientemente del tamaño de la ventana y desplazamiento de la misma se define un parámetro de relleno o *padding*:

$$\hat{G}_{k,l,n} = \sum_{i,j,m} \hat{K}_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m} \tag{3.34}$$

donde $\hat{G}$ las características de salida para la convolución estándar, F las características de entrada, y $\hat{K}$ el *kernel* de convolución de tamaño $m \cdot n$.

El coste computacional de las redes convolucionales está mayoritariamente ligado a la cantidad de multiplicaciones que deben realizarse durante su ejecución. En la Ecuación (3.35) se muestra la expresión para el cálculo del costo de la operación de convolución:

$$Coste = D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f \tag{3.35}$$

siendo M el número de canales de entrada, N el número de canales de salida, $D_k \cdot D_k$ el tamaño del filtro y $D_f \cdot D_f$ el tamaño del mapa de características. Por otro lado, en la Figura 3.12 es posible observar el esquema de procesamiento de las convoluciones 2D y sus correspondientes salidas:

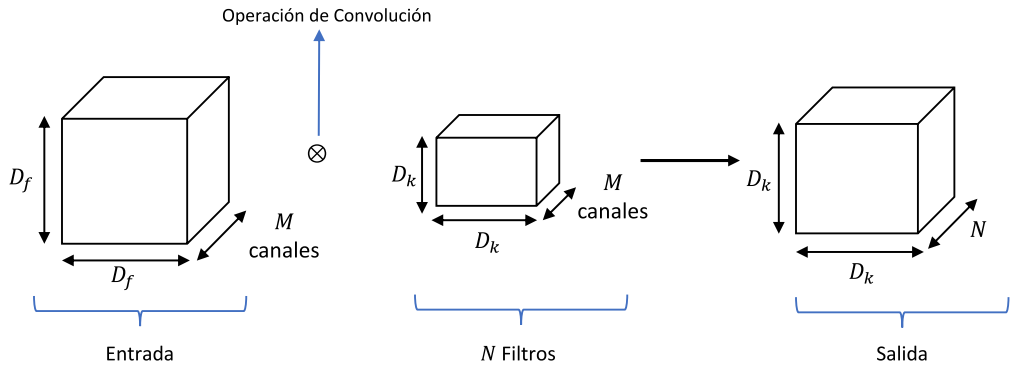


Figura 3.12.: Convolución 2D. Esquema que detalla la operación de convolución aplicada sobre los datos de entrada y su salida. Imagen adaptada de [172].

Dado que las redes convolucionales habitualmente utilizadas en problemas reales implican arquitecturas con un gran número de capas y parámetros, se han desarrollado diferentes estrategias para reducir el número de parámetros requeridos

en las operaciones de convolución y la cantidad de memoria necesaria. Estas técnicas, fundamentalmente convoluciones separables y convoluciones separables en profundidad, se explican en las secciones siguientes.

Convoluciones separables

Las convoluciones estándar pueden ser descompuestas en varias operaciones sin que esta división afecte al resultado final. De aquí surge la idea de las convoluciones separables, cuya implementación se ha descrito en *Mobilenet* [172] y *Xception* [173]. Algunas ventajas de utilizar convoluciones separables son:

- Cuentan con menor cantidad de parámetros a ajustar en comparación con las convoluciones estándar, lo que implica que existe menor posibilidad de producirse un sobreajuste.
- Tienen menor coste computacional ya que precisan menor cantidad de cálculos, lo cual la hace adecuada para arquitecturas de cómputo reducido.
- Al implicar un menor número de parámetros, consiguen una reducción importante en la cantidad de memoria necesaria para almacenar un modelo.

Algunos tipos de convolución son:

- **Convolución puntual**. Es la operación de convolución estándar. Su particularidad radica en el tamaño del filtro de convolución que utiliza ($1 \times 1 \times M$), donde M es el número de mapas de características que serán obtenidas. La profundidad del filtro está dada por la cantidad de canales de la imagen de entrada. En la Figura 3.13 es posible observar la estructura de funcionamiento de la operación de convolución de tipo puntual.

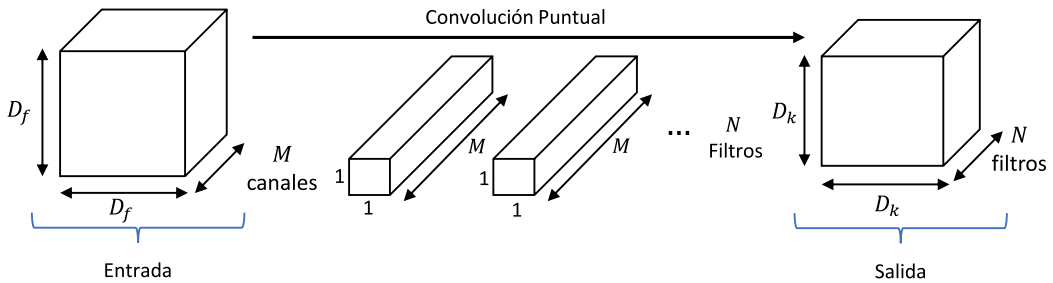


Figura 3.13.: Convolución puntual. Estructura de la operación convolución que utiliza filtros convolucionales de $1 \times 1 \times M$. Imagen adaptada de [172].

- **Convolución espacial separable**. Permite aplicar convoluciones separadas sobre cada uno de los ejes espaciales. En el caso de una imagen, sus dimensiones espaciales están dadas por el ancho y el alto de la misma. La ventaja de su utilización radica en la reducción del coste computacional debido al menor número de parámetros y cantidad de multiplicaciones de matrices que deben ejecutarse en comparación a las convoluciones tradicionales. Sin embargo, su uso no es extensible a todos los filtros convolucionales ya que existen filtros que no pueden ser descompuestos en dos núcleos de menor tamaño. En la Figura 3.14 se puede observar el esquema de funcionamiento de la convolución separable.

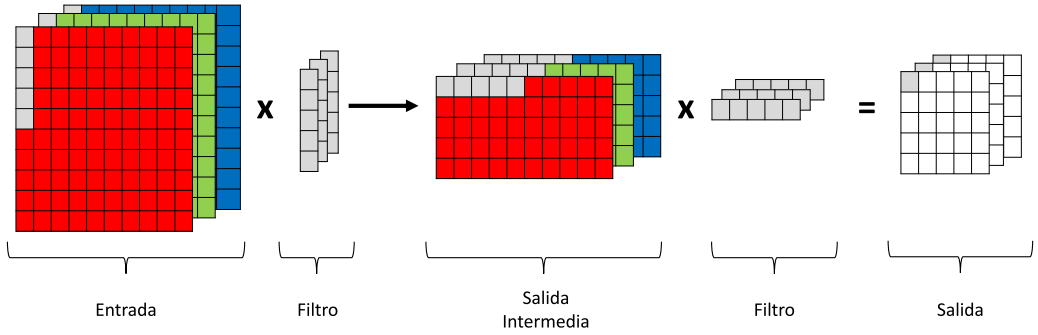


Figura 3.14.: Convolución separable. Ejemplo aplicado a una imagen RGB.

- **Convolución en profundidad**. Consiste en aplicar individualmente un filtro convolucional a cada uno de los canales de entrada para mantener aislado cada canal. El proceso consiste en dividir la entrada y los filtros en los canales correspondientes. Luego, se aplica la convolución sobre cada una de las entradas con su correspondiente filtro. Finalmente, se apilan los resultados obtenidos por la convoluciones realizadas anteriormente. La Ecuación (3.36) describe formalmente la convolución en profundidad:

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} \cdot F_{k+i-1,l+j-1,m} \tag{3.36}$$

donde $\hat{K}$ es el filtro de la convolución en profundidad de tamaño $D_k \times D_k \times M$. Así, el m-ésimo filtro $\hat{K}$ se aplica al m-ésimo canal en F para producir el m-ésimo canal del mapa de características de salida. El coste de la convolución en profundidad sería:

$$Coste = D_k \cdot D_k \cdot M \cdot D_f \cdot D_f \tag{3.37}$$

Por tanto, resulta notoria la mejora en la eficiencia que conlleva la utilización de la convolución en profundidad en comparación con una convolución estándar. Por otra parte, resulta importante destacar que este tipo de convoluciones sólo permiten el filtrado de los canales de entrada y no realizan combinaciones entre dichos canales para generar nuevas características. Por esta razón, es necesaria una capa adicional que permita computar la combinación lineal de la salida de la convolución en profundidad mediante una convolución de $1 \times 1 \times M$ que se encarga de producir las nuevas características. El esquema de procesamiento de las convoluciones en profundidad puede verse en la Figura 3.15:

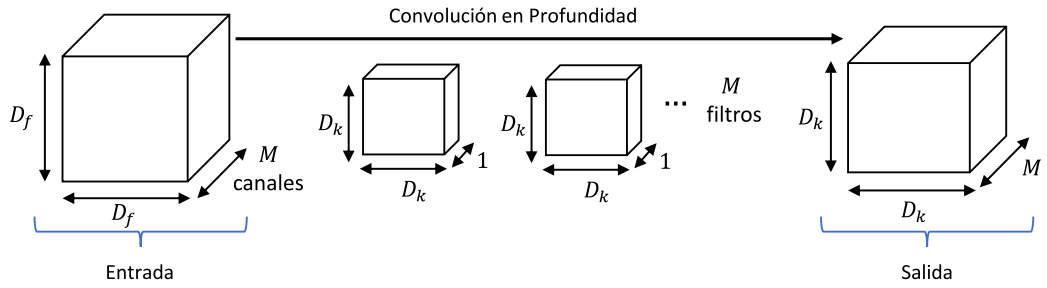


Figura 3.15.: Convolución en profundidad. Estructura de procesamiento. Imagen adaptada de [172].

La convolución separable y la convolución en profundidad pueden combinarse para realizar el procesamiento de los datos de entrada en términos de sus dimensiones espaciales y de profundidad [174]. Esto es un proceso que consta de dos partes: en primer lugar, la operación convolucional corresponde a una convolución separable espacial que se aplica a cada canal de los datos de entrada. Luego, a la salida de la primera convolución se le aplica la operación convolucional puntual para proyectar la salida del canal dentro de un nuevo espacio. En relación al coste computacional de la convolución separable en profundidad, su cálculo se lleva a cabo mediante la suma de los costes correspondientes a la convolución puntual y la convolución en profundidad:

$$Coste = D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f \tag{3.38}$$

Teniendo esto en cuenta, la reducción de la complejidad de la convolución separable en profundidad en comparación con las convoluciones 2D sería la siguiente:

$$\frac{D_k \cdot D_k \cdot M \cdot D_f \cdot D_f + M \cdot N \cdot D_f \cdot D_f}{D_k \cdot D_k \cdot M \cdot N \cdot D_f \cdot D_f} = \frac{1}{N} + \frac{1}{D_k^2} \tag{3.39}$$

En la Figura 3.16 es posible observar el esquema de funcionamiento de la convolución separable en profundidad que incluye las convoluciones separables y puntual:

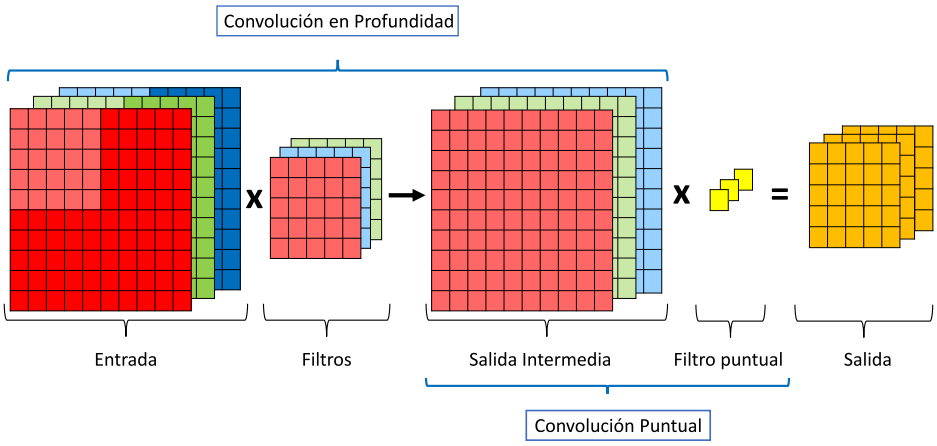


Figura 3.16.: Convolución separable en profundidad. Ejemplo aplicado a una imagen RGB. Imagen adaptada de [175].

Por otro lado, las capas convolucionales también cuentan con otros parámetros que deben ser configurados antes de iniciar el entrenamiento de una CNN. Estos parámetros son conocidos por el nombre de hiperparámetros. Algunos de ellos son:

- **Filtro (*kernel*)**. Matriz cuadrada o rectangular de dimensión (M, N), la cual será menor a la de la imagen original que recorre la imagen.
- **Relleno (*padding*)**. Agregación de píxeles en los bordes de la imagen. El objetivo es evitar la pérdida de información relevante presente en los bordes o esquinas de la imagen (ver Figura 3.17).

0	0	0	0	0	0
0	5	12	6	10	0
0	4	3	9	11	0
0	7	19	17	2	0
0	16	14	6	3	0
0	0	0	0	0	0

Figura 3.17.: Método de relleno (*padding*).

- **Zancada (*stride*)**. Número de píxeles que la ventana será desplazada en los dos ejes (define también el solapamiento de la ventana).

Capas de agrupación (*pooling*)

La capa de agrupación es adyacente a la capa convolucional y tiene como tarea reducir la dimensionalidad. Es una matriz que recorre la imagen, y dependiendo del tipo de operación de *pooling* utilizada, escoge su valor máximo (*max-pooling*) o medio (*average-pooling*) (ver Figura 3.18).

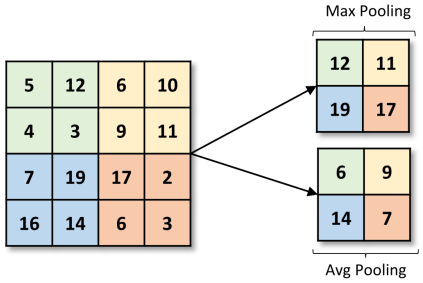


Figura 3.18.: Capa de agrupamiento (*pooling*).

No obstante, las implementaciones actuales sustituyen el uso de la capa de agregación por un tamaño de zancada mayor de 1.

Normalización por lotes

La normalización por lotes es una técnica utilizada durante el proceso de entrenamiento de las ANNs para evitar problemas de convergencia debidos a las diferencias de escala existentes entre las capas ocultas que la componen [176]. Esta técnica consiste en normalizar las entradas de cada capa de la red mediante la utilización de la media μ y la varianza σ^2 (Ecuaciones (3.40) y (3.41)) de cada lote de entrada:

$$\mu = \frac{1}{m} \cdot \sum_{i=1}^{m} h_i \tag{3.40}$$

$$\sigma^2 = \frac{1}{m} \cdot \sum_{i=1}^{m} (h_i - \mu)^2 \tag{3.41}$$

siendo h la activación de una neurona de la capa oculta y m la cantidad de instancias del lote. La normalización del lote de datos h está dada por h_{bn} según la siguiente expresión:

$$h_{bn} = \frac{h - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{3.42}$$

donde ϵ es un pequeño valor que se utiliza para evitar que se produzcan divisiones entre cero. Posteriormente, al resultado de la normalización h_{bn} se le añade una transformación lineal multiplicando por γ y sumando β con el fin de evitar posible linealidad:

$$BN_{\gamma\beta} = \gamma h_{bn} + \beta \tag{3.43}$$

Finalmente, es importante destacar que la normalización por lotes puede ser aplicada de forma previa o posterior al paso de los datos por la función de activación.

Capa totalmente conectada

La capa totalmente conectada [177] se utiliza para implementar una red perceptrón multi-capa (MLP) [178] que se encarga de la clasificación de las características extraídas por las capas convolucionales. Puede expresarse como:

$$y_{fc}[i] = b_{fc}[j] + \sum_{j=0}^{N-1} W_{fc}[ij] \cdot x_j \qquad \text{con} \quad 0 \leq i \leq N \tag{3.44}$$

donde N es el número de neuronas de la última capa. En la expresión anterior, $y_{fc}[i]$ indica la salida de la neurona i-ésima, x_j es la entrada j-ésima a dicha neurona, W_{fc} es la matriz de pesos y $b_{fc}[j]$ es el sesgo correspondiente a la neurona j-ésima.

3.6.2. Clasificación de series temporales mediante arquitecturas convolucionales 1D

Las redes neuronales convolucionales 1D (CNN-1D) son utilizadas para el procesamiento de series temporales [179-182]. Su funcionamiento es análogo a las tradicionales CNN-2D que han sido desarrolladas para el procesamiento de imágenes y vídeos. No obstante, la principal diferencia radica en que el filtro se desliza a través de una única dimensión mientras que en las capas convolucionales 2D la ventana se desliza en dos dimensiones. Aunque pueden realizarse convoluciones 1D utilizando capas convolucionales 2D y considerando que una de las dimensiones es 1, el uso de implementaciones específicas para convoluciones 1D tiene algunas ventajas con respecto a las CNN-2D. Concretamente, permiten el diseño de arquitecturas menos profundas con menor cantidad de parámetros, permitiendo simplificar el proceso de

implementación así como el de entrenamiento. Además, las CNN-1D implementadas utilizando capas convolucionales 1D requieren de menor cantidad de recursos en comparación a las redes CNN-2D.

En la Figura 3.19 se presenta una propuesta de arquitectura de CNN que implementa capas convolucionales 1D. Específicamente, la señal de entrada tiene dimensión 1×100 donde la configuración que se utiliza [171] incluye tres capas convolucionales 1D y capas de tipo *pooling*. Además, se incluyen dos capas totalmente conectadas y finalmente la capa de salida. Por otra parte, los hiperparámetros que han sido ajustados considerando la arquitectura propuesta son: el número de capas ocultas convolucionales y las capas que formaran parte del MLP, cuyos valores serán 3 y 2 respectivamente. El tamaño del filtro en todas las capas es de 41.

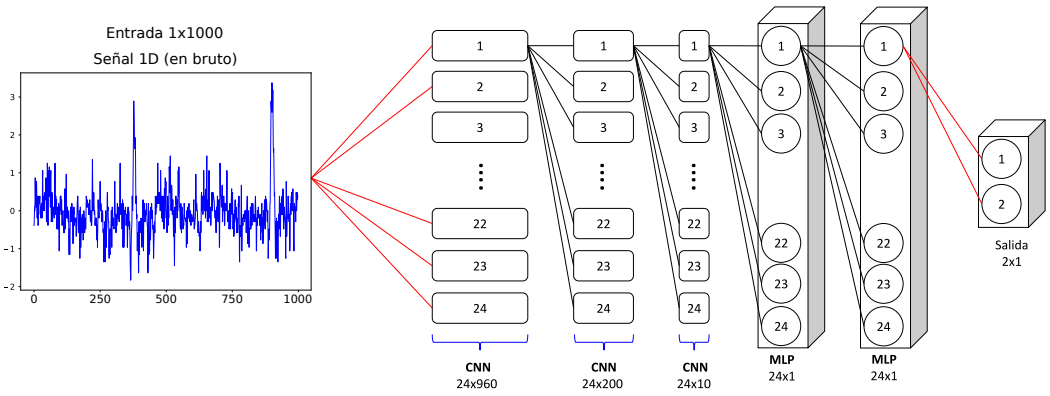


Figura 3.19.: Arquitectura de una CNN-1D. Configuración que consta de tres capas convolucionales 1D, el MLP compuesto por dos capas totalmente conectadas y la capa de salida. Imagen adaptada de [171].

Así, el cambio fundamental que se produce entre las CNN-1D y las 2D es el reemplazo de las matrices 2D a las matrices 1D en filtros y mapas de características.

Arquitecturas convolucionales 1D para clasificación de señales EEG

Actualmente existe un amplio espectro de aplicaciones basadas en DL para el procesamiento de señales biomédicas, siendo posible encontrar una gran cantidad de trabajos donde se utilizan CNNs profundas para tareas de clasificación. Específicamente, en el ámbito de los sistemas BCI con base en señales EEG, las CNNs han sido implementadas para la predicción y seguimiento de la epilepsia [183], la clasificación de movimiento imaginario motor [184] y la detección de las respuestas visuales evocada [185].

El enfoque de los estudios citados previamente denota la falta de consenso en diversos aspectos tales como los relativos al diseño de la red o la normalización de los datos. Además, los estudios suelen enfocarse en tareas específicas para BCI. Consecuentemente, no existe un enfoque general que unifique los criterios en el diseño de arquitecturas CNN para el procesamiento de señales EEG que provienen de sistemas BCI. En ese sentido, la investigación realizada en [186] propone dos diferentes tipos de arquitecturas CNN (profunda y superficial) para la clasificación de tareas imaginadas o ejecutadas mediante la utilización de señales EEG: *DeepConvNet* y *ShallowConvNet*. El objetivo principal de la propuesta consiste en que la arquitectura de las CNNs sea genérica, es decir, que utilicen elementos básicos de las CNNs sin requerir de conocimiento experto para su diseño pero que permita al mismo tiempo alcanzar buenos resultados de clasificación.

La *DeepConvNet* es una CNN profunda para extracción de características cuya arquitectura cuenta con cuatro bloques compuestos de una capa convolucional estándar y otra *Max-Pooling*. El quinto bloque corresponde a la capa de clasificación. En la Figura 3.20 se observa el diseño de su arquitectura. Además, la arquitectura *DeepConvNet* contempla dentro del primer bloque convolucional el uso de dos capas convolucionales que le permiten administrar la gran cantidad de canales de entrada provenientes de cada uno de los electrodos. En la primera capa convolucional los filtros realizan una convolución en el tiempo, mientras que en la segunda se ejecuta un filtro espacial sobre la salida de la convolución anterior. Entre ambas capas convolucionales del primer bloque no se utilizan funciones de activación. En cada uno de los bloques siguientes la función de activación utilizada es la *ELU*.

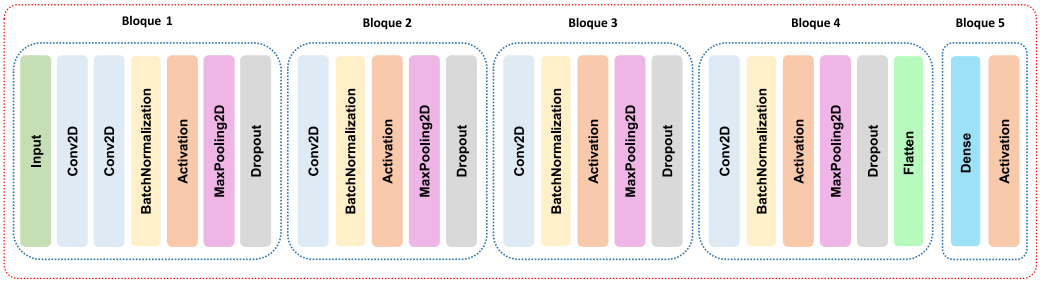


Figura 3.20.: DeepConvNet. Arquitectura de CNN profunda propuesta en [186].

La *ShallowConvNet* es una arquitectura cuyo primer bloque es análogo al de la *DeepConvNet*, realizando una convolución temporal seguida de una convolución espacial. La diferencia de la *ShallowConvNet* con respecto a la *DeepConvNet* viene dada por la menor cantidad de bloques con el que cuenta. Además, el tamaño del filtro temporal que utiliza *ShallowConvNet* es menor. En la Figura 3.21 se observa el diseño de la arquitectura de la *ShallowConvNet*:

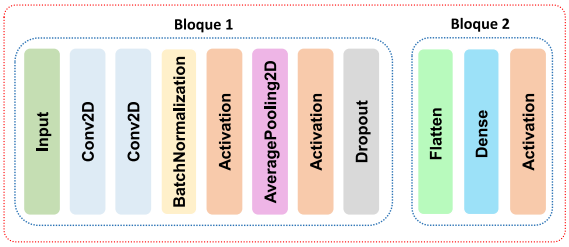


Figura 3.21.: ShallowConvNet. Arquitectura de CNN superficial propuesta en [186].

Combinando las ideas de las redes anteriores, Lawhern et al. en [187] propone una arquitectura CNN denominada *EEGNet* para clasificación de señales EEG en sistemas BCI. En la arquitectura *EEGNet* se incorporan las capas de convoluciones separables y en profundidad para realizar la extracción de características más relevantes de la señal EEG, permitiendo disponer de una red más profunda y con menor número de parámetros. En la Figura 3.22 se muestra el diseño de la arquitectura *EEGNet*:

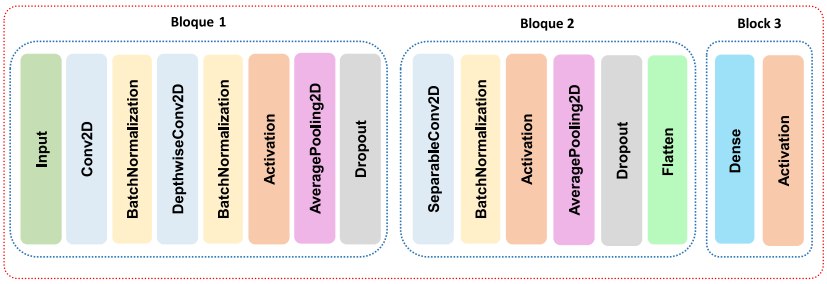


Figura 3.22.: EEGNet. Arquitectura de la CNN propuesta en [187].

La arquitectura *EEGNet* consta de tres bloques principales. La primera capa utilizada es una capa convolucional 2D que permite capturar información temporal. Posteriormente, la capa convolucional en profundidad *(depthwise)* permite extraer información espacial. Los datos de salida de cada una de las capas convolucionales implementadas en este bloque pasan por un proceso de normalización por lotes. En este punto, se procede a reducir la dimensionalidad del conjunto de datos mediante su paso a través de la capa de agrupación *max-pooling* y, finalmente, se incorpora la capa *dropout* para su regularización.

El segundo bloque de la *EEGNet* sigue dedicado a la extracción de características. Se utiliza para ello una convolución separable que permite reducir el número de parámetros que deben ajustarse y facilita la representación de los datos. Luego, los valores de salida de la capa convolucional separable son normalizados. Los nuevos valores normalizados pasan por la capa de activación para introducir la no-linealidad. Posteriormente, se utiliza la capa *dropout* para la regularización y finalmente los

valores obtenidos se colapsan en una matriz unidimensional para que puedan ser procesados por el clasificador.

El último bloque corresponde al clasificador e incorpora una capa totalmente conectada que utiliza en su salida una activación de tipo *softmax*. Dicha función se encarga de especificar la probabilidad de activación de cada una de las neuronas de la salida que tienen relación directa con las distintas clases de las señales procesadas.

4 Optimización multi-objetivo

Vivimos en un mundo en constante evolución con recursos limitados y necesidades crecientes [188], por lo que cada vez tiene mayor importancia aprovechar eficientemente los recursos para dar soluciones adecuadas a diferentes problemas. Solucionar problemas de optimización significa minimizar determinadas variables (costes, complejidad, tiempos, errores...) y maximizar otras (beneficios, calidad, eficiencia...) que, además, suelen estar sujetas a restricciones. En los problemas reales se pueden encontrar varios objetivos que deben optimizarse simultáneamente y que suelen estar en conflicto entre sí. Esto es, que no es posible encontrar un óptimo para cada uno de los objetivos. Dichos problemas se conocen como problemas de optimización multi-objetivo (MOO) y están presentes en prácticamente cualquier campo de las ingenierías, ciencias, gestión, toma de decisión, entre otros. Las ciencias de la computación no es ajena a esta situación, pues es una de las áreas donde se aplica de forma intensa toda la teoría de la optimización multi-objetivo para alcanzar las soluciones propuestas.

La resolución de los MOO puede darse a través de dos métodos diferentes: exactos y aproximados (heurísticos). Los métodos exactos implican un gran coste en términos de cómputo y tiempo, lo cual en ocasiones puede resultar inviable. Sin embargo, en el caso de alcanzar una solución, ésta corresponderá a la solución óptima. En contrapartida, con los métodos aproximados el problema puede resolverse de forma más sencilla y con menor costo, aunque no es posible determinar una única solución que garantice su optimalidad.

La naturaleza ha servido de fuente de inspiración para el desarrollo de paradigmas que permitan la resolución de MOO mediante la imitación del comportamiento de los sistemas naturales [189]. Dado que esta tesis tiene relación con los algoritmos bioinspirados, en este capítulo se abordarán los conceptos más importantes de los algoritmos genéticos (GA) [190]. Dichos procedimientos se basan en la teoría de la evolución [191] y resultan de especial utilidad en problemas donde el espacio de búsqueda es demasiado amplio como para utilizar métodos de búsqueda exactos. Su funcionamiento es simple: iterar a lo largo de varias generaciones para mejorar las soluciones propuestas mediante los llamados operadores genéticos. Estos operadores

imitan los mecanismos de cruce, mutación, copia y selección que tienen lugar en la adaptación del genoma de las especies durante el proceso evolutivo.

4.1. Enfoques para el problema de optimización

Obtener la solución óptima a un problema consiste en encontrar la mejor solución de entre un conjunto de posibles soluciones [192]. Para evaluar la bondad de una solución se utiliza una función de evaluación (FE) (*fitness*) que sirve como índice de calidad de acuerdo a un criterio establecido.

4.1.1. Métodos de optimización

Métodos exactos

Los métodos exactos tienen como objetivo encontrar una única solución (óptima) para el problema de optimización. Sin embargo, en la mayoría de los casos prácticos, el problema de optimización es complejo y el tiempo requerido para encontrar esta solución suele ser muy largo, por lo que su aplicación puede resultar inviable. De acuerdo a [193], los métodos exactos funcionan de forma efectiva en problemas donde el espacio de búsqueda es pequeño. Estos métodos son:

- **Programación dinámica**. Se trata de un método matemático de optimización mediante un proceso secuencial de toma de decisiones que permite alcanzar la solución óptima [194]. En otras palabras, es un método iterativo de descomposición del problema en distintas etapas o subproblemas mas sencillos donde se toman decisiones parciales que formarán parte del conjunto total de decisiones. La principal ventaja de este método es que evita la exploración de todo el espacio de búsqueda mediante la depuración de las decisiones que no permitan obtener la solución óptima.
- **Ramificación y poda (B&B)**. Permite la optimización global imponiendo restricciones en forma de cotas superiores e inferiores sobre el valor objetivo óptimo. Este método suele ser bastante lento ya que requiere de un gran esfuerzo computacional que además crece exponencialmente con el tamaño del espacio de búsqueda [195].

- **Algoritmo A***. Desarrollado para abordar una única función objetivo. No obstante, posteriormente fue extendido para tratar múltiples objetivos [196, 197]. El funcionamiento del algoritmo está basado en determinar la mejor ruta desde un punto inicial o nodo hasta el punto final (objetivo).
- **Programación con restricciones**. Permite resolver problemas de optimización con restricciones. La definición del problema de optimización se lleva cabo mediante la determinación de un conjunto de variables que se vinculan con el conjunto de restricciones. Si bien es cierto que las restricciones limitan el espacio de búsqueda, lo cual se traduce en una mayor eficiencia en términos de tiempo, también dificultan la búsqueda de la solución óptima por la pérdida del criterio de optimalidad [198].

Métodos aproximados

Los métodos aproximados, también conocidos como métodos heurísticos, permiten alcanzar soluciones de calidad aun cuando no se pueda garantizar que de entre estas soluciones se encuentre la óptima. Cabe resaltar que el tiempo de procesamiento es de gran importancia, por lo que normalmente se busca un compromiso entre la calidad de las soluciones y el tiempo empleado en llegar a ellas. Estos métodos, además, pueden ser clasificados en heurísticas simples y metaheurísticas. Las heurísticas simples tienen como objetivo la resolución de un problema especifico y su principal ventaja radica en su mayor velocidad de ejecución. No obstante, su principal desventaja es la facilidad con la que tienden a quedarse estancadas en mínimos locales. Por otro lado, las metaheurísticas pueden ser consideradas como algoritmos de propósito general que permiten la resolución de casi cualquier tipo de problema de optimización, y su principal ventaja radica en la gran capacidad que tienen para la resolución de problemas de gran complejidad (espacios de búsqueda grandes o múltiples objetivos). Algunos ejemplos típicos de metaheurísticas son los algoritmos bioinspirados y los algoritmos basados en comportamientos de enjambres. A continuación se detallan algunas de las principales metaheurísticas utilizadas con gran éxito a lo largo de los últimos años:

- **Algoritmos evolutivos**. Inspirados en los procesos evolutivos de las especies donde la mejora constante de los individuos se produce a través de sucesivas generaciones [199].
- **Algoritmos de enfriamiento simulado (*simulated annealing*)**. Inspirado en el proceso térmico de recocido del acero y otros materiales. Este algoritmo

permite encontrar buenas aproximaciones a la solución óptima en vastos espacios de búsqueda [200].

- **Enjambres de partículas**. Inspirados en el comportamiento de las bandadas de pájaros o bancos de peces [46]. El proceso de optimización se basa en mover de posición las soluciones candidatas (partículas) dentro del espacio de búsqueda. El movimiento de cada solución está determinado por su mejor posición local encontrada y por las mejores posiciones globales encontradas por otras soluciones.
- **Búsqueda tabú**. Inspirada en el concepto de memoria para guiar las búsquedas. Mediante un registro histórico de búsqueda se puede extraer información de lo sucedido previamente y actuar en consecuencia [201].
- **Colonias de hormigas**. Simulan el comportamiento de las hormigas para buscar el camino más corto entre la comida y el hormiguero [202].
- **Algoritmos híbridos**. Se producir al combinar diversas metaheurísticas.

A continuación se presenta una figura que esquematiza la organización de los diferentes métodos de optimización:

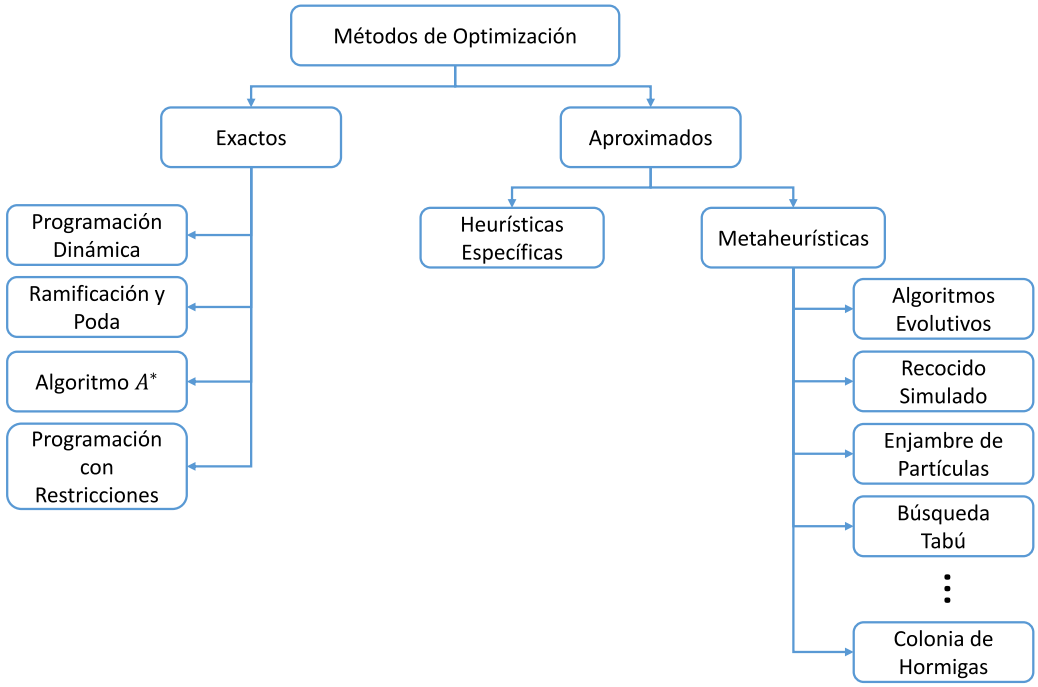


Figura 4.1.: Clasificación de los métodos de optimización. Esquema que presenta los diferentes métodos de optimización agrupados por tipo.

4.1.2. Optimización mono y multi-objetivo

Los problemas de optimización que consideran un único objetivo se centran en mejorar la calidad de la solución mediante una única función de coste (fitness) que debe ser minimizada o maximizada. En el caso de múltiples objetivos existirán múltiples funciones que tendrán que ser optimizadas de forma simultánea. La optimización puede llevarse a cabo de dos formas:

1. Combinando las funciones fitness f_i de cada objetivo para optimizar una única función f. Cada una de las funciones podría ser multiplicada por un peso λ_i que modulará la importancia relativa de cada objetivo. Dicha función puede expresarse de la forma $f = \lambda_1 \cdot f_1 + \lambda_2 \cdot f_2 + \cdots + \lambda_n \cdot f_n$ para el caso genérico de n objetivos.

2. Utilizando algoritmos evolutivos que utilizan el concepto de hipervolumen como métrica de calidad de las soluciones encontradas en el Frente de Pareto. Algunos ejemplos de estos algoritmos son el algoritmo genético de ordenación no dominada (NSGA-II y NSGA-III) [203-205] o el algoritmo evolutivo de Pareto fuerte (SPEA-II y SPEA-III) [206].

La Ecuación 4.1 define formalmente el problema de optimización multi-objetivo:

$$\begin{aligned} &\underset{f(x)}{\text{Minimizar}} && (f_1(x), f_2(x), \cdots, f_M(x)) \\ &\text{sujeto a} && x \in X \rightarrow \mathbb{R} \wedge k \geq 2 \end{aligned} \tag{4.1}$$

donde M es la cantidad de funciones objetivo y X el conjunto de posibles soluciones dentro del espacio de búsqueda del problema multi-objetivo. La función de coste definida por el vector f, se utiliza para evaluar de la calidad de cada una de las soluciones $(x_1, x_2, \ldots, x_n)$.

Un concepto importante en los problemas de optimización multi-objetivo es el de dominancia. La definición de dominancia establece que para determinar si una solución es mejor que otra, la solución evaluada deberá ser mejor en todos los objetivos del problema y en ningún caso deberá ser peor en cualquier otro objetivo. Formalmente, la definición de las dos condiciones necesarias para establecer la relación de dominancia se detallan a continuación:

$$
\begin{aligned}
&(1) \quad f_i(x) \leq f_i(y) \quad \forall i \in (1 \ldots M) \\
&(2) \quad \nexists i / f_i(x) > f_i(y)
\end{aligned}
$$

Si alguna de estas dos condiciones no se cumple entonces la solución x no dominará a la solución y. Dado que en problemas de optimización multi-objetivo los diferentes objetivos están en conflicto, se obtiene un conjunto de soluciones factibles que pueden considerarse óptimas para un problema en concreto. En la Figura 4.2 se muestra por ejemplo la inexistencia de una relación de dominancia entre los puntos A y B pero por el contrario el punto C se encuentra dominado tanto por A como por B.

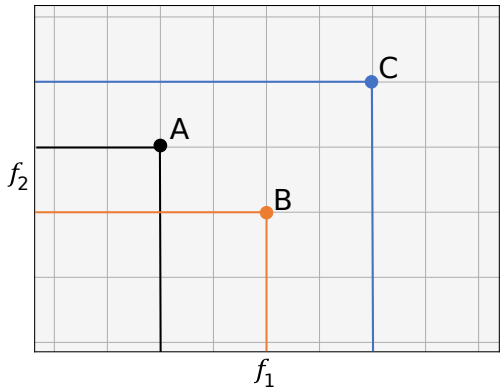


Figura 4.2.: Relación de dominancia entre los puntos A, B y C.

A partir del concepto de dominancia se define el frente de Pareto (FP) como el espacio geométrico de las soluciones que no se encuentran dominadas por otras soluciones. Formalmente, el frente de Pareto se define como [207, 208]:

Definición 4.1.1 (Óptimo de Pareto)**.** Una solución $x^* \in X$ se denomina óptimo de Pareto si no existe otra solución que la domine. Es decir, que $\nexists x \in X$ tal que $f_i(x) \leq f_i(x^*) \wedge f_i(x) < f_i(x^*); \forall_i, j \in \{1, \cdots, M\}$. Esto se denota como $f(x) \nprec f(x^*)$.

Por tanto, el principal objetivo es encontrar el frente de Pareto, el cual se expresa matemáticamente como:

$$
\begin{aligned}
&FP = \{F(x); x \in X\} \\
&\text{sujeto a } \nexists x \in X \text{ tal que } f(x) \nprec f(x^*)
\end{aligned}
\tag{4.2}
$$

En la Figura 4.3 se muestra un ejemplo de frente de Pareto para un problema de optimización con dos objetivos, representados por las funciones de coste f_1 y f_2. El número de soluciones situadas en el frente de Pareto dependerá del problema específico de optimización, de la población inicial de soluciones y del algoritmo utilizado. Como puede verse en la Figura 4.3, no es posible minimizar los dos objetivos de forma simultánea. No obstante, el codo de la curva, también conocido como óptimo de Pareto, representa la solución de compromiso entre ambos objetivos.

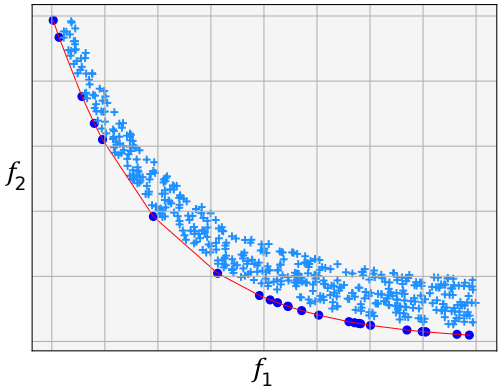


Figura 4.3.: Ejemplo de frente de Pareto para 2 objetivos representados por las funciones de coste f_1 y f_2.

4.2. Algoritmos evolutivos

Charles Darwin es considerado el padre de la teoría de la evolución. En su libro "El origen de la especies" de 1859 postula que los individuos que mejor se adaptan al medio son los que sobreviven, tienen mayor capacidad para reproducirse y más probabilidad de perpetuar la especie transmitiendo su información genética a las siguientes generaciones. No obstante, la teoría evolutiva moderna, conocida también como el paradigma neo-darwiniano, se apoya en tres teorías que se complementan. Esta teorías son: (ii) la teoría evolutiva de Darwin; (ii) las teorías de Mendel acerca de la transmisión de características de una generación a otra y (iii) la teoría Weismann, que considera el fenotipo (rasgos) y el genotipo (información genética) de los individuos. Este paradigma establece que los ciclos biológicos de mantenimiento de la vida pueden ser modelados como procesos estadísticos en los que interactúan las especies y sus poblaciones mediante mecanismos de reproducción, mutación, competencia y selección, lo cual da lugar a la evolución de las especies.

Los algoritmos evolutivos (EAs) están basados en la teoría evolutiva de los organismos biológicos, es decir, imitan el proceso evolutivo de una población de individuos.

Inicialmente, la población se genera de forma aleatoria dando lugar a un conjunto de soluciones aleatorias. Tras evaluar la bondad de las soluciones mediante las correspondientes funciones de coste se aplican los operadores genéticos de cruce, mutación y selección, produciendo una nueva generación que contendrá soluciones mejoradas (es decir, adaptadas al problema). La utilización de los EA para la resolución de problemas de optimización se encuentra muy extendida dado que alcanzan soluciones satisfactorias sin ningún conocimiento previo acerca del problema en cuestión. En este contexto, los EAs cuentan con tres tipos distintos de paradigmas: las estrategias evolutivas (EEs), la programación evolutiva (EP) y los algoritmos genéticos (GAs). Este capítulo se centra en los algoritmos genéticos pues ha sido el paradigma utilizado en esta Tesis.

4.2.1. Algoritmos genéticos

Los GAs [209] fueron propuestos en el año 1962 por John Holland [210] como estrategia para la resolución de problemas de optimización. Sin embargo, no fue hasta el año 1975 cuando De Jong [211] utilizó por primera vez los GAs para la resolución de un problema de optimización paramétrico. Aún así, realmente la gran popularidad de los GAs se produjo hacia finales de la década de 1980 con la publicación del libro de Golberg [209]. Los GAs utilizan conjuntos de individuos (soluciones) que son representados (codificados) como cromosomas, y es importante resaltar que no aseguran la convergencia [212] a un conjunto de soluciones óptimas. En la Figura 4.4 se muestra el esquema básico de funcionamiento de un GA.

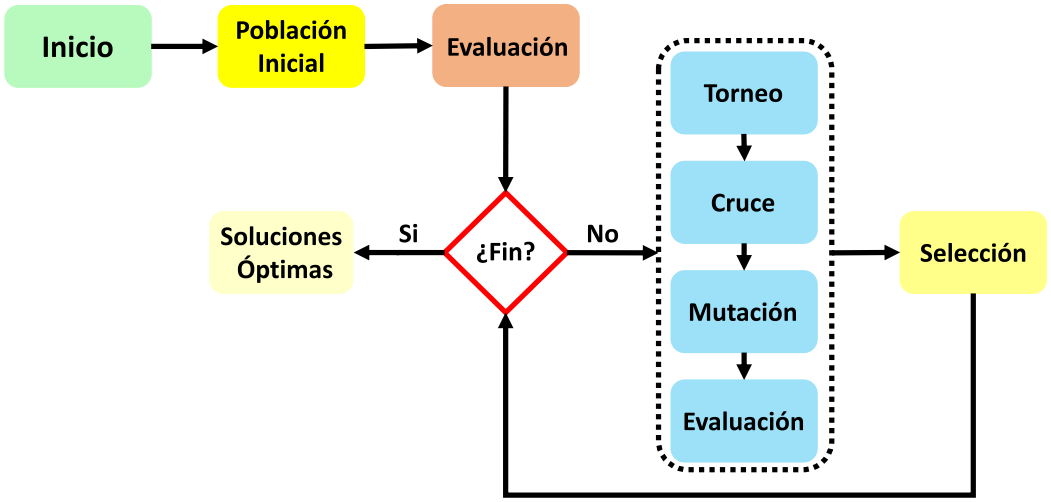


Figura 4.4.: Algoritmo genético. Esquema básico de su funcionamiento.

Representación y función de evaluación

Los problemas de optimización que implementan GAs requieren tomar una serie de decisiones previas teniendo en consideración el problema específico a resolver. En este sentido, la cantidad de genes del cromosoma y su representación (codificación, tipo de datos, etc.) son cuestiones que deben resolverse previamente a la ejecución del GA. La cantidad de genes de un cromosoma se determina en base al espacio de decisión del problema de interés, mientras que los genes pueden estar representados por valores reales, enteros o binarios. Otra cuestión esencial que debe ser resuelta previamente es la función de evaluación o coste (FE), la cual se encarga de evaluar la calidad de cada una de las soluciones alcanzadas durante el proceso evolutivo. En los problemas de maximización, cuanto mayor es el valor retornado por la función mayor será la calidad de la solución. En problemas de minimización un valor inferior será una mejor solución.

Otra decisión que se debe tomar es la condición de parada. Aunque sea posible seguir explorando el espacio de búsqueda debe establecerse algún criterio de parada que limite el coste computacional del procedimiento. Algunos de los criterios de parada empleados en los GAs son:

1. Parar cuando se alcance un número de generaciones determinado a priori (definido previamente).
2. Parar cuando las soluciones superen un umbral de calidad (definido previamente).
3. Parar cuando tras computar cierto número de generaciones no haya mejoría en la calidad de las soluciones.

También se debe decidir cómo evolucionar las soluciones para adaptarse al problema, es decir, cuáles serán los operadores de cruce, mutación y selección a utilizar.

Operadores de cruce

La función principal de los operadores de cruce es obtener una nueva descendencia de individuos a partir de la última población existente. Para alcanzar este propósito se seleccionan pares de individuos de la generación actual que actuarán como padres del nuevo individuo al combinar la información contenida de sus cromosomas. Es importante tener en cuenta la representación cromosómica de los individuos ya que es un factor que condicione el tipo de operador genético de cruce que deberá ser

implementado. En este sentido, los operadores genéticos de cruce pueden combinar cromosomas representados con números enteros o reales. A continuación se detallan algunos de los operadores de cruce más utilizados para cromosomas que tienen representación binaria:

- **Basado en un punto**. Selecciona aleatoriamente un gen del cromosoma de los padres. El gen seleccionado dividirá al cromosoma de los padres en dos partes (derecha e izquierda). La parte izquierda del cromosoma del primer hijo contendrá los genes de la parte izquierda de uno de los padres mientras que en la derecha tendrá la parte equivalente del otro padre. El cromosoma del segundo hijo es justo lo contrario. En la Figura 4.5 es posible observar el funcionamiento de este operador de cruce:

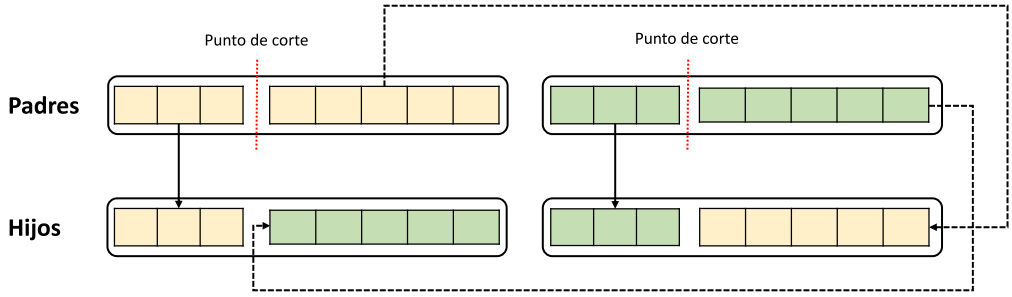


Figura 4.5.: Cruce basado en un punto. Esquema básico de su funcionamiento.

- **Basado en dos puntos**. Este operador actúa de forma similar al operador anterior (basado en un punto). La principal diferencia radica en que el cromosoma se divide en tres partes, para lo cual es necesario seleccionar dos genes aleatorios distintos que indicarán el punto de corte del cromosoma. A continuación se genera el primer hijo: su cromosoma estará compuesto por los genes del primer padre tanto en la parte derecha como en la izquierda, mientras que la parte central del cromosoma adquirirá los genes del segundo padre. Siguiendo la lógica, el cromosoma del segundo hijo tendrá en su parte derecha e izquierda los genes del segundo padre, mientas que la parte central estará compuesta por los genes del primer padre. En la Figura 4.6 se puede observar su funcionamiento:

- **Uniforme**. Este operador realiza una comparación individual entre los genes de los padres. En caso de encontrar diferencias entre los genes, éstos son individualizados y asignados con una cierta probabilidad de ser intercambiados. Por lo tanto, el cromosoma de los hijos tendrá los mismos genes de uno de sus progenitores con excepción de aquellos genes que han sido intercambiados

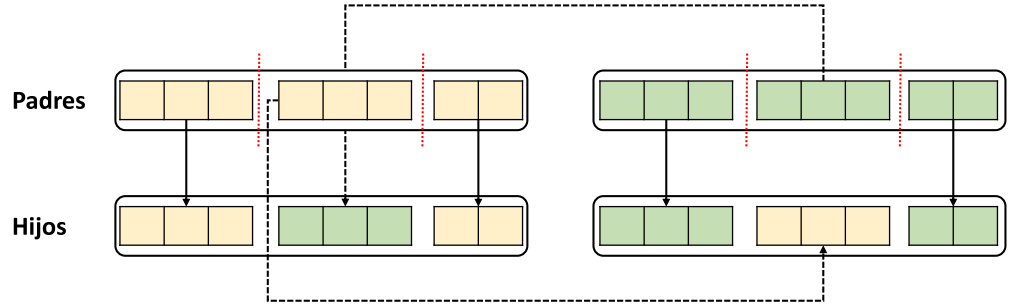


Figura 4.6.: Cruce basado en dos puntos. Esquema básico del funcionamiento del operador.

por los genes del otro progenitor. En la Figura 4.7 es posible observar el funcionamiento del operador.

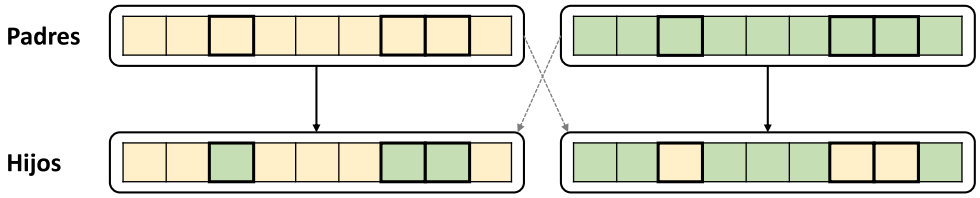


Figura 4.7.: Cruce uniforme. Esquema básico del funcionamiento del operador.

Por otra parte, los cromosomas codificados en notación real cuentan con otros tipos de operadores de cruce diseñados específicamente para este tipo de representación. Uno de los más populares operadores de este tipo es el operador de cruce binario simulado (SBX) [213] debido a su notable rendimiento, razón por la cual han sido propuestas diversas variantes y extensiones. Dicho operador trata de hacer que la descendencia sea cercana a los padres. Para ello, implementa una función de distribución [214] que utiliza el parámetro η para delimitar el espacio de exploración de las soluciones. Por tanto, índices de distribución pequeños incrementan la probabilidad de obtener soluciones alejadas de sus padres, mientras que índices de distribución elevados aumentan la probabilidad de alcanzar soluciones cercanas a sus padres. De esta forma, se define β como el factor de dispersión que considera el valor absoluto de la diferencia entre los hijos C_1 y C_2 en proporción a la diferencia existente entre los padres P_1 y P_2, tal y como se muestra en la siguiente ecuación:

$$\beta = \left| \frac{C_1 - C_2}{P_1 - P_2} \right| \tag{4.3}$$

Por tanto, la probabilidad de distribución β puede ser calculada como:

$$P(\beta) = \begin{cases} 0.5 \cdot (\eta + 1) \cdot \beta^{\eta}, & \text{si } \beta \leq 1 \\ 0.5 \cdot (\eta + 1) \cdot \frac{1}{\beta^{\eta+2}}, & \text{si } \beta > 1 \end{cases} \tag{4.4}$$

donde los hijos C_1 y C_2 pueden definirse como:

$$C_1 = 0.5 \cdot [(1 + \beta) \cdot P_1 + (1 - \beta) \cdot P_2] \tag{4.5}$$

$$C_2 = 0.5 \cdot [(1 - \beta) \cdot P_1 + (1 + \beta) \cdot P_2] \tag{4.6}$$

Operadores de mutación y selección

El proceso de adaptación de los individuos suelen generar cambios en su genética, algo que se conoce como mutación. Normalmente, estos cambios no suelen producirse en breves periodos de tiempo sino que suelen requerir de múltiples generaciones para que se produzcan. La mutación de los genes se establece en función de una probabilidad determinada. Aunque la mutación se encarga de proveer diversidad al espacio de búsqueda y evitar un temprano estancamiento en mínimos locales, su valor suele ser muy pequeño para no hacer que el algoritmo "no se pierda". De forma similar al proceso de cruce anteriormente descrito, la mutación se implementa de forma diferente dependiendo del tipo de codificación de los genes. Así, cuando los cromosomas utilizan codificación binaria, la mutación consiste en invertir el gen en cuestión. Sin embargo, cuando los cromosomas utilizan codificación real existen dos opciones: modificar el valor del gen por un valor constante o utilizar una función de probabilidad que permita la modificación del gen de forma aleatoria dentro de un rango de valores previamente establecidos.

Por otro lado, los operadores de selección tienen como principal objetivo la elección de los individuos que formarán parte de la siguiente generación. Con el fin de realizar su cometido, los operadores de selección asignan un valor de calidad a cada una de las soluciones. En problemas con un sólo objetivo se suele utilizar el valor retornado por la función de evaluación. En el caso multi-objetivo, encontrar una única solución resulta bastante improbable, más aún cuando la cantidad de individuos de la población o el número de objetivos incrementa. En consecuencia, dada la imposibilidad de determinar un orden absoluto para las soluciones propuestas resulta vital aplicar

otros operadores de selección. Como se verá en la próxima sección, este problema se puede resolver con los algoritmos basados en la ordenación no dominada.

4.2.2. Algoritmos genéticos de ordenación no dominada

El algoritmo genético de ordenación no dominada (NSGA) [215] fue propuesto en el año 1994 por Srinivas y Deb. Este algoritmo fue uno de los primeros EAs y sigue los pasos tradicionales de los GAs pero incorpora algunas características propias. El funcionamiento del algoritmo NSGA comienza con la clasificación de los individuos de la población en base al nivel de no dominancia, el valor de aptitud y el resultado obtenido al aplicar el parámetro de distribución. De esta forma, una vez que se haya realizado la primera clasificación de individuos se debe repetir el proceso hasta que todos los individuos de la población hayan sido clasificados. Los individuos que tengan los mejores valores de aptitud tendrán mayor probabilidad de generar descendencia y, por lo tanto, posibilita búsquedas más exhaustivas en regiones no dominadas. No obstante, el algoritmo NSGA presenta algunos importantes inconvenientes que impactan negativamente en sus posibilidades y eficiencia. De hecho, uno de los principales puntos débiles del NSGA es su complejidad computacional como consecuencia del método de ordenación no dominada que utiliza ($O(M \cdot N^3)$), siendo M el número de objetivos a optimizar y N el tamaño de la población. Debido a este problema, Deb et al. [203, 204] presentan en el año 2000 el algoritmo NSGA-II con el objetivo de sortear las limitaciones y desventajas de su antecesor.

NSGA-II

El NSGA-II es un algoritmo de gran popularidad cuya complejidad algorítmica es $\mathcal{O}(M \cdot N^2)$. Dado que está basado en el concepto de elitismo, permite conservar a los mejores individuos encontrados en cada nueva generación, es decir, que tanto padres como progenitores pueden formar parte del frente de Pareto.

El primer enfoque propuesto en [203] para clasificar las soluciones del primer frente no dominado de la población de tamaño N requiere realizar comparaciones entre cada una de las soluciones con respecto al resto de las soluciones con el fin de determinar si la solución es dominada. En definitiva, este es un procedimiento sencillo pero de muy lenta ejecución pues requiere de $\mathcal{O}(M \cdot N)$ comparaciones por cada solución. La complejidad computacional para la búsqueda de las soluciones no dominadas del primer frente es $\mathcal{O}(M \cdot N^2)$. De este modo, para seguir con la búsqueda de las soluciones que corresponden al siguiente frente de dominancia

es necesario que las soluciones del primer frente no dominado sean descartadas temporalmente y, a continuación, se repita el proceso descrito anteriormente.

El segundo enfoque propuesto en [203] para la clasificación de soluciones no dominadas corresponde al algoritmo de clasificación rápida no dominada. La cantidad de comparaciones necesarias para llevar a cabo esta clasificación es del orden $\mathcal{O}(M \cdot N^2)$. El Algoritmo 1 detalla el pseudocódigo que describe el funcionamiento de este enfoque. En primer lugar, es necesario calcular para cada solución p la cantidad de soluciones que la dominan (n_p) y la cantidad de soluciones a las que p domina (S_p). El contador de dominancia n_p debe ser inicializado a cero para todas las soluciones del primer frente no dominado.

Algoritmo 1: Clasificación rápida no dominada

```
para cada (p ∈ P) hacer
    S_p = ∅ ;
    n_p = 0 ;
    para cada (q ∈ P) hacer
        si (p ≺ q) entonces
            S_p = S_p ∪ {q} ;
        si no, si (q ≺ p) entonces
            n_p = n_p + 1 ;
        fin
    fin
    si (n_p = 0) entonces
        p_rank = 1 ;
        F_1 = F_1 ∪ {p} ;
    fin
fin
i = 1 ;
mientras (F_i ≠ ∅) hacer
    Q = ∅;
    para cada (p ∈ F_i) hacer
        para cada (q ∈ S_p) hacer
            n_p = n_p − 1 ;
            si (n_p = 0) entonces
                q_rank = i + 1 ;
                Q = Q ∪ {q} ;
            fin
        fin
    fin
    i = i + 1 ;
    F_i = Q ;
fin
```

En el primer frente de dominancia, cada solución p consta de un atributo p_{rank} que especifica el frente al que pertenece dicha solución. Seguidamente, en cada solución p con $n_p = 0$ se visita a cada miembro q que pertenece al conjunto S_p y además se disminuye el contador n_q en uno. Como consecuencia, si para alguno de los miembros de la población q el contador n_q alcanza el valor cero, entonces q se incluye en una lista Q. Los miembros de dicha lista corresponden al segundo frente no dominado. Este proceso se repite hasta que se identifiquen todos los frentes.

Diversidad

Mantener la diversidad genética de la población es sumamente importante puesto que el cruce entre individuos de una población homogénea no permite generar nuevas soluciones. Concretamente, la diversidad implica una correcta distribución de las soluciones que pertenecen al conjunto de soluciones obtenidas por el EA. El algoritmo NSGA implementa una función para el mantenimiento de la diversidad conocida en inglés como *sharing function*, la cual se ocupa de realizar los ajustes adecuados de los parámetros asociados. El parámetro σ_{share} es el que permite indicar el máximo valor de la métrica de distancia donde de forma indistinta ambas soluciones comparten el valor de aptitud.

El parámetro σ_{share} puede considerarse un hiperparámetro del algoritmo de optimización y debe ser seleccionado adecuadamente de acuerdo a las indicaciones que se presentan en [216]. No obstante, existen un par de inconvenientes relacionados con el enfoque de diversidad propuesto por el NSGA original:

1. El desempeño final de la función tiene una gran dependencia con el valor del σ_{share} seleccionado.
2. La complejidad $\mathcal{O}(N^2)$ de la función de intercambio producida por la cantidad de comparaciones que deben realizarse.

Por ello, el algoritmo NSGA-II implementa un enfoque distinto que utiliza las distancias de hacinamiento (distancia de *crowding*) para mantener cierto nivel de diversidad en las soluciones. No obstante, este enfoque requiere definir la métrica de estimación de la densidad si posteriormente se quiere utilizar un operador de comparación de hacinamiento. La estimación de densidad se define como el cálculo de la densidad aproximada entre aquellas soluciones que se encuentran próximas a una determinada solución de la población. Concretamente, la distancia de *crowding* $i_{distancia}$ se obtiene calculando el perímetro del paralelepípedo (cuboide) que tendrá por vértices a los vecinos más cercanos de una solución en particular. La Figura 4.8

muestra gráficamente el cálculo de la distancia de crowding para el caso de dos objetivos. El cálculo de la distancia de *crowding* en la i-ésima solución del frente corresponde a la media de la longitud de los lados del cuboide (denotado como un cuadro de lineas discontinuas de color azul).

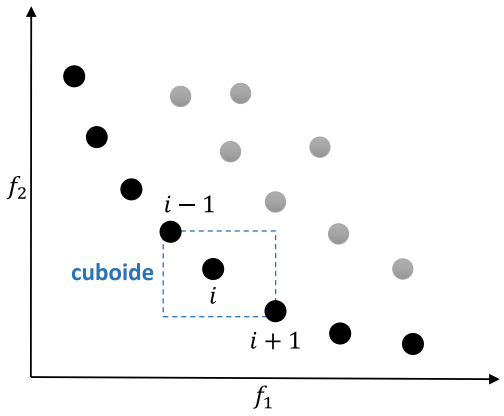


Figura 4.8.: Distancia de *crowding* en un problema con dos objetivos.

Es importante destacar que el cómputo de la distancia de *crowding* requiere que todos los individuos de la población actual sean clasificados y organizados de forma ascendente tomando como referencia al valor en cada función objetivo. Posteriormente, se asigna el valor infinito a aquellas distancias que correspondan a las soluciones de contorno de cada función objetivo. El resto de valores de la función (soluciones intermedias) serán asignadas con una distancia que equivale a la diferencia absoluta normalizada entre dos soluciones contiguas. Todo este proceso habría que realizarlo con las demás funciones objetivo. Finalmente, el cálculo de la distancia de *crowding* total se realiza sumando el valor de las distancias obtenidas para cada uno de los objetivos. En el Algoritmo 2 se describe el procedimiento para el calcular la distancia de *crowding* del conjunto de soluciones no dominadas I.

Algoritmo 2: Asignación de la distancia de *crowding*

```
1  l = I ;
2  para cada i = 0 hacer
3  |   I[i]_distancia = 0 ;
4  fin
5  para cada objetivo m hacer
6  |   I = ordenar(I, m);
7  |   I[1]_distancia = I[i]_distancia = ∞;
8  |   para i = 2 hasta (l − 1) hacer
9  |   |   I[i]_distancia = I[i]_distancia + (I[i+1] · M − I[i−1] · M)/(f_M^max − f_M^min);
10 |   fin
11 fin
```

En el Algoritmo 2, $I[i] \cdot M$ indica el valor de la función objetivo m-ésima del individuo i-ésimo correspondiente al conjunto de soluciones no dominadas I, y los parámetros f_M^{max} y f_M^{min} son los valores máximos y mínimos de la función objetivo m-ésima. La complejidad de este procedimiento viene determinada por el algoritmo de clasificación, dado que son necesarias M ejecuciones independientes del algoritmo de clasificación. Finalmente, la complejidad computacional del algoritmo es de $\mathcal{O}(M \cdot N \cdot \log(N))$.

De esta forma, una vez que todos los individuos de la población del conjunto I cuenten con el valor de distancia correspondiente se procede a realizar la comparación entre pares de soluciones en relación a su grado de proximidad con otras soluciones. Lo que se espera es que la solución que alcance el menor valor de distancia se encuentre más apiñada por otras soluciones. El operador que utiliza el NSGA-II, denotado por $(\prec_n)$, se encarga justo de realizar esta comparación. Además, el NSGA-II asigna dos atributos a cada uno de los individuos de la población i: el i_{rank}, que corresponde al rango no dominado y la $i_{distancia}$, que es la distancia de *crowding* anteriormente comentada. Concretamente, el algoritmo permite establecer el orden parcial $\prec_n$ como se indica en la Ecuación 4.7:

$$i \prec_n j \; ; \quad \text{si } (i_{rank} < j_{rank}) \text{ ó } \Big((i_{rank} = j_{rank}) \text{ y } (i_{distancia} > j_{distancia}) \Big) \tag{4.7}$$

Esta condición establece que cuando dos soluciones tengan distintos rangos de no dominancia se preferirá aquella que tenga el menor rango. En caso contrario, si el par de soluciones corresponden al mismo frente entonces se preferirá la solución que se encuentre en una región de menor densidad poblacional.

4.2.3. Ciclo principal del NSGA-II

Los tres pilares fundamentales con los que se construye el ciclo principal del NSGA-II son: el procedimiento de clasificación rápida no dominada, el operador de asignación de distancia de crowding y el operador de comparación de crowding. En principio, el NSGA-II requiere que se genere una población aleatoria P_0 que posteriormente debe ser ordenada considerando los diferentes frentes dominados y el valor de aptitud de cada una de las soluciones. Las comparaciones para decidir qué individuos sobreviven para la siguiente generación se basan en el concepto de elitismo, el cual es detallado en el Algoritmo 3.

Algoritmo 3: Generación poblacional elitista

$R_t = P_t \cup Q_t$;
$F = Clasificación\_rápida\_no\_dominada(R_t)$;
$P_{t+1} = \emptyset$;
$i = 1$;
mientras $(|P_{t+1}| + |F_i| \leq N)$ **hacer**
    $Asignación\_distancia\_crowding(R_t)$;
    $P_{t+1} = P_{t+1} \cup F_i$;
    $i = i + 1$;
fin
$ordenar(F_i \prec n)$;
$P_{t+1} = P_{t+1}$;
$Q_{t+1} = crear\_nueva\_población(P_{t+1})$;
$t = t + 1$;

El procedimiento de selección de individuos basado en elitismo parte de una población R_t de tamaño $(2 \cdot N)$. Esta población es el resultado de unificar la población correspondiente a la anterior iteración P_t y los descendientes Q_t de dicha población. El elitismo está asegurado durante la ordenación no dominada ya que en R_t están incluidos los miembros anteriores y actuales de la población. El cálculo de la complejidad de un ciclo completo del NSGA-II se lleva a cabo tomando en cuenta las complejidades de cada una de las operaciones básicas que suceden dentro del algoritmo, que son:

- Clasificación rápida no dominada: $\mathcal{O}(M \cdot (2 \cdot N)^2)$.
- Asignación de la distancia de crowding: $\mathcal{O}(2 \cdot N \cdot M \cdot \log(2 \cdot N))$.
- Procedimiento de clasificación en rangos: $(\prec_n)$: $\mathcal{O}(2 \cdot N \cdot \log(2 \cdot N))$.

Teniendo esto en cuenta, la complejidad algorítmica del NSGA-II es $\mathcal{O}(M \cdot N^2)$ al estar condicionada por la clasificación de las soluciones no dominadas. También hay que decir que una población de tamaño $(2 \cdot N)$ no requiere ser ordenada completamente en el caso en que el procedimiento alcance la cantidad de frentes necesarios para obtener N miembros en P_{t+1}. En la Figura 4.9 se muestra el esquema de funcionamiento del NSGA-II desarrollado en [203, 204].

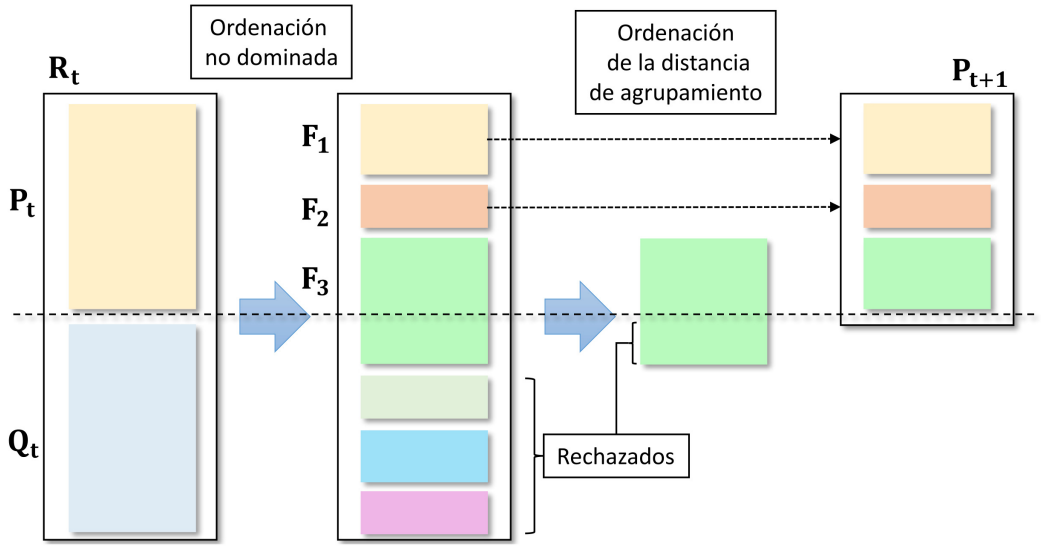


Figura 4.9.: NSGA-II. Esquema de su funcionamiento. Imagen adaptada de [203].

4.3. Métricas de evaluación

Las métricas utilizadas para evaluar los algoritmos de optimización multi-objetivo proporcionan a las soluciones obtenidas una medida de calidad. Al evaluar las soluciones se busca minimizar la distancia entre el frente de Pareto obtenido por el algoritmo y el exacto del problema en cuestión y alcanzar una distribución uniforme del frente de Pareto maximizando la distancia entre las soluciones sobre dicho frente. A lo largo de los años han sido desarrolladas diferentes métricas de calidad basadas en los dos criterios citados anteriormente donde además también se ha incluido como criterio su esfuerzo computacional. A continuación se citan algunas de estas métricas:

- Número de soluciones en el frente óptimo de Pareto.
- Número de ejecuciones de la función objetivo.
- Distancia generacional y generacional inversa [217].
- Dispersión de las soluciones dentro del espacio de los objetivos: hipervolumen [218, 219] y efficient set spacing (ESS) [220].

El hipervolumen surge como una de los métricas de calidad más utilizadas en optimización multi-objetivo. Este indicador de calidad está especialmente diseñado para evaluar en términos de convergencia y diversidad el frente de Pareto de los problemas de minimización. Concretamente, el hipervolumen permite calcular el volumen delimitado por el conjunto Q de soluciones no dominadas dentro del

espacio de objetivos. Formalmente, para cada solución no dominada $i \in Q$ se define el correspondiente hipercubo v_i que tiene a W como punto de referencia. Las coordenadas de este punto serán los valores máximos teóricos que una función objetivo pudiera alcanzar. Teniendo todo esto en cuenta, la expresión matemática para el cálculo del hipervolumen es la siguiente:

$$HV = volumen\left(\bigcup_{i=1}^{|F|} v_i\right) \tag{4.8}$$

Cuanto mayor sea el valor del hipervolumen mayor será la calidad de las soluciones proporcionadas por el algoritmo de optimización. En la Figura 4.10 es posible observar el hipervolumen correspondiente a dos funciones objetivo:

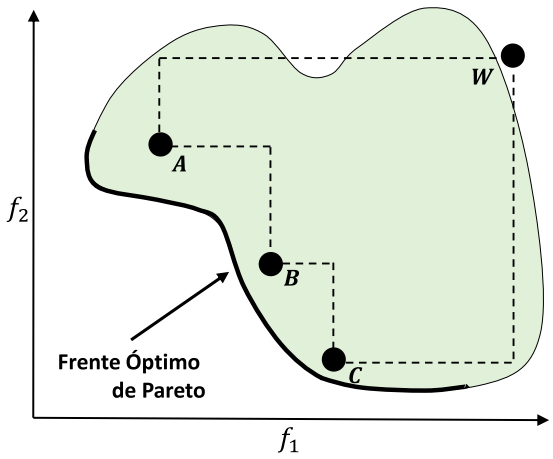


Figura 4.10.: Hipervolumen. Ejemplo para un problema de minimización de dos objetivos.

5 Framework para optimizar arquitecturas de aprendizaje profundo

Diseñar arquitecturas de aprendizaje profundo es una tarea compleja ya que habitualmente suele realizarse de forma manual. Aunque la experiencia del diseñador puede proporcionar una primera aproximación, no existen reglas de diseño estándar ni garantía de que la propuesta proporcione las prestaciones deseadas. Por tanto, el ajuste de la arquitectura y de los hiperparámetros requiere de un proceso de ensayo y error muy costoso en tiempo. El proceso de optimización de las arquitecturas suele estar centrado en las prestaciones del modelo (como por ejemplo la tasa de clasificación) y obvia otros objetivos que pueden ser de interés (coste computacional, velocidad de inferencia, memoria consumida, consumo energético, etc.). Con esto en mente, en este capítulo se propone un framework para automatizar la búsqueda de arquitecturas de aprendizaje profundo óptimas que permitan resolver el problema de interés pero que también tengan en cuenta aspectos como la energía consumida o la complejidad del modelo.

5.1. Propuesta de optimización multi-objetivo basada en computación evolutiva

En esta Tesis se han diseñado e implementado procedimientos para optimizar arquitecturas de ANNs considerando múltiples objetivos de diseño simultáneos e incorporando restricciones que limiten el espacio de búsqueda. Para ello, se utilizan algoritmos de optimización basados en computación evolutiva que tienen un bajo coste computacional pero que permiten alcanzar los mejores resultados en un tiempo razonable. Resumiendo, los objetivos principales que serán abordados por el algoritmo evolutivo son:

- Minimizar la complejidad computacional mediante la reducción de la cantidad de parámetros entrenables de las ANNs.

- Maximizar el resultado en los índices de clasificación de las ANNs.

En la Figura 5.1 se muestra el esquema de los componentes principales del framework propuesto y las interacciones que existen entre los diferentes elementos:

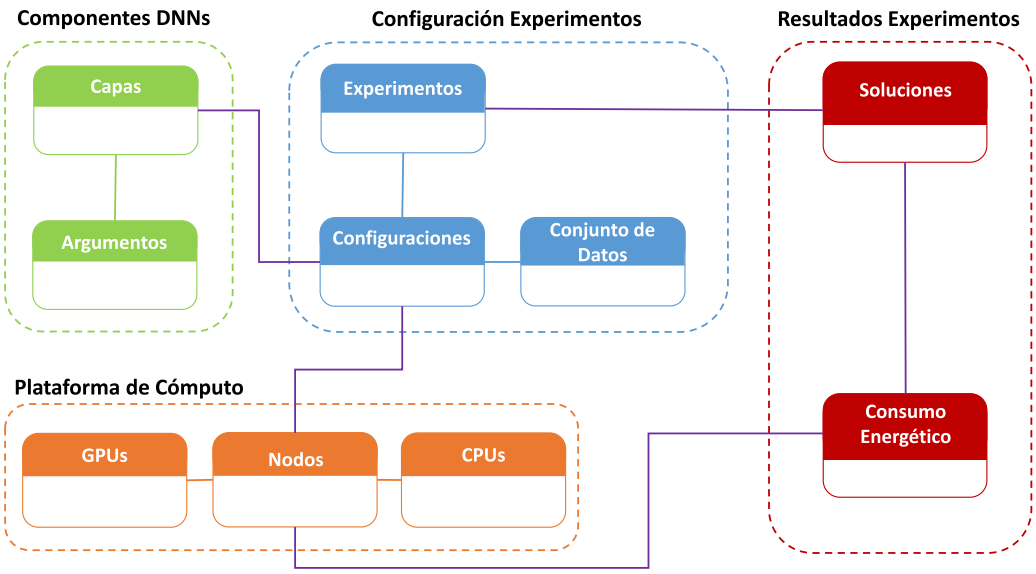


Figura 5.1.: Esquema de componentes. Diseño de los diferentes elementos del framework de optimización.

El procedimiento de optimización desarrollado se ha integrado en un framework capaz de producir ANNs que alcanzan una precisión óptima en tareas de clasificación con la mínima cantidad de parámetros entrenables. Además, el framework tiene la capacidad de administrar el uso de las plataformas heterogéneas de cómputo para la ejecución del experimento propuesto aprovechando el alto grado de paralelismo existente en los computadores actuales. Esto se refiere principalmente al uso de núcleos CPU, procesadores gráficos (GPUs) y unidades de procesamiento tensorial (TPUs). Además, la herramienta generada es totalmente configurable y permite utilizar diferentes objetivos, seleccionar el algoritmo evolutivo a utilizar durante el proceso de optimización e imponer cualquier restricción. Para demostrar el funcionamiento del framework se ha considerado como objetivo maximizar el índice kappa de las tasas de clasificación de la ANNs y minimizar la cantidad de parámetros entrenables. Asimismo, con el fin de aprovechar las arquitecturas de cómputo heterogéneas de los sistemas actuales, el procedimiento de optimización se ha ejecutado en las CPUs mientras que la función de evaluación se ejecutará en las GPUs disponibles (lo que incluye el entrenamiento de las arquitecturas propuestas y el proceso de inferencia).

El procedimiento de optimización comienza generando la población inicial. El cromosoma de un individuo representa a una arquitectura de ANN en particular donde cada gen corresponde a algún componente específico de dicha arquitectura. Estos componentes podrían ser: una capa, su tipo, algún parámetro suyo o el rango de valores que puede tomar un parámetro de la capa. Todos los genes que codifican las soluciones (arquitecturas propuestas) están descritos más adelante.

De entre las principales funciones del framework se encuentra la de generar de forma automática código *Tensorflow* [221] para implementar cada una de las soluciones propuestas. De esta forma, es posible construir arquitecturas de ANNs basadas en el cromosoma suministrado por el algoritmo de optimización. Cada individuo de la población (solución) se evalúa mediante la correspondiente función de evaluación multi-objetiva tras realizar el entrenamiento de la ANN que codifica. Así, el *fitness* de cada individuo estará compuesto por dos valores: la tasa de clasificación de la ANN que dicho individuo codifica y la cantidad de parámetros usados para entrenar la red neuronal. Posteriormente, se aplican los operadores genéticos de cruce, mutación y selección para producir la nueva generación de individuos. Todo este proceso se repetirá hasta alcanzar el criterio de parada o la cantidad máxima de generaciones que haya sido determinada previamente (parámetros configurables en el framework).

Para toda la gestión del framework, su configuración y monitorización del proceso de optimización se utiliza una base de datos de tipo relacional [222] (DBMS) que almacena las tablas y estructuras de datos necesarias para el framework. El DBMS utilizado ha sido Postgresql [223], y tiene especial importancia en las múltiples funcionalidades necesarias para la configuración de los experimentos. También proporciona gran capacidad de almacenamiento y una gestión eficiente en los accesos a los datos mediante el uso del lenguaje de consulta estructurada (SQL) [224]. En las siguientes secciones se procederá a detallar de forma exhaustiva el framework propuesto en esta tesis, sus componentes y la interacciones que se producen entre ellos.

5.2. Funciones principales del framework

En esta sección se describen las principales funciones del framework para optimizar las ANNs: lanzamiento y control de ejecución de procesos, optimización de ANNs basada en computación evolutiva y la generación automática de código fuente óptimo.

5.2.1. Lanzamiento y control de ejecución de procesos

El procedimiento implementado para el lanzamiento y control de la ejecución de procesos de optimización de ANNs ha utilizado el modelo Maestro/Servidor [225]. Este paradigma asigna a un nodo la ejecución del proceso maestro, el cual tendrá control sobre uno o más dispositivos de cómputo heterogéneo que se denominan servidores y que estarán encargados de la ejecución de los experimentos de optimización. En la Figura 5.2 se muestra el esquema de lanzamiento y control de ejecución de procesos. El nodo maestro tienen alojada la base de datos (DB) que almacena los detalles del experimento a ejecutar. Desde el nodo maestro se lanzan los procesos para su ejecución en los nodos disponibles. La disponibilidad de cada uno de los nodos se denota en el esquema con un circulo en verde, mientras que un circulo rojo indica que el nodo se encuentra ocupado. En la figura también se pueden apreciar los dispositivos de cómputo CPU y GPU disponibles en cada nodo. Finalmente, el flujo de datos entre el nodo maestro, los servidores y la DB es visualizado mediante las correspondientes flechas direccionadas.

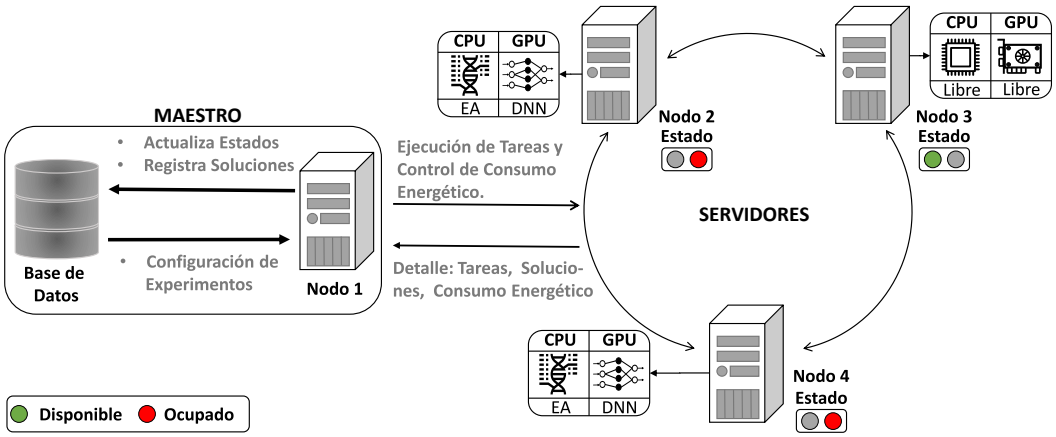


Figura 5.2.: Esquema de lanzamiento y control de ejecución de procesos. Incluye los recursos computacionales, su gestión y la base de datos.

Para conseguir el óptimo funcionamiento del algoritmo de optimización multi-objetivo se realiza lo siguiente: en primer lugar, cuando se inicia el ciclo del procedimiento de optimización, se identifica el nodo computacional de ejecución (maestro), el número de proceso asignado por el sistema operativo, el directorio de ejecución y la dirección de la DB desde donde se extraerán detalles de los recursos computacionales disponibles para el experimento. Posteriormente, se establece la conexión con la DB y se ejecutará en paralelo el algoritmo que permitirá el registro del consumo energético de los nodos computacionales utilizados durante el experimento. En este

punto comienza el ciclo de ejecución continua que permitirá verificar constantemente el estado de utilización de los recursos computacionales y comprobar las tareas pendientes en la cola de ejecución. En caso de que existan recursos disponibles y tareas pendientes de ejecución el nodo maestro ordenará al nodo servidor disponible ejecutar una de ellas. La DB almacena la cantidad de ejecuciones realizadas y tareas pendientes y contiene un atributo que indica si el recurso computacional está siendo utilizado o no. El ciclo de control de tareas pendientes concluye cuando todas las ejecuciones previstas para el experimento de optimización hayan finalizado. Cuando se produce esta condición también se para la ejecución del algoritmo que registra el consumo energético. El pseudocódigo del procedimiento de lanzamiento y control de ejecución de procesos se muestra en el Algoritmo 4:

Algoritmo 4: Pseudocódigo del procedimiento de lanzamiento y control de ejecución de procesos.

```
1  Conexion_DB();
2  Registra_detalles_identificacion_procedimiento;
3  Ejecución_paralela_algoritmo_registro_consumo_energía;
4  mientras (existan tareas de optimización pendientes) hacer
5      dispositivo = Verifica_disponibilidad_dispositivos();
6      tarea = Verifica_tareas_pendientes();
7      si (existe dispositivo disponible y tarea pendiente) entonces
8          Ejecución_tareas_pendientes(dispositivo, tarea);
9          Actualiza_estado_tareas_pendientes(tarea);
10         Actualiza_estado_dispositivos_disponibles(dispositivo);
11     fin
12 fin
```

5.2.2. Estimación del consumo energético

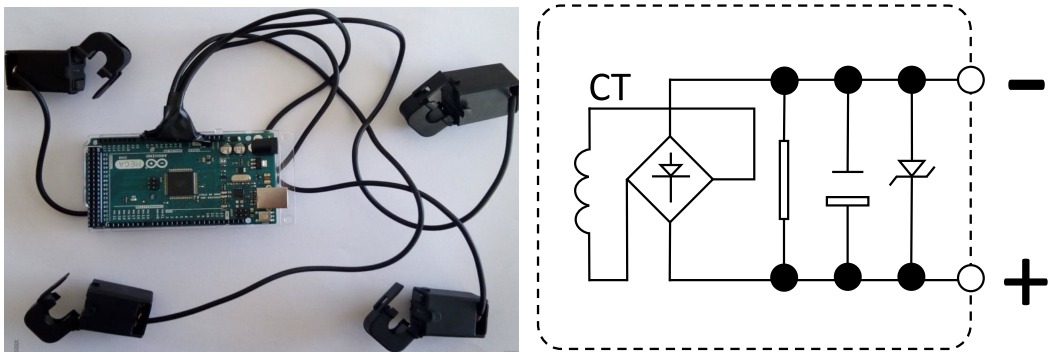
Las mediciones de energía de los nodos se obtienen mediante un vatímetro construido específicamente para los experimentos. Para ello, se ha utilizado un placa *Arduino Mega* [226] que proporciona a cada segundo y para cada nodo de cómputo un valor de potencia instantánea P(W) y otro de energía acumulada, E(W · h). Los valores de energía del switch Gigabit Ethernet no han tenido en cuenta por ser muy inferiores a los de los nodos (por debajo de 5W). Los sensores de medición de la placa *Arduino* obtienen la corriente eléctrica que circula a través del cable que alimenta la plataforma. Al realizar la medición en este punto se puede determinar su consumo real, incluyendo todos los componentes activos así como las pérdidas de conversión de la fuente de alimentación. Entre sus diferentes utilidades, la medición

del consumo de energía permite identificar el perfil energético de una aplicación y realizar las mejoras necesarias en función de las observaciones.

Descripción del *hardware*

El vatímetro se compone de una placa *Arduino Mega* y un conjunto de sensores conectados a las entradas analógicas. El cable de alimentación de cada uno de los equipos cuenta con un sensor diferente y se cierra la pinza para fijarla, detectando la corriente que circula por el cable. Posteriormente, la placa *Arduino* es conectada al puerto USB del nodo para transmitir los datos por el puerto serie creado en la interfaz */dev/ttyACM0*.

El modelo de sensor utilizado se denomina *SCTD010T-5A* y ha sido suministrado por *YHDC Electronics Co., Ltd. Ltd*. El sensor permite medir hasta 5 A y proporciona una tensión de salida de entre 0 y 5 V. Por otra parte, su precisión está en torno al $\pm 2\,\%$ y la temperatura de trabajo soportada se sitúa entre -20 a $+50^{o}$ Celsius. Las señales analógicas emitidas por los sensores son procesadas por la *Arduino* a través de su convertidor interno de 10 bits. Además, para aprovechar mejor su rango dinámico se utiliza la referencia interna de 2.56 V, que sólo está disponible en el modelo *Mega*. Aunque con esto se reduce la medición a una entrada máxima de 2.56 V, en la práctica permite valores de potencia instantánea de hasta 588 W. Dado que las plataformas presentan valores más bajos, la reducción de la tensión máxima no implica limitación alguna para obtener las mediciones. En la Figura 5.3 se puede observar el vatímetro y el diagrama interno de los sensores de corriente:



(a) Placa *Arduino Mega* y sensores de corriente. **(b)** Esquema interno del sensor de corriente.

Figura 5.3.: Vatímetro basado en *Arduino Mega* usado para medir la energía.

Descripción del *software*

El *software* está desarrollado en *Python* y permite que diferentes usuarios puedan obtener mediciones de energía de forma independiente y simultánea. Como la placa *Arduino* envía la información al puerto serie a través del USB, es necesario compartir los datos recibidos entre los diferentes usuarios. Para ello, se utiliza el sistema de mensajes *ZeroMQ*[1] bajo el mecanismo de publicación-suscripción: un proceso maestro llamado *master_meter.py* abre el puerto serie y activa con *ZeroMQ* un proceso de publicación de datos a través del puerto TCP 5214 (en el que se retransmiten los datos recibidos). Cada usuario que quiera recibir datos ejecuta un proceso en *Python* llamado *show_consumption.py* para suscribirse al sistema, de forma que los datos son recibidos desde la *Arduino* en tiempo real. Dado que es posible que varios procesos se conecten al sistema de envío todos recibirán la información simultáneamente. En la Figura 5.4 se puede observar esquema general del proceso de publicación-suscripción:

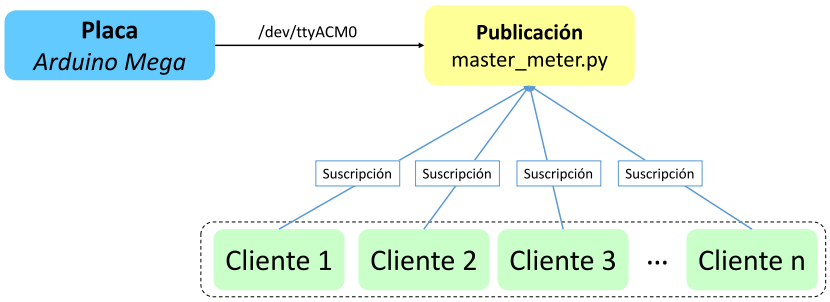


Figura 5.4.: Mecanismo de publicación-suscripción. Esquema de su funcionamiento.

Cada usuario tiene en su directorio $HOME un archivo que contiene el consumo de energía acumulado tras la primera llamada a *show_consumption.py*. En la siguiente llamada, el usuario obtendrá la diferencia entre el valor actual y el almacenado en el fichero. Para el usuario esto es útil porque permite controlar un intervalo de tiempo en concreto (como el inicio y fin de un experimento). Los valores pueden ser reinicializados a cero en cualquier momento para realizar una nueva medición.

5.2.3. Procedimiento de optimización de ANNs

En la Figura 5.5 se observa el diagrama de bloques del framework de optimización propuesto. En ella se puede apreciar los objetivos a optimizar, las plataformas heterogéneas de cómputo sobre las que se ejecutan diferentes procesos, el conjunto

[1]Biblioteca de mensajería asíncrona utilizada en aplicaciones distribuidas o concurrentes

de datos utilizado, el ciclo de optimización de arquitecturas profundas, la evaluación de los individuos y los operadores genéticos que permiten producir una nueva generación de individuos.

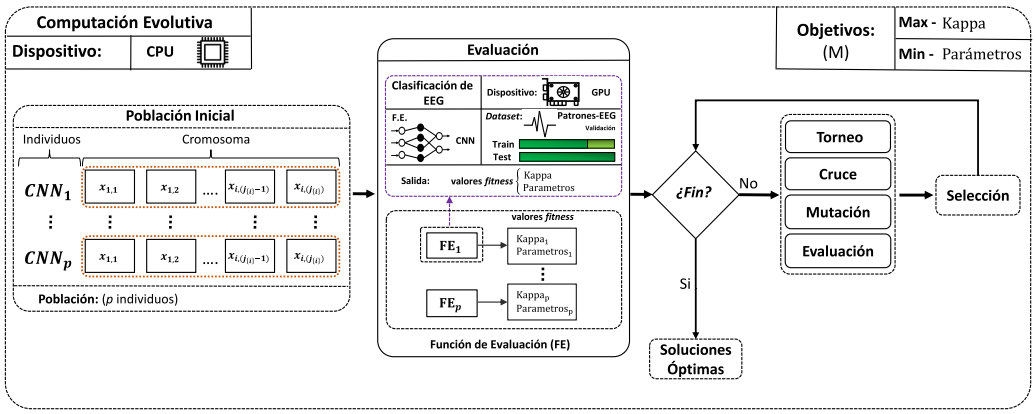


Figura 5.5.: Procedimiento evolutivo multi-objetivo para optimización de arquitecturas profundas. $x_{ij_{\{i\}}}$ indica el j-ésimo gen del cromosoma que ha sido utilizado para codificar los parámetros de la i-ésima capa.

Configuraciones previas

En primer lugar, el procedimiento de optimización recibe como parámetros de entrada los valores que han sido enviados desde el nodo maestro, el cual administra la ejecución de las tareas pendientes. De entre los parámetros recibidos se encuentra el código identificador del experimento en ejecución. Posteriormente, se establece la conexión con la DB y utilizando el código identificador del experimento se extraen detalles más exhaustivos sobre su configuración. Acto seguido se crean dos tablas de forma automática en la DB: en la primera se registran los cromosomas, el valor de sus respectivos genes y los resultados obtenidos por las ANNs codificadas por dichos cromosoma. La segunda tabla es utilizada para guardar el código fuente de (*Tensorflow*) generado automáticamente tomando como parámetro los genes del correspondiente cromosoma. Por otro lado, se crea un archivo de texto plano para registrar de forma detallada la ejecución de las diferentes soluciones propuestas. Finalmente, se crea un directorio especifico para el experimento en cuestión que almacena los pesos finales de cada una de las redes neuronales optimizadas.

Utilizando el código del experimento se puede extraer de la DB la configuración de las capas y atributos que serán objeto de optimización así como el nombre y ubicación de los conjuntos de entrenamiento y evaluación necesarios. Además, antes de comenzar el algoritmo evolutivo hay que indicar el valor de algunos

parámetros: la cantidad de objetivos, las variables de decisión, las probabilidades de cruce y mutación, el tamaño de la población y la cantidad de generaciones. Todos estos valores son obtenidos también de la DB. Por último, se declara la función de evaluación, la cual consiste en entrenar cada ANN en la GPU que haya sido asignada al experimento. La función tiene por objetivo minimizar la cantidad de parámetros de la ANN y maximizar el índice kappa resultante del proceso de clasificación.

Ejecución del procedimiento de optimización

Una vez configurado el programa se pone en marcha el algoritmo de optimización e inmediatamente se registra la fecha y hora de inicio. Al principio se genera una población aleatoria de individuos con sus respectivos cromosomas y genes. Luego, se procede a evaluar cada uno los individuos, con este propósito se debe verificar si el cromosoma propuesto por el EA permite la construcción de una ANN. La Figura 4.4 muestra el proceso de generación de la arquitectura de una ANN a partir el cromosoma y sus respectivos genes. Además, se indica en cada una de las capas los atributos que corresponden a cada gen.

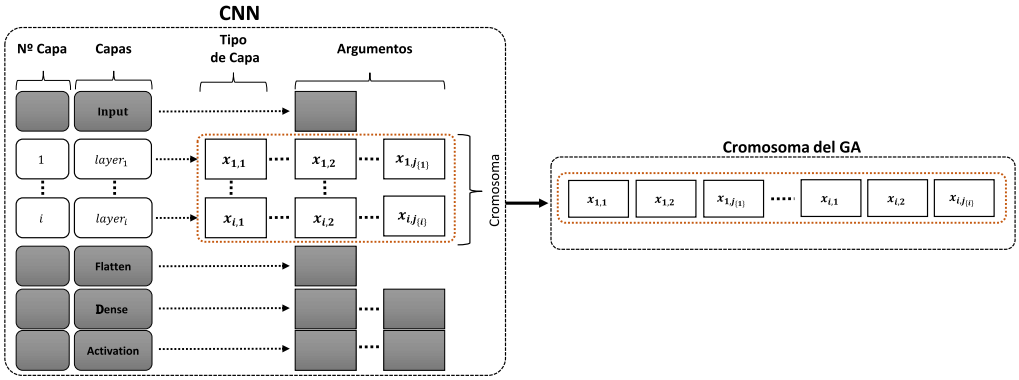


Figura 5.6.: Codificación del cromosoma. Soluciones del algoritmo de optimización evolutivo y valores utilizados en la arquitectura de ANN de interés.

En caso de que no haya sido posible generar una ANN a partir del cromosoma del individuo evaluado se procede a su registro en la DB asignando los valores $kappa = -1$ y $parametros\_ann = \infty$. Con esto se consigue que los valores de evaluación de la solución sean muy desfavorables como para ser considerados en las siguientes etapas del EA. Por el contrario, si ha sido posible generar la ANN a partir del cromosoma del individuo evaluado se verifica el número resultante de parámetros entrenables: si el número excede la cantidad máxima establecida para el experimento en cuestión entonces la ANN no se entrenará y automáticamente

le serán asignados los valores de $kappa = -1$ y $parametros\_ann = \infty$. Esto quiere decir que sólo se entrenará la red neuronal cuando sea posible generar la ANN y la cantidad de parámetros no exceda la cantidad máxima establecida. Para entrenar la ANN, el conjunto de datos de entrenamiento se divide en un 90 % para el proceso de entrenamiento y 10 % para la validación. Tras concluir el entrenamiento se evalúa el modelo generado con el conjunto de datos de validación para obtener el valor del índice kappa que determina el desempeño del modelo generado para la tarea de clasificación. Finalmente, se registra en la DB la fecha y hora de la finalización del entrenamiento y los resultados de evaluar la ANN. El pseudocódigo para generar las ANNs se encuentra descrito en el Algoritmo 5. En él se detalla el proceso utilizado para ejecutar la función evaluación, la cual se encarga de entrenar la ANN y posteriormente retornar los correspondientes los valores del *fitness*.

Algoritmo 5: Evaluación de la función de aptitud (*fitness*).

Entrada: cromosoma.
Salida: kappa, parametros_ann
$array = cromosoma[x_{(1,1)}, \ldots, x_{(i,2)}, x_{(i,j_{(i)})}]$;
$datos\_entrenamiento = archivo\_conjunto\_train$;
$datos\_prueba = archivo\_conjunto\_eval$;
$genera\_ann = Generación\_ann(array)$;
si $(genera\_ann == Falso)$ **entonces**
    $kappa = -1$;
    $parametros\_ann = \infty$;
si no, si $(genera\_ann == Verdadero)$ **entonces**
    $parametros\_ann = Verifica\_cantidad\_parámetros\_ann$;
    si $(parametros\_ann > max\_parametros)$ **entonces**
        $kappa = -1$;
        $parametros\_ann = \infty$;
    si no, si $(parametros\_ann \leq max\_parametros)$ **entonces**
        $[modelo, parametros\_ann] = Entrenamiento\_ann(datos\_entrenamiento)$;
        $kappa = Evaluación\_ann(modelo, datos\_prueba)$;
    fin
fin
Resultado: $[kappa, parametros\_ann]$;

La evolución de las soluciones se lleva a cabo mediante la aplicación de los operadores de selección, cruce y mutación. Este proceso se repite hasta alcanzar la cantidad máxima de generaciones (obtenida de la DB). Al finalizar, se obtiene la población resultante, se extraen las soluciones no dominadas del frente óptimo de Pareto y se guardan debidamente en la DB. Del conjunto de soluciones no dominadas que corresponden al experimento en cuestión se selecciona aquella que se encuentra

en el codo del frente óptimo de Pareto ya que en dicho punto se compensan ambos objetivos (índice Kappa y número de parámetros de la ANN). Para finalizar, se procede a actualizar en la DB el estado de la GPU y el nodo para liberar dichos recursos y se procede a cerrar la conexión a la base de datos. El pseudocódigo del framework se muestra en el Algoritmo 6:

Algoritmo 6: Framework de optimización multi-objetivo de ANNs.

```
Conexion_DB();
Consulta_configuración_experimento_DB();
Creación_tablas_registro_detalles();
Generacion_población_inicial_aleatoria;
Evaluación_función_aptitud;
Ordenamiento_población_no_dominada;
mientras (no se alcance el criterio de parada) hacer
    Seleccion_padres;
    Cruce;
    Mutación;
    Evaluación_función_aptitud;
    Ordenamiento_población_no_dominada;
    Estrategia_elitista;
    Nueva_población;
fin
Población_final;
```

Resultado: Soluciones óptimas finales;

5.2.4. Base de datos de configuración y seguimiento

El diseño de la DB relacional que incorpora el framework utiliza varios elementos para modelar la estructura de datos. A continuación se detallan brevemente algunos de los componentes utilizados:

- **Tablas**. Estructuras que son definidas para registrar información relevante de un objeto. Por ejemplo, el objeto que almacena los diferentes tipos de capas que pueden utilizar las ANNs corresponderá a la tabla "Capas". Los datos registrados se organizan mediante filas y columnas.
- **Atributos**. Características o propiedades de cada una de las tablas. Los elementos que se registran en las tablas cuentan con los mismos atributos y adquieren un valor único para cada elemento. Por ejemplo, en la tabla "GPUs" se incluyen las características que la definen: identificador, nodo computacional donde se encuentra instalada, nombre, capacidad, etc.

- **Relaciones**. Determinan las asociaciones existentes entre las diferentes tablas. Específicamente, el vínculo o relación entre las tablas permite definir las dependencias existentes entre varias tablas. En consecuencia, algunas tablas deberán compartir determinados atributos. Las claves son las encargadas de establecer las relaciones entre las tablas y existen dos tipos:

    1. **Clave primaria (PK)**. Atributo de la tabla que identifica de forma única a un determinado registro y que además permite establecer relaciones entre tablas.

    2. **Clave foránea (FK)**. Atributo de la tabla que permite establecer relaciones entre los datos de un registro de la tabla con los de otra tabla. También es posible relacionar registros distintos dentro de la misma tabla.

- **Cardinalidad**. Especifica la cantidad de elementos o instancias que tienen vínculo con dos tablas distintas. Es decir, determina el tipo de relación que tiene una instancia de la tabla A con una instancia de la tabla B y viceversa. Existen tres tipos de cardinalidades:

    1. **Uno a uno (1:1)**. Cuando el elemento que relaciona a ambas tablas aparece una única vez en cada una de las tablas.

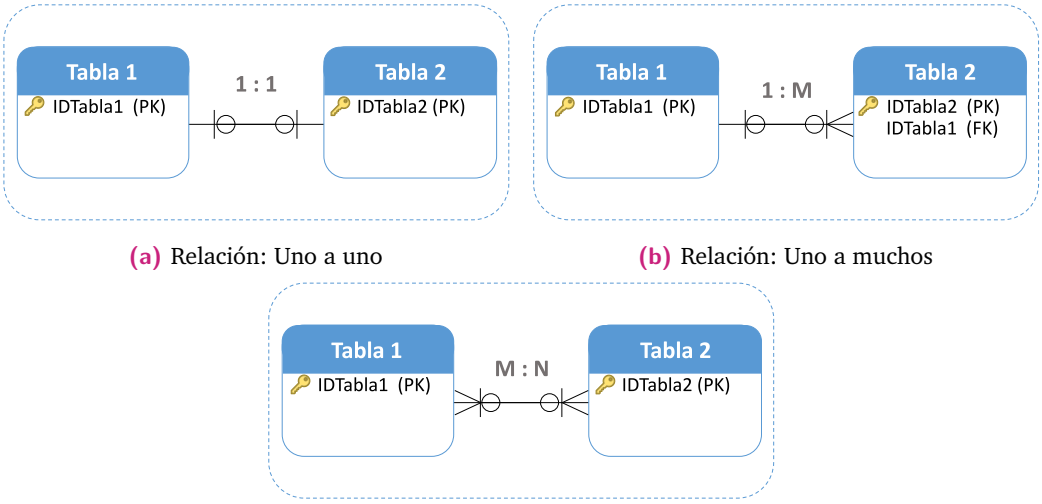
    2. **Uno a muchos (1:M)**. Cuando el registro de una de las tablas puede estar asociado a uno o más registros de la otra tabla.

    3. **Muchos a muchos (N:M)**. Cuando existen múltiples registros de una tabla que se encuentran asociados a múltiples registros de la otra tabla.

En la Figura 5.7 se puede ver los distintos elementos utilizados para el diseño de una DB. Entre ellos se encuentran las tablas, los diferentes tipos de relaciones que existen entre ellas y la nomenclatura utilizada para expresar cada relación y su cardinalidad. A continuación se detalla cuáles son las funciones principales de la DB:

1. Guardar los componentes de las ANNs: los distintos tipos de capas y el conjunto global de parámetros correspondientes a cada tipo de capa. Para ello se ha utilizado la nomenclatura de componentes que utiliza *Tensorflow*.

2. Guardar información de los componentes hardware: nodos, CPUs y GPUs que estarán disponibles durante el transcurso del proceso de optimización. Además, se registra el estado de uso de cada uno de los componentes durante la ejecución del procedimiento de optimización.



(a) Relación: Uno a uno

(b) Relación: Uno a muchos

(c) Relación: Muchos a muchos

Figura 5.7.: Elementos de la DB. Diagrama que indica los diferentes tipos de relaciones entre tablas y la nomenclatura utilizada.

3. Registrar la arquitectura de la ANN a optimizar. Aquí se incluyen la cantidad de capas que la forman, los tipos de capas a utilizar, el orden de las capas y el rango determinado de valores que pueden asumir cada uno de los parámetros del modelo durante el proceso de optimización.

4. Registrar las restricciones relacionadas con la cantidad de capas. Estas restricciones incluyen el tipo de capa, parámetros de cada una de las capas y el rango de valores que podrán asumir cada uno de los parámetros.

5. Guardar los parámetros de configuración de las ANNs que hayan sido evaluadas por el procedimiento de optimización.

6. Generar automáticamente el código para construir y compilar los modelos de arquitecturas ANN en base a los genes de los cromosomas de cada uno de los individuos.

7. Registrar el *fitness* de la soluciones alcanzadas por el procedimiento de optimización y el tiempo de ejecución, nodo y la correspondiente GPU en la que ha sido ejecutada la función de evaluación. También se registra el consumo energético de cada entrenamiento.

En la Figura 5.8 se muestra el modelado lógico de la DB diseñado específicamente para el framework de optimización de ANNs y las estructuras encargadas de almacenar los datos.

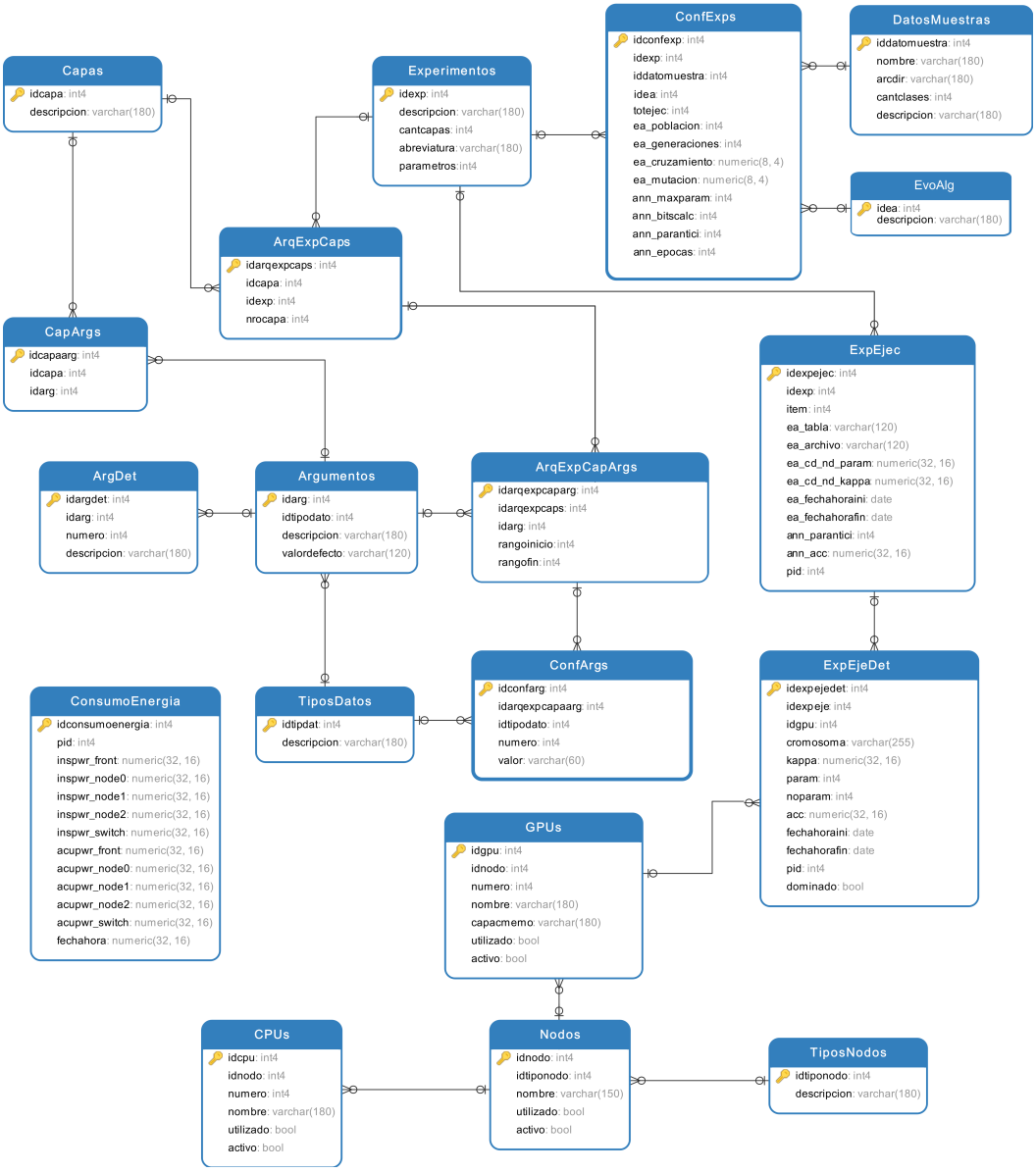


Figura 5.8.: Diseño de la DB utilizada en el framework propuesto. Esquema que incluye todas las tablas y sus relaciones.

Modelado de datos para la gestión de recursos computacionales

En esta sección se describe la gestión de los recursos computacionales y se indica el procedimiento necesario para registrar los datos de los dispositivos de cómputo que ejecutarán los experimentos. Además, se incluyen detalles sobre las funciones encargadas de controlar el estado de los diferentes componentes de la plataforma computacional. La Figura 5.9 ilustra las tablas y relaciones de la DB que brindan soporte a las funciones que gestionan de forma eficiente los recursos de cómputo.

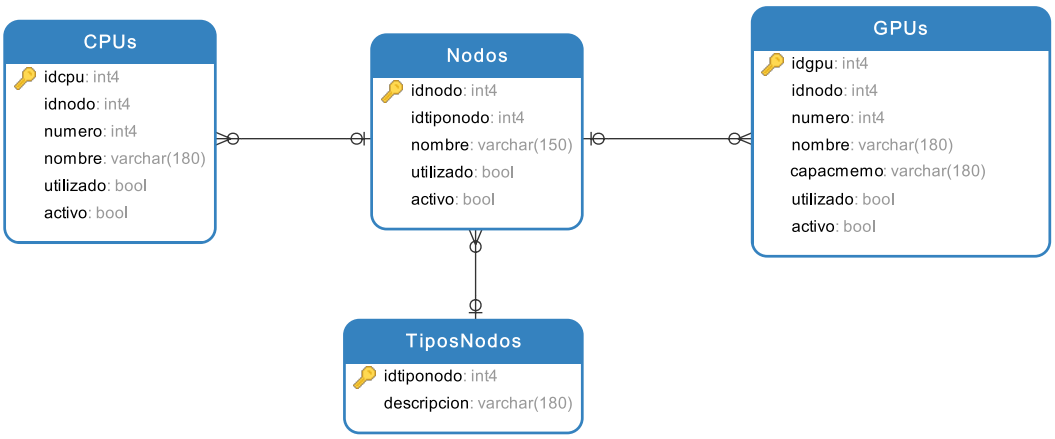


Figura 5.9.: Plataformas de cómputo. Estructura de las tablas que registran los recursos computacionales disponibles para los experimentos.

A continuación se comenta brevemente el contenido de cada una de las tablas:

- **TiposNodos**. Registra si un nodo computacional va a encargarse de gestionar los procesos de optimización (maestro) o si por el contrario el nodo estará dedicado a optimizar ANNs. En la Tabla A.1 se pueden consultar más detalles.
- **Nodos**. Registra la disponibilidad de los nodos de cómputo. En la Tabla A.2 se pueden consultar más detalles.
- **GPUs**. Guarda información de la GPU y si ésta estará disponible para optimizar ANNs. En la Tabla A.3 se pueden consultar más detalles.
- **CPUs**. Guarda información de la CPU y si ésta estará disponible para optimizar ANNs. En la Tabla A.4 se pueden consultar más detalles.

Finalmente, una vez registrados correctamente los datos de la plataforma de cómputo disponible, el framework contará con la información necesaria para que la funciones implementadas sean capaces de controlar el uso de los recursos durante la ejecución de los experimentos.

Modelado de datos para la estructura de componentes de las ANNs

A continuación se detalla el diseño de la DB en cuanto a la estructura de componentes que forman una ANN y el procedimiento a realizar para registrar los parámetros de configuración de las ANNs, los cuales están basados en *Tensorflow*. La Figura 5.10 muestra las tablas y relaciones del modelo de DB utilizadas para construir el código fuente *Tensorflow* que permite generar las ANNs.

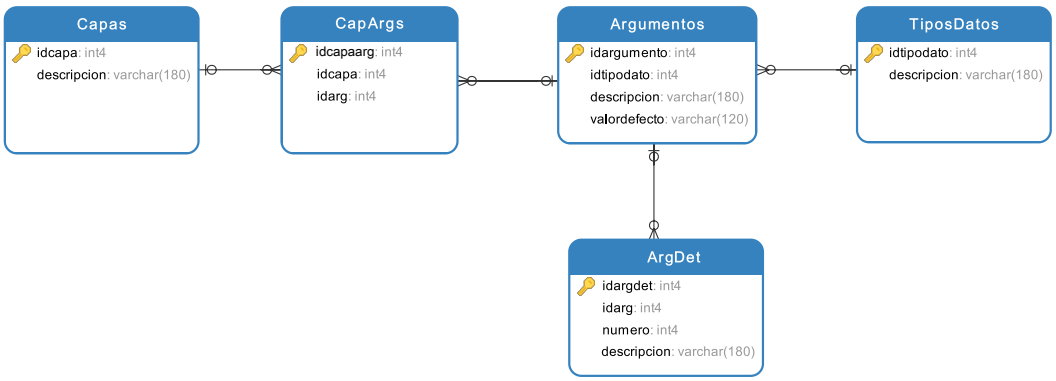


Figura 5.10.: Componentes de las ANNs. Diagrama que detalla la estructura de la DB utilizada para almacenar el código *Tensorflow* de los diferentes elementos de las ANNs.

A continuación se comenta brevemente el contenido de cada una de las tablas:

- **Capas**. Registra los tipos de capas soportadas por la librería *Tensorflow*. Algunos ejemplos de capas son la *Conv2D*, *DepthwiseConv2D* o *SeparableConv2D*. Los atributos de esta tabla se detallan en la Tabla A.5.
- **Argumentos**. Registra los datos del conjunto global de argumentos de los diferentes tipos de capas. Los atributos de esta tabla se detallan en la Tabla A.6.
- **CapArgs**. Guarda los argumentos que corresponden a una capa específica. Por ejemplo, algunos argumentos de la capa *Conv2D* son: *filters*, *kernel_size*, *strides* y *padding*. Los atributos de esta tabla se detallan en la Tabla A.7.
- **ArgDet**. Almacena el rango de valores que pueden ser asignados al argumento de una capa. Los atributos de esta tabla se detallan en la Tabla A.8.

Una vez registrados los datos correspondientes a los elementos que construyen las ANNs el framework contará con la información necesaria para que las funciones implementadas tengan la capacidad de generar automáticamente el código fuente.

Modelado de datos para la configuración de experimentos

En esta sección se describe el diseño de la DB necesario para los experimentos de optimización de ANNs. También se especifican las indicaciones que deben llevarse a cabo para registrar los datos de los parámetros y restricciones que permiten definir los requisitos y el alcance del experimento. La Figura 5.11 ilustra las entidades y las relaciones entre las tablas involucradas:

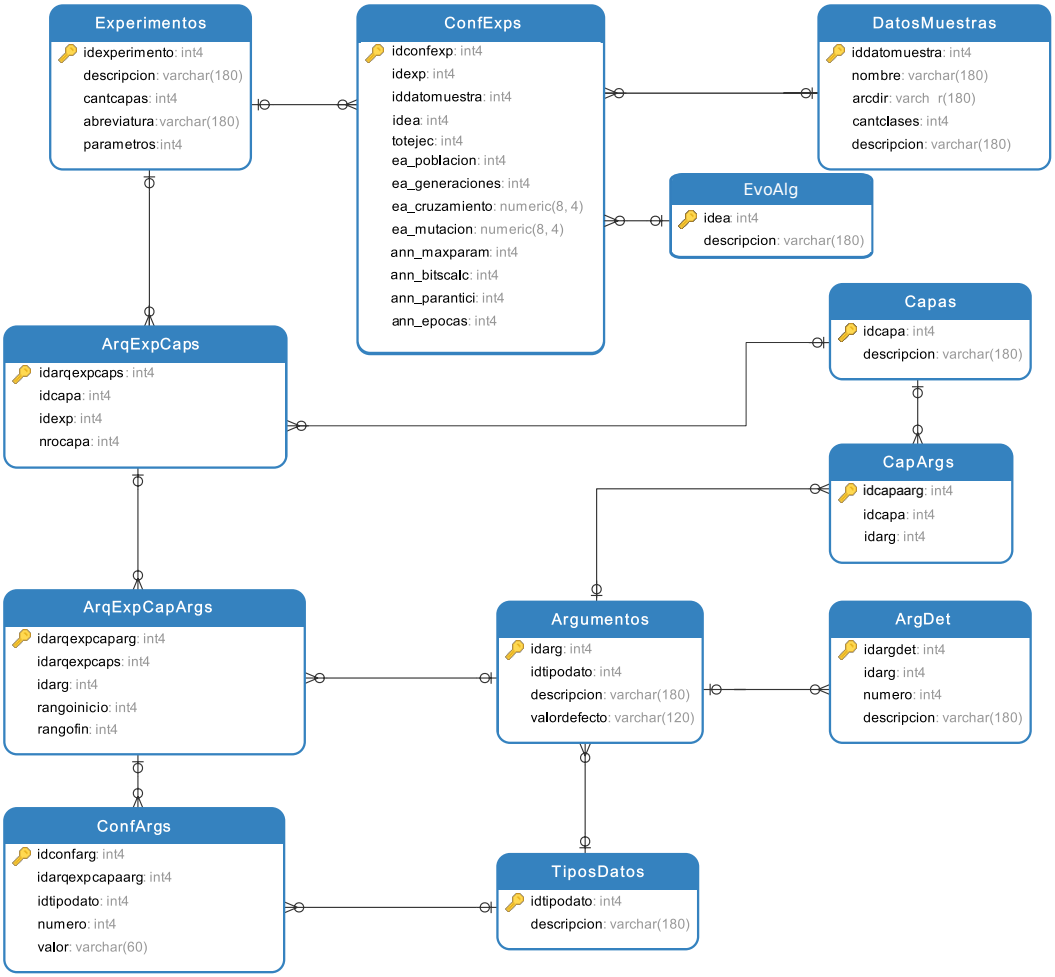


Figura 5.11.: Configuración de experimentos. Diagrama que detalla la estructura de la DB y las tablas que son utilizadas para especificar los experimentos y su alcance.

A continuación se comenta brevemente el significado de cada una de las tablas:

- **DatosMuestras**. Registra información referente al conjunto de datos que será utilizado para el entrenamiento y validación de las ANNs. Los atributos de esta tabla y sus valores se detallan en la Tabla A.9.
- **TiposDatos**. Registra los tipos de datos involucrados en el procedimiento de optimización. Los atributos de esta tabla y sus valores se detallan en la Tabla A.10.
- **EvoAlg**. Registra los EAs que pueden ser configurados por el procedimiento de optimización. Los atributos de esta tabla y sus valores se detallan en la Tabla A.11.

- **Experimentos**. Registra los diferentes tipos de experimentos que serán ejecutados. Los atributos de esta tabla y sus valores se detallan en la Tabla A.12.
- **ConfExp**. Registra la configuración de un experimento en cuestión. Los atributos de esta tabla y sus valores se detallan en la Tabla A.13.
- **ArqExpCaps**. Registra las capas de la ANN que serán optimizadas durante el experimento. Permite también indicar los diferentes tipos de capa que pueden ser seleccionados por el algoritmo de optimización y su posición específica. Los atributos de esta tabla y sus valores se detallan en la Tabla A.14.
- **ArqExpCapArgs**. Registra el rango de posibles valores para el gen del cromosoma del argumento de la capa seleccionada. Los atributos de esta tabla y sus valores se detallan en la Tabla A.15.
- **ConfArgs**. Registra el valor del argumento para la capa seleccionada que será utilizado en la ANN. Los atributos de esta tabla y sus valores se detallan en la Tabla A.16.

Modelado de datos para registrar los resultados experimentales

Para registrar los resultados experimentales y algunos detalles de las diferentes arquitecturas ANN evaluadas también se han creado las tablas correspondientes. Además de los resultados también se almacena el consumo energético de los dispositivos CPU y GPU durante la ejecución del experimento. En la Figura 5.12 se puede ver el modelo de tablas creado para este fin:

A continuación se describe la estructura de las tablas, sus atributos y relaciones:

- **ExpEjec**. Registra datos referentes a los experimentos y las diferentes ejecuciones que han sido realizadas por el procedimiento de optimización. Concretamente, en cada registro se almacenan los mejores resultados de cada ejecución, la cantidad de parámetros, índice de clasificación y detalles del modelo de ANN propuesto. Los atributos de esta tabla y sus valores se detallan en la Tabla A.17.
- **ExpEjeDet**. Registra información detallada sobre los cromosomas de todas las soluciones propuestas por el algoritmo de optimización y los resultados de evaluar cada uno de los individuos. También indica si la solución pertenece al frente óptimo de Pareto o no. Los atributos de esta tabla y sus valores se detallan en la Tabla A.18.

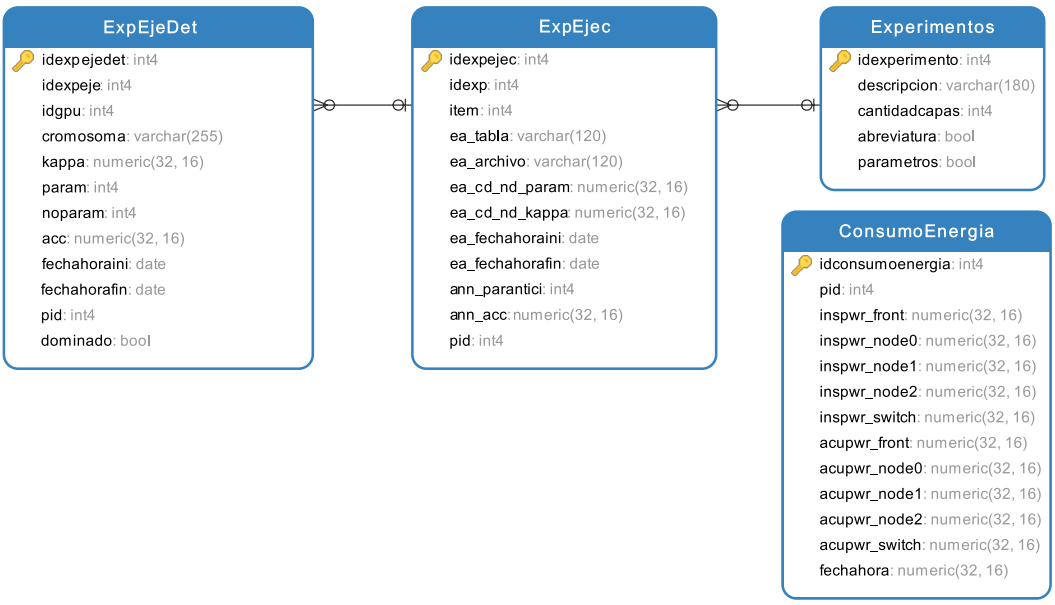


Figura 5.12.: Estructura de la DB utilizada para registrar los resultados experimentales.

- **ConsumoEnergia**. Registra el consumo energético de cada una de las soluciones evaluadas. Los atributos de esta tabla y sus valores se detallan en la Tabla A.19.

Resultados experimentales

6

En este capítulo se presentan los resultados que se han obtenido mediante dos aproximaciones diferentes: la primera, utilizando técnicas de aprendizaje automático para encontrar patrones a partir de características estadísticas extraídas de las señales en el dominio de la frecuencia. Estas características han sido escogidas a priori basándose en el hecho de que es habitual en neurofisiología utilizar características del espectro de potencia de la señal en cada banda (alfa, beta, gamma, delta y theta) para diferenciar diferentes estados neurológicos o diferentes niveles de activación cortical. Así, se ha diseñado un método de detección de anomalías basado en descriptores espectrales para clasificar los sujetos en dos clases en base a las señales EEG adquiridas de cada uno. En este caso, se ha utilizado una base de datos que contiene adquisiciones EEG de 32 canales de sujetos controles y disléxicos. En un segundo experimento, se ha utilizado un conjunto de datos de EEG/BCI que contiene 3 movimientos imaginados diferentes. En este problema de clasificación multi-clase no se ha realizado ninguna elección a priori de los descriptores que pueden resultar más discriminantes, sino que se ha utilizado el framework propuesto en el Capítulo 5 para hacer que una red neuronal sea capaz de extraer y seleccionar descriptores por sí misma y realizar la clasificación. Para ello, se ha tomado como punto de partida una red neuronal con capas convolucionales unidimensionales denominada *EEGNet*, utilizando tres conjuntos de datos que corresponden a tres sujetos diferentes. En las siguientes secciones se detallan los conjuntos de datos utilizados, los experimentos llevados a cabo, la metodología seguida y los resultados obtenidos.

6.1. Clasificación de señales EEG mediante descriptores espectrales

En este primer experimento se utilizan descriptores espectrales extraídos de las señales EEG para caracterizar el espectro en las diferentes bandas (alfa, beta, gamma, delta y theta). Estos descriptores, desarrollados en el Capítulo 2, son en realidad características genéricas no calculadas para tener un especial poder discriminante en este experimento. Concretamente, se utilizan señales EEG para la búsqueda

de patrones de activación cerebral que guarden relación con la dislexia (DD) y que puedan ser de utilidad en el diagnóstico diferencial [77] de la misma. La principal limitación de esta aproximación es que no podemos saber a priori si el conjunto de descriptores elegido va a permitir encontrar patrones con suficiente poder discriminante.

6.1.1. Conjunto de datos

Las señales EEG utilizadas durante los experimentos han sido proporcionadas por el grupo de investigación Leeduca de la Universidad de Málaga [227]. El procedimiento para establecer los grupos de control y experimental ha sido realizado de acuerdo a los criterios y normas estándar utilizados previamente en estudios similares y los Servicios Escolares de Educación Especial (SESS). El procedimiento de selección y adquisición cuenta con las garantías específicas del estudio longitudinal desarrollado y aplicado por el Proyecto Leeduca, el cual ha puesto en marcha un sistema de respuesta a la intervención (RtI) que ha sido aplicado durante 20 años en los Estados Unidos y 10 años en Finlandia y se basa en aplicar evaluación dinámica a grandes muestras de población tres veces al año, desde los 4 hasta los 8 años. Concretamente, los grupos de control y experimentación proceden de una cohorte ($N = 700$). El SESS dispone de una evaluación dinámica longitudinal de los sujetos más un cuestionario de autoinforme sobre dificultades de lectoescritura para adultos. También disponen de un censo oficial para otros tipos de trastornos de neurodesarrollo como los trastornos del lenguaje (LI), del habla (SSD), autismo, déficit de atención e hiperactividad (TDAH) y otros déficits sensoriales auditivos o visuales.

La dislexia es un trastorno del aprendizaje que suele caracterizarse por un déficit de conciencia fonológica [228]. Modelos recientes de la codificación neuronal del habla sugieren que la dislexia se origina por problemas de sincronía entre diferentes regiones del cerebro con los ritmos de muestreo de las sílabas (banda theta, 4-8 Hz) o de fonemas (banda gamma, 12-40 Hz). Los estímulos utilizados en este experimento están basados en esta teoría, y por lo tanto, consisten en ruido blanco modulado en amplitud a diferentes frecuencias (2 Hz, 4 Hz y 20 Hz). De esta forma, las señales EEG se registraron a una frecuencia de muestreo de 500 Hz durante sesiones de 15 minutos mientras los participantes recibían estímulos auditivos utilizando auriculares. El equipo que ha sido utilizado para las adquisiciones de EEG es el Brainvision actiCHamp Plus con un amplificador de 32 canales que permite una frecuencia de muestreo de hasta 100 kHz. Además, se utilizaron electrodos activos (actiCAP, Brain Products GmbH y Alemania) ya que permitían un mayor rango de

impedancias, mejorando la SNR y haciendo que el sistema sea más robusto frente a los movimientos. Por otra parte, el equipo se alimentó con baterías para garantizar el aislamiento con la línea eléctrica y para reducir el ruido de adquisición. Las señales EEG obtenidas se preprocesaron para eliminar artefactos relacionados con el parpadeo de los ojos y las variaciones de impedancia debidas a los movimientos. Dado que la señal de parpadeo se registra junto con las señales de EEG a través de un electrodo específico, estos artefactos se eliminan mediante separación ciega de fuentes utilizando el análisis de componentes independientes (ICA). Otros artefactos fueron eliminados manualmente por un neurofisiólogo experto.

Los participantes que tomaron parte de este estudio son en total cuarenta y ocho personas, entre las que se incluyen 32 lectores expertos (17 hombres) y 16 lectores con DD (siete varones) emparejados en edad $[t(1) = -1,4, p > 0,05,$ rango de edad: $88 - 100$ meses$]$. La media de edad del grupo de control era de $94,1 \pm 3,3$ meses y de $95,6 \pm 2,9$ meses para el grupo DD. Además, todos los participantes eran diestros, hablantes nativos de español sin problemas de audición y con una visión normal o corregida. Los niños disléxicos de este estudio recibieron el diagnóstico clínico formal.

6.1.2. Extracción de descriptores

Los descriptores que se han extraído para la clasificación de las señales están basados en sus características espectrales. Así, el primer paso consiste en estimar la densidad espectral de potencia (PSD). Como se ha explicado en el Capítulo 2, ésta suele calcularse mediante la transformada de Fourier. Sin embargo, la fiabilidad de la PSD calculada por este método se ve reducida por: (i) la alta varianza de la estimación, que hace el espectro ruidoso y (ii) el sesgo creado por la fuga de energía a través de las frecuencias. Por tanto, tal y como se expone en el Capítulo 2, se utilizan ventanas (también llamadas *tappers*) en el dominio del tiempo, reduciendo la fuga producida por los múltiples lóbulos laterales de una ventana en el dominio de la frecuencia. La energía total de estos *tappers* se normaliza para mantener la energía total invariable. Este enfoque puede ampliarse para reducir la varianza de la estimación en cada frecuencia utilizando múltiples *tappers*. Una vez calculada la PSD con el citado método (*multitapper*) [77, 94, 99] se extraen dos descriptores que caracterizan el espectro en cada banda y para cada electrodo. La primera característica es el centroide espectral, (SC), que indica la ubicación del centro de masa del espectro (es decir, la frecuencia donde se concentra la PSD). El segundo descriptor es la PSD media por banda. De esta forma, se compone el vector de características siguiente para las bandas alfa, beta, gamma, delta y theta:

$$f_l = (SC_l^{\Delta}, PSD_l^{\Delta}, SC_l^{\theta}, PSD_l^{\theta}, SC_l^{\alpha}, PSD_l^{\alpha}, SC_l^{\beta}, PSD_l^{\beta}, SC_l^{\gamma}, PSD_l^{\gamma}) \quad (6.1)$$

6.1.3. Selección de descriptores

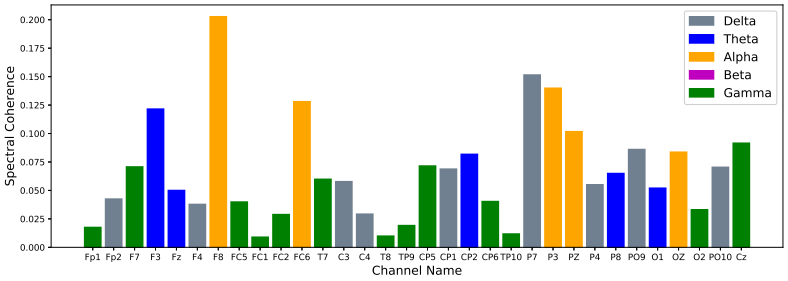
La selección de características se ha realizado utilizando la coherencia espectral entre los mismos canales de sujetos controles y con dislexia. La coherencia espectral es un estadístico con múltiples aplicaciones en neurociencia [229] que mide la relación entre las señales adquiridas de dos electrodos $x(t)$ y $y(t)$:

$$C_{xy} = \frac{|C_{xy}|^2}{C_{xx} \cdot C_{yy}} \quad (6.2)$$

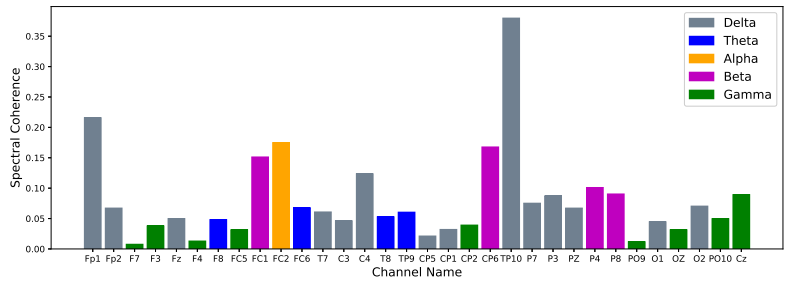
donde C_{xx} y C_{yy} son las densidades espectrales de potencia de las señales x e y, respectivamente, y C_{xy} es la densidad espectral cruzada, que puede calcularse como el espectro de potencia de la función de correlación cruzada entre las señales x e y. Tal y como se muestra en la Figura 6.1, distintos electrodos presentan diferentes valores de coherencia en función de la banda de frecuencias que se analiza. Esto indica que las señales adquiridas por diferentes electrodos contienen información relativa a diferentes bandas. Por tanto, la selección de electrodos puede realizarse manteniendo los electrodos que presentan la menor coherencia cuando se comparan los sujetos CN con los DD. La Figura 6.1 muestra la coherencia sólo para las bandas que presentan los valores más bajos de coherencia espectral.

6.1.4. Clasificación mediante detección de anomalías

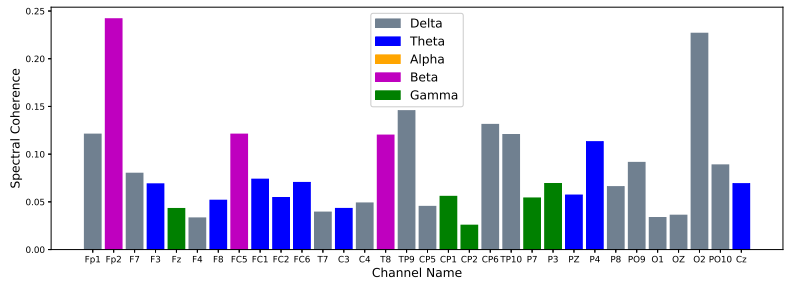
Esta sección presenta los resultados experimentales obtenidos al clasificar los sujetos mediante las características extraídas de las señales EEG. Para ello, se ha utilizado el *One-Class SVM* [230], una variante del clasificador de vectores de soporte (SVC) [231] que fue concebido para identificar valores atípicos en el conjunto de datos de entrenamiento. El método separa todos los puntos de datos del conjunto de entrenamiento maximizando la distancia del hiperplano calculado con respecto al origen. Este problema de optimización se aborda mediante programación cuadrática acorde a la Ecuación 6.3:



(a)



(b)



(c)

Figura 6.1.: Bandas de mínima coherencia espectral en cada electrodo. Estímulos de (a) 2 Hz, (b) 8 Hz y (c) 20 Hz.

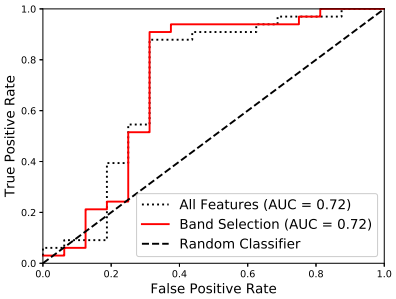
$$\begin{aligned} \underset{\omega,\xi_i,b}{\text{Minimizar}} \quad & \left\{ \frac{1}{2} \cdot \|\omega\| + \frac{a}{\nu \cdot N} \cdot \sum_{i=1}^{N} \xi_i - b \right\} \\ \text{sujeto a} \quad & (\omega \cdot \phi(x_i)) \geq b - \xi_i \\ & \xi_i > 0, \nu \in (0, 1] \end{aligned} \tag{6.3}$$

donde ξ_i son variables distintas de cero para controlar el margen, ϕ es el núcleo de la función y ν controla el número de vectores de soporte y la fracción de muestras de entrenamiento consideradas como valores atípicos. Por lo tanto, una función de decisión puede ser construida para que proporcione valores diferentes para las mues-

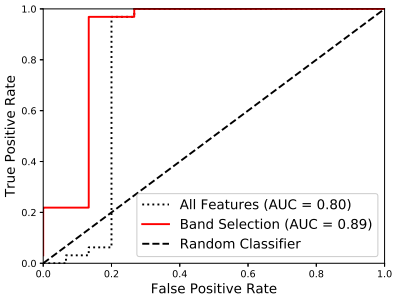
tras que pertenecen a la misma clase con la que se ha realizado el entrenamiento, al igual que para muestras que pertenezcan a otra clase mediante la utilización del hiperplano que se define por los parámetros ω y b. Además, se utiliza una función núcleo de base radial para permitir un hiperplano de separación no lineal:

$$f(z) = sign\{(\omega \cdot \phi(z)) - b\} \tag{6.4}$$

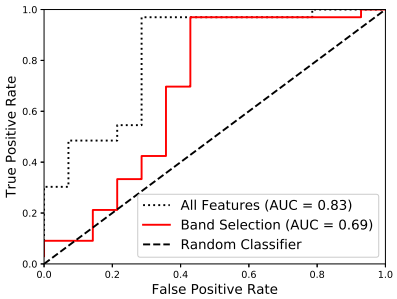
Por otra parte, para llevar a cabo los experimentos de clasificación se han utilizado las características del EEG de las señales adquiridas durante los estímulos de 2 Hz, 8 Hz y 20 Hz. El método propuesto ha sido evaluado mediante validación cruzada estratificada (*k-fold*), con $k = 5$, para garantizar la independencia del conjunto de datos y comprobar la capacidad de generalización del clasificador. En las Figuras 6.2a, 6.2b y 6.2c se pueden ver las curvas ROC obtenidas al clasificar los sujetos mediante estímulos de 2 Hz, 8 Hz y 20 Hz, respectivamente:



(a) 2 Hz



(b) 8 Hz



(c) 20 Hz

Figura 6.2.: Curvas ROC obtenidas con estímulos de (a) 2 Hz, (b) 8 Hz y (c) 20 Hz.

El método de selección de características basado en el uso de la banda que muestra la menor coherencia entre los sujetos CN y DD mejora el rendimiento del clasificador con respecto al uso de todas las características para 2 Hz y 8 Hz. La mejora del rendimiento se debe a la reducción de la dimensión del espacio de características

y al uso de características más discriminantes. Sin embargo, el uso de todas las características (es decir, todas las bandas para todos los electrodos) proporciona valores AUC más altos para el estímulo de 20 Hz. La Tabla 6.1 muestra el rendimiento de la clasificación en términos de precisión, sensibilidad y especificidad. Como se indica en dicha tabla, el método de selección de características mejora la sensibilidad y especificidad para 2 Hz y 8 Hz pero disminuye el rendimiento en el caso de 20 Hz.

Tabla 6.1.: Resultados de clasificación.

Estímulos	Exactitud	Sensitividad	Especificidad	AUC
2 Hz (Todas las características)	0,62	0,66	0,60	0,72
2 Hz (Selección de banda)	0,70	0,66	0,69	0,72
8 Hz (Todas las características)	0,63	0,80	0,55	0,80
8 Hz (Selección de banda)	0,66	0,86	0,56	0,89
20 Hz (Todas las características)	0,78	0,66	0,81	0,83
20 Hz (Selección de banda)	0,71	0,53	0,78	0,69

6.2. Framework de optimización aplicado a la clasificación de señales EEG

En esta sección se muestran los resultados del experimento realizado para la clasificación multi-clase de señales EEG sin utilizar descriptores conocidos a priori. Precisamente, el framework desarrollado en esta tesis y descrito en el capítulo anterior tiene como objetivo encontrar de forma automática arquitecturas que sean capaces de extraer características ad hoc para un determinado problema, seleccionarlas y clasificar los patrones encontrados. Por tanto, en lo sucesivo se mostrará la utilidad de dicho framework y la mejora que supone las soluciones optimizadas encontradas.

Para llevar a cabo los experimentos se ha configurado el framework para utilizar el algoritmo NSGA-II en el procedimiento de optimización multi-objetivo debido al buen balance entre prestaciones y carga computacional que proporciona. Además, el NSGA-II permite alcanzar óptimos resultados cuando se abordan problemas con múltiples objetivos [232] (pero no excesivamente alto). No obstante, es importante destacar que el framework otorga bastante flexibilidad para ejecutar el procedimiento de optimización de ANNs utilizando diversos métodos de optimización basados en metaheurísticas, considerando que inclusive es posible seleccionar el método

de optimización a ser utilizado, ya que éste constituye un parámetro más del framework. Como se ha mencionado anteriormente, la principal ventaja de las ANNs con respecto a otros métodos es que una única arquitectura encapsula las fases de extracción, selección de características y clasificación. En la Figura 6.3 se observa el diagrama de bloques del framework con el NSGA-II incorporado y su interacción con el resto de componentes durante el proceso de optimización:

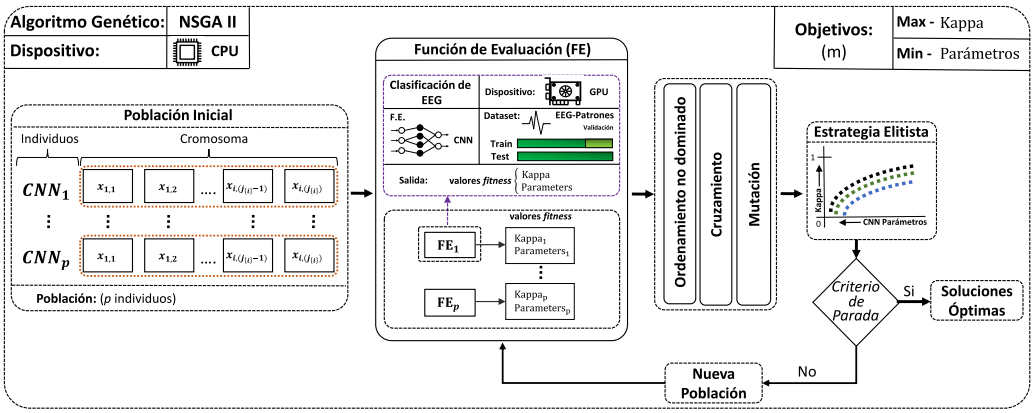


Figura 6.3.: Procedimiento evolutivo multi-objetivo. Esquema de la implementación para la optimización de la CNN utilizando el NSGA-II.

El procedimiento de la figura se aplica directamente para la optimización de la red *EEGNet* [187], la cual ha sido tomada como referencia. Las entidades de la red (capas, regularizadores de capas de normalización), los hiperparámetros que se incluyen en la *EEGNet* y los rangos de valores sobre los que pueden variar cada uno de los argumentos han sido registrados previamente en la DB. El framework tiene capacidad de optimizar todos los hiperparámetros de la red e incluso optimizar la arquitectura añadiendo o eliminando capas según las restricciones establecidas, las cuales se encargan de determinar el alcance del procedimiento de optimización (modificar la estructura de la red o limitándose únicamente a la optimización de los hiperparámetros de una ANN preexistente). La Figura 6.4 muestra los genes que componen los cromosomas. La configuración del procedimiento de optimización permite la selección de los diferentes parámetros indicados en dicha figura según las capas que se utilizaron en la red de referencia. Los genes del cromosoma son de tipo entero para acelerar el proceso de búsqueda y limitar el rango de cada gen mediante una tabla de búsqueda donde cada entrada se encuentra codificada como un valor entero.

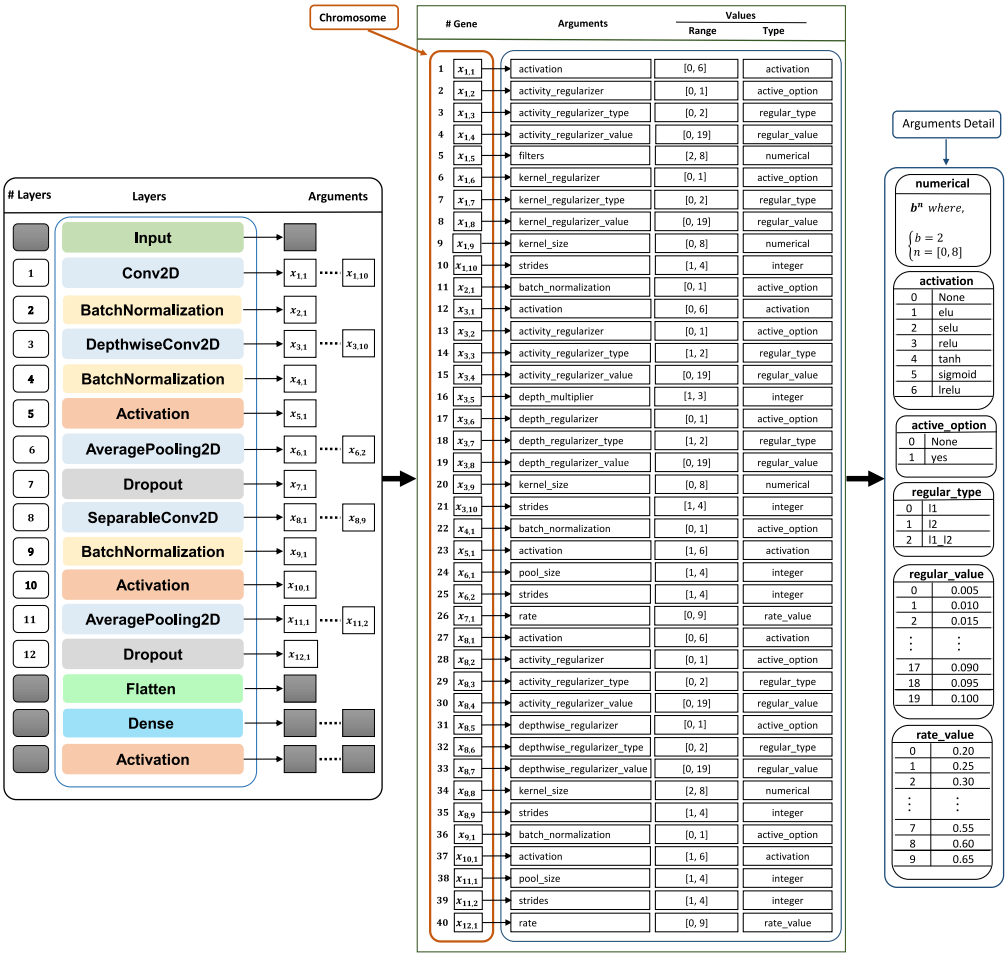


Figura 6.4.: **Codificación del cromosoma**. Vista detallada de los valores utilizados para la optimización de la arquitectura *EEGNet*. No obstante, es posible configurar en la DB otras funciones de activación, rangos específicos o tipos de regularizaciónen dependiendo del problema de optimización específico.

6.2.1. Conjunto de datos

Los datos utilizados en esta tesis han sido obtenidos por el laboratorio de BCI de la Universidad de Essex, Reino Unido [136]. Cabe destacar que la selección de los participantes humanos se produjo tratando de mantener una distribución equilibrada entre géneros y sin experiencia en BCI: los 12 sujetos seleccionados para este experimento son mujeres con edades comprendidas entre los 24 y los 50 años. Los participantes tenían buena salud, se les aconsejó que durmieran bien antes de la recogida de datos y recibieron una compensación económica por su participación. Además, de manera previa a la realización del experimento dieron su consentimiento expreso mediante un formulario que fue aprobado por el Comité de Ética de la

Universidad de Essex. Cada EEG del conjunto de datos corresponde a una serie temporal que consta de 5120 muestras registradas a una frecuencia de muestreo de 256 Hz. Para ello, se colocaron 32 electrodos en el cuero cabelludo durante la recogida de datos, aunque finalmente se han utilizado sólo los 15 electrodos mostrados en la Figura 6.5.

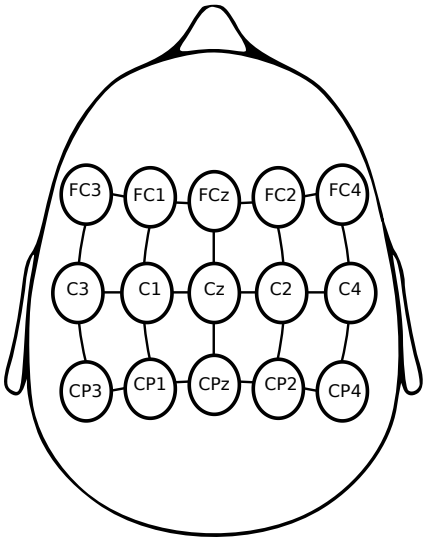


Figura 6.5.: Sistema de coordenadas 10/20. Posiciones de los electrodos utilizados para la adquisición de las señales de electroencefalografía (EEG).

Cada conjunto de datos se compone de 178 patrones de entrenamiento y 179 para test. Cada patrón se etiqueta según la clase BCI correspondiente: imaginación motora de la mano izquierda, de la mano derecha o de los pies (ver Tabla 6.2). Del total de sujetos participantes en la toma de muestras se han seleccionado únicamente tres sujetos codificados como 104, 107 y 110. Esto se debe a que en trabajos previos [233, 234] se comprobó que son los que proporcionan mejor rendimiento.

Tabla 6.2.: Número de muestras para cada sujeto de las diferentes imaginaciones motoras.

Sujeto	Mano izquierda		Mano derecha		Pie		Total	
	Train	Test	Train	Test	Train	Test	Train	Test
104	60	56	66	65	52	58	178	179
107	56	58	57	58	65	63	178	179
110	58	59	60	60	60	60	178	179

6.2.2. Estrategia para optimización y distribución CPU-GPU de la carga de trabajo

El objetivo principal del proceso de optimización consiste en aumentar el rendimiento de la clasificación y reducir al mismo tiempo el número de parámetros. Estos objetivos están en conflicto pues redes más grandes suelen conseguir un mejor ajuste sobre los datos de entrenamiento. Es por ello que es preferible mejorar el rendimiento de la generalización ya que redes más grandes también serán más propensas al sobreajuste. Como se ha mencionado anteriormente, las funciones fitness utilizadas para evaluar las soluciones son el índice Kappa y el número total de parámetros de la red neuronal. Cabe destacar que la evaluación de la aptitud de una solución requiere el entrenamiento y la validación de una red neuronal utilizando para ello el conjunto de datos de entrenamiento. Aunque estas tareas consumen una gran cantidad tiempo, pueden ser aceleradas aprovechando el paralelismo que ofrecen las GPUs y los diversos núcleos CPU. Como el framework propuesto permite distribuir tareas entre los dispositivos de un nodo de cómputo, en la base de datos de configuración del framework es posible configurar la combinación específica de CPUs y GPUs a utilizar. De este modo, se pueden aprovechar los recursos computacionales con el objetivo de disminuir el tiempo de procesamiento. Específicamente, la CPU se encarga de ejecutar el algoritmo evolutivo y la GPU de evaluar las soluciones. Durante la experimentación se han utilizado dos equipos diferentes: uno de ellos contiene dos CPUs Intel Xeon E5-2640 v4 y una GPU Nvidia TITAN Xp. El otro, dos CPUs Intel Xeon Silver 4214 y una GPU Nvidia Quadro RTX 6000.

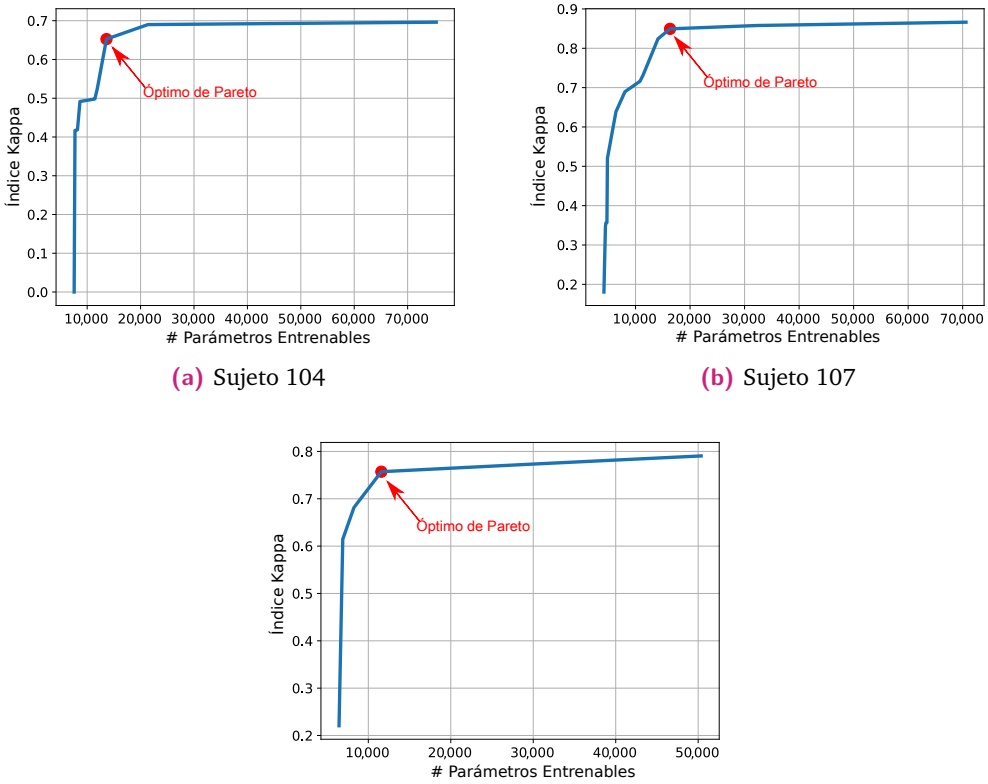
En cuanto al proceso de optimización, éste se resume de la siguiente forma: la población inicial que se encuentra formada por un conjunto de soluciones aleatorias empezará su proceso evolutivo mediante la aplicación de los operadores genéticos, tal y como se explica en la Sección 4.2.1. La evaluación de cada solución proporcionará las soluciones no dominadas. A continuación, se seleccionan las mejores soluciones (no dominadas) en función de los objetivos a optimizar y se aplican de nuevo los operadores de cruce y mutación. El framework de optimización almacena en la DB el conjunto de soluciones que sobreviven después de cada generación y sus respectivos valores de fitness. En la Tabla 6.3 se encuentran detallados los parámetros utilizados para el algoritmo NSGA-II:

Tabla 6.3.: Parámetros del NSGA-II.

Parámetros	Valor
Extensión del cromosoma (genes)	40
Tamaño de población	100
Número de generaciones	100
Probabilidad de mutación	0,05
Probabilidad de cruce	0,5

6.2.3. Resultados

Los experimentos para optimizar cada conjunto de datos han tomado como punto de partida la red *EEGNet*. Como resultado se han obtenido los frentes de Pareto de la Figura 6.6. Se recuerda que en este frente se encuentran las soluciones generadas durante el proceso evolutivo que no pueden dominarse entre sí. En la Figura 6.6 el óptimo de Pareto se encuentra indicado con un punto rojo, y corresponde a la solución que proporciona el compromiso entre ambos objetivos (índice Kappa y número de parámetros del modelo):



(a) Sujeto 104

(b) Sujeto 107

(c) Sujeto 110

Figura 6.6.: **Frente de Pareto para cada sujeto tras optimizar la red neuronal.**

La Tabla 6.4 expone los modelos de estas soluciones y las capas que los componen. El procedimiento de optimización no sólo selecciona los valores de los hiperparámetros sino que también puede decidir si se incluyen o no determinadas capas, si utiliza regularización y el método para ello en caso afirmativo.

Tabla 6.4.: Modelos correspondientes al óptimo de Pareto.

Capas	Parámetros	EEGNet (baseline)	Soluciones optimizadas 104	107	110
Conv2D	activation	No	selu	sigmoid	selu
	filters	8	4	4	4
	kernel_regul	No	No	Sí	No
	kernel_regul_type	No	No	l1	No
	kernel_regul_value	No	No	0,065	No
	kernel_size	(1, 32)	(1, 1)	(1, 2)	(1, 1)
	strides	1	1	1	3
BatchNormalization	batch_normal	Sí	Sí	Sí	Sí
DepthwiseConv2D	activation	No	relu	tanh	No
	filters	16	4	4	4
	depth_multiplier	2	1	1	1
	depthwise_regul	No	Sí	No	No
	depthwise_regul_type	No	l2	No	No
	depthwise_regul_value	No	0,065	No	No
	kernel_size	1	1	1	1
	strides	No	3	4	2
BatchNormalization	batch_normal	Sí	Sí	Sí	Sí
Activation	activation	elu	elu	tanh	relu
AveragePooling2D	pool_size	(1, 8)	(1, 4)	(1, 1)	(1, 3)
	strides	No	3	1	3
Dropout	rate	0,50	0,50	0,40	0,40
SeparableConv2D	activation	No	relu	relu	tanh
	activity_regul	No	Sí	Sí	No
	activity_regul_type	No	l2	l2	No
	activity_regul_value	No	0,040	0,025	No
	depthwise_regul	No	Sí	Sí	Sí
	depthwise_regul_type	No	l2	l2	l2
	depthwise_regul_value	No	0,040	0,070	0,045
	kernel_size	(1,8)	(1, 4)	(1, 16)	(1, 8)
	strides	No	4	4	2
BatchNormalization	batch_normal	Sí	Sí	Sí	Sí
Activation	activation	elu	selu	relu	relu
AveragePooling2D	pool_size	(1,8)	(1, 4)	(1, 3)	(1, 4)
	strides	No	2	2	3
Dropout	rate	0,50	0,55	0,40	0,25

Los modelos producidos por el framework de optimización se entrenaron utilizando el archivo de datos de entrenamiento completo (sin la división de validación) y luego se probaron con el archivo de datos de test. Los resultados que ofrecidos en la Tabla 6.5 muestran una clara mejora en la precisión de los modelos optimizados con respecto a los modelos originales tomados como referencia (*EEGNet* y *DeepConvNet*). Estos resultados muestran que los modelos optimizados consiguen una mejora media en el índice Kappa del 43,9 %, 87,2 % y 27,5 % respecto a la *EEGNet* original para los sujetos 104, 107 y 110, respectivamente.

Con el fin de demostrar estadísticamente la superioridad de la solución optimizada con respecto a las dos redes utilizadas como referencia (*EEGNet* y *DeepConvNet*), se ha realizado un test de hipótesis no paramétrico mediante la prueba de Mann-Whitney [163]. Con este método se obtiene la probabilidad de que los resultados obtenidos para cada sujeto sean diferentes. Se han realizado 15 ejecuciones con el fin de tener en cuenta la varianza debida a la inicialización aleatoria de los algoritmos involucrados en el procedimiento de optimización. La comparación realizada mediante el test de Mann-Whitney indica p-valores inferiores a 10^7, lo que permite rechazar con un alto nivel de significancia la hipótesis nula (que indicaría que ambos grupos tienen la misma media). Cabe destacar que en la Tabla 6.5 los p-valores se encuentran indicados únicamente en el bloque correspondiente a la solución optimizada ya que el test de hipótesis se ha realizado entre los dos grupos que proporcionan mejores prestaciones (*EEGNet* y la solución optimizada proporcionada por el framework).

Tabla 6.5.: Resultados de clasificación y validación estadística.

Sujeto	**Exactitud**		**Índice Kappa**		***p*-valores**
	Promedio	**Dev. std.**	**Promedio**	**Dev. std.**	
DeepConvNet [187]					
104	0,58	0,03	0,38	0,04	
107	0,66	0,04	0,48	0,06	
110	0,48	0,02	0,22	0,03	
***EEGNet* (baseline)** [187]					
104	0,63	0,05	0,44	0,08	
107	0,63	0,06	0,44	0,08	
110	0,72	0,05	0,58	0,08	
Solución optimizada					
104	0,75	0,01	0,63	0,01	$p < 1,70 \cdot 10^{-6}$
107	0,88	0,02	0,82	0,02	$p < 1,69 \cdot 10^{-6}$
110	0,83	0,01	0,74	0,01	$p < 1,64 \cdot 10^{-6}$

La Figura 6.7 muestra una comparativa entre los diferentes modelos de *EEGNet* para los tres sujetos utilizados en este trabajo. Como se puede ver, la versión optimizada siempre supera en prestaciones a los modelos de referencia. Las redes basadas en *EEGNet* que se comparan en la Tabla 6.5 difieren principalmente en el número de filtros utilizados en las capas inferiores: mientras que la llamada *Compact-CNN* [235] utiliza 96 filtros, la *EEGNet* tomada como baseline [187] utiliza 16 filtros. Esta es la razón principal del por qué dicha red tiene menor número de parámetros. Por otra parte, el método de optimización incluye capas de regularización que actúan sobre los núcleos y funciones de activación y cuya necesidad está motivada por la mejora en la capacidad de generalización de la red (es decir, para reducir el sobreajuste). En total, el proceso de optimización de las ANNs ha requerido 116,99 horas para el sujeto 104, 139,02 horas para el 107 y 142,81 horas para el 110 utilizando el equipo que contiene la CPU Intel Xeon E5-2640 v4 y la GPU Nvidia TITAN Xp.

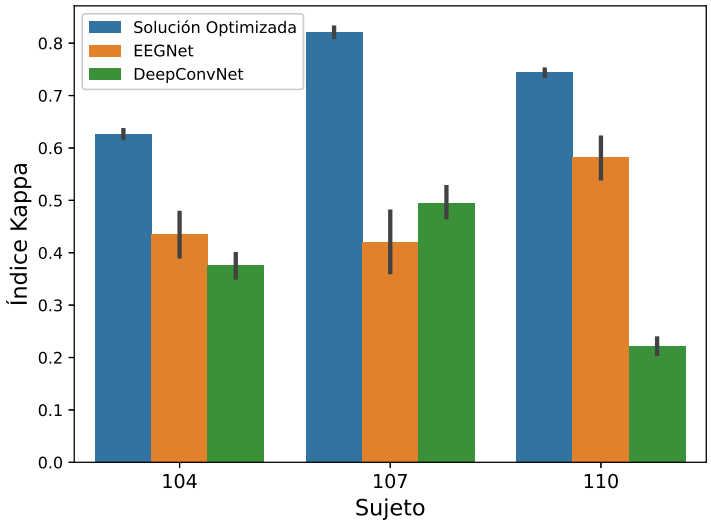
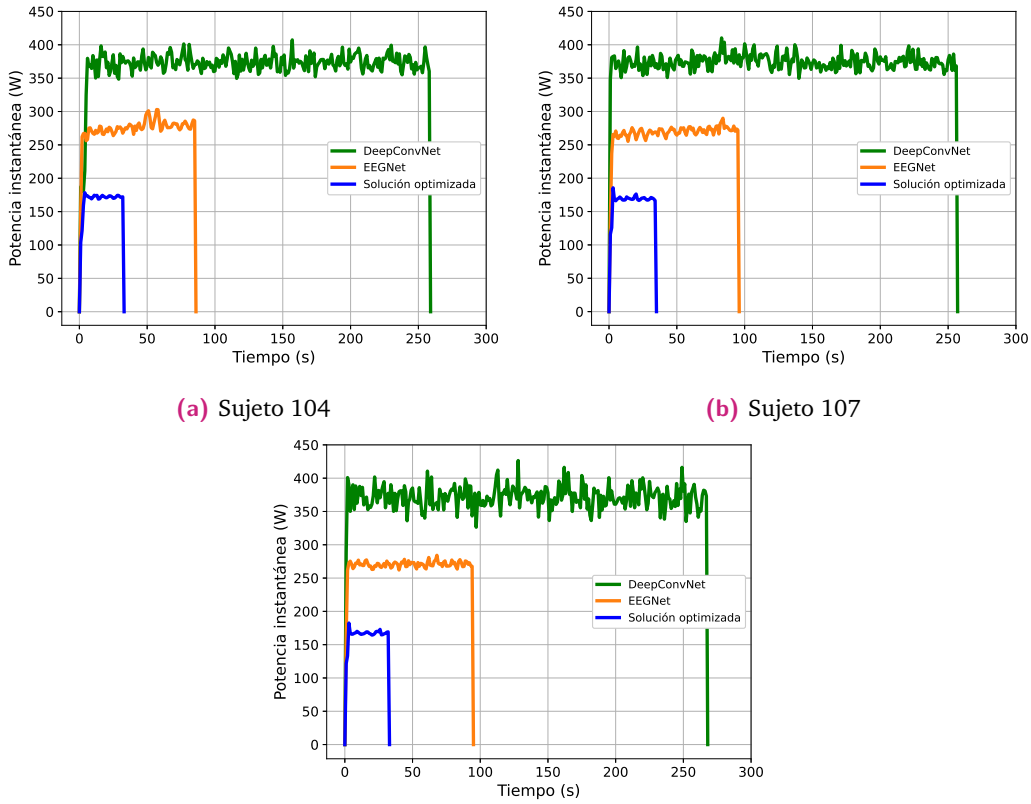


Figura 6.7.: Comparativa de rendimiento entre diferentes modelos de *EEGNet*. Incluye la arquitectura optimizada generada por el framework de optimización correspondiente al óptimo de Pareto.

6.2.4. Eficiencia energética de las soluciones optimizadas

En esta sección se analiza el consumo energético de las diferentes redes utilizadas. Con ello, se pretende evaluar la eficiencia energética de la red optimizada con respecto a las que han sido tomadas como referencia. En la Figura 6.8 se muestra el perfil de potencia instantánea cuando las redes son entrenadas utilizando los datos EEG de los sujetos 104, 107 y 110. La GPU utilizada para este experimento ha sido la Nvidia TITAN Xp. Tal y como se muestra en esta figura, la alternativa

optimizada tarda menos en entrenarse independientemente del sujeto y reduce considerablemente la potencia instantánea. Este resultado es el esperado ya que al reducir el número de parámetros también lo hace la complejidad de la red y el estrés hacia el dispositivo de cómputo (GPU), lo cual repercute directamente no sólo en el tiempo de ejecución sino también en el consumo energético.



(a) Sujeto 104

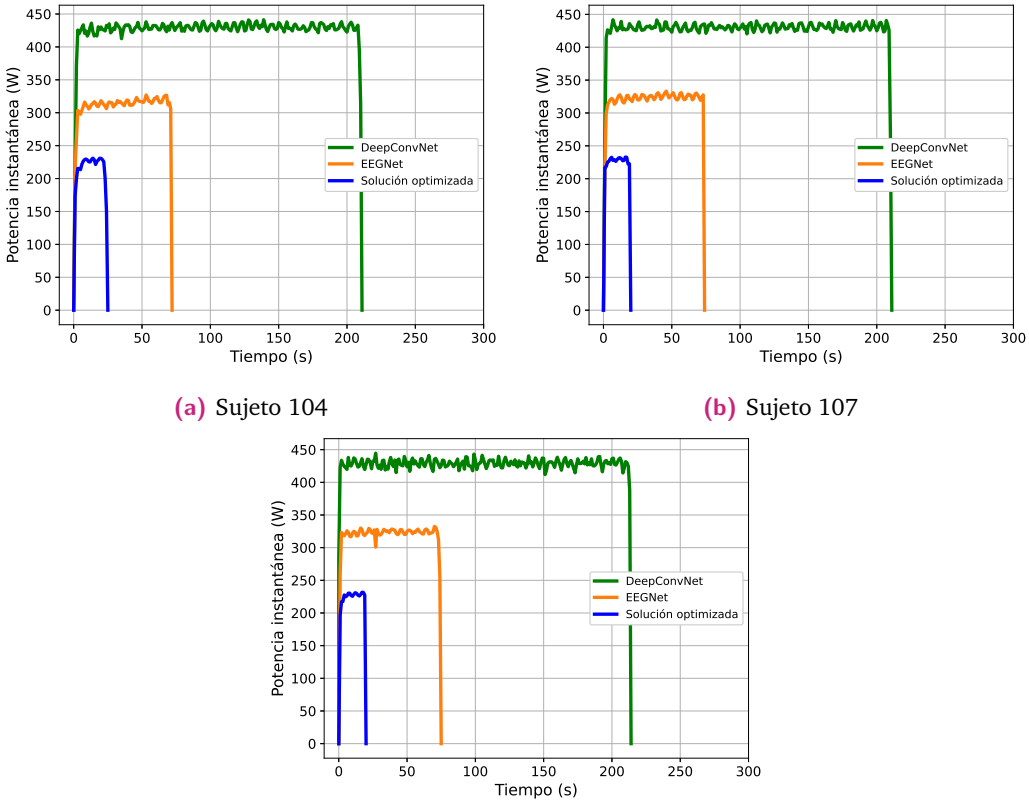
(b) Sujeto 107

(c) Sujeto 110

Figura 6.8.: Potencia instantánea obtenida durante el entrenamiento de diferentes modelos y sujetos al utilizar la GPU Nvidia TITAN Xp.

De forma similar, en la Figura 6.9 se muestra la potencia instantánea al entrenar las redes pero esta vez se ha utilizado la GPU Nvidia Quadro RTX 6000. El objetivo es analizar cuál es el comportamiento energía-tiempo al utilizar una GPU con una arquitectura más moderna. De la figura claramente se observa que el tiempo de ejecución ha disminuido notablemente para todas las redes y la proporción de tiempo que las separa se ha conservado. Sin embargo, la potencia instantánea se ha incrementado ligeramente en todos los casos, lo cual es normal ya que la GPU más moderna tiene una potencia de diseño térmico (TDP) de 295W frente a los 250W de la Nvidia TITAN Xp. Esto no quiere decir que para entrenar las redes se haya requerido más energía, pues la energía total consumida depende también del tiempo de ejecución. En la Tabla 6.6 se muestra el consumo de energía acumulado para

cada GPU. Fácilmente se puede ver que al utilizar la GPU más moderna el consumo energético es siempre inferior y además con la ventaja de haber reducido en gran medida el tiempo de ejecución tal y como se había comentado en la Figura 6.9.



(a) Sujeto 104

(b) Sujeto 107

(c) Sujeto 110

Figura 6.9.: Potencia instantánea obtenida durante el entrenamiento de diferentes modelos y sujetos al utilizar la GPU Nvidia Quadro RTX 6000.

Tabla 6.6.: Consumo de energía en W · h para cada GPU durante el entrenamiento de diferentes modelos y sujetos.

Modelo	Sujeto		
	104	107	110
Nvidia Titan Xp			
DeepConvNet [187]	26,51	26,52	27,49
EEGNet (base) [187]	6,46	7,05	6,99
Solución Optimizada	1,46	1,54	1,44
Nvidia Quadro RTX 6000			
DeepConvNet [187]	24,89	24,94	25,37
EEGNet (base) [187]	6,11	6,47	6,54
Solución Optimizada	**1,18**	**1,14**	**1,13**

6.3. Conclusiones

En este capítulo se han presentado los resultados obtenidos utilizando descriptores conocidos a priori y utilizando el framework de optimización propuesto. Como se ha comentado, el principal problema de utilizar descriptores estadísticos conocidos a priori es que su elección resulta compleja y nunca se tiene garantía de que su poder discriminante sea óptimo para un problema concreto. Es por ello que se ha utilizado el framework propuesto, el cual ha permitido optimizar arquitecturas de aprendizaje profundo que se encargan del proceso de extracción, selección y clasificación de características. La configuración del experimento ha sido realizada no únicamente pensando en el ajuste de los hiperparámetros, sino también para habilitar o deshabilitar algunas capas como las que implementan la regularización. Como resultado, la arquitectura de los modelos generados puede diferir de la original. Se han utilizado dos métricas de rendimiento: el índice Kappa para medir el rendimiento de la clasificación multi-clase y el número de parámetros, el cual obliga al algoritmo a seleccionar soluciones con un menor número de parámetros. La optimización de estas métricas está basada en los resultados de validación con el objetivo de mejorar la capacidad de generalización del modelo óptimo. El framework propuesto se ha evaluado utilizando datos de EEG para BCI compuestos por 15 canales. Posteriormente, se ha utilizado como baseline la red *EEGNet* para optimizar su rendimiento. Los resultados alcanzados han demostrado claras mejoras con respecto a la red original, consiguiendo hasta un 87 % de mejora con hasta un 33 % menos de parámetros entrenables. Además, la solución optimizada también reduce drásticamente el consumo energético y tiempo de ejecución necesario para entrenar las redes.

También en este capítulo se ha visto cómo las técnicas de autoML contribuyen al campo del Deep Learning con procedimientos para desarrollar u optimizar automáticamente redes existentes para resolver un problema específico. Este es el caso de la clasificación de señales EEG, donde las arquitecturas habituales basadas en CNN para el procesamiento de imágenes no son apropiadas debido a las características especiales de las señales de EEG. Además, la inclusión de capas que permiten convoluciones espaciales y temporales añade nuevos hiperparámetros difíciles de seleccionar. El framework de optimización presentado en esta tesis aporta una alternativa flexible para optimizar las redes existentes y mejorar el rendimiento de la clasificación para un problema específico.

7 Conclusiones y principales aportaciones

En este tesis se ha abordado el problema de diseñar arquitecturas de redes neuronales para dar solución a problemas específicos. La inexistencia de reglas de diseño hace que dicha tarea sea costosa en tiempo y que normalmente requiera de un proceso de ensayo y error. Dicho problema es aún más importante en el caso de las arquitecturas profundas ya que el número de capas que pueden combinarse es muy alto y existe una gran cantidad de hiperparámetros de cuya elección depende en gran medida el rendimiento del modelo final. Una de las principales aplicaciones de las redes profundas es la clasificación de instancias de datos de diferente naturaleza. Un ejemplo es la clasificación de imágenes, tarea para la cual se han utilizado extensivamente en los últimos año dada su eficiencia. Otra de las ventajas del uso de arquitecturas profundas es su capacidad para aprender a extraer descriptores con un poder discriminante óptimo para un determinado problema. Los métodos de clasificación clásicos utilizan descriptores estadísticos conocidos a priori para conformar un espacio de características que será posteriormente utilizado por el algoritmo de clasificación. Por tanto, las prestaciones del clasificador dependen en gran medida de las características con las que estos se alimentan, la estructura del espacio vectorial de dichas características, y su distribución estadística. Las arquitecturas profundas extraen las características más apropiadas para un problema concreto mediante un proceso de aprendizaje, evitando tener que conocer descriptores a priori o seleccionarlos en base a su poder discriminante o de representación de los datos. Lo expuesto anteriormente motiva la necesidad de disponer de métodos para optimizar e incluso generar de forma automática modelos basados en arquitecturas de redes neuronales profundas que evite, por un lado, utilizar descriptores conocidos a priori, y por otro, el proceso de ensayo y error necesario para ajustar la arquitectura y sus hiperparámetros. Esta línea puede enmarcarse en lo que hoy día se conoce como Auto Machine Learning (autoML).

Con ese objetivo, en esta tesis se ha desarrollado un framework de optimización basado en computación evolutiva que permite mediante una base de datos configurar los parámetros de las ANNs a optimizar y el rango de valores de cada uno de sus parámetros. Además, permite configurar las restricciones que delimitan el espacio

de búsqueda para encontrar soluciones factibles y reducir el tiempo de búsqueda. Dado que el framework es totalmente configurable, se podría utilizar para generar ANNs desde cero o para optimizar arquitecturas preexistentes.

Concretamente, en esta tesis se presenta el problema de clasificación de señales EEG, el cual generalmente es un problema complejo porque la extracción de los patrones que permiten clasificar a la señal están habitualmente basados en la selección de descriptores estadísticos en tiempo, frecuencia o tiempo-frecuencia conocidos a priori. Si se utilizan descriptores que realmente no permitan caracterizar correctamente a los datos no se podrán encontrar patrones. De esta forma, el problema no se reduce únicamente a encontrar características sino también de seleccionar aquellas que permitirán clasificar las señales con más éxito. En este ámbito, se han realizado diferentes tipos de experimentos utilizando varios conjuntos de datos. En primer lugar, se han realizado experimentos clásicos extrayendo características espectrales de las señales EEG, mostrando que la clasificación de señales EEG es una cuestión compleja y manifestando la influencia que tienen las características seleccionadas en la capacidad discriminante del clasificador. Por otro lado, se han realizado otros experimentos con señales EEG/BCI utilizando un conjunto de datos multi-clase. En concreto, los experimentos realizados tenían como objetivo optimizar una arquitectura preexistente diseñada originalmente para el procesamiento de señales de EEG denominada *EEGNet*.

Los resultados han permitido observar también, que el procedimiento de optimización realmente mejora las prestaciones que brinda la red tomada como baseline (*EEGNet*). Al ser evaluada con el mismo conjunto de señales EEG se superan los resultados obtenidos por la *EEGNet* original, con hasta un 87 % de mejora en el índice de clasificación y hasta un 33 % menos en cantidad de parámetros entrenables. Además, se han realizado medidas de energía donde se ha podido corroborar que las soluciones o redes que proporcionan el framework de optimización no solamente mejoran en precisión sino también en eficiencia computacional y por tanto en consumo energético. Esto se debe a que las ANN optimizadas tardan menos en entrenarse por tener menos parámetros, lo cual se traduce en que utilizan menos recursos de los dispositivos CPU y GPU del sistema heterogéneo. Por último, se ha demostrado que utilizar arquitecturas hardware más modernas permite reducir de forma simultánea tanto el tiempo de ejecución como el consumo energético. Es por todo esto que actualmente, debido a temas medioambientales y en medio de la actual crisis del cambio climático, proporcionar soluciones eficientes en energía podría considerarse una forma de aliviar el problema.

Bibliografía

[1] M. Castells. *The information age: Economy, society and culture (Vol. 1–3)*. Oxford Blackwell Publishers, 2000 (vid. pág. 1).

[2] F. Webster y M. A. Ruggles. *Theories of the information society*. Vol. 23. 1. Edmonton: Canadian Journal of Sociology, 1998, pág. 125 (vid. pág. 1).

[3] Cisco Systems. *Global - 2021 Forecast Highlights*. Inf. téc. San José, CA, 2021, págs. 1-6. URL: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf (visitado 10 de ene. de 2022) (vid. pág. 1).

[4] Cisco Systems. *Spain - 2021 Forecast Highlights*. Inf. téc. San José, CA, 2021, págs. 1-5. URL: https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf (visitado 10 de ene. de 2022) (vid. pág. 1).

[5] K. Miettinen. *Nonlinear multiobjective optimization*. Vol. 12. Springer Science & Business Media, 2012 (vid. pág. 1).

[6] Y. Sawaragi, H. Nakayama y T. Tanino. *Theory of multiobjective optimization*. Elsevier, 1985 (vid. pág. 1).

[7] L. N. de Castro. "Fundamentals of natural computing: an overview". En: *Physics of Life Reviews* 4.1 (2007), págs. 1-36 (vid. pág. 2).

[8] J. Branke, J. Branke, K. Deb, K. Miettinen y R. Slowiński. *Multiobjective optimization: Interactive and evolutionary approaches*. Vol. 5252. Springer Science & Business Media, 2008 (vid. pág. 2).

[9] M. Ehrgott y X. Gandibleux. "A survey and annotated bibliography of multiobjective combinatorial optimization". En: *OR-Spektrum* 22.4 (nov. de 2000), págs. 425-460 (vid. pág. 2).

[10] W. S. McCulloch y W. Pitts. "A logical calculus of the ideas immanent in nervous activity". En: *The bulletin of mathematical biophysics* 5.4 (dic. de 1943), págs. 115-133 (vid. págs. 3, 45).

[11] M. Wistuba. "Practical Deep Learning Architecture Optimization". En: *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*. 2018, págs. 263-272 (vid. pág. 3).

[12] K. Doi. "Computer-aided diagnosis in medical imaging: historical review, current status and future potential". En: *Computerized medical imaging and graphics* 31.4-5 (2007), págs. 198-211 (vid. pág. 4).

[13] T. W. Freer y M. J. Ulissey. “Screening mammography with computer-aided detection: prospective study of 12,860 patients in a community breast center”. En: *Radiology* 220.3 (2001), págs. 781-786 (vid. pág. 4).

[14] M. R. K. Mookiah, U. R. Acharya, C. K. Chua, C. M. Lim, E. Ng y A. Laude. “Computer-aided diagnosis of diabetic retinopathy: A review”. En: *Computers in biology and medicine* 43.12 (2013), págs. 2136-2155 (vid. pág. 4).

[15] G. S. Lodwick, C. L. Haun, W. E. Smith, R. F. Keller y E. D. Robertson. “Computer Diagnosis of Primary Bone Tumors”. En: *Radiology* 80.2 (1963), págs. 273-275 (vid. pág. 4).

[16] U. R. Acharya, S. Bhat, H. Adeli, A. Adeli y col. “Computer-aided diagnosis of alcoholism-related EEG signals”. En: *Epilepsy & Behavior* 41 (2014), págs. 257-263 (vid. pág. 4).

[17] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller y T. M. Vaughan. “Brain–computer interfaces for communication and control”. En: *Clinical neurophysiology* 113.6 (2002), págs. 767-791 (vid. pág. 4).

[18] S. Aggarwal y N. Chugh. “Signal processing techniques for motor imagery brain computer interface: A review”. En: *Array* 1 (2019), pág. 100003 (vid. pág. 5).

[19] S. Krishnan e Y. Athavale. “Trends in biomedical signal feature extraction”. En: *Biomedical Signal Processing and Control* 43 (2018), págs. 41-63 (vid. pág. 6).

[20] E. Tessy, P. M. Shanir y S. Manafuddin. “Time domain analysis of epileptic EEG for seizure detection”. En: *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*. IEEE. 2016, págs. 1-4 (vid. pág. 6).

[21] F. O.K. y R. R. “Time-domain exponential energy for epileptic EEG signal classification”. En: *Neuroscience Letters* 694 (2019), págs. 1-8 (vid. pág. 6).

[22] A. Khorshidtalab, M. J. E. Salami y M. Hamedi. “Evaluation of time-domain features for motor imagery movements using FCM and SVM”. En: *2012 Ninth International Conference on Computer Science and Software Engineering (JCSSE)*. 2012, págs. 17-22 (vid. pág. 6).

[23] R. G. Willison. “A method of measuring motor unit activity in human muscle”. En: *Journal of Physiology-London*. Vol. 168. 2. Cambridge Univ. Press 40 West 20th Street, New York, NY 10011-4211. 1963, P35 (vid. pág. 6).

[24] S. Negi, Y. Kumar y V. M. Mishra. “Feature extraction and classification for EMG signals using linear discriminant analysis”. En: *2016 2nd International Conference on Advances in Computing, Communication, Automation (ICACCA) (Fall)*. 2016, págs. 1-6 (vid. pág. 6).

[25] A. Khorshidtalab, M.-J. E. Salami y M. Hamedi. “Robust classification of motor imagery EEG signals using statistical time–domain features”. En: *Physiological measurement* 34.11 (2013), pág. 1563 (vid. pág. 6).

[26] B. Remeseiro y V. Bolon-Canedo. “A review of feature selection methods in medical applications”. En: *Computers in Biology and Medicine* 112 (2019), pág. 103375 (vid. pág. 6).

[27] L. Lu, J. Yan y C. W. de Silva. “Feature selection for ECG signal processing using improved genetic algorithm and empirical mode decomposition”. En: *Measurement* 94 (2016), págs. 372-381 (vid. pág. 6).

[28] F. M. Noori, N. Naseer, N. K. Qureshi, H. Nazeer y R. A. Khan. “Optimal feature selection from fNIRS signals using genetic algorithms for BCI”. En: *Neuroscience Letters* 647 (2017), págs. 61-66 (vid. pág. 6).

[29] K.-L. Du y M. N. Swamy. *Neural networks and statistical learning*. Springer Science & Business Media, 2013 (vid. pág. 6).

[30] F. Lotte, L. Bougrain, A. Cichocki, M. Clerc, M. Congedo, A. Rakotomamonjy y F. Yger. “A review of classification algorithms for EEG-based brain-computer interfaces: a 10 year update”. En: *Journal of Neural Engineering* 15.3 () (vid. págs. 6, 42).

[31] F. Hutter, L. Kotthoff y J. Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019 (vid. págs. 7, 8).

[32] T. Elsken, J. H. Metzen y F. Hutter. “Neural Architecture Search: A Survey”. English. En: *Journal of Machine Learning Research* 20 (2019) (vid. pág. 8).

[33] B. Baker, O. Gupta, N. Naik y R. Raskar. “Designing neural network architectures using reinforcement learning”. En: *arXiv preprint arXiv:1611.02167* (2016) (vid. pág. 8).

[34] B. Zoph y Q. V. Le. “Neural architecture search with reinforcement learning”. En: *arXiv preprint arXiv:1611.01578* (2016) (vid. pág. 8).

[35] C.-H. Hsu, S.-H. Chang, J.-H. Liang, H.-P. Chou y col. “Monas: Multi-objective neural architecture search using reinforcement learning”. En: *arXiv preprint arXiv:1806.10332* (2018) (vid. pág. 8).

[36] H. Cai, T. Chen, W. Zhang, Y. Yu y J. Wang. “Efficient architecture search by network transformation”. En: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018 (vid. pág. 8).

[37] D. Whitley, T. Starkweather y C. Bogart. “Genetic algorithms and neural networks: Optimizing connections and connectivity”. En: *Parallel computing* 14.3 (1990), págs. 347-361 (vid. pág. 8).

[38] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman y W. Banzhaf. “Nsga-net: neural architecture search using multi-objective genetic algorithm”. En: *Proceedings of the Genetic and Evolutionary Computation Conference*. 2019, págs. 419-427 (vid. pág. 8).

[39] E. Miahi, S. Mirroshandel y A. Nasr. “Genetic Neural Architecture Search for automatic assessment of human sperm images”. En: *Expert Systems with Applications* (2021), pág. 115937 (vid. pág. 8).

[40] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal y col. “Chapter 15 - Evolving Deep Neural Networks”. En: *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Ed. por R. Kozma, C. Alippi, Y. Choe y F. C. Morabito. Academic Press, 2019, págs. 293-312 (vid. pág. 8).

[41] M. Feurer y F. Hutter. “Hyperparameter optimization”. En: *Automated machine learning*. Springer, Cham, 2019, págs. 3-33 (vid. pág. 8).

[42] S. D. Binitha, S. S. Sathya y col. “A survey of bio inspired optimization algorithms”. En: *International journal of soft computing and engineering* 2.2 (2012), págs. 137-151 (vid. pág. 8).

[43] E. Bonabeau, G. Theraulaz y M. Dorigo. *Swarm intelligence*. Springer, 1999 (vid. pág. 8).

[44] T. Back. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996 (vid. págs. 9, 10).

[45] C. Darwin. *On the origin of species, 1859*. Routledge, 2004 (vid. pág. 9).

[46] J. Kennedy y R. Eberhart. “Particle swarm optimization”. En: *Proceedings of ICNN’95-international conference on neural networks*. Vol. 4. IEEE. 1995, págs. 1942-1948 (vid. págs. 9, 84).

[47] C. Zhang, H. Shao e Y. Li. “Particle swarm optimisation for evolving artificial neural network”. En: *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.’cybernetics evolving to systems, humans, organizations, and their complex interactions’(cat. no. 0*. Vol. 4. IEEE. 2000, págs. 2487-2490 (vid. págs. 9, 10).

[48] M. Carvalho y T. B. Ludermir. “Particle swarm optimization of neural network architectures and weights”. En: *7th International Conference on Hybrid Intelligent Systems (HIS 2007)*. IEEE. 2007, págs. 336-339 (vid. págs. 9, 10).

[49] J. Yu, L. Xi y S. Wang. “An improved particle swarm optimization for evolving feedforward artificial neural networks”. En: *Neural Processing Letters* 26.3 (2007), págs. 217-231 (vid. págs. 9, 10).

[50] T. Sinha, A. Haidar y B. Verma. “Particle swarm optimization based approach for finding optimal values of convolutional neural network parameters”. En: *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2018, págs. 1-6 (vid. pág. 9).

[51] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li y H. Han. “Efficient network architecture search via multiobjective particle swarm optimization based on decomposition”. En: *Neural Networks* 123 (2020), págs. 305-316 (vid. pág. 9).

[52] B. Wang, Y. Sun, B. Xue y M. Zhang. “Evolving Deep Convolutional Neural Networks by Variable-Length Particle Swarm Optimization for Image Classification”. En: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, págs. 1-8 (vid. págs. 9, 10).

[53] F. E. F. Junior y G. G. Yen. “Particle swarm optimization of deep neural networks architectures for image classification”. En: *Swarm and Evolutionary Computation* 49 (2019), págs. 62-74 (vid. págs. 9, 10).

[54] T. Y. Tan, L. Zhang y C. P. Lim. “Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models”. En: *Applied Soft Computing* 84 (2019), pág. 105725 (vid. págs. 9, 10).

[55] J. Huang, B. Xue, Y. Sun y M. Zhang. “A Flexible Variable-length Particle Swarm Optimization Approach to Convolutional Neural Network Architecture Design”. En: *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2021, págs. 934-941 (vid. págs. 9, 10).

[56] X. Liu, C. Zhang, Z. Cai, J. Yang, Z. Zhou y X. Gong. “Continuous Particle Swarm Optimization-Based Deep Learning Architecture Search for Hyperspectral Image Classification”. En: *Remote Sensing* 13.6 (2021), pág. 1082 (vid. pág. 9).

[57] Y. Li, J. Xiao, Y. Chen y L. Jiao. “Evolving deep convolutional neural networks by quantum behaved particle swarm optimization with binary encoding for image classification”. En: *Neurocomputing* 362 (2019), págs. 156-165 (vid. pág. 9).

[58] J. Sun, W. Xu y B. Feng. “A global search strategy of quantum-behaved particle swarm optimization”. En: *IEEE Conference on Cybernetics and Intelligent Systems, 2004.* Vol. 1. 2004, 111-116 vol.1 (vid. pág. 9).

[59] H. Larochelle, D. Erhan, A. Courville, J. Bergstra e Y. Bengio. “An empirical evaluation of deep architectures on problems with many factors of variation”. En: *Proceedings of the 24th international conference on Machine learning*. 2007, págs. 473-480 (vid. pág. 9).

[60] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner. “Gradient-based learning applied to document recognition”. En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324 (vid. pág. 9).

[61] L. Prechelt y col. *Proben1: A set of neural network benchmark problems and benchmarking rules*. Technical Report 21/94. Fakultät für Informatik, Universität Karlsruhe, Germany, 1994 (vid. pág. 10).

[62] D. Dua y C. Graff. *UCI Machine Learning Repository*. 2017. URL: `http://archive.ics.uci.edu/ml` (visitado 8 de ene. de 2022) (vid. pág. 10).

[63] J. León, J. J. Escobar, A. Ortiz, J. Ortega y col. “Deep learning for EEG-based Motor Imagery classification: Accuracy-cost trade-off”. En: *Plos one* 15.6 (2020), e0234178 (vid. págs. 11, 43).

[64] E. Rapaport, O. Shriki y R. Puzis. “EEGNAS: Neural architecture search for electroencephalography data analysis and decoding”. En: *International Workshop on Human Brain and Artificial Intelligence*. Springer. 2019, págs. 3-20 (vid. pág. 11).

[65] M. Baldeon-Calisto y S. K. Lai-Yuen. “AdaResU-Net: Multiobjective adaptive convolutional neural network for medical image segmentation”. En: *Neurocomputing* 392 (2020), págs. 325-340 (vid. pág. 11).

[66] O. Ronneberger, P. Fischer y T. Brox. “U-net: Convolutional networks for biomedical image segmentation”. En: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, págs. 234-241 (vid. pág. 11).

[67] H. Louati, S. Bechikh, A. Louati, A. Aldaej y L. B. Said. “Evolutionary Optimization of Convolutional Neural Network Architecture Design for Thoracic X-Ray Image Classification”. En: *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*. Springer. 2021, págs. 121-132 (vid. pág. 11).

[68] K. O. Stanley, J. Clune, J. Lehman y R. Miikkulainen. “Designing neural networks through neuroevolution”. En: *Nature Machine Intelligence* 1.1 (2019), págs. 24-35 (vid. pág. 11).

[69] Z. Fan, J. Wei, G. Zhu, J. Mo y W. Li. “Evolutionary neural architecture search for retinal vessel segmentation”. En: *arXiv preprint arXiv:2001.06678* (2020) (vid. pág. 12).

[70] D. Kimovski, J. Ortega, A. Ortiz y R. Banos. “Feature selection in high-dimensional EEG data by parallel multi-objective optimization”. En: *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2014, págs. 314-322 (vid. pág. 12).

[71] P. Martín-Smith, J. Ortega, J. Asensio-Cubero, J. Q. Gan y A. Ortiz. “A label-aided filter method for multi-objective feature selection in EEG classification for BCI”. En: *International Work-Conference on Artificial Neural Networks*. Springer. 2015, págs. 133-144 (vid. págs. 12, 43).

[72] J. Ortega, J. Asensio-Cubero, J. Q. Gan y A. Ortiz. “Classification of motor imagery tasks for BCI with multiresolution analysis and multiobjective feature selection”. En: *Biomedical engineering online* 15.1 (2016), págs. 149-164 (vid. págs. 12, 43).

[73] P. Martín-Smith, J. Ortega, J. Asensio-Cubero, J. Q. Gan y A. Ortiz. “A supervised filter method for multi-objective feature selection in EEG classification based on multi-resolution analysis for BCI”. En: *Neurocomputing* 250 (2017), págs. 45-56 (vid. págs. 12, 43).

[74] J. González, J. Ortega, M. Damas, P. Martín-Smith y J. Q. Gan. “A new multi-objective wrapper method for feature selection – Accuracy and stability analysis for BCI”. En: *Neurocomputing* 333 (2019), págs. 407-418 (vid. pág. 12).

[75] D. Aquino-Brítez, A. Ortiz, J. Ortega, J. León, M. A. Formoso, J. Q. Gan y J. J. Escobar. “Optimization of Deep Architectures for EEG Signal Classification: An AutoML Approach Using Evolutionary Algorithms”. En: *Sensors* 21.6, 2096 (2021) (vid. pág. 14).

[76] M. Formoso, A. Ortiz, F. J. Martínez-Murcia, D. Aquino-Brítez, J. J. Escobar y J. L. Luque. “Temporal phase synchrony disruption in Dyslexia: anomaly patterns in auditory processing”. En: *International Work-Conference on the Interplay Between Natural and Artificial Computation*. 2022 (vid. pág. 14).

[77] A. Ortiz, P. J. López, J. L. Luque, F. J. Martínez-Murcia, D. Aquino-Britez y J. Ortega. “An Anomaly Detection Approach for Dyslexia Diagnosis Using EEG Signals”. En: *Understanding the Brain Function and Emotions*. Ed. por J. M. Ferrández Vicente, J. R. Álvarez-Sánchez, F. de la Paz López, J. Toledo Moreo y H. Adeli. Cham: Springer International Publishing, 2019, págs. 369-378 (vid. págs. 14, 122, 123).

[78] E. Donchin, K. Spencer y R. Wijesinghe. “The mental prosthesis: assessing the speed of a P300-based brain-computer interface”. En: *IEEE Transactions on Rehabilitation Engineering* 8.2 (2000), págs. 174-179 (vid. pág. 17).

[79] H. Berger. “Über das Elektrenkephalogramm des Menschen. XIV.” En: *Archiv für Psychiatrie und Nervenkrankheiten* (1938) (vid. pág. 19).

[80] O. Foerster. “The motor cortex in man in the light of Hughlings Jackson’s doctrines”. En: *Brain* 59.2 (1936), págs. 135-159 (vid. pág. 19).

[81] L. Sarikcioglu. “Otfrid Foerster (1873-1941): one of the distinguished neuroscientists of his time”. En: *Journal of neurology, neurosurgery, and psychiatry* 78.6 (2007), págs. 650-650 (vid. pág. 19).

[82] I. B. Levitan, I. B. Levitan, L. K. Kaczmarek y col. *The neuron: cell and molecular biology*. Oxford University Press, USA, 2002 (vid. pág. 20).

[83] A. Gramfort. “Mapping, timing and tracking cortical activations with MEG and EEG: Methods and application to human vision”. Tesis doct. Ecole nationale supérieure des telecommunications-ENST, 2009. URL: `https://tel.archives-ouvertes.fr/tel-00426852` (visitado 4 de ene. de 2022) (vid. pág. 20).

[84] C. Lustenberger y R. Huber. “High Density Electroencephalography in Sleep Research: Potential, Problems, Future Perspective”. En: *Frontiers in neurology* 3 (2012), pág. 77 (vid. pág. 22).

[85] F. Sepulveda y A. Barrera. “Brain-actuated control of robot navigation”. En: *Advances in Robot Navigation* 8 (2011) (vid. pág. 22).

[86] W. E. Bingaman y J. Bulacio. “Placement of subdural grids in pediatric patients: technique and results”. En: *Child’s Nervous System* 30.11 (2014), págs. 1897-1904 (vid. pág. 22).

[87] Y. Mohan, S. S. Chee, D. K. P. Xin y L. P. Foong. “Artificial neural network for classification of depressive and normal in EEG”. En: *2016 IEEE EMBS conference on biomedical engineering and sciences (IECBES)*. IEEE. 2016, págs. 286-290 (vid. pág. 24).

[88] A. C. Atencio, T. F. B. Filho, A. Ferreira y A. B. Benevides. “Evaluation of ERD/ERS caused by unpleasant sounds to be applied in BCIs”. En: *2013 ISSNIP Biosignals and Biorobotics Conference: Biosignals and Robotics for Better and Safer Living (BRC)*. 2013, págs. 1-5 (vid. pág. 26).

[89] C. S. Nam, A. Nijholt y F. Lotte. *Brain–computer interfaces handbook: technological and theoretical advances*. CRC Press, 2018 (vid. pág. 28).

[90] S. S. Alam y M. I. H. Bhuiyan. “Detection of seizure and epilepsy using higher order statistics in the EMD domain”. En: *IEEE journal of biomedical and health informatics* 17.2 (2013), págs. 312-318 (vid. pág. 29).

[91] A. T. Tzallas, M. G. Tsipouras y D. I. Fotiadis. “Automatic seizure detection based on time-frequency analysis and artificial neural networks”. En: *Computational intelligence and neuroscience* 2007 (2007) (vid. pág. 30).

[92] H. Adeli, S. Ghosh-Dastidar y N. Dadmehr. “A wavelet-chaos methodology for analysis of EEGs and EEG subbands to detect seizure and epilepsy”. En: *IEEE Transactions on Biomedical Engineering* 54.2 (2007), págs. 205-211 (vid. pág. 30).

[93] B. Schack, H. Witte y G. Grießbach. “Parametrische Methoden der dynamischen Spektralanalyse und ihre Anwendung in der Biosignalanalyse”. En: (1993) (vid. pág. 30).

[94] A. Ortiz, F. J. Martinez-Murcia, J. L. Luque, A. Giménez, R. Morales-Ortega y J. Ortega. “Dyslexia diagnosis by eeg temporal and spectral descriptors: An anomaly detection approach”. En: *International Journal of Neural Systems* 30.07 (2020), pág. 2050029 (vid. págs. 30, 123).

[95] N. Hazarika, J. Z. Chen, A. C. Tsoi y A. Sergejew. “Classification of EEG signals using the wavelet transform”. En: *Signal processing* 59.1 (1997), págs. 61-72 (vid. pág. 30).

[96] D. P. Subha, P. K. Joseph, R. Acharya U., C. M. Lim y col. “EEG signal analysis: a survey”. En: *Journal of medical systems* 34.2 (2010), págs. 195-212 (vid. pág. 30).

[97] K. K. Parhi y M. Ayinala. “Low-complexity Welch power spectral density computation”. En: *IEEE Transactions on Circuits and Systems I: Regular Papers* 61.1 (2013), págs. 172-182 (vid. pág. 30).

[98] P. Welch. “The use of fast Fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms”. En: *IEEE Transactions on audio and electroacoustics* 15.2 (1967), págs. 70-73 (vid. pág. 30).

[99] D. J. Thomson. “Spectrum estimation and harmonic analysis”. En: *Proceedings of the IEEE* 70.9 (1982), págs. 1055-1096 (vid. págs. 31, 123).

[100] D. Gabor. “Theory of communication. Part 1: The analysis of information”. En: *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering* 93.26 (1946), págs. 429-441 (vid. pág. 32).

[101] M. X. Cohen. *Analyzing neural time series data: theory and practice*. MIT press, 2014 (vid. pág. 32).

[102] P. S. Addison, J. Walker y R. C. Guido. “Time–frequency analysis of biosignals”. En: *IEEE Engineering in Medicine and Biology Magazine* 28.5 (2009), págs. 14-29 (vid. pág. 32).

[103] T. Gandhi, B. K. Panigrahi y S. Anand. “A comparative study of wavelet families for EEG signal classification”. En: *Neurocomputing* 74.17 (2011), págs. 3051-3057 (vid. pág. 32).

[104] D. Cvetkovic, E. D. Übeyli e I. Cosic. “Wavelet transform feature extraction from human PPG, ECG, and EEG signal responses to ELF PEMF exposures: A pilot study”. En: *Digital signal processing* 18.5 (2008), págs. 861-874 (vid. pág. 32).

[105] B. P. Lathi y R. A. Green. *Linear systems and signals*. Vol. 2. Oxford University Press New York, 2005 (vid. pág. 33).

[106] A. D. Poularikas. *Transforms and applications handbook*. CRC press, 2018 (vid. pág. 34).

[107] S. Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". En: *Fundamental Papers in Wavelet Theory*. Princeton University Press, 2009, págs. 494-513 (vid. págs. 35, 37).

[108] S. Mallat. *A wavelet tour of signal processing*. Elsevier, 1999 (vid. pág. 35).

[109] C. Guger, G. Edlinger, W. Harkam, I. Niedermayer y G. Pfurtscheller. "How many people are able to operate an EEG-based brain-computer interface (BCI)?" En: *IEEE transactions on neural systems and rehabilitation engineering* 11.2 (2003), págs. 145-147 (vid. págs. 37, 39).

[110] L. F. Nicolas-Alonso y J. Gomez-Gil. "Brain computer interfaces, a review". En: *sensors* 12.2 (2012), págs. 1211-1279 (vid. pág. 37).

[111] A. Bhardwaj, A. Gupta, P. Jain, A. Rani y J. Yadav. "Classification of human emotions from EEG signals using SVM and LDA Classifiers". En: *2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE. 2015, págs. 180-185 (vid. págs. 38, 43).

[112] B. Graimann, G. Townsend, J. E. Huggins, A. Schlögl, S. Levine y G. Pfurtscheller. "A comparison between using ECoG and EEG for direct brain communication". En: *Proceedings of the EMBEC05* (2005) (vid. pág. 39).

[113] M. M. Moore. "Real-world applications for brain-computer interface technology". En: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 11.2 (2003), págs. 162-165 (vid. pág. 39).

[114] M. H. Lee, C. D. Smyser y J. S. Shimony. "Resting-state fMRI: a review of methods and clinical applications". En: *American Journal of neuroradiology* 34.10 (2013), págs. 1866-1872 (vid. pág. 39).

[115] M. Ferrari y V. Quaresima. "A brief review on the history of human functional near-infrared spectroscopy (fNIRS) development and fields of application". En: *Neuroimage* 63.2 (2012), págs. 921-935 (vid. pág. 40).

[116] J. R. Wolpaw, D. J. McFarland, G. W. Neat y C. A. Forneris. "An EEG-based brain-computer interface for cursor control". En: *Electroencephalography and clinical neurophysiology* 78.3 (1991), págs. 252-259 (vid. pág. 40).

[117] E. Donchin, K. M. Spencer y R. Wijesinghe. "The mental prosthesis: assessing the speed of a P300-based brain-computer interface". En: *IEEE transactions on rehabilitation engineering* 8.2 (2000), págs. 174-179 (vid. pág. 40).

[118] F. Schneider, B. Rockstroh, H. Heimann, W. Lutzenberger y col. "Self-regulation of slow cortical potentials in psychiatric patients: Schizophrenia". En: *Biofeedback and Self-regulation* 17.4 (1992), págs. 277-292 (vid. pág. 40).

[119] T. Hinterberger, S. Schmidt, N. Neumann, J. Mellinger, B. Blankertz, G. Curio y N. Birbaumer. "Brain-computer communication and slow cortical potentials". En: *IEEE Transactions on Biomedical Engineering* 51.6 (2004), págs. 1011-1018 (vid. pág. 41).

[120] T. Mulder. “Motor imagery and action observation: cognitive tools for rehabilitation”. En: *Journal of neural transmission* 114.10 (2007), págs. 1265-1278 (vid. pág. 41).

[121] G. Pfurtscheller y C. Neuper. “Motor imagery activates primary sensorimotor area in humans”. En: *Neuroscience letters* 239.2-3 (1997), págs. 65-68 (vid. pág. 41).

[122] R. Abiri, S. Borhani, E. W. Sellers, Y. Jiang y X. Zhao. “A comprehensive review of EEG-based brain–computer interface paradigms”. En: *Journal of Neural Engineering* 16.1 (2019), pág. 011001 (vid. pág. 41).

[123] J. R. Wolpaw, D. J. McFarland, T. M. Vaughan y G. Schalk. “The Wadsworth Center brain-computer interface (BCI) research and development program”. En: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 11.2 (2003), págs. 1-4 (vid. pág. 42).

[124] S. Siuly e Y. Zhang. “Medical big data: neurological diseases diagnosis through medical data analysis”. En: *Data Science and Engineering* 1.2 (2016), págs. 54-64 (vid. pág. 42).

[125] R. Palaniappan, C. S. Syan y R. Paramesran. “Current practices in electroencephalogram-based brain-computer interfaces”. En: *Encyclopedia of Information Science and Technology, Second Edition*. IGI Global, 2009, págs. 888-901 (vid. pág. 42).

[126] K.-R. Muller, C. W. Anderson y G. E. Birch. “Linear and nonlinear methods for brain-computer interfaces”. En: *IEEE transactions on neural systems and rehabilitation engineering* 11.2 (2003), págs. 165-169 (vid. pág. 43).

[127] S. Suthaharan. “Support vector machine”. En: *Machine learning models and algorithms for big data classification*. Springer, 2016, págs. 207-235 (vid. pág. 43).

[128] R. A. Fisher. “The use of multiple measurements in taxonomic problems”. En: *Annals of eugenics* 7.2 (1936), págs. 179-188 (vid. pág. 43).

[129] A. Craik, Y. He y J. L. Contreras-Vidal. “Deep learning for electroencephalogram (EEG) classification tasks: a review”. En: *Journal of neural engineering* 16.3 (2019), pág. 031001 (vid. pág. 43).

[130] G. Costantini, M. Todisco, D. Casali, M. Carota y col. “SVM classification of EEG signals for brain computer interface”. En: *Neural Nets WIRN09*. IOS Press, 2009, págs. 229-233 (vid. pág. 43).

[131] X. Li, X. Chen, Y. Yan, W. Wei y Z. J. Wang. “Classification of EEG signals using a multiple kernel learning support vector machine”. En: *Sensors* 14.7 (2014), págs. 12784-12802 (vid. pág. 43).

[132] K. Mebarkia y A. Reffad. “Multi optimized SVM classifiers for motor imagery left and right hand movement identification”. En: *Australasian physical & engineering sciences in medicine* 42.4 (2019), págs. 949-958 (vid. pág. 43).

[133] L. Zhiwei y S. Minfen. “Classification of mental task EEG signals using wavelet packet entropy and SVM”. En: *2007 8th international conference on electronic measurement and instruments*. IEEE. 2007, págs. 3-906 (vid. pág. 43).

[134] L. Wang, X. Zhang, X. Zhong e Y. Zhang. “Analysis and classification of speech imagery EEG for BCI”. En: *Biomedical signal processing and control* 8.6 (2013), págs. 901-908 (vid. pág. 43).

[135] R. Boostani, B. Graimann, M. H. Moradi y G. Pfurtscheller. “A comparison approach toward finding the best feature and classifier in cue-based BCI”. En: *Medical & biological engineering & computing* 45.4 (2007), págs. 403-412 (vid. pág. 43).

[136] J. Asensio-Cubero, J. Q. Gan y R. Palaniappan. “Multiresolution analysis over simple graphs for brain computer interfaces”. En: *Journal of neural engineering* 10.4 (2013), pág. 046014 (vid. págs. 43, 129).

[137] Y. Wang y K. C. Veluvolu. “Evolutionary algorithm based feature optimization for multi-channel EEG classification”. En: *Frontiers in neuroscience* 11 (2017), pág. 28 (vid. pág. 43).

[138] J. León, J. Ortega y A. Ortiz. “Convolutional neural networks and feature selection for BCI with multiresolution analysis”. En: *International Work-Conference on Artificial Neural Networks*. Springer. 2019, págs. 883-894 (vid. pág. 43).

[139] Y. R. Tabar y U. Halici. “A novel deep learning approach for classification of EEG motor imagery signals”. En: *Journal of neural engineering* 14.1 (2016), pág. 016003 (vid. pág. 43).

[140] J. León, A. Ortiz, M. Damas, J. González y J. Ortega. “Cost-Efficiency of Convolutional Neural Networks for High-Dimensional EEG Classification”. En: *International Conference on Hybrid Artificial Intelligence Systems*. Springer. 2020, págs. 774-785 (vid. pág. 43).

[141] J. León, J. J. Escobar, J. González, J. Ortega, F. M. Arrabal-Campos, A. Ortiz y M. Damas. “Recurrent Neural Networks and Efficiency in High-Dimensional EEG Classification”. En: *International Conference on Bioengineering and Biomedical Signal and Image Processing*. Springer. 2021, págs. 297-310 (vid. pág. 43).

[142] Y. LeCun, Y. Bengio y G. Hinton. “Deep learning”. En: *Nature* 521.7553 (2015), págs. 436-444 (vid. pág. 45).

[143] F. Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain.” En: *Psychological Review* 65.6 (1958), págs. 386-408 (vid. pág. 46).

[144] M. Minsky y S. A. Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017 (vid. pág. 46).

[145] P. J. Werbos. “Backpropagation through time: what it does and how to do it”. En: *Proceedings of the IEEE* 78.10 (1990), págs. 1550-1560 (vid. pág. 46).

[146] D. E. Rumelhart, G. E. Hinton y R. J. Williams. “Learning representations by back-propagating errors”. En: *nature* 323.6088 (1986), págs. 533-536 (vid. pág. 46).

[147] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard y L. D. Jackel. “Backpropagation applied to handwritten zip code recognition”. En: *Neural computation* 1.4 (1989), págs. 541-551 (vid. pág. 46).

[148] S. Hochreiter y J. Schmidhuber. “Long short-term memory”. En: *Neural computation* 9.8 (1997), págs. 1735-1780 (vid. pág. 47).

[149] G. E. Hinton, S. Osindero e Y.-W. Teh. “A fast learning algorithm for deep belief nets”. En: *Neural computation* 18.7 (2006), págs. 1527-1554 (vid. pág. 47).

[150] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu y col. “Generative Adversarial Nets”. En: 27 (2014). Ed. por Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence y K. Weinberger (vid. pág. 47).

[151] V. Nair y G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. En: *Icml*. 2010 (vid. pág. 49).

[152] Y. Bengio, I. Goodfellow y A. Courville. *Deep learning*. Vol. 1. MIT press Cambridge, MA, USA, 2017 (vid. págs. 49, 58).

[153] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo y col. *Tensorflow Documentation: Module: tf.keras.losses*. 2015. URL: `https://www.tensorflow.org/api_docs/python/tf/keras/losses` (visitado 5 de ene. de 2022) (vid. pág. 56).

[154] J. Duchi, E. Hazan e Y. Singer. “Adaptive subgradient methods for online learning and stochastic optimization.” En: *Journal of machine learning research* 12.7 (2011) (vid. pág. 58).

[155] H. B. McMahan y M. Streeter. “Adaptive bound optimization for online convex optimization”. En: *arXiv preprint arXiv:1002.4908* (2010) (vid. pág. 58).

[156] D. P. Kingma y J. Ba. “Adam: A method for stochastic optimization”. En: *arXiv preprint arXiv:1412.6980* (2014) (vid. pág. 59).

[157] D. M. Hawkins. “The Problem of Overfitting”. En: *Journal of Chemical Information and Computer Sciences* 44.1 (2004), págs. 1-12 (vid. pág. 61).

[158] S. J. Raudys, A. K. Jain y col. “Small sample size effects in statistical pattern recognition: Recommendations for practitioners”. En: *IEEE Transactions on pattern analysis and machine intelligence* 13.3 (1991), págs. 252-264 (vid. pág. 61).

[159] R. P. Duin. “Classifiers in almost empty spaces”. En: *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*. Vol. 2. IEEE. 2000, págs. 1-7 (vid. pág. 61).

[160] S.-i. Amari, N. Murata, K.-R. Muller, M. Finke y H. H. Yang. “Asymptotic statistical theory of overtraining and cross-validation”. En: *IEEE Transactions on Neural Networks* 8.5 (1997), págs. 985-996 (vid. pág. 61).

[161] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting”. En: *The journal of machine learning research* 15.1 (2014), págs. 1929-1958 (vid. pág. 61).

[162] L. Prechelt. “Early stopping-but when?” En: *Neural Networks: Tricks of the trade*. Springer, 1998, págs. 55-69 (vid. pág. 62).

[163] W. C. Navidi. *Statistics for engineers and scientists*. Vol. 2. 4. McGraw-Hill Higher Education New York, NY, USA, 2008 (vid. págs. 63, 134).

[164] J. Cohen. “A coefficient of agreement for nominal scales”. En: *Educational and psychological measurement* 20.1 (1960), págs. 37-46 (vid. pág. 66).

[165] Q. Ke, J. Liu, M. Bennamoun, S. An, F. Sohel y F. Boussaid. “Chapter 5 - Computer Vision for Human–Machine Interaction”. En: *Computer Vision for Assistive Healthcare*. Ed. por M. Leo y G. M. Farinella. Computer Vision and Pattern Recognition. Academic Press, 2018, págs. 127-145 (vid. pág. 66).

[166] A. Krizhevsky, I. Sutskever y G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. En: *Advances in neural information processing systems* 25 (2012), págs. 1097-1105 (vid. pág. 66).

[167] K. Simonyan y A. Zisserman. “Very deep convolutional networks for large-scale image recognition”. En: *arXiv preprint arXiv:1409.1556* (2014) (vid. pág. 66).

[168] G. Huang, Z. Liu, L. Van Der Maaten y K. Q. Weinberger. “Densely connected convolutional networks”. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, págs. 4700-4708 (vid. pág. 66).

[169] J. Schmidhuber. “Deep learning in neural networks: An overview”. En: *Neural Networks* 61 (2015), págs. 85-117 (vid. pág. 66).

[170] L. Hertel, E. Barth, T. Käster y T. Martinetz. “Deep convolutional neural networks as generic feature extractors”. En: *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015, págs. 1-4 (vid. pág. 66).

[171] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj y D. J. Inman. “1D convolutional neural networks and applications: A survey”. En: *Mechanical systems and signal processing* 151 (2021), pág. 107398 (vid. págs. 67, 76).

[172] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko y col. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. En: *arXiv preprint arXiv:1704.04861* (2017) (vid. págs. 69, 70, 72).

[173] F. Chollet. “Xception: Deep learning with depthwise separable convolutions”. En: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, págs. 1251-1258 (vid. pág. 70).

[174] L. Sifre. “Rigid-motion scattering for image classification author”. Tesis doct. 2014. URL: `https://www.di.ens.fr/data/publications/papers/phd_sifre.pdf` (visitado 28 de ene. de 2022) (vid. pág. 72).

[175] K. Paupamah, S. James y R. Klein. “Quantisation and pruning for neural network compression and regularisation”. En: *2020 International SAUPEC/RobMech/PRASA Conference*. IEEE. 2020, págs. 1-6 (vid. pág. 73).

[176] S. Ioffe y C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. En: *International conference on machine learning*. PMLR. 2015, págs. 448-456 (vid. pág. 74).

[177] S. Albawi, T. A. Mohammed y S. Al-Zawi. “Understanding of a convolutional neural network”. En: *2017 International Conference on Engineering and Technology (ICET)*. 2017, págs. 1-6 (vid. pág. 75).

[178] S. K. Pal. “Multilayer Perceptron, Fuzzy Sets, and Classification”. En: *IEEE Transactions on Neural Networks* 3.5 (1992), pág. 683 (vid. pág. 75).

[179] S. Kiranyaz, T. Ince, R. Hamila y M. Gabbouj. “Convolutional neural networks for patient-specific ECG classification”. En: *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE. 2015, págs. 2608-2611 (vid. pág. 75).

[180] T. Ince, S. Kiranyaz, L. Eren, M. Askar y M. Gabbouj. “Real-time motor fault detection by 1-D convolutional neural networks”. En: *IEEE Transactions on Industrial Electronics* 63.11 (2016), págs. 7067-7075 (vid. pág. 75).

[181] O. Abdeljaber, O. Avci, M. S. Kiranyaz, B. Boashash, H. Sodano y D. J. Inman. “1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data”. En: *Neurocomputing* 275 (2018), págs. 1308-1317 (vid. pág. 75).

[182] S. Kiranyaz, A. Gastli, L. Ben-Brahim, N. Al-Emadi y M. Gabbouj. “Real-time fault detection and identification for MMC using 1-D convolutional neural networks”. En: *IEEE Transactions on Industrial Electronics* 66.11 (2018), págs. 8760-8771 (vid. pág. 75).

[183] A. Antoniades, L. Spyrou, C. C. Took y S. Sanei. “Deep learning for epileptic intracranial EEG data”. En: *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2016, págs. 1-6 (vid. pág. 76).

[184] R. T. Schirrmeister, J. T. Springenberg, L. D. J. Fiederer, M. Glasstetter y col. “Deep learning with convolutional neural networks for EEG decoding and visualization”. En: *Human Brain Mapping* 38.11 (2017), págs. 5391-5420. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1002/hbm.23730` (vid. pág. 76).

[185] J. Shamwell, Y. Lee, H. Kwon, A. R. Marathe, V. Lawhern y W. Nothwang. “Single-Trial EEG RSVP Classification using Convolutional Neural Networks”. En: ed. por T. George, A. K. Dutta y M. S. Islam. Vol. 9836. Conference on Micro- and Nanotechnology Sensors, Systems, and Applications VIII. 2016 (vid. pág. 76).

[186] S. R. Tibor, S. J. Tobias, F. L. D. Josef, G. Martin y col. “Deep learning with convolutional neural networks for EEG decoding and visualization”. En: *Human Brain Mapping* 38.11 (), págs. 5391-5420 (vid. págs. 77, 78).

[187] V. J. Lawhern, A. J. Solon, N. R. Waytowich, S. M. Gordon, C. P. Hung y B. J. Lance. “EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces”. En: *Journal of Neural Engineering* 15.5 (2018), pág. 056013 (vid. págs. 78, 128, 134, 135, 137).

[188] M. Luptacik. *Mathematical Optimization and Economic Analysis*. Vol. 36. Mathematical Optimization and Economic Analysis. Springer, 2010, págs. 1-292 (vid. pág. 81).

[189] N. Forbes. *Imitation of life: how biology is inspiring computing*. Mit Press, 2004 (vid. pág. 81).

[190] S. Mirjalili. “Genetic algorithm”. En: *Evolutionary algorithms and neural networks*. Springer, 2019, págs. 43-55 (vid. pág. 81).

[191] M. Abdel-Basset, L. Abdel-Fatah y A. K. Sangaiah. “Metaheuristic algorithms: A comprehensive review”. En: *Computational intelligence for multimedia big data on the cloud with engineering applications* (2018), págs. 185-231 (vid. pág. 81).

[192] N. Siddique y H. Adeli. “Nature inspired computing: an overview and some future directions”. En: *Cognitive computation* 7.6 (2015), págs. 706-714 (vid. pág. 82).

[193] M. Basseur, E.-G. Talbi, A. Nebro y E. Alba. “Metaheuristics for Multiobjective Combinatorial Optimization Problems: Review and recent issues”. En: (2006) (vid. pág. 82).

[194] D. J. White. “The set of efficient solutions for multiple objective shortest path problems”. En: *Computers & Operations Research* 9.2 (1982), págs. 101-107 (vid. pág. 82).

[195] T. Sen, F. M. Raiszadeh y P. Dileepan. “Note—A Branch-and-Bound Approach to the Bicriterion Scheduling Problem Involving Total Flowtime and Range of Lateness”. En: *Management Science* 34.2 (1988), págs. 254-260 (vid. pág. 82).

[196] L. Mandow y E. Milián. “Goal programming and heuristic search”. En: *Advances in multiple objective and goal programming*. Springer, 1997, págs. 48-56 (vid. pág. 83).

[197] D. Navinchandra. *Exploration and innovation in design: towards a computational model*. Springer Science & Business Media, 2012 (vid. pág. 83).

[198] F. Rossi, P. Van Beek y T. Walsh. *Handbook of constraint programming*. Elsevier, 2006, págs. 1-955 (vid. pág. 83).

[199] X. Yu y M. Gen. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010 (vid. pág. 83).

[200] P. J. Van Laarhoven y E. H. Aarts. “Simulated annealing”. En: *Simulated annealing: Theory and applications*. Springer, 1987, págs. 7-15 (vid. pág. 84).

[201] F. Glover y M. Laguna. “Tabu search”. En: *Handbook of combinatorial optimization*. Springer, 1998, págs. 2093-2229 (vid. pág. 84).

[202] M. Dorigo, M. Birattari y T. Stutzle. “Ant colony optimization”. En: *IEEE computational intelligence magazine* 1.4 (2006), págs. 28-39 (vid. pág. 84).

[203] K. Deb, A. Pratap, S. Agarwal y T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. En: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), págs. 182-197 (vid. págs. 85, 93, 94, 98, 99).

[204] K. Deb, S. Agrawal, A. Pratap y T. Meyarivan. “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II”. En: *Parallel Problem Solving from Nature PPSN VI*. Ed. por M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo y H.-P. Schwefel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, págs. 849-858 (vid. págs. 85, 93, 98).

[205] K. Deb y H. Jain. “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints”. En: *IEEE transactions on evolutionary computation* 18.4 (2013), págs. 577-601 (vid. pág. 85).

[206] E. Zitzler, M. Laumanns y L. Thiele. “SPEA2: Improving the strength Pareto evolutionary algorithm”. En: *TIK-report* 103 (2001) (vid. pág. 85).

[207] V. Pareto. *Cours d’économie politique*. Vol. 1. Librairie Droz, 1964 (vid. pág. 86).

[208] V. Pareto y col. *Manual of political economy*. AM Kelley, 1971 (vid. pág. 86).

[209] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. USA: Addison-Wesley Longman Publishing Co., Inc., 1989 (vid. pág. 88).

[210] J. H. Holland. “Outline for a logical theory of adaptive systems”. En: *Journal of the ACM (JACM)* 9.3 (1962), págs. 297-314 (vid. pág. 88).

[211] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. University of Michigan, 1975 (vid. pág. 88).

[212] G. Rudolph. “Convergence analysis of canonical genetic algorithms”. En: *IEEE transactions on neural networks* 5.1 (1994), págs. 96-101 (vid. pág. 88).

[213] K. Deb, R. B. Agrawal y col. “Simulated binary crossover for continuous search space”. En: *Complex systems* 9.2 (1995), págs. 115-148 (vid. pág. 91).

[214] F. Herrera, M. Lozano y A. M. Sánchez. “A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study”. En: *International Journal of Intelligent Systems* 18.3 (2003), págs. 309-338 (vid. pág. 91).

[215] N. Srinivas y K. Deb. “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms”. En: *Evolutionary Computation* 2.3 (1994), págs. 221-248 (vid. pág. 93).

[216] K. Deb y D. E. Goldberg. “An Investigation of Niche and Species Formation in Genetic Function Optimization”. En: *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, págs. 42-50 (vid. pág. 95).

[217] D. A. Van Veldhuizen. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. Air Force Institute of Technology, 1999 (vid. pág. 99).

[218] A. Auger, J. Bader, D. Brockhoff y E. Zitzler. “Theory of the hypervolume indicator: optimal μ-distributions and the choice of the reference point”. En: *Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms*. 2009, págs. 87-102 (vid. pág. 99).

[219] Q. Yang y S. Ding. “Novel algorithm to calculate hypervolume indicator of Pareto approximation set”. En: *International Conference on Intelligent Computing*. Springer. 2007, págs. 235-244 (vid. pág. 99).

[220] J. R. Schott. “Fault tolerant design using single and multicriteria genetic algorithm optimization”. Tesis doct. Massachusetts Institute of Technology, 1995. URL: `http://hdl.handle.net/1721.1/11582` (visitado 18 de ene. de 2022) (vid. pág. 99).

[221] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo y col. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software disponible desde tensorflow.org. 2015. URL: https://www.tensorflow.org/ (visitado 5 de ene. de 2022) (vid. pág. 103).

[222] P. Atzeni y V. De Antonellis. *Relational database theory*. Benjamin-Cummings Publishing Co., Inc., 1993 (vid. pág. 103).

[223] M. Stonebraker. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. 1996. URL: https://www.postgresql.org/ (visitado 28 de ene. de 2022) (vid. pág. 103).

[224] S. L. Emerson, M. Darnovsky y J. Bowman. *The practical SQL handbook: using structured query language*. Addison-Wesley Longman Publishing Co., Inc., 1989 (vid. pág. 103).

[225] G. Shao, F. Berman y R. Wolski. "Master/slave computing on the grid". En: *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000)(Cat. No. PR00556)*. IEEE. 2000, págs. 3-16 (vid. pág. 104).

[226] Arduino. *Arduino Mega Information*. URL: https://www.arduino.cc/en/Main/arduinoBoardMega2560/ (visitado 21 de ene. de 2022) (vid. pág. 105).

[227] G. Leeduca. *Grupo de investigación Leeduca*. 2022. URL: www.leeduca.uma.es (visitado 8 de ene. de 2022) (vid. pág. 122).

[228] G. R. Lyon, S. E. Shaywitz y B. A. Shaywitz. "A definition of dyslexia". En: *Annals of Dyslexia* 53.1 (2003), págs. 1-14 (vid. pág. 122).

[229] A. K. Engel, P. Fries y W. Singer. "Dynamic predictions: Oscillations and synchrony in top–down processing". En: *Nature Reviews Neuroscience* 2.10 (2001), págs. 704-716 (vid. pág. 124).

[230] B. Schölkopf, R. C. Williamson, A. Smola, J. Shawe-Taylor y J. Platt. "Support Vector Method for Novelty Detection". En: 12 (1999). Ed. por S. Solla, T. Leen y K. Müller (vid. pág. 124).

[231] V. N. Vapnick. *Statistical learning theory*. Wiley, New York, 1998 (vid. pág. 124).

[232] X. Zhang, Y. Tian e Y. Jin. "A knee point-driven evolutionary algorithm for many-objective optimization". En: *IEEE Transactions on Evolutionary Computation* 19.6 (2014), págs. 761-776 (vid. pág. 127).

[233] J. León, J. J. Escobar, A. Ortiz, J. Ortega y col. "Deep learning for EEG-based Motor Imagery classification: Accuracy-cost trade-off". En: *PLOS ONE* 15.6 (2020), págs. 1-30 (vid. pág. 130).

[234] P. Martín-Smith, J. Ortega, J. Asensio-Cubero, J. Q. Gan y A. Ortiz. "A supervised filter method for multi-objective feature selection in EEG classification based on multi-resolution analysis for BCI". En: *Neurocomputing* 250 (2017), págs. 45-56 (vid. pág. 130).

[235] N. Waytowich, V. J. Lawhern, J. O. Garcia, J. Cummings, J. Faller, P. Sajda y J. M. Vettel. “Compact convolutional neural networks for classification of asynchronous steady-state visual evoked potentials”. En: *Journal of Neural Engineering* 15.6 (2018), pág. 066031 (vid. pág. 135).

Apéndice. Tablas de la base de datos del framework de optimización

A

En este apéndice se detallan los atributos de cada una de las tablas utilizadas en la base de datos del framework. La nomenclatura para la columna que denota el tipo de dato utilizado ("Tipo") es la siguiente: (B) Booleano; (C) Cadena de caracteres; (E) Entero; (F) Flotante y (FH) Fecha y Hora.

Tabla A.1.: Descripción detallada de los atributos de la tabla "TiposNodos".

Atributo	Tipo	Descripción
idtiponodo (PK)	E	Identificador de la tabla que se genera de forma automática.
descripcion	C	Descripción del tipo de nodo (maestro o trabajador). El usuario se encarga de indicar el valor del atributo.

Tabla A.2.: Descripción detallada de los atributos de la tabla "Nodos".

Atributo	Tipo	Descripción
idnodo (PK)	E	Identificador de la tabla que se genera de forma automática.
idtiponodo (FK)	E	Tipo de nodo al que corresponde. Tiene dependencia con la tabla "TiposNodos". El usuario se encarga de indicar el valor del atributo.
nombre	C	Denominación del nodo. El usuario se encarga de indicar el valor del atributo.
utilizado	B	Indicador del estado actual del nodo (si está siendo utilizado o no). El valor del atributo puede ser Falso o Verdadero aunque por defecto es Falso. El valor del atributo es asignado automáticamente.
activo	B	Indicador que registra si el nodo puede ser utilizado durante el proceso de optimización. El valor del atributo puede ser Falso o Verdadero. El usuario se encarga de indicar el valor del atributo.

Tabla A.3.: Descripción detallada de los atributos de la tabla "GPUs".

Atributo	Tipo	Descripción
idgpu (PK)	E	Identificador de la tabla que se genera de forma automática.
idnodo (FK)	E	Indica el nodo al que corresponde la GPU. Tiene dependencia con la tabla "Nodos".
numero	E	Indica el número asignado a la GPU. El usuario se encarga de indicar el valor del atributo.
nombre	C	Denominación dada a la GPU. El usuario se encarga de indicar el valor del atributo.
capacmemo	E	Indicador de la capacidad de memoria de la GPU. El usuario se encarga de indicar el valor del atributo.
utilizado	B	Indicador del estado actual de la GPU (si está siendo utilizada o no). El valor del atributo puede ser Falso o Verdadero aunque su valor por defecto es Falso. El valor del atributo es asignado automáticamente.
activo	B	Indicador que registra si la GPU puede ser utilizada durante el proceso de optimización. El valor del atributo puede ser Falso o Verdadero. El usuario se encarga de indicar el valor del atributo.

Tabla A.4.: Descripción detallada de los atributos de la tabla "CPUs".

Atributo	Tipo	Descripción
idcpu (PK)	E	Identificador de la tabla que se genera de forma automática.
idnodo (FK)	E	Indica el nodo al que corresponde la CPU. Tiene dependencia con la tabla "Nodos". El usuario se encarga de indicar el valor del atributo.
numero	E	Indica el número asignado a la CPU. El usuario se encarga de indicar el valor del atributo.
nombre	C	Denominación dada a la CPU. El usuario se encarga de indicar el valor del atributo.
utilizado	B	Indicador del estado actual de la CPU (si está siendo utilizada o no). El valor del atributo puede ser Falso o Verdadero aunque su valor por defecto es Falso. El valor del atributo es asignado automáticamente.
activo	B	Indica si la CPU puede ser utilizada durante el proceso de optimización. El valor del atributo puede ser Falso o Verdadero. El usuario se encarga de indicar el valor del atributo.

Tabla A.5.: Descripción detallada de los atributos de la tabla “Capas”.

Atributo	Tipo	Descripción
idcapa (PK)	E	Identificador de la tabla que se genera de forma automática.
descripcion	C	Denominación de la capa en base a la sintaxis de Tensorflow. El usuario se encarga de indicar el valor del atributo.

Tabla A.6.: Descripción detallada de los atributos de la tabla “Argumentos”.

Atributo	Tipo	Descripción
idarg (PK)	E	Identificador de la tabla que se genera de forma automática.
idtipodato (FK)	E	Tipo de dato que puede ser asignado al argumento. Tiene dependencia con la tabla “TiposDatos”. El usuario se encarga de indicar el valor del atributo.
descripcion	C	Denominación del argumento en base a la sintaxis de Tensorflow. El usuario se encarga de indicar el valor del atributo.
valordefecto	C	Valor por defecto que tiene el atributo.

Tabla A.7.: Descripción detallada de los atributos de la tabla “CapArgs”.

Atributo	Tipo	Descripción
idcapaarg (PK)	E	Identificador de la tabla que se genera de forma automática.
idarg (FK)	E	Indica el código del argumento que corresponde a la capa. Tiene dependencia con la tabla “Argumentos”. El usuario se encarga de indicar el valor del atributo.
idcapa (FK)	E	Tipo de capa al que corresponde el argumento. Tiene dependencia con la tabla “Capas”. El usuario se encarga de indicar el valor del atributo.

Tabla A.8.: Descripción detallada de los atributos de la tabla "ArgDet".

Atributo	Tipo	Descripción
idargdet (PK)	E	Identificador de la tabla que se genera de forma automática.
idarg (FK)	E	Indica el argumento al que corresponde. Tiene dependencia con la tabla "Argumentos". El usuario se encarga de indicar el valor del atributo.
numero	E	Número entero que representará en el cromosoma al valor definido en el atributo "descripcion" y que corresponde al valor de un argumento específico de la ANN. El usuario se encarga de indicar el valor del atributo.
descripcion	C	Valor que será utilizado por el argumento de la ANN. El usuario se encarga de indicar el valor del atributo.

Tabla A.9.: Descripción detallada de los atributos de la tabla "DatosMuestras".

Atributo	Tipo	Descripción
iddatomuestra (PK)	E	Identificador de la tabla que se genera de forma automática.
nombre	C	Nombre del archivo del conjunto de datos. El usuario se encarga de indicar el valor del atributo.
arcdir	C	Ruta en el disco donde se encuentra el archivo de datos. El usuario se encarga de indicar el valor del atributo.
cantclases	E	Número de clases de los conjuntos de datos. El usuario se encarga de indicar el valor del atributo.
descripcion	C	Descripción del archivo de conjuntos de datos. El usuario se encarga de indicar el valor del atributo.

Tabla A.10.: Descripción detallada de los atributos de la tabla "TiposDatos".

Atributo	Tipo	Descripción
idtipodato (PK)	E	Identificador de la tabla que se genera de forma automática.
descripcion	C	Denominación de los diferentes tipos de datos que pueden ser utilizados. El usuario se encarga de indicar el valor del atributo.

Tabla A.11.: Descripción detallada de los atributos de la tabla "EvoAlg".

Atributo	Tipo	Descripción
idea (PK)	E	Identificador de la tabla que se genera de forma automática.
descripcion	C	Nombre de los EA que pueden ser utilizados por el procedimiento de optimización. El usuario se encarga de indicar el valor del atributo de acuerdo a los EAs disponibles en la librería de optimización multiobjetivo.

Tabla A.12.: Descripción detallada de los atributos de la tabla "Experimentos".

Atributo	Tipo	Descripción
idexp (PK)	E	Identificador de la tabla que se genera de forma automática.
descripcion	C	Descripción del experimento que será realizado. El usuario se encarga de indicar el valor del atributo.
cantcapas	E	Número máximo de capas que puede tener las ANNs generadas por el procedimiento de optimización. El usuario se encarga de indicar el valor del atributo.
abreviatura	C	Abreviatura que será asignada a un experimento determinado y utilizada como prefijo de las diferentes soluciones alcanzadas. El usuario se encarga de indicar el valor del atributo.
parametros	E	Número máximo de parámetros que pueden tener las ANNs generadas por el procedimiento de optimización. El usuario se encarga de indicar el valor del atributo.

Tabla A.13.: Descripción detallada de los atributos de la tabla “ConfExps”.

Atributo	Tipo	Descripción
idconfexp(PK)	E	Identificador de la tabla que se genera de forma automática.
idexp (FK)	E	Indica el experimento al que corresponde el registro de configuración. Tiene dependencia con la tabla “Experimentos”. El usuario se encarga de indicar el valor del atributo.
iddatomuestra (FK)	E	Indica el conjunto de datos que será utilizado para el experimento de optimización. Tiene dependencia con la tabla “DatosMuestras”. El usuario se encarga de indicar el valor del atributo.
idea (FK)	E	Indica el EA que será utilizado para el experimento. Tiene dependencia con la tabla “EvoAlg”. El usuario se encarga de indicar el valor del atributo.
totejec	E	Número de ejecuciones que se realizaran del experimento. El usuario se encarga de indicar el valor del atributo.
ea_poblacion	E	Número de individuos de la población. El valor asignado es utilizado por el algoritmo de optimización. El usuario se encarga de indicar el valor del atributo.
ea_generaciones	E	Número de generaciones. El valor asignado es utilizado por el algoritmo de optimización. El usuario se encarga de indicar el valor del atributo.
ea_cruzamiento	F	Probabilidad de cruzamiento. El valor asignado es utilizado por el algoritmo de optimización. El usuario se encarga de indicar el valor del atributo.
ea_mutacion	F	Probabilidad de mutación. El valor asignado es utilizado por el algoritmo de optimización. El usuario se encarga de indicar el valor del atributo.
ann_maxparam	E	Cantidad máxima de parámetros de la ANN que serán considerados para que la solución sea utilizada por el algoritmo de optimización. El usuario se encarga de indicar el valor del atributo.
ann_bitscalc	E	Número de bits utilizados para el cálculo. El usuario se encarga de indicar el valor del atributo.
ann_parantici	E	Cantidad máxima de iteraciones de entrenamiento en la métrica monitorizada hasta que haya dejado de mejorar. El usuario se encarga de indicar el valor del atributo.
ann_epocas	E	Cantidad máxima de épocas de entrenamiento de la ANN. El usuario se encarga de indicar el valor del atributo.

Tabla A.14.: Descripción detallada de los atributos de la tabla "ArqExpCaps".

Atributo	Tipo	Descripción
idarqexpcaps (PK)	E	Identificador de la tabla que se genera de forma automática.
idcapa (FK)	E	Tipo de capa que puede ser asignado a la posición dentro del orden de capas de la ANN. Tiene dependencia con la tabla "Capas". El usuario se encarga de indicar el valor del atributo.
idexp (FK)	E	Indica el experimento al que corresponden la configuraciones de las capas realizadas. Tiene dependencia con la tabla "Experimentos". El usuario se encarga de indicar el valor del atributo.
nrocapa	E	Indica el número de capa que corresponde en la ANN. El usuario se encarga de indicar el valor del atributo.

Tabla A.15.: Descripción detallada de los atributos de la tabla "ArqExpCapArgs".

Atributo	Tipo	Descripción
idarqexpecaparg (PK)	E	Identificador de la tabla que se genera de forma automática.
idarqexpcaps (FK)	E	Indica la capa especifica de la ANN que será optimizada en el marco del experimento al que corresponde. Tiene dependencia con la tabla "ArqExpCap". El usuario se encarga de indicar el valor del atributo.
idarg (FK)	E	Argumento que será optimizado de una capa especifica de la ANN. Tiene dependencia con la tabla "Argumentos". El usuario se encarga de indicar el valor del atributo.
rangoinicio	E	Valor inicial del rango de argumentos correspondiente a la tabla "ConfArgs" que serán objeto de optimización por parte del procedimiento. El usuario se encarga de indicar el valor del atributo.
rangofin	E	Valor final del rango de argumentos correspondiente a la tabla "ConfArgs" que serán objeto de optimización por parte del procedimiento. El usuario se encarga de indicar el valor del atributo.

Tabla A.16.: Descripción detallada de los atributos de la tabla “ConfArgs”.

Atributo	Tipo	Descripción
idconfarg (PK)	E	Identificador de la tabla que se genera de forma automática.
idarqexpcapaarg (FK)	E	Tipo de capa que puede ser asignado a la posición dentro del orden de capas de la ANN . Tiene dependencia con la tabla “ArqExpCapArgs”. El usuario se encarga de indicar el valor del atributo.
idtipodato (FK)	E	Tipo de dato al que corresponderá el valor del argumento. Tiene dependencia con la tabla “TiposDatos”. El usuario se encarga de indicar el valor del atributo.
numero	E	Número entero con el que ha sido codificado el valor del argumento para la capa seleccionada. El usuario se encarga de indicar el valor del atributo.
valor	C	Valor que será asignado al argumento de la capa seleccionada. El usuario se encarga de indicar el valor del atributo.

Tabla A.17.: Descripción detallada de los atributos de la tabla "ExpsEjec".

Atributo	Tipo	Descripción
idexpejec (PK)	E	Identificador de la tabla que se genera de forma automática.
idexp	E	Indica el experimento al que corresponde las ejecuciones del procedimiento de optimización de ANNs. Tiene dependencia con la tabla "Experimentos". El usuario se encarga de indicar el valor del atributo.
item	E	Número de ejecución del procedimiento de optimización de las ANNs. El valor del atributo es asignado automáticamente.
ea_tabla	C	Nombre de la tabla donde se alojaran las soluciones correspondientes a un ciclo de ejecución del experimento. El valor del atributo es asignado automáticamente.
ea_archivo	C	Nombre del archivo donde se registran detalles de la ejecución del experimento. El valor del atributo es asignado automáticamente.
ea_cd_nd_param	E	Número de parámetros entrenables de la ANN que corresponde a la solución que se ubica en el codo del frente óptimo de Pareto. El valor del atributo es asignado automáticamente.
ea_cd_nd_kappa	F	Valor del índice kappa de la ANN que ha sido evaluada con el conjunto de datos de test y que corresponde a la solución que se ubica en el codo del frente óptimo de Pareto. El valor del atributo es asignado automáticamente.
ea_fechahoraini	FH	Fecha y hora de inicio de la ejecución del procedimiento de optimización. El valor del atributo es asignado automáticamente.
ea_fechahorafin	FH	Fecha y hora del final de la ejecución del procedimiento de optimización. El valor del atributo es asignado automáticamente.
ann_parantici	E	Número de épocas de entrenamiento que han sido ejecutadas para generar el modelo de la ANN.
ann_acc	F	Valor de exactitud (*accuracy*) alcanzado por el modelo de la ANN.
pid	E	Código que identifica al proceso en ejecución asignado por el sistema operativo. El valor del atributo es asignado automáticamente.
dominado	E	Indica si la solución corresponde al frente óptimo de Pareto. El valor del atributo es asignado automáticamente.

Tabla A.18.: Descripción detallada de los atributos de la tabla "ExpEjeDet".

Atributo	Tipo	Descripción
idexpejedet (PK)	E	Identificador de la tabla que se genera de forma automática.
idexpeje	E	Indica el experimento y la ejecución correspondiente con el que guarda relación. Tiene dependencia con la tabla "ExpEjec". El valor del atributo es asignado automáticamente.
idgpu	E	GPU donde se ha evaluado la solución propuesta por el algoritmo de optimización. Tiene dependencia con la tabla "GPUs". El valor del atributo es asignado automáticamente.
cromosoma	C	Valor de los distintos genes que corresponden a un cromosoma. El valor del atributo es asignado automáticamente.
kappa	F	Indica el valor del índice de clasificación alcanzado por la solución propuesta. El valor del atributo es asignado automáticamente.
parametros	E	Número de parámetros entrenables de la ANN o solución propuesta. El valor del atributo es asignado automáticamente.
noparam	E	Número de parámetros no entrenables de la ANN o solución propuesta. El valor del atributo es asignado automáticamente.
acc	F	Valor de exactitud alcanzado por la solución propuesta. El valor del atributo es asignado automáticamente.
fechahoraini	FH	Fecha y hora de inicio de la evaluación de la solución. El valor del atributo es asignado automáticamente.
fechahorafin	FH	Fecha y hora de fin de la evaluación de la solución. El valor del atributo es asignado automáticamente.
pid	E	Código que identifica al proceso que se ejecuta y es asignado por el sistema operativo. El valor del atributo es asignado automáticamente.
dominado	E	Indica si la solución corresponde al frente óptimo de Pareto. El valor del atributo es asignado automáticamente.

Tabla A.19.: Descripción detallada de los atributos de la tabla “ConsumoEnergia”.

Atributo	Tipo	Descripción
idconsumoenergia(PK)	E	Identificador de la tabla que se genera de forma automática.
pid	E	Código que ha sido asignado por el sistema operativo a la función que captura los datos acerca del consumo energético. El valor del atributo es asignado automáticamente.
inspwr_front	F	Potencia eléctrica instantánea consumida por el nodo “front-end”. El valor del atributo es asignado automáticamente.
inspwr_node0	F	Potencia eléctrica instantánea consumida por el nodo computacional denominado “node0”. El valor del atributo es asignado automáticamente.
inspwr_node1	F	Potencia eléctrica instantánea consumida por el nodo computacional denominado “node1”. El valor del atributo es asignado automáticamente.
inspwr_node2	F	Potencia eléctrica instantánea consumida por el nodo computacional denominado “node2”. El valor del atributo es asignado automáticamente.
acupwr_front	F	Energía acumulada consumida por el nodo “front-end”. El valor del atributo es asignado automáticamente.
acupwr_node0	F	Energía acumulada consumida por el nodo computacional denominado “node0”. El valor del atributo es asignado automáticamente.
acupwr_node1	F	Energía acumulada consumida por el nodo computacional denominado “node1”. El valor del atributo es asignado automáticamente.
acupwr_node2	F	Energía acumulada consumida por el nodo computacional denominado “node2”. El valor del atributo es asignado automáticamente.
fechahora	FH	Fecha y hora del registro. El valor del atributo es asignado automáticamente.