UNIVERSIDAD DE VALLADOLID

ESCUELA TÉCNICA SUPERIOR

DE INGENIEROS DE TELECOMUNICACIÓN

Trabajo Fin De Grado

GRADO EN INGENIERÍA DE TECNOLOGÍAS ESPECÍFICAS DE TELECOMUNICACIÓN, MENCIÓN EN SISTEMAS DE TELECOMUNICACIÓN

Aplicación de técnicas de *deep learning* en la ayuda al diagnóstico de la enfermedad de Alzheimer

Autora:

Dña. Arancha Abad Martín

Tutores:

Dr. D. Jesús Poza Crespo

D. Fernando Vaquerizo Villar

TÍTULO: **Aplicación de técnicas de *deep learning* en la ayuda al diagnóstico de la enfermedad de Alzheimer**

AUTOR: **Dña. Arancha Abad Martín**

TUTORES: **Dr. D. Jesús Poza Crespo**

D. Fernando Vaquerizo Villar

DEPARTAMENTO: **Teoría de la Señal y Comunicaciones e Ingeniería Telemática**

TRIBUNAL

PRESIDENTE: **Dra. Dña. María García Gadañón**

SECRETARIO: **Dr. D. Carlos Gómez Peña**

VOCAL: **Dr. D. Jesús Poza Crespo**

SUPLENTE 1: **Dr. D. Roberto Hornero Sánchez**

SUPLENTE 2: **Dr. D. Miguel López-Coronado Sánchez-Fortún**

FECHA:

CALIFICACIÓN:

Agradecimientos

Para comenzar, me gustaría agradecer a mis tutores, Jesús Poza Crespo y Fernando Vaquerizo Villar por darme la oportunidad de realizar este trabajo y ayudarme en todo lo que he necesitado.

A mi familia, en especial a mis padres, María Jesús y José Ignacio, mis abuelos, María Jesús y Teodoro, mi pareja, Pablo y mi hermana Jessica. Gracias por apoyarme en este largo recorrido, por estar siempre ahí en los buenos y los malos momentos porque, sin vosotros, no habría llegado a donde estoy hoy.

A mis amigos, algunos de ellos compañeros de carrera. Gracias por el apoyo, el compañerismo, la ayuda y el cariño mostrado a lo largo de todos estos años y, en especial, a lo largo del desarrollo de este trabajo fin de grado.

Resumen

La enfermedad de Alzheimer (EA) es una patología neurodegenerativa, progresiva e irreversible caracterizada por provocar alteraciones tanto a nivel cognitivo como a nivel de conducta. Es la demencia más común y sus principales factores de riesgo son la edad y la genética. Provoca lesiones en la corteza cerebral, principalmente depósitos de proteína beta-amiloide y ovillos neurofibrilares de proteína tau anormalmente fosforilada. Estas lesiones fomentan la aparición de alteraciones en la actividad eléctrica cortical, generando cambios en la actividad electroencefalográfica (EEG). Diversas investigaciones apuntan a que estas alteraciones aparecen en fases incipientes de la demencia; sin embargo, la complejidad de los cambios neuronales ha implicado que apenas haya estudios que traten de integrarlos para mejorar la identificación precoz de la EA. Es por ello, que en el presente Trabajo de Fin de Grado (TFG) se propone una metodología de *deep learning*, con el objetivo de extraer de manera automática características de la actividad EEG que ayuden a identificar esta enfermedad en los distintos sujetos de este estudio.

En este trabajo se han analizado señales EEG de dos bases de datos distintas con un total de 449 sujetos divididos en tres categorías principales: sujetos de control de edad avanzada, pacientes con deterioro cognitivo leve (DCL) y pacientes con demencia debida a EA. La primera base de datos proviene del Hospital Universitario Río Hortega de Valladolid; está formada por un total de 196 sujetos: 45 controles, 69 pacientes con DCL y 82 pacientes con demencia por EA. La segunda base de datos ha sido registrada en el marco de un proyecto europeo desarrollado por el Grupo de Ingeniería Biomédica; está formada por 253 sujetos: 51 controles, 51 pacientes con DCL y 151 pacientes con EA. En un primer paso, los registros EEG de las dos bases de datos se preprocesaron para eliminar los posibles artefactos de la señal, dividiéndose en segmentos de 1 y 5 s. Tras este preprocesado, los registros EEG se dividieron en tres conjuntos de datos: entrenamiento, validación y test, que se utilizaron para diseñar y validar las distintas arquitecturas de *deep learning*. En concreto, se utilizaron arquitecturas basadas en redes neuronales convolucionales de una dimensión, desarrolladas utilizando *Python* y la librería *Keras.*

Se desarrollaron dos casos de estudio diferentes: un problema de clasificación binaria con sujetos controles y enfermos de Alzheimer; y un problema de clasificación múltiple con sujetos controles, pacientes con DCL y pacientes con EA. Los valores de precisión obtenidos por los distintos modelos desarrollados para el primer caso de estudio fueron del 70% aproximadamente para las dos bases de datos. En el caso del segundo caso de estudio, la precisión disminuyó hasta un 40% aproximadamente. Se observó que a medida que se aumentaba el número de clases, disminuía la precisión obtenida. Estos resultados de precisión bajos pueden deberse a dos motivos principales: (*i*) a que el número de sujetos en las bases de datos utilizadas sea pequeño, lo cual se podría solventar con una base de datos más grande; (*ii*) a que la señal EEG es demasiado compleja para introducirla en crudo a una red neuronal convolucional 1-D, por lo que realizar una transformación sobre

ésta ayudará a obtener mejores resultados. Por tanto, estudios futuros podrían ayudar a mejorar la detección automática de la EA.

Palabras Clave

Enfermedad de Alzheimer, deterioro cognitivo leve, electroencefalograma, *deep learning*, redes neuronales convolucionales.

Abstract

Alzheimer's disease (AD) is a neurodegenerative, progressive and irreversible pathology. This disease causes both cognitive and behavioral alterations. It is the most common form of dementia and its risk factors are age and genetics. AD causes damage to the cerebral cortex, mainly deposits of beta-amyloid protein and neurofibrillary tangles of abnormally phosphorylated tau protein. This damage increases the chance of suffering alterations on the cortical electrical activity, which are reflected on the electroencephalogram (EEG). Several studies indicate that these alterations appear in the early stages of the disease. Nonetheless, there are few studies that integrate these changes in the early detection of EA due to its complexity. Accordingly, in this study we propose a deep-learning methodology, aimed at automatically detect the disease on the different subjects under study.

To achieve this goal, two different databases have been analyzed, including 449 subjects divided into three main categories: control, mild cognitive impairment (MCI) and AD. The first database, Hospital Universitario Río Hortega, is composed of 196 subjects: 45 control subjects, 69 MCI patients (27 mild MCI, 23 moderated MCI and 19 severe MCI) and 82 AD patients. The second database, POCTEP (Programa Operativo de Cooperación Transfronteriza España – Portugal), comprehends 253 subjects: 51 control subjects, 51 MCI patients and 151 AD patients (51 mild AD, 50 moderated AD and 50 severe AD). This data is first preprocessed in order to eliminate every potential artifact present in the signal. Once the data is artifact free, it is divided into segments of one and five seconds each. After this preprocessing, the data is divided into three different sets: training, validation and test, that will be used to design and validate the different deep-learning architectures. These architectures are one dimensional convolutional neural networks developed using Python and keras library.

Two case studies have been developed. The first one is a binary classification problem with the groups Control and AD. The second one is a multiclass classification problem with the classes Control, MCI and AD. For the first case study, the accuracy values obtained by the different developed models are values around 70%, with a better identification of AD segments. For the second study case, the value for this metric decreases up to 40%, still obtaining better results for the AD class. A drop on the accuracy value can be observed as the number of classes increases. The low accuracy obtained for the automated detection of the disease may be caused by two main issues: (*i*) the number of subjects under study is low, what could be overcome with a larger sample size; (*ii*) the EEG signal is too complex to be introduced raw as input for a 1D convolutional neural network, hence performing a pretreatment of this signal could lead to better results. Thus, future research could help to improve the automated detection of EA.

Key Words

Alzheimer’s disease, mild cognitive impairment, electroencephalogram, deep learning, convolutional neural networks, accuracy.

Índice general

1 Introducción 21

1.1 Procesado de señales biomédicas 21

1.2 La enfermedad de Alzheimer 22

1.3 Aprendizaje profundo o *Deep Learning* 23

1.4 Hipótesis 24

1.5 Objetivos 24

1.6 Descripción del documento 24

2 La Enfermedad de Alzheimer 27

2.1 Epidemiología y etiología 27

2.2 Características neuropatológicas 29

2.3 Clínica y fases 31

2.4 Diagnóstico 32

3 Electroencefalografía 35

3.1 Introducción a la electroencefalografía 35

3.2 Historia de la electroencefalografía 35

3.3 Neurofisiología 35

3.4 Registros EEG 36

3.5 Alteraciones de las señales EEG debido a la evolución de la enfermedad de Alzheimer 38

4 Aprendizaje profundo o Deep Learning 41

4.1 Introducción 41

4.2 Redes neuronales convolucionales 45

4.2.1 Etapa convolucional 48

4.2.2 Etapa de activación 49

4.2.3 Etapa de submuestreo o *pooling* 49

4.3 Arquitectura seleccionada y algoritmos utilizados 50

4.3.1 Arquitecturas utilizadas 54

4.3.2 Algoritmos de optimización, métodos de regularización para la prevención del overfitting y funciones de pérdida 55

5 Análisis de los resultados 57

5.1 Bases de datos utilizadas 58

5.1.1 Base de datos del Hospital Universitario Río Hortega 58
5.1.2Base de datos POCTEP..... 59
5.2 Preprocesado de los datos 60
5.2.1 Base de datos del Hospital Universitario Río Hortega 60
5.2.2 Base de datos del proyecto POCTEP 60
5.2.3 Procesado común a las dos bases de datos 61
5.3 Análisis de precisión y análisis de clasificación de sujetos mediante matrices de confusión62
5.3.1 Resultados obtenidos para el problema de clasificación binaria en la base de datos del Hospital Universitario Río Hortega 63
5.3.1.1 Resultados obtenidos para segmentos de cinco segundos..... 64
5.3.1.2 Resultados obtenidos para segmentos de un segundo..... 70
5.3.1.3 Resultados obtenidos para la aplicación de transfer learning 75
5.3.2 Resultados obtenidos para el problema de clasificación binaria en la base de datos del proyecto POCTEP 75
5.3.2.1 Resultados obtenidos para segmentos de cinco segundos..... 75
5.3.2.2 Resultados obtenidos para segmentos de un segundo..... 81
5.3.2.3 Resultados obtenidos para la aplicación de transfer learning 85
5.3.3 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del Hospital Universitario Río Hortega 86
5.3.3.1 Resultados obtenidos para segmentos de cinco segundos..... 86
5.3.3.2 Resultados obtenidos para segmentos de un segundo..... 93
5.3.3.3 Resultados obtenidos para la aplicación de transfer learning 99
5.3.4 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del proyecto POCTEP 100
5.3.4.1 Resultados obtenidos para segmentos de cinco segundos..... 100
5.3.4.2 Resultados obtenidos para segmentos de un segundo..... 105
5.3.4.3 Resultados obtenidos para la aplicación de transfer learning 109

6 Discusión 110

6.1 Introducción 111
6.2 Problema de clasificación binaria..... 112
6.3 Problema de clasificación múltiple 114
6.4 Comparativa con otros estudios de la literatura..... 116
6.5 Limitaciones 118

7 Conclusiones y líneas futuras..... 120

7.1 Cumplimiento de los objetivos del trabajo fin de grado..... 121

7.2 Conclusiones..... 121

7.3 Líneas futuras..... 122

Glosario de siglas y acrónimos

Bibliografía

Índice de figuras

1. Taxonomía de la inteligencia artificial (Alom *et ál*., 2018)..... 23
2. Prevalencia de la EA con la edad en distintos países y ciudades (Chengxuan *et ál*., 2009) 28
3. Lóbulos cerebrales. Figura adaptada de (Purves *et ál*., 2001) 29
4. Aparición de depósitos de proteína amiloide Aβ en las distintas etapas de la enfermedad, siendo A las etapas iniciales, B las etapas intermedias y C las etapas finales (Braak & Braak, 1991). 30
5. Lesiones producidas por la EA observadas en placas a nivel microscópico. Figura obtenida de (DeTure & Dickson, 2019). Las placas seniles se muestran en A y B y las placas neuríticas y los ovillos neurofibrilares en C y D, los últimos, señalados por flechas blancas. 31
6. Evolución de las fases de la EA. Figura adaptada de (Sperling *et ál*., 2011). 32
7. Distribución de electrodos utilizada en Sistema Internacional 10-20. Figura obtenida de (Song *et ál*., 2015) 37
8. Taxonomía y definiciones de AI, ML y DL. Figura adaptada de (Nouri, 2012). 41
9. Estructura de un problema de clasificación para ML (A) y DL (B). Figura adaptada de (Nouri, 2012). 42
10. Rendimiento de DL y ML con la cantidad de datos utilizada 42
11. Paradigmas de aprendizaje asociados a las distintas zonas cerebrales. Figura adaptada de (Hu *et ál*., 2007)..... 43
12. Esquema de funcionamiento del paradigma aprendizaje reforzado. Figura adaptada de (Raschka & Mirjalili, 2017) 44
13. Aplicaciones principales de los tres paradigmas de aprendizaje: supervisado, no-supervisado y reforzado. 44
14. Diferencia entre los procesos de aprendizaje tradicionales de ML y DL (a) y el utilizado en transfer learning (b). Figura adaptada de (Pan & Yang, 2010)..... 45
15. Conexión entre neuronas de dos capas de una red neuronal tradicional. Figura de (Goodfellow *et ál*., 2016) 46
16. Conexión entre neuronas de dos capas de una red neuronal convolucional con un tamaño de kernel 3. Figura de (Goodfellow *et ál*., 2016). 46
17. La flecha negra indica que el parámetro de la matriz de pesos correspondiente a esa neurona se utiliza una única vez. Figura de (Goodfellow *et ál*., 2016) 46
18. Las flechas negras implican que al aprender un único conjunto de parámetros para cada ubicación gracias al uso del parameter sharing todas las neuronas de la red utilizarán ese conjunto. Figura de (Goodfellow *et ál*., 2016). 47
19. Arquitectura de una capa convolucional. Figura adaptada de (Goodfellow *et ál*., 2016). 47
20. Tipos de padding. Figura adaptada de (Raschka & Mirjalili, 2017)..... 48
21. Convolución discreta 1D. Figura adaptada de (Raschka & Mirjalili, 2017). 49

22. Ejemplo de convolución 2D. Figura de (Goodfellow *et ál.*, 2016). 50
23. Ejemplo de MaxPooling y AveragePooling. 51
24. Ejemplo de red neuronal con capas totalmente conectadas que utiliza la técnica de Dropout. Figura de (Vasilev *et ál.*, 2019). 52
25. Representación gráfica de la función de activación Sigmoid. 54
26. Representación gráfica de la función de activación Sigmoid. 54
27. Ejemplo de la primera arquitectura seleccionada. 54
28. Ejemplo de la arquitectura por bloques. 55
29. Ejemplo de codificación utilizando LabelEncoder. 62
30. Aplicación de la función np_utils de la librería keras para finalizar la codificación de las etiquetas. 62
31. Arquitectura del modelo 3 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH. 65
32. Arquitectura del modelo 5 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH. 66
33. Arquitectura del modelo 6 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH. 66
34. Arquitectura del modelo 7 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH. 67
35. Porcentaje de precisión obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y z-score para el problema de segmentos de 5 s en la base de datos del HURH. 69
36. Arquitectura de los modelos 3 y 4 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH. 72
37. Arquitectura del modelo 5 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH. 72
38. Arquitectura del modelo 6 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH. 73
39. Porcentaje de precisión obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y z-score, para el problema de segmentos de 1 s en la base de datos del HURH. 74
40. Arquitectura del modelo 1 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 76
41. Arquitectura del modelo 2 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 77
42. Arquitectura de los modelos 3 y 4 utilizados en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 77
43. Porcentaje de precisión obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y z-score para el problema de clasificación binaria con segmentos de 5 s en la base de datos del proyecto POCTEP. 79

44. Arquitectura de los modelos 4 y 5 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP. 81
45. Arquitectura del modelo 6 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP. 82
46. Porcentaje de precisión obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y z-score para el problema de clasificación binaria con segmentos de 1 s en la base de datos del proyecto POCTEP. 83
47. Arquitectura del modelo 5 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH. 86
48. Arquitectura del modelo 6 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH. 87
49. Arquitectura del modelo 8 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH. 87
50. Arquitectura del modelo 9 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH. 88
51. Arquitectura del modelo 10 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH. 88
52. Porcentaje de precisión obtenido por los modelos para el caso de segmentos de 5 s en el problema de clasificación múltiple en la base de datos del HURH. 92
53. Arquitectura del modelo 3 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH. 94
54. Arquitectura de los modelos 8 y 9 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH. 95
55. Arquitectura de los modelos 10 y 11 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH. 95
56. Porcentaje de precisión obtenido por los modelos para el *caso* de segmentos de1 s en el problema de clasificación múltiple en la base de datos del HURH. 97
57. Arquitectura del modelo 1 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 101
58. Arquitectura del modelo 2 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 101
59. Arquitectura del modelo 3 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 102
60. Arquitectura del modelo 4 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP. 102
61. Porcentaje de precisión obtenido por los modelos para el caso de segmentos de 5 s en el problema de clasificación múltiple en la base de datos del proyecto POCTEP. 103
62. Arquitectura de los modelos 5 y 6 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP. 105
63. Arquitectura del modelo 7 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP. 106

64. Porcentaje de precisión obtenido por los modelos para el caso de segmentos de 1 s en el problema de clasificación múltiple en la base de datos del proyecto POCTEP. 107

Índice de tablas

1. Hipótesis etiológicas y su evidencia epidemiológica, adaptada de (Chengxuan *et ál.*, 2009; Livingston *et ál.*, 2020) **27**
2. Datos sociodemográficos y clínicos de la base de datos del HURH. N indica el número de sujetos y SD la desviación estándar. **58**
3. Datos sociodemográficos y clínicos de la base de datos del proyecto POCTEP. N indica el número de sujetos y SD, la desviación estándar. **59**
4. Resumen de resultados de precisión en tanto por ciento (%) y valor de kappa para modelos entrenados con segmentos de 5 s en la base de datos del HURH. **67**
5. Matriz de confusión de clasificación de sujetos obtenida por el modelo 7 entrenado con segmentos de 5 s en la base de datos del HURH. **68**
6. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s en la base de datos del HURH. **69**
7. Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 5 s en la base de datos del HURH. **69**
8. Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 5 s en la base de datos del HURH. **70**
9. Resumen de resultados de precisión en tanto por ciento (%) y valor de kappa para modelos entrenados con segmentos de 1 s en la base de datos del HURH. **71**
10. Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 1 s en la base de datos del HURH. **73**
11. Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 1 s en la base de datos del HURH. **73**
12. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s en la base de datos del HURH. **74**
13. Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s en la base de datos del HURH. **75**
14. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del proyecto *POCEP.* **78**
15. Matriz de confusión obtenida por el modelo 1 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP. **80**
16. Matriz de confusión obtenida por el modelo 2 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP. **80**
17. Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP **80**
18. Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP. **81**
19. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del proyecto POCTEP. **83**
20. Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP. **84**
21. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP. **84**
22. Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP. **85**

23. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del HURH. **90**
24. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH. **91**
25. Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH. **91**
26. Matriz de confusión obtenida por el modelo 8 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH. **92**
27. Matriz de confusión obtenida por el modelo 9 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH. **92**
28. Matriz de confusión obtenida por el modelo 10 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH. **93**
29. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del HURH. **96**
30. Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH. **98**
31. Matriz de confusión obtenida por el modelo 8 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH. **98**
32. Matriz de confusión obtenida por el modelo 9 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH. **99**
33. Matriz de confusión obtenida por el modelo 10 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH. **99**
34. Matriz de confusión obtenida por el modelo 11 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH. **99**
35. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del proyecto POCTEP. **100**
36. Matriz de confusión obtenida por el modelo 1 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **104**
37. Matriz de confusión obtenida por los modelos 2 y 3 entrenados con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **104**
38. Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **104**
39. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **105**
40. Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del proyecto POCTEP. **106**
41. Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **108**
42. Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **108**
43. Matriz de confusión obtenida por el modelo 7 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP. **109**

Capítulo 1

1. Introducción

1.1 Procesado de señales biomédicas **21**
1.2 La enfermedad de Alzheimer **22**
1.3 Aprendizaje profundo o *Deep Learning* **23**
1.4 Hipótesis **24**
1.5 Objetivos **24**
1.6 Descripción del documento **24**

1.1 Procesado de señales biomédicas

La Ingeniería Biomédica puede ser definida como la aplicación de distintas técnicas, principios y métodos, utilizados en el campo de la ingeniería, al campo de la biología. Las distintas técnicas o principios utilizados permiten entender, modificar y controlar diferentes sistemas biológicos, así como diseñar y producir dispositivos que ayudan en el diagnóstico y posterior tratamiento de pacientes. (Bronzino, 2006a)

Este Trabajo Fin de Grado (TFG) se desarrolla en el ámbito de la Ingeniería Biomédica, dentro del área del tratamiento de señales biomédicas. Esta área engloba, entre otras, funciones como la interpretación diagnóstica mediante técnicas de procesamiento de señal. Otra función importante, utilizada en este TFG, es la del análisis de datos relacionados con el paciente y la toma de decisiones clínicas mediante el uso de inteligencia artificial (Bronzino, 2006a).

Una señal biomédica difiere de una señal corriente únicamente en su aplicación, es decir, una señal biomédica es lo mismo que una señal corriente, es un fenómeno que transmite información aplicado al campo de la biomedicina (Bronzino, 2006b). Dependiendo de su origen o fuente de adquisición, se tienen distintos tipos de señales biomédicas, algunas de ellas son: las señales eléctricas, como el electroencefalograma (EEG), generalmente producidas por los campos eléctricos generados por células musculares o nerviosas; las señales acústicas, generadas por el flujo de sangre o aire en los distintos órganos, como una ecografía; y las señales magnéticas, como el magnetoencefalograma (MEG), producidas por los campos magnéticos generados por algunos órganos (Bronzino, 2006b).

En el ámbito de la Ingeniería Biomédica, las señales biomédicas *per se* no suelen ofrecer información relevante debido a múltiples factores como, por ejemplo, el ruido introducido por los sistemas de adquisición. Es por esto que deben ser procesadas para eliminar posibles interferencias o artefactos y hacer visible la información de interés. De forma general, existen tres etapas en el procesado de la señal biomédica (Poza, 2008):

1. **Obtención y registro**
    a. Adquisición, muestreo, cuantificación y posterior digitalización de las señales biomédicas.
    b. Preprocesado de la señal que elimina artefactos y ruidos propios de este tipo de señales.
    c. Almacenamiento y/o transformación de la señal obtenida tras el preprocesado.
2. **Procesado**
    a. División en segmentos de la señal preprocesada
    b. Filtrado y/o transformación de la señal
    c. Detección de los patrones deseados
3. **Clasificación**
    a. Extracción de características
    b. Clasificación de la señal

La señal biomédica utilizada en este TFG consiste en el registro de la actividad eléctrica producida por el cerebro. Concretamente, se han utilizado registros de EEG correspondientes a personas cognitivamente sanas de edad avanzada, conocidas como controles, pacientes que padecen de deterioro cognitivo leve (DCL), y pacientes que padecen demencia debida a la enfermedad de Alzheimer (EA).

1.2 La enfermedad de Alzheimer

La EA es un trastorno progresivo que produce un desgaste de las células cerebrales que hace que éstas se degeneren y mueran. Es la causa más común de demencia a nivel mundial, es decir, en torno al 50% – 60% de los casos de demencia diagnosticados son EA (Blennow *et ál.*, 2006).

Existen diversos factores de riesgo asociados a la EA, como factores genéticos o de género, pero uno de los factores más comunes e importantes es la edad. Se ha observado una prevalencia inferior del 1% en personas menores de 65 años mientras que, en personas mayores de 85 años, la prevalencia incrementa de forma exponencial hasta llegar al 30% (Blennow *et ál.*, 2006).

Esta enfermedad hace que disminuyan habilidades cognitivas, de comportamiento y sociales, interrumpiendo así la capacidad de una persona para valerse por sí misma. Los primeros signos de esta enfermedad son pequeñas pérdidas de memoria y comportamientos extraños. A medida que avanza, las alteraciones de memoria empeoran y aparecen otros síntomas más graves como problemas de orientación y cambios de humor sumados a un deterioro del lenguaje o afasia y a una apraxia o deterioro de la movilidad. En los casos más graves, el enfermo no sólo no podrá realizar tareas cotidianas si no que perderá la capacidad de moverse (Blennow *et ál.*, 2006).

A nivel estructural, los cambios en la enfermedad se deben a la aparición de placas neuríticas y ovillos neurofibrilares. Las placas seniles están formadas por depósitos de la proteína beta-amiloide (Aβ) y los ovillos de proteína tau anormalmente fosforilada (Blennow *et ál.*, 2006). Estas lesiones son típicas de dos regiones cerebrales: el hipocampo y la zona cortical. No obstante, se desconoce si han sido resultado o causa de la EA (Blennow *et ál.*, 2006).

Existe un posible estado previo a la demencia por EA, el DCL, que es un estado intermedio entre el deterioro propio de la edad o envejecimiento y el deterioro causado por una demencia como la EA. El DCL no es lo suficientemente grave para ser considerado EA (Petersen, 2004), pero en ocasiones y dependiendo del sujeto que lo padezca, puede presentar distintos síntomas que preceden a esta enfermedad (Blennow *et ál.*, 2006).

Es importante destacar que no existe un tratamiento que cure la EA o que altere el proceso de dicha enfermedad en el cerebro, por lo que, normalmente, las personas que la sufren en las etapas más avanzadas morirán (Blennow *et ál.*, 2006; Chengxuan *et ál.*, 2009). La investigación de los últimos años ha hecho posible el desarrollo de algunos tratamientos que permiten ralentizar el avance de la enfermedad y suavizar los síntomas del deterioro que ésta produce (Ashford, 2019). Por lo tanto, es

fundamental continuar con la investigación de nuevos biomarcadores que, a partir de distintas modalidades, tanto neurofísicas como de neuroimagen y metabólicas, ayuden a un diagnóstico precoz de la EA (Sabbagh *et ál*., 2017).

1.3 Aprendizaje profundo o *Deep Learning*

Para poder entender qué es el aprendizaje profundo (*Deep Learning,* DL) es necesario conocer conceptos como inteligencia artificial (IA) y aprendizaje automático (*Machine Learning*, ML). Existen distintas definiciones del concepto IA. La definición más común define la IA como un área de estudio en el campo de la informática que permite el desarrollo de ordenadores capaces de participar en procesos de pensamiento similares a los humanos como, por ejemplo, el aprendizaje, el razonamiento, la adaptación o la autocorrección (Kok, 2009). En la Figura 1 (Alom *et ál*., 2018) se observa la taxonomía de la inteligencia artificial.

Una vez conocida la definición de IA, el ML se define como un subconjunto de métodos de IA cuyo objetivo es desarrollar algoritmos que aprenden principios de interpretación a partir de muestras de entrenamiento. Una vez realizado este entrenamiento, se aplica el algoritmo a nuevos datos de un mismo dominio permitiendo así la toma de decisiones informadas. A su vez, las redes neuronales (*Neural Networks*, NN) son un subcampo del aprendizaje automático; concretamente, el subcampo a partir del cual se generó lo que se conoce como DL. El DL es una clase o subconjunto del ML consistente en estimar los parámetros de un modelo o algoritmo que aprenda a realizar una tarea específica (Alom *et ál*., 2018; Chassagnon *et ál*., 2020). Al utilizar ML, se deben extraer las características de interés creando extractores de manera manual. Esto requiere tiempo, gran habilidad y experiencia. En cambio, el DL es capaz de aprender estas características de forma automática. Esta es la principal razón por la que el DL se está imponiendo al ML(LeCun *et ál*., 2015).

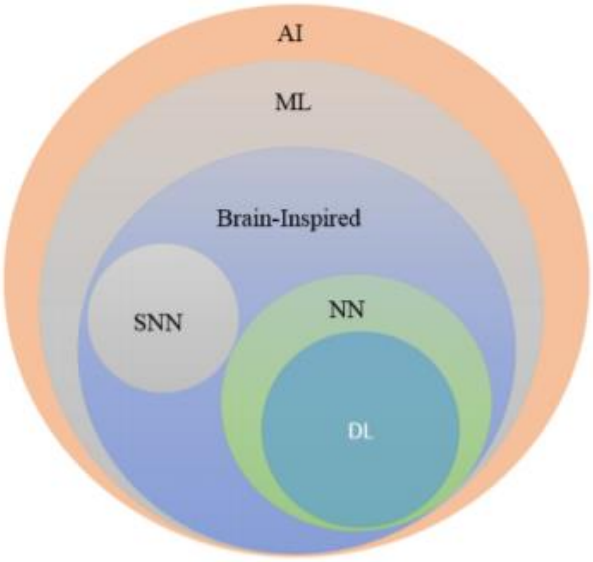

Figura 1: Taxonomía de la inteligencia artificial (Alom *et ál*., 2018).

En el Capítulo 4 se profundiza en el aprendizaje profundo (*deep learning*) y en las redes neuronales con arquitectura convolucional, puesto que es la arquitectura utilizada en este TFG.

1.4 Hipótesis

Los métodos tradicionales utilizados en el análisis de señales EEG para el diagnóstico de la EA se basan en la extracción de características complejas que no son capaz de caracterizar de manera exhaustiva las alteraciones de las propiedades dinámicas de la actividad neuronal. Resultados preliminares sugieren que la aplicación de técncias de DL para el análisis de la actividad cerebral, como imágenes de resonancia magnética (MRI) o tomografía por emisión de positrones (PET), permite obtener precisiones elevadas al clasificar entre controles y enfermos con EA (problema binario) (Liu *et ál.*, 2014), así como en problemas de clasificación múltiple entre controles, pacientes con DCL y enfermos con EA (Ding *et ál.*, 2018). Por ello, en este TFG planteamos la hipótesis de que la aplicación de una red neuronal convolucional (*Convolutional Neural Networks*, CNN) a señales EEG podría extraer información cerebral relevante para clasificar entre tres categorías de sujetos: controles, pacientes con DCL y enfermos con EA.

1.5 Objetivos

El objetivo principal de este TFG es evaluar la utilidad de aplicar técnicas de DL en la ayuda al diagnóstico de la EA. Para conseguir este objetivo se han creado distintos modelos de DL utilizando arquitecturas CNN con dos bases de datos distintas. Los pasos a seguir para conseguir el objetivo son los siguientes:

1. Búsqueda de bibliografía para familiarizarse con las enfermedades con las que se trabaja (DCL y EA), las señales que se utilizan (EEG) y la metodología que se aplica (DL).
2. Aprendizaje del lenguaje de programación Python y del manejo de las librerías necesarias para implementar las CNN.
3. Análisis y evaluación de las bases de datos utilizadas, que incluyen tres grupos: sujetos de control, pacientes con DCL y pacientes con demencia por EA.
4. Diseño, desarrollo y aplicación de las redes CNN a los registros EEG.
5. Obtención de distintas métricas de rendimiento diagnóstico en el entrenamiento y posterior testeo de los modelos implementados que servirán para comprobar los resultados obtenidos.
6. Análisis de los resultados obtenidos comparación con los publicados en otros estudios e identificación de las limitaciones del estudio.
7. Extracción de conclusiones y posibles líneas futuras de investigación.

1.6 Descripción del documento

Este punto versará sobre la estructura de este documento, que está dividido en siete capítulos, incluyendo el capítulo actual que es la Introducción al TFG. A continuación, se presenta una breve descripción de los siguientes capítulos:

- **Capítulo 2. La enfermedad de Alzheimer.** Este capítulo contiene las principales características fisiológicas de la enfermedad junto con los principales métodos de diagnóstico.
- **Capítulo 3. Electroencefalografía.** El Capítulo 3 comienza con una breve introducción a la electroencefalografía, seguida de la neurofisiología de la señal EEG y las distintas alteraciones producidas por la EA en esta señal.
- **Capítulo 4. Introducción al aprendizaje profundo o *Deep Learning.*** Este capítulo ofrece una explicación sobre el DL, las redes neuronales en general y las CNN en particular, además de una descripción sobre la arquitectura seleccionada y los algoritmos empleados.
- **Capítulo 5. Resultados.** Este capítulo englobará los materiales y métodos utilizados, así como una descripción de las dos bases de datos utilizadas y de los resultados obtenidos tras la aplicación de los algoritmos.
- **Capítulo 6. Discusión**. El Capítulo 6 presenta un análisis de los resultados desde un punto de vista autocrítico, así como una comparación de los mismos con los resultados existentes en la literatura y la exposición de las principales limitaciones del estudio.
- **Capítulo 7**. **Conclusiones y líneas futuras.** Este capítulo contiene las conclusiones extraídas de la realización de este TFG, junto con las líneas futuras de investigación.

Capítulo 2

2. La enfermedad de Alzheimer

2.1 Epidemiología y etiología 27
2.2 Características neuropatológicas 29
2.3 Clínica y fases 31
2.4 Diagnóstico 32

2.1 Epidemiología y etiología

La EA fue descrita en 1906 por Alois Alzheimer en la XXXVII Conferencia de Psiquiatras del Sudoeste Alemán (Möller & Graeber, 1998). Actualmente, es la demencia más común y con más prevalencia en el mundo occidental ya que, del total de casos de demencia diagnosticados, ésta representa entre el 50% y el 70% de los casos (Chengxuan *et ál*., 2009; Park *et ál*., 2020). En los países desarrollados, la prevalencia es del 1% para personas menores de 65 años (Blennow *et ál*., 2006), del 4.4% para mayores de 65 años y aumenta de manera exponencial a medida que aumenta la edad (Blennow *et ál*., 2006; Chengxuan *et ál*., 2009). En los países en vías de desarrollo, las cifras descienden hasta el 1.6% de África (Chengxuan *et ál*., 2009) debido a que la esperanza de vida es menor que la existente en los países desarrollados. En la Figura 2 se observan los porcentajes y el crecimiento en la prevalencia con la edad, llegando al 30% para mayores de 85-90 años.

La Tabla 1 resume las hipótesis etiológicas, los factores de riesgo y/o protección y el grado de evidencia epidemiológica de cada una de las hipótesis.

Hipótesis Etiológica	Factores de riesgo y/o de protección	Evidencia epidemiológica
Factores genéticos	**Riesgo:** Incremento del alelo APOE ε4.	Fuerte
Problemas vasculares	**Riesgo:** Hipertensión, alto índice de masa corporal, diabetes, accidentes cerebrovasculares. **Protección:** leve consumo de alcohol, tratamiento de la hipertensión.	Moderada o suficiente
Factores psicosociales	**Protección:** nivel de educación alto, actividades estimulantes para el cerebro, actividad física.	Moderada o suficiente
Factores nutricionales	**Riesgo:** deficiencia de vitamina B12 y de antioxidantes. **Protección:** consumir ácidos grasos (omega-3) y verduras.	Insuficiente o limitada
Otros (factores tóxicos o inflamatorios)	**Riesgo:** traumatismos craneales, exposición a toxinas. **Protección:** antiinflamatorios.	Insuficiente o limitada

Tabla 1: Hipótesis etiológicas y su evidencia epidemiológica, adaptada de (Chengxuan *et ál*., 2009; Livingston *et ál*., 2020)

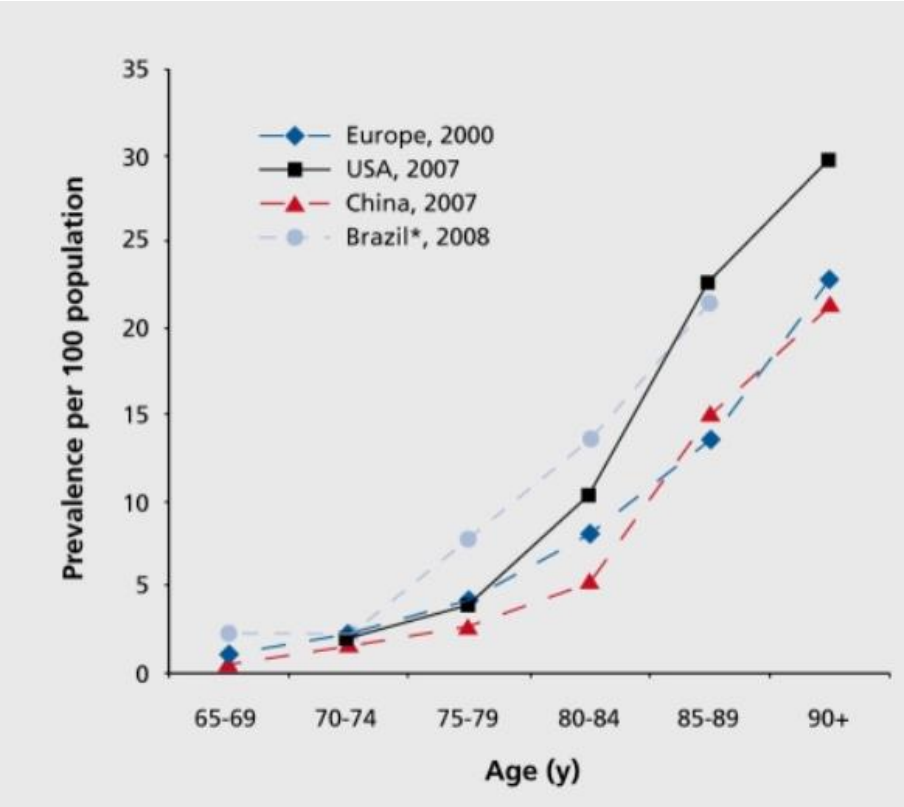

Figura 2: Prevalencia de la EA con la edad en distintos países y ciudades (Chengxuan *et ál.*, 2009).

Los factores genéticos son una de las causas más comunes de la EA, los familiares de primer orden de enfermos de Alzheimer tienen mayor riesgo a desarrollar la enfermedad que la población general. Algunos estudios afirman que lo que provoca este riesgo es el alelo ε4 de la apolipoproteína E (APOE), que es el primer gen de riesgo identificado y el que mayor impacto tiene (Chengxuan *et ál.*, 2009). El hecho de que sea este gen el que lo provoque, hace que existan otros genes susceptibles que pueden estar involucrados en el desarrollo de la enfermedad (Chengxuan *et ál.*, 2009). Al aumentar este gen, aumentan las probabilidades de desarrollar la enfermedad, es decir, si una persona hereda dos alelos ε4 de sus progenitores, tendrá un mayor riesgo de desarrollar la EA (Chengxuan *et ál.*, 2009).

Los problemas vasculares como hipertensión, hipercolesterolemia o la obesidad, así como enfermedades derivadas de estos problemas como la diabetes, están asociadas a un incremento del riesgo de desarrollo de la enfermedad. Otros factores como el tabaco o el alcohol suelen empeorar la situación, incrementando el riesgo de desarrollo de la enfermedad. Existe una excepción en el consumo leve de alcohol, que está asociada a una menor incidencia de la enfermedad (Chengxuan *et ál.*, 2009, Livingston *et ál.*, 2020).

La hipótesis de los factores nutricionales afirma que, siguiendo una dieta rica en antioxidantes como la vitamina E y la C y la dieta Mediterránea, el riesgo de desarrollar la EA disminuye. Sin embargo, esta hipótesis no tiene suficiente evidencia epidemiológica (Chengxuan *et ál.*, 2009, Livingston *et ál.*, 2020).

Los factores psicosociales presentan una evidencia epidemiológica moderada ya que se ha demostrado que, manteniendo el cerebro activo, realizando actividad física y llevando una vida social activa, el riesgo de EA disminuye. Por tanto, es más probable que ancianos que viven solos y prácticamente aislados, sin ningún tipo de contacto con otras personas, desarrollen la enfermedad (Chengxuan *et ál.*, 2009; Livingston *et ál.*, 2020).

Por último, otros factores, como la inflamación o la exposición a sustancias tóxicas, no tienen suficiente evidencia científica (Chengxuan *et ál.*, 2009; Livingston *et ál.*, 2020).

2.2 Características neuropatológicas

Alois Alzheimer describió las lesiones histopatológicas cerebrales ligadas a la EA en 1906 y, actualmente, estas lesiones siguen teniendo vigencia como marcadores neuropatológicos de la enfermedad (Castellanos-Aguilar *et ál.*, 2017; Rodrigo *et ál.*, 2007). Las principales características neuropatológicas son (Rodrigo *et ál.*, 2007):

1. Aparición de depósitos de la proteína beta-amiloide (Aβ) tanto intra como extracelulares.
2. Formación de placas seniles y placas neuríticas.
3. Formación de ovillos neurofibrilares debido a la proteína tau.
4. Atrofia cerebral.
5. Astrogliosis.

De las características neuropatológicas expuestas en la lista anterior, dos de ellas son las lesiones más típicas de la EA: depósitos Aβ y ovillos neurofibrilares. Estas lesiones aparecen en distintas zonas del cerebro. Normalmente, el número de lesiones en las zonas cerebrales aumentará conforme avance la enfermedad, por lo que será distinto en las diferentes fases o etapas de la EA (Braak & Braak, 1991; Rodrigo *et ál.*, 2007). La Figura 3 representa la anatomía superficial del hemisferio cerebral, los cuatro lóbulos cerebrales junto con las fisuras y surcos principales que ayudan a definir sus límites. Esta figura explica dónde está situado cada uno de los lóbulos cerebrales: el lóbulo frontal, el parietal, el temporal y el occipital. Cada uno de estos lóbulos presentará lesiones a lo largo del desarrollo de la EA (Braak & Braak, 1991).

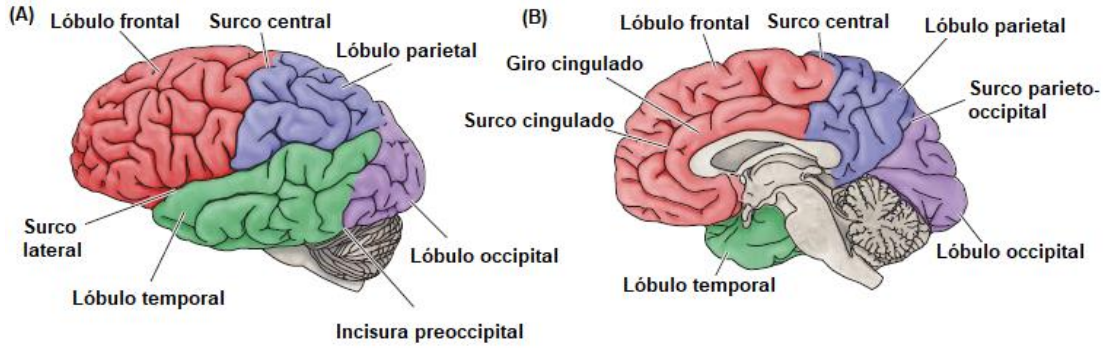

Figura 3: Lóbulos cerebrales. Figura adaptada de (Purves *et ál.*, 2001)

En las etapas iniciales de la enfermedad (ver Figura 4 A), los depósitos de proteína Aβ son los primeros en aparecer. Son depósitos de baja densidad de proteína Aβ y se encuentran distribuidos a lo largo del lóbulo frontal, el lóbulo temporal y el occipital (Braak & Braak, 1991). A medida que avanza la enfermedad (ver Figura 4 B), estos depósitos son más densos y comienzan a extenderse, ocupando una mayor área que la ocupada en las etapas iniciales. El lóbulo temporal es el que más depósitos acumula, mientras que el lóbulo parietal no se ve afectado (Braak & Braak, 1991). Finalmente, en las últimas etapas de la enfermedad, todos los lóbulos estarán afectados por unos depósitos de proteína Aβ densos. Todos los lóbulos presentan un gran número de depósitos, tal y como se observa en la Figura 4 C (Braak & Braak, 1991).

Las placas neuríticas son las placas seniles que aparecen, generalmente, en las últimas etapas de la enfermedad, como las placas representadas en la Figura 4 C (Blennow *et ál*., 2006; Braak & Braak, 1991; Rodrigo *et ál*., 2007). Son lesiones multicelulares que se encuentran en los lóbulos cerebrales (Rodrigo *et ál*., 2007), destacando su presencia en el lóbulo temporal (Blennow *et ál*., 2006; Braak & Braak, 1991). Se caracterizan por presentar axones y dendritas degeneradas o en proceso de degeneración, situadas dentro o alrededor de los depósitos de proteína Aβ (Braak & Braak, 1991). Estas dos lesiones son las responsables de la pérdida de sinapsis y de la pérdida neuronal (Blennow *et ál*., 2006; Rodrigo *et ál*., 2007).

Por otra parte, los ovillos neurofibrilares, formados por proteína tau fosforilada, son los principales responsables de la pérdida de neuronas (Alzheimer's Association, 2018; Barthélemy *et ál*., 2020; Blennow *et ál*., 2006; Rodrigo *et ál*., 2007), ya que bloquean el transporte de nutrientes y otras moléculas esenciales al interior de las mismas(Alzheimer's Association, 2018). Las neuronas piramidales son las responsables de la hiperfosforilación de la proteína tau, que causa la formación de estas lesiones. Estas neuronas son un factor clave en la disfunción cognitiva de la EA ya que son las más afectadas por estos ovillos y se degeneran rápidamente (Rodrigo *et ál*., 2007).

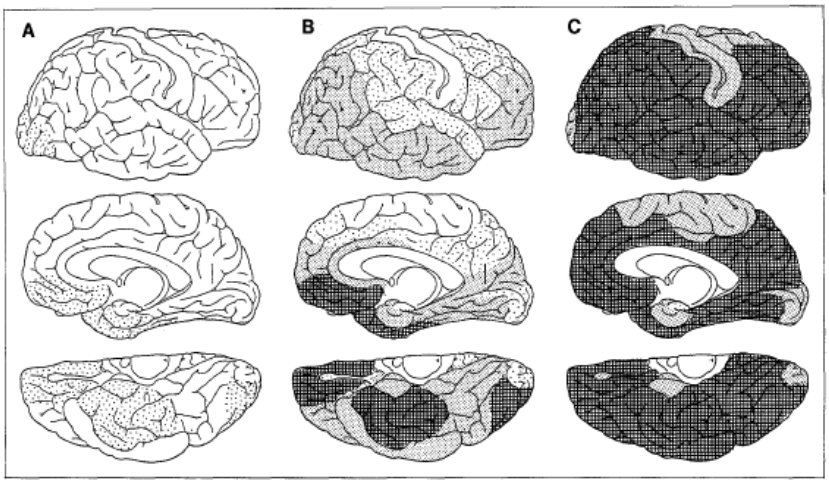

Figura 4: Aparición de depósitos de proteína amiloide Aβ en las distintas etapas de la enfermedad, siendo A las etapas iniciales, B las etapas intermedias y C las etapas finales (Braak & Braak, 1991).

Todas las lesiones explicadas, placas seniles y neuríticas, y ovillos neurofribrilares, pueden observarse a nivel microscópico tal y como se muestra en la Figura 5. Las Figuras 5 A y B representan placas seniles, mientras que las Figuras 5 C y D representan placas neuríticas y ovillos neurofribrilares, éstos últimos marcados con flechas blancas (DeTure & Dickson, 2019).

2.3 Clínica y fases

La EA se asocia con tres fases distintas: fase preclínica, DCL y demencia por EA. La última de estas fases, puede subdividirse en otras tres: demencia leve por EA, demencia moderada por EA y demencia grave o severa por EA (Petersen, 2014). Cada una de estas fases se caracteriza por la aparición de ciertas lesiones histopatológicas y diferentes síntomas asociados, aunque esto puede variar en cada sujeto. Normalmente, las fases suelen evolucionar con la edad, como se muestra en la Figura 6, y la función cognitiva disminuye con dicha evolución. (DeTure & Dickson, 2019; Sperling *et ál.*, 2011).

La fase preclínica se caracteriza por ser una fase asintomática que aparece al menos unos 10 años antes del diagnóstico de EA (Dubois *et ál.*, 2016; Sperling *et ál.*, 2011). Es una fase en la que el proceso fisiopatológico progresa comenzando la neurodegeneración provocada por la aparición de las lesiones típicas de la enfermedad. Estas lesiones aumentan con la evolución de la enfermedad, por lo que en esta fase no existirá un número suficiente de lesiones para provocar síntomas (Blennow *et ál.*, 2006; DeTure & Dickson, 2019; Rodrigo *et ál.*, 2007).

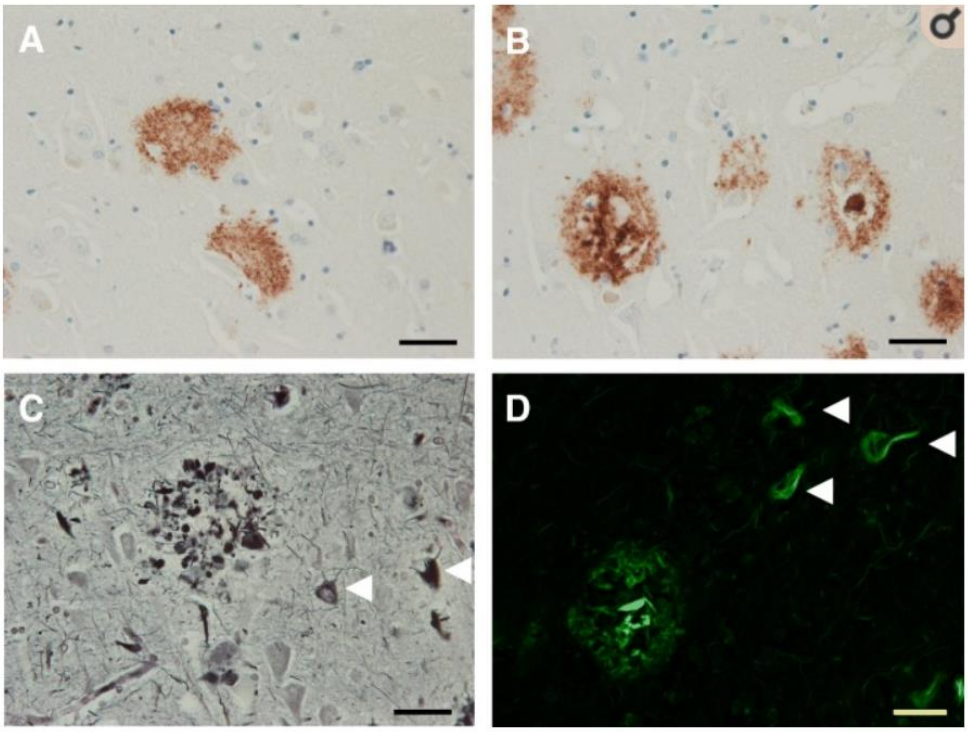

Figura 5: Lesiones producidas por la EA observadas en placas a nivel microscópico. Figura obtenida de (DeTure & Dickson, 2019). Las placas seniles se muestran en A y B y las placas neuríticas y los ovillos neurofibrilares en C y D, los últimos, señalados por flechas blancas.

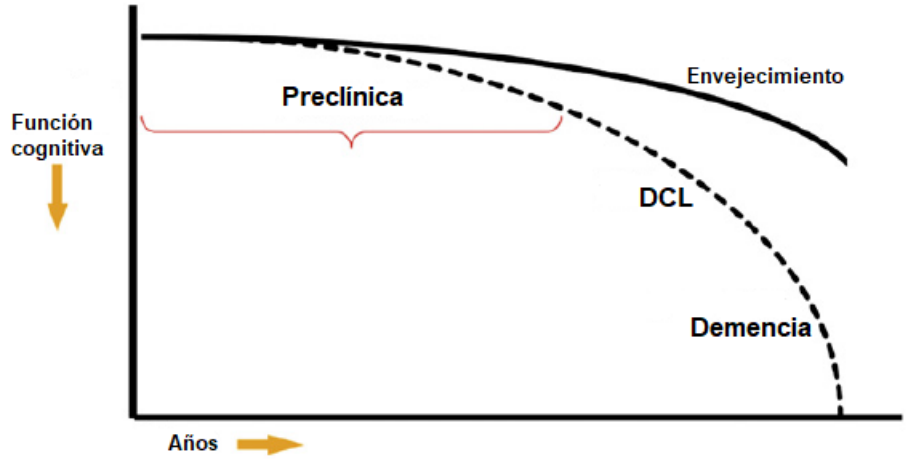

Figura 6: Evolución de las fases de la EA. Figura adaptada de (Sperling *et ál.*, 2011).

A medida que las lesiones aumentan y se extienden a distintas zonas del cerebro, los primeros síntomas comienzan a aparecer. Se presentan como fallos de memoria, cambios de humor y/o confusión, entre otros, y son distinguibles de posibles déficits del envejecimiento normal, pero no cumplen los criterios de la EA (Allegri *et ál.*, 2011; Blennow *et ál.*, 2006). Una vez aparecen estos síntomas, la enfermedad pasa a la fase de DCL. Aunque se postule el DCL como una patología por sí mismo, las alteraciones neurológicas que comparte con la EA y los casos que evolucionan hacia esta enfermedad hacen que sea considerada como la fase previa a la misma (Petersen, 2004, 2014). A medida que avanza esta fase, aumentan los síntomas pasando a ser más graves y afectando a las zonas encargadas del razonamiento y el lenguaje. Fruto de las lesiones en estas áreas cerebrales, se desarrollará afasia, las pérdidas de memoria empeorarán y la toma de decisiones se verá nublada, haciendo que el sujeto que padezca la enfermedad no pueda llevar a cabo su vida diaria (Alzheimer's Association, 2018).

Por último, en la fase de demencia por EA se verán afectadas más zonas cerebrales, como la encargada de la motricidad haciendo que se desarrolle apraxia. En general, a medida que avanza esta fase, aparecerá una atrofia cerebral global por la que el enfermo será incapaz de ser independiente. Los síntomas de las fases anteriores empeorarán y aparecerán nuevos síntomas más graves y agresivos, como problemas para tragar o moverse. Estos dos síntomas son la causa de muerte más común de los enfermos de EA (Alzheimer's Association, 2018). Por una parte, los problemas para tragar pueden derivar en asfixia o en una neumonía por aspiración y, por otra, los problemas para moverse harán que el enfermo esté postrado en una cama, lo que favorecerá la aparición de coágulos, infecciones de piel o sepsis que provocará un síndrome de disfunción multiorgánica que a su vez derivará en la muerte del sujeto (Alzheimer's Association, 2018).

2.4 Diagnóstico

El diagnóstico en las primeras etapas de la enfermedad es fundamental por lo que se buscan biomarcadores que puedan aportar información sobre el estadio de la enfermedad en el que se encuentre el sujeto. Estos biomarcadores son:

concentración de la proteína beta-amiloide, concentración de la proteína tau y concentración de la proteína tau fosforilada (Allegri *et ál*., 2011; Blennow *et ál*., 2006; Dubois *et ál*., 2016; Jack *et ál*., 2018). El diagnóstico puede ser llevado a cabo mediante neuroimágenes obtenidas mediante técnicas de neuroimagen como MRI o PET. Estas imágenes pueden detectar los cambios estructurales del cerebro, es decir, pueden detectar los biomarcadores gracias a la detección de las posibles lesiones producidas por la enfermedad (Allegri *et ál*., 2011; Blennow *et ál*., 2006; Dubois *et ál*., 2016; Jack *et ál*., 2018). También pueden realizarse extracciones de sangre para un posterior análisis en búsqueda de la presencia del gen APOE (Allegri *et ál*., 2011; Blennow *et ál*., 2006; Dubois *et ál*., 2016; Jack *et ál*., 2018).

Además de estas técnicas, existen diferentes test neuropsicológicos que ayudan al diagnóstico de la EA. Uno de los más comunes es el *Mini Mental State Examination* (MMSE) (Folstein *et ál*., 1975). Sin embargo, este test no tiene mucha precisión en el diagnóstico (Sabbagh *et ál*., 2017), por lo que suele complementarse con el test del reloj y la comprobación de la fluencia semántica (Sabbagh *et ál*., 2017) . En general, cuando se emplea este test, suele hacerse una evaluación cognitiva completa que alcanzará mayor precisión diagnóstica cuanto mayor sea el número de funciones cognitivas evaluadas. Las funciones cognitivas a evaluar son: atención, memoria, lenguaje, funciones ejecutivas, abstracción y razonamiento (Allegri *et ál*., 2011; Jack *et ál*., 2018).

Otra alternativa diagnóstica que está siendo estudiada desde hace se basa en el EEG (Jeong, 2004), que lleva décadas siendo estudiado como un biomarcador potencial para la EA Numerosos estudios sugieren que los registros EEG contienen información neuronal fundamental que puede ser de gran utilidad para detectar la evolución de la enfermedad (Allegri *et ál*., 2011; Dubois *et ál*., 2016; Jeong, 2004). No obstante, hasta el momento, el uso del EEG como herramienta de ayuda al diagnóstico de la EA está en fase de investigación y no está recomendado en los criterios de diagnóstico (Jack *et ál*., 2018).

Capítulo 3

3. Electroencefalografía

3.1 Introducción a la electroencefalografía 35
3.2 Historia de la electroencefalografía 35
3.3 Neurofisiología 35
3.4 Registros EEG 36
3.5 Alteraciones de las señales EEG debido a la evolución de la enfermedad de Alzheimer .. 38

3.1 Introducción a la electroencefalografía

La electroencefalografía hace referencia a la técnica que se centra en la medida de la actividad eléctrica del cerebro. Es una técnica no invasiva utilizada en la investigación y en distintos entornos clínicos por ser una señal con una gran resolución temporal (<1ms), lo cual es muy adecuado para investigar los dinámicos ritmos cerebrales. No sólo tiene una alta resolución temporal, sino que también es fácil de adquirir y tiene un bajo coste (Babiloni *et ál.*, 2009; Cohen, 2017).

La alta resolución temporal de la señal y su bajo coste de adquisición hacen que esta técnica supere a algunas técnicas como la imagen de resonancia magnética (*Magnetic Resonance Imaging*, MRI) funcional o la tomografía por emisión de positrones (*Positron emission tomography*, PET). Una de sus principales desventajas es que tiene una resolución espacial baja (5-9 cm), que depende del número de electrodos utilizados para obtener la señal (Babiloni *et ál.*, 2009). Las configuraciones más comunes utilizadas para la adquisición de la señal son las que utilizan menos de 128 sensores, configuraciones de 16, 32 y 64 electrodos (Song *et ál.*, 2015).

La actividad cerebral recogida por el EEG es la generada por la actividad postsináptica de las neuronas piramidales perpendiculares a la superficie cerebral (Babiloni *et ál.*, 2009; Song *et ál.*, 2015). Las medidas que recoge esta señal son medidas diferenciales con respecto a una referencia, como por ejemplo, el lóbulo de la oreja (Babiloni *et ál.*, 2009). Esto hace que las señales sean dependientes de esta referencia; esta dependencia, junto con la alta sensibilidad al ruido de esta técnica, son las principales limitaciones del EEG (Babiloni *et ál.*, 2009).

3.2 Historia de la electroencefalografía

En la década de 1870 los primeros ritmos eléctricos espontáneos del cerebro de animales mamíferos fueron demostrados por el fisiólogo inglés Richard Caton (Millett, 2001). Fue el mismo fisiólogo quién en 1875 estudió la actividad eléctrica cerebral como consecuencia de ciertas corrientes generadas por la actividad neural (Gil-Nagel, 2001). Posteriormente, el EEG fue medido en humanos por primera vez en 1929 gracias a Hans Berger, un fisiólogo y psiquiatra alemán. Gracias a este psiquiatra el EEG consiguió ser reconocido y aceptado por la comunidad científica como un método para la obtención de señales cerebrales de sujetos sanos o con patologías (Babiloni *et ál.*, 2009; Gil-Nagel, 2001). Actualmente, el EEG sigue siendo una de las técnicas de diagnóstico más presentes en el ámbito clínico (Gil-Nagel, 2001). Además, los últimos avances tecnológicos permiten que el EEG pueda competir con técnicas de neuroimagen ampliamente consolidadas, como la MRI (Gil-Nagel, 2001).

3.3 Neurofisiología

La actividad eléctrica recogida por el EEG es la que se produce en el intercambio de señales electroquímicas a través de la sinapsis. El EEG registra las corrientes extracelulares que reflejan los potenciales postsinápticos sumados en miles o

millones de células piramidales alineadas entre sí de forma paralela y perpendicularmente con la superficie cerebral (Babiloni *et ál*., 2009; Cohen, 2017; Gil-Nagel, 2001). Estos potenciales pueden ser excitadores y/o inhibidores. Para que estos potenciales puedan ser detectados las neuronas vecinas deben estar sincronizadas de manera que produzcan una actividad eléctrica mínima. Debe existir actividad sináptica en, al menos, 6 cm^2 de corteza cerebral (Gil-Nagel, 2001). Estos 6 cm^2 de corteza cerebral capaces de generar actividad eléctrica son considerados generadores. Existen tres áreas de generación (Gil-Nagel, 2001):

- **Generador A.** Este generador está situado a aproximadamente 500 μm de la superficie cortical y genera ondas negativas en superficie.
- **Generador B.** Está ubicado a 900 μm de la superficie cortical por lo que es una zona más profunda que la anterior. En este caso, produce ondas positivas.
- **Generador C.** Está ubicado a la misma profundidad que en generador B, a 900 μm y produce ondas negativas.

Estas ondas generadas por las distintas zonas de la superficie cortical, se agrupan en diferentes frecuencias formando los ritmos cerebrales u oscilaciones (Cohen, 2017). Los principales ritmos cerebrales son los siguientes (Dauwels *et ál*., 2011; Guevara *et ál*., 2010):

- **Ritmos delta (δ).** Son los ritmos más lentos, con frecuencias inferiores a 4 Hz. Predominan en la zona frontal o parieto-occipital con amplitudes que varían entre los 100 y los 200 μV y están relacionados con el sueño profundo, la hiperventilación o con enfermedades cerebrales graves.
- **Ritmos zeta (θ).** Están situados en la banda de 4 a 8 Hz, predominan en la zona occipital. Es la componente principal del EEG en niños y aparecen en diversas situaciones como mantener los ojos abiertos o dormir. Tienen una amplitud de 70μV.
- **Ritmos alfa (α).** Oscilan entre los 8 y los 13 Hz predominando en la zona occipital. Aparecen en sujetos normales despiertos que no realizan ninguna actividad y mantienen los ojos cerrados o en estados comatosos o sueño REM. Su amplitud se aproxima a los 50μV.

  Ritmos beta (β). Estos ritmos oscilan entre los 13 y los 30 Hz. Predominan en la zona frontal con una amplitud de entre 20 y 30 μV. Aparecen en situaciones como dormir o abrir los ojos. En condiciones como administración de medicamentos o estados comatosos se dividen en:
  - Ondas β_1 que oscilan entre 13 y 18 Hz.
  - Ondas β_2 que oscilan entre 18 y 30 Hz.
- **Ritmos gamma (γ).** Oscilan en la banda de 30 a 100 Hz, predominan en las zonas central y frontal y son las señales que menos amplitud presentan, de entre 10 y 20 μV.

3.4 Registros EEG

El procedimiento necesario para la adquisición de las señales EEG es conocido como registro electroencefalográfico (Gil-Nagel, 2001; Song *et ál.*, 2015). Este proceso puede llevarse en determinadas situaciones clínicas donde el estado de los sujetos puede variar. Destacan dos situaciones, la primera es en la que el sujeto está en reposo, relajado y, generalmente, con los ojos cerrados; en esta situación se consigue un registro basal o en estado de reposo (Bachiller, 2012). La segunda situación destacable es en la que se somete al sujeto a estímulos externos, visuales o auditivos; esto se conoce como potenciales evocados (*Event Related Potentials*, ERP) donde se busca obtener la respuesta al estímulo. Un ejemplo de ERP sería el potencial P300, caracterizado por un pico en la amplitud de la señal que aparece a los 300 ms (Bachiller, 2012).

Existen diferentes configuraciones de electrodos utilizadas para la obtención de esta señal. La posición de los electrodos en cada una de estas configuraciones es importante, ya que las señales bioeléctricas tienen una alta variabilidad. La configuración más común es el Sistema Internacional 10-20, que utiliza 21 electrodos situados en distintas zonas cerebrales. Esta configuración fue presentada por Henri Jasper en 1949 y aceptada como convenio para la posición de los distintos canales o electrodos (Gil-Nagel, 2001). Esta configuración sigue un criterio fijo para nombrar cada electrodo utilizado. En primer lugar, se indica la zona cerebral en la que se coloca cada electrodo y, en segundo lugar, se indica su posición con respecto a la línea media sagital del cráneo. La Figura 7 muestra la posición exacta y el nombre de cada uno de los 21 electrodos utilizados en la configuración 10-20.

Las letras utilizadas para indicar la zona cerebral son (Gil-Nagel, 2001):

- **O:** utilizada en los electrodos situados en la zona occipital.
- **P:** utilizada en los electrodos situados en la zona parietal.
- **T:** utilizada en los electrodos situados en la zona temporal.
- **C:** utilizada en los electrodos situados en la zona central.
- **F:** utilizada en los electrodos situados en la zona frontal.
- **Fp:** utilizada en los electrodos situados en la zona frontopolar.

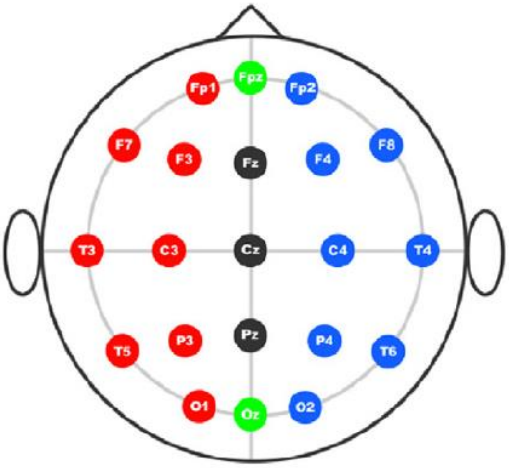

Figura 7: Distribución de electrodos utilizada en Sistema Internacional 10-20. Figura obtenida de (Song *et ál.*, 2015)

En cuanto a la posición con respecto a la línea media sagital:

- **Número par:** el electrodo está situado a la derecha de la línea media.
- **Número impar:** el electrodo está situado a la izquierda de la línea media.
- **Z:** el electrodo está situado en la línea media sagital.

Como ya se ha explicado en este capítulo, el EEG recoge medidas diferenciales, por lo que el uso de una referencia es necesario. Existen dos tipos de registros principales dependiendo del tipo de referencia utilizada (Gil-Nagel, 2001):

- **Registro monopolar.** Este tipo de registro obtiene las medidas entre cada electrodo, de manera independiente, y una referencia. Las principales opciones para crear esta referencia son: (*i*) utilizar un electrodo situado en el mentón o en el lóbulo de una oreja; (*ii*) crear un punto de referencia mediante el cálculo del promedio de la actividad registrada por todos los electrodos utilizados.
- **Registro bipolar.** Este tipo de registro suele utilizarse cuando la configuración elegida utiliza un número de electrodos bajo. En este caso, se emparejan los electrodos de dos en dos y se estima la diferencia de potencial existente entre la señal obtenida por cada uno de ellos.

3.5 Alteraciones de las señales EEG debido a la evolución de la enfermedad de Alzheimer

La EA se caracteriza por ser un trastorno neurodegenerativo progresivo que produce un deterioro cognitivo caracterizado por la pérdida tanto de memoria como de las distintas capacidades del sujeto impidiendo que éste pueda desarrollar con normalidad su vida diaria (Cummings, 2004). Al tratarse de un deterioro cognitivo, se esperan cambios a nivel cerebral, ya que esta enfermedad afecta a la actividad neuronal. Estos cambios afectarán a la actividad eléctrica cerebral, por lo que podrán ser recogidos en señales EEG. Los EEG de personas que padecen la enfermedad presentan diversas alteraciones destacables con respecto a los EEG de personas sanas: lentificación del espectro, desconexión entre áreas cerebrales, disminución de la coherencia, pérdida o disminución de complejidad y la irregularidad de los registros (Dauwels *et ál.*, 2011; Jeong, 2004).

La alteración más estudiada es la lentificación del espectro. Se caracteriza por una reducción de los ritmos alfa y beta, es decir, una disminución de la actividad eléctrica en las bandas de 8 a 13 Hz y de 13 a 30 Hz (Dauwels *et ál.*, 2011). Existe una correlación, tanto lineal como no lineal, entre esta lentificación y el grado de demencia asociado a la EA medido mediante el MMSE (Babiloni *et ál.*, 2009; Jeong, 2004).

En cuanto a la complejidad de la señal EEG, existe una pérdida significativa de la misma en enfermos de EA, que no es tan clara en el caso de enfermos con DCL (Dauwels *et ál.*, 2011). Para evaluar estas diferencias, medir la complejidad y demostrar la disminución de ésta a medida que avanza la enfermedad, puede utilizarse la complejidad de Lempel-Ziv (Abásolo & Simons, 2014) que, además de

demostrar la disminución de la complejidad en la banda alfa, demuestra la existencia de un aumento de ésta en bandas de frecuencias bajas (Abásolo & Simons, 2014; Dauwels *et ál.*, 2011). Por otra parte, la regularidad de las señales EEG puede medirse calculando la entropía de los registros obtenidos. Con este cálculo, se detecta una disminución de la entropía en registros de personas que padecen EA, lo que implica un aumento de la regularidad de la señal. Esta pérdida de irregularidad puede estar relacionada con la pérdida de complejidad de la señal (Dauwels *et ál.*, 2011).

La EA ha sido definida como síndrome de desconexión cortical que implica una pérdida cognitiva derivada de la degeneración de las conexiones corticales que conectan las áreas cerebrales más lejanas (Leutcher *et ál.*, 1992). Esto es debido a que las placas seniles y los ovillos neurofibrilares afectan a la sinapsis de las fibras en dichas conexiones corticales (Jeong, 2004). La degeneración de las conexiones corticales hace que se hayan desarrollado diferentes estudios con el fin de obtener distintas formas para determinar la conectividad cerebral. Algunos de estos estudios se basan en distintos parámetros de la Teoría de Grafos como el coeficiente de agrupamiento y la longitud de camino característico (Si *et ál.*, 2019). El primero de estos parámetros se emplea para obtener la segregación de la red. Con este parámetro se determina el número de triángulos cerrados que rodean un nodo, si este número es alto entonces se considera que segregación (Rubinov & Sporns, 2010; Si *et ál.*, 2019). En cuanto al segundo parámetro, se utiliza para obtener la integración de la red. La longitud de camino característico indica la secuencia de nodos y enlaces recorridos para llegar de un nodo de la red a otro. Cuanto mayor sea este camino, menor será la integración de la red (Rubinov & Sporns, 2010).

Por último, cabe destacar que a lo largo de los últimos años se han realizado numerosos estudios dirigidos a identificar nuevos biomarcadores de la EA mediante el uso de señales EEG, por su bajo coste de adquisición y por ser un método no invasivo (Babiloni *et ál.*, 2009; Rossini *et ál.*, 2020). Sin embargo, la complejidad de la enfermedad, la dificultad de aplicación de las metodologías propuestas y la utilización de bases de datos aisladas ha hecho que los resultados no alcancen las precisiones de clasificación y los niveles de robustez diagnóstica requeridos (Rossini *et ál.*, 2020). Por ello, es necesario seguir profundizando en nuevos enfoques metodológicos, suficientemente generalizables, que sean capaces de dar el salto requerido para que el EEG se constituya como una alternativa real a otras técnicas de neuroimagen como el PET o el MRI.

Capítulo 4

4. Aprendizaje profundo o *Deep Learning*

4.1 Introducción **41**

4.2 Redes neuronales convolucionales **45**

4.2.1 Etapa convolucional **48**

4.2.2 Etapa de detección **49**

4.2.3 Etapa de submuestreo o *pooling* **49**

4.3 Arquitectura seleccionada y algoritmos utilizados **50**

4.3.1 Arquitecturas utilizadas **54**

4.3.2 Algoritmos de optimización, métodos de regularización para la prevención del overfitting y funciones de pérdida **55**

4.1 Introducción

Para comprender correctamente qué es el *Deep Learning* o aprendizaje profundo es necesario explicar qué es el *Machine Learning*, así como las similitudes y diferencias existentes entre ambos. En el Capítulo 1 se han definido tanto ML como DL dentro del conjunto de AI, tal y como se resume en la Figura 8.

Normalmente, la estructura general de modelos tanto de ML como de DL consiste en una o varias capas de entrada, una serie de capas escondidas y una o varias capas de salida. La diferencia principal entre estas estructuras es la profundidad de las capas intermedias y la necesidad de intervención humana (Nouri, 2012). La Figura 9 muestra el mismo problema de clasificación para ML y para DL.

En la Figura 9 A se observa que la intervención humana es necesaria para la extracción de las características. Los datos deben ser procesados para poder extraer estas características y deben crearse algoritmos complejos que realicen dicha extracción. En el caso de la Figura 9 B, esta intervención no es necesaria, ya que el DL es capaz de obtener las características a partir de los datos en crudo (LeCun *et ál.*, 2015; Nouri, 2012). No existe la necesidad de crear capas para realizar la extracción, pues las características son aprendidas o extraídas de los propios datos (LeCun *et ál.*, 2015). Esta es la principal diferencia entre ML y DL, pero no es la única. Otra de las diferencias más importantes es la mostrada en la Figura 10: la cantidad de datos necesaria para que los algoritmos de DL obtengan buenos resultados debe ser mucho mayor que la necesaria para los algoritmos de ML. Por último, otra de las ventajas del DL es que alcanza un mayor nivel de abstracción. Esto permite identificar patrones más completos que los identificados por los métodos de extracción de características convencionales (LeCun *et ál.*, 2015).

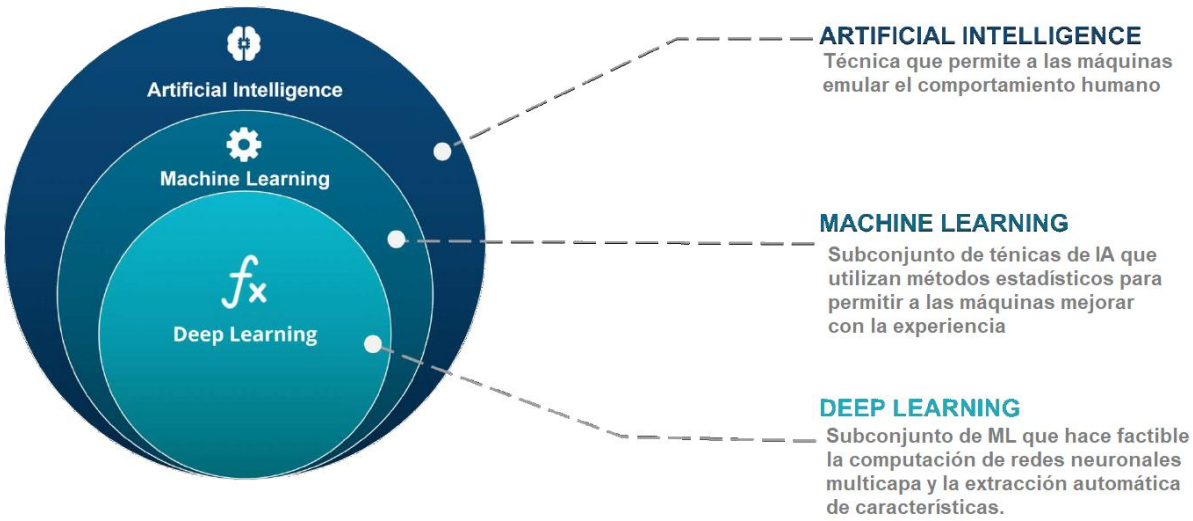

Figura 8: Taxonomía y definiciones de AI, ML y DL. Figura adaptada de (Nouri, 2012).

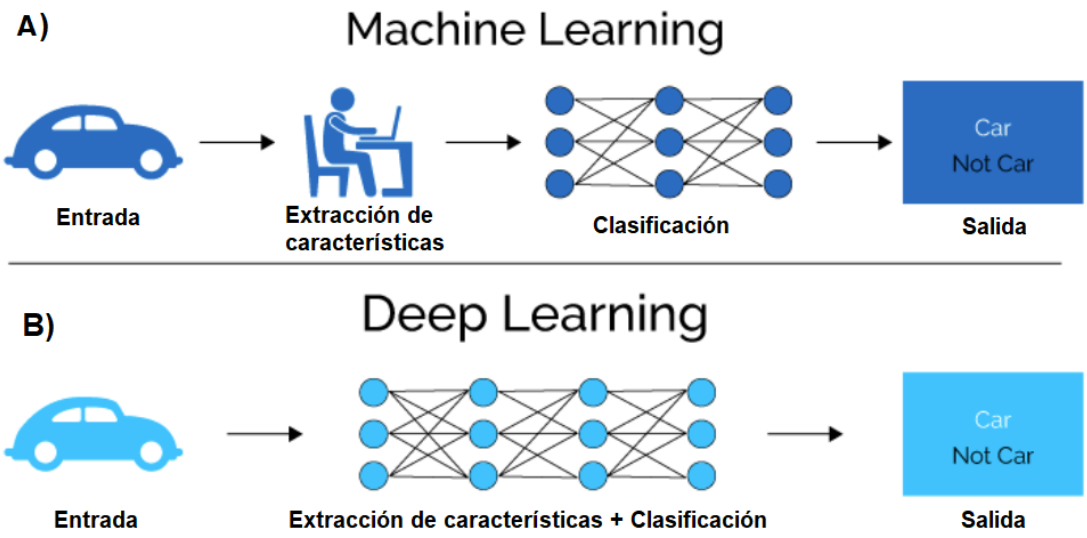

Figura 9: Estructura de un problema de clasificación para ML (A) y DL (B). Figura adaptada de (Nouri, 2012).

En cuanto a las similitudes, las redes o modelos utilizados tanto en ML como en DL necesitan ser entrenados. Una regresión logística (método de ML) o una red neuronal profunda (método de DL) necesitan este proceso para poder aprender a realizar la tarea que se les ha encomendado. Existen varios tipos de aprendizaje utilizados en el entrenamiento de estos modelos o redes: aprendizaje supervisado (*supervised learning)*, aprendizaje no supervisado (*unsupervised learning)*, aprendizaje reforzado (*reinforcement learning)* y transferencia de aprendizaje (*transfer learning*) (Goodfellow *et ál*., 2016; LeCun *et ál*., 2015; Raschka & Mirjalili, 2017). Cada uno de estos tres primeros métodos de aprendizaje está basado en el funcionamiento de distintas zonas cerebrales como se muestra en la Figura 11 Así pues, el aprendizaje supervisado se corresponde con el cerebelo, el no supervisado con el córtex cerebral y el reforzado con los ganglios basales (Hu *et ál*., 2007).

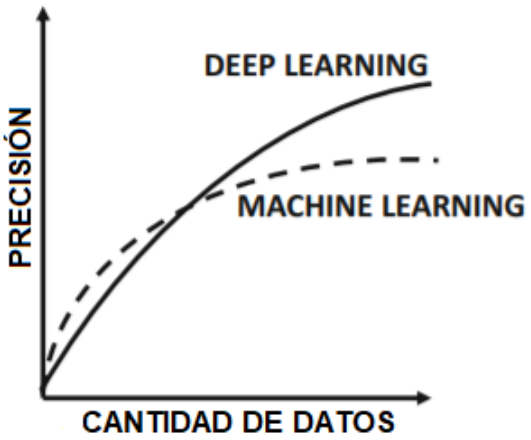

Figura 10: Rendimiento de DL y ML con la cantidad de datos utilizada. Figura adaptada de (Goodfellow *et ál*., 2016).

El paradigma de aprendizaje más común para ambos subconjuntos, ML y DL, es el aprendizaje supervisado (LeCun *et ál.*, 2015). Este paradigma se caracteriza por utilizar conjuntos de datos etiquetados. Esto implica utilizar características con una etiqueta que indica la clase a la que pertenece, es decir, la salida que debería tener el modelo para esa característica (Alom *et ál.*, 2018; Goodfellow *et ál.*, 2016; Raschka & Mirjalili, 2017). El objetivo de este paradigma es entrenar un modelo o agente utilizando estos datos etiquetados que sea capaz de realizar predicciones sobre conjuntos de datos nuevos o sin etiquetar (Raschka & Mirjalili, 2017). El aprendizaje supervisado suele utilizarse en algoritmos de clasificación y regresión (Alom *et ál.*, 2018; Raschka & Mirjalili, 2017).

Otro paradigma es el aprendizaje no supervisado, que se caracteriza por no utilizar datos etiquetados. En este caso, el modelo o agente será capaz de aprender la representación interna de los datos o extraer características importantes para obtener relaciones o posibles patrones que poseen los datos utilizados como entrada al modelo. Este paradigma suele utilizarse en problemas de *clustering* o reducción de dimensionalidad (Alom *et ál.*, 2018; Raschka & Mirjalili, 2017).

El paradigma de aprendizaje reforzado, también conocido como aprendizaje semi-supervisado, se caracteriza por utilizar tanto datos etiquetados como datos sin etiquetar. Este paradigma funciona como un sistema de recompensa. El objetivo de este aprendizaje es crear un sistema o agente que mejore su funcionamiento basándose en interacciones con el entorno. Para conseguir este objetivo, se utiliza una métrica de recompensa de forma que el agente aprenderá mejor o peor en función de lo buena o mala que ésta sea. La Figura 12 muestra el esquema de funcionamiento de este paradigma.

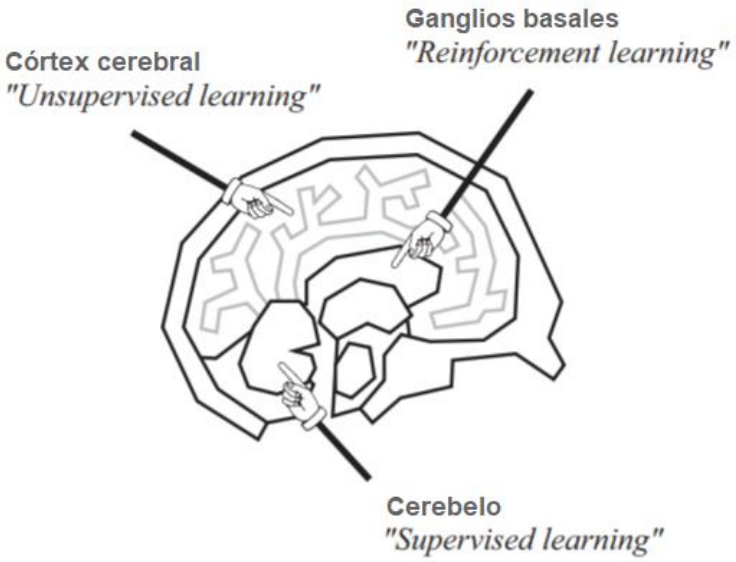

Figura 11: Paradigmas de aprendizaje asociados a las distintas zonas cerebrales. Figura adaptada de (Hu *et ál.*, 2007)

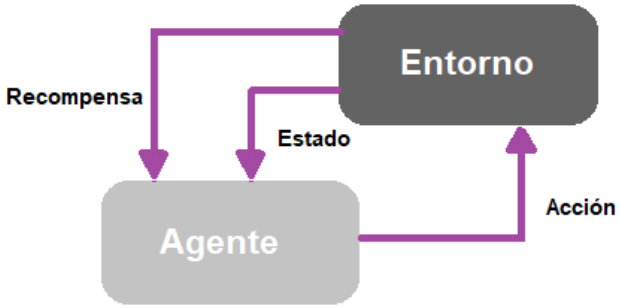

Figura 12: Esquema de funcionamiento del paradigma aprendizaje reforzado. Figura adaptada de (Raschka & Mirjalili, 2017)

A través de su interacción con el entorno, un agente utilizará este paradigma para aprender una serie de acciones que maximicen la recompensa mediante ensayo y error o incluso con cierta planificación, cambiando la estrategia utilizada en interacciones anteriores (Alom *et ál.*, 2018; Raschka & Mirjalili, 2017).

La Figura 13 representa estos tres paradigmas con sus principales aplicaciones.

Por último, existe otro paradigma llamado *transfer learning* o transferencia de aprendizaje. Este paradigma implica utilizar el conocimiento adquirido por un modelo, diseñado para una tarea específica y entrenado con cualquiera de los otros tres paradigmas ya explicados, para resolver otro tipo de tareas. Para esto, se utilizará un modelo ya entrenado para un determinado fin, utilizando ese mismo modelo para otro fin distintos y con nuevos datos (Pan & Yang, 2010). La Figura 14 muestra las principales diferencias entre el proceso de aprendizaje tradicional de DL o ML, Figura 14 A, y el utilizado en *transfer learning*, Figura 14 B.

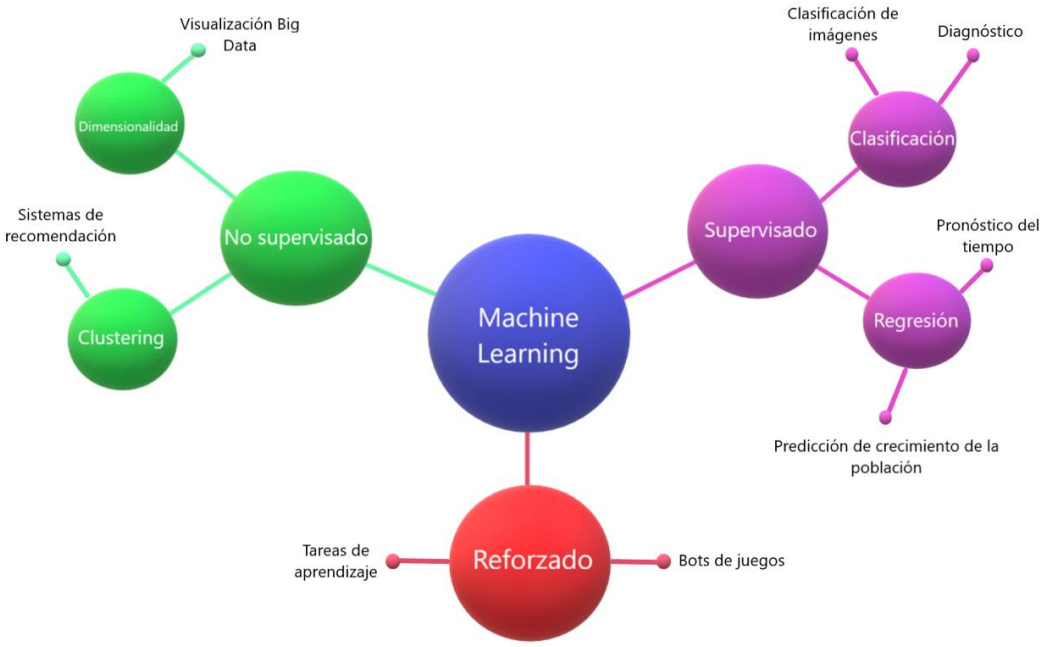

Figura 13: Aplicaciones principales de los tres paradigmas de aprendizaje: supervisado, no-supervisado y reforzado.

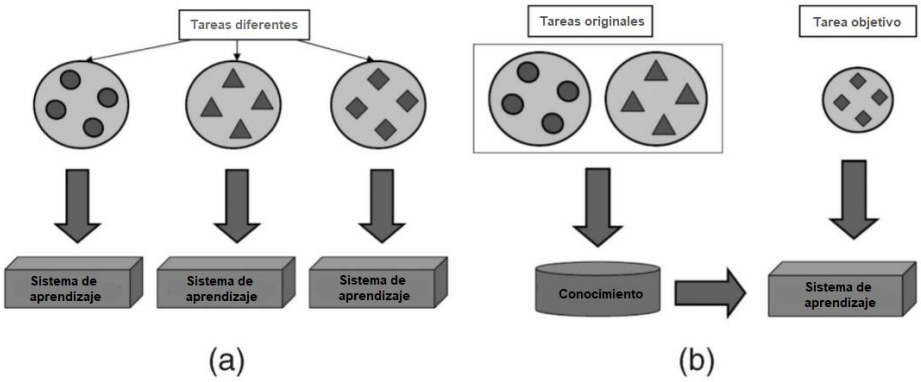

Figura 14: Diferencia entre los procesos de aprendizaje tradicionales de ML y DL (a) y el utilizado en *transfer learning* (b). Figura adaptada de (Pan & Yang, 2010).

4.2 Redes neuronales convolucionales

Una red neuronal convolucional es un tipo de red neuronal profunda. Gracias a su gran eficiencia en tareas como la clasificación de imágenes, es uno de los tipos de redes más utilizados actualmente y el primero que consiguió posicionar al DL en el lugar donde se encuentra hoy (Goodfellow *et ál*., 2016; Raschka & Mirjalili, 2017). Este tipo de redes ha conseguido igualar e incluso superar la eficiencia conseguida por los humanos en determinados tipos de tareas como: reconocimiento de voz y facial, traducción de textos y tareas de clasificación en general (Aggarwal, 2018; LeCun *et ál*., 2015). Se caracteriza por ser capaz de modelar correlaciones tanto espaciales como temporales en señales con varias dimensiones. Además, son redes capaces de considerar y utilizar la topología de los datos de entrada (Kok, 2009). En el ámbito del procesado de señales biomédicas, la representación tanto espectral como tiempo-frecuencia de las distintas señales utilizadas muestra una fuerte correlación. Las CNNs son capaces de modelar esta correlación de manera sencilla gracias al *weight sharing* o los pesos compartidos, por lo que este tipo de red será ampliamente utilizada en este ámbito (Kok, 2009). Este tipo de red neuronal está inspirada en el córtex visual primario (Aggarwal, 2018; Goodfellow *et ál*., 2016; Kok, 2009). Crea una jerarquía de características combinando características de bajo nivel en distintas capas para formar características de alto nivel. Si se trabaja con imágenes, las primeras capas identificarían los bordes, las siguientes manchas o zonas con color y combinando las características que extrae cada capa, se forma una imagen final (Goodfellow *et ál*., 2016).

Existen dos ideas fundamentales relacionadas con las CNNs: la *sparse connectivity* o *sparse interaction* y el *weight sharing* o *parameter sharing.* En el caso de la primera idea, las redes neuronales tradicionales utilizan una conexión de uno con todo, denominada *full conectivity*, lo cual implica que cada neurona de las distintas capas está conectada con todas las neuronas de la capa siguiente como se muestra en la Figura 15 (Goodfellow *et ál*., 2016). En el caso de las redes convolucionales, se utiliza *sparse connectivity*, que implica que cada neurona de una capa de la red estará conectada con tantas neuronas como se indique en el filtro o *kernel,* tal y como se muestra en la Figura 16.

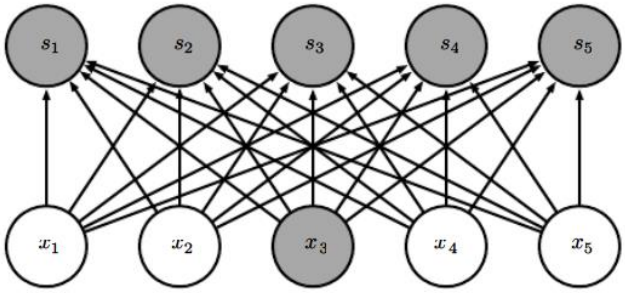

Figura 15: Conexión entre neuronas de dos capas de una red neuronal tradicional. Figura de (Goodfellow *et ál*., 2016)

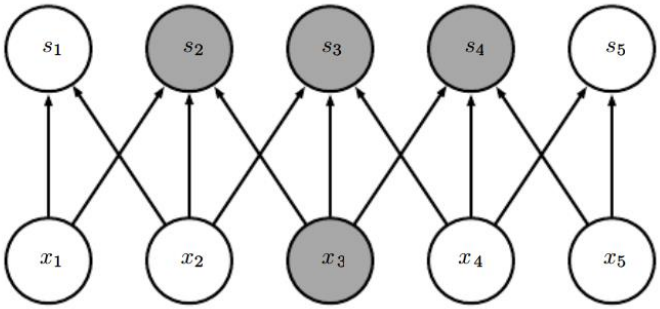

Figura 16: Conexión entre neuronas de dos capas de una red neuronal convolucional con un tamaño de kernel 3. Figura de (Goodfellow *et ál*., 2016).

Esto se consigue haciendo que el *kernel* tenga dimensiones menores a la dimensión de la entrada (Goodfellow *et ál*., 2016). Esta nueva forma de conexión entre neuronas permite que la red describa de manera eficiente interacciones complejas entre muchas variables, mediante la creación de pequeños bloques de interacciones de menor tamaño (Goodfellow *et ál*., 2016).

La otra idea fundamental es el *parameter sharing* representado en la Figura 17. Tal y como ocurre con la *sparse connectivity*, en las redes neuronales tradicionales cada elemento de la matriz de pesos se utiliza una única vez cuando se calcula la salida de la capa (Goodfellow *et ál*., 2016). Las CNNs hacen un uso compartido de parámetros cuando realizan las convoluciones, tal y como se observa en la Figura 18. Esto implica que, en lugar de tener que aprender un conjunto de parámetros para cada ubicación del mapa de características, se aprende un único conjunto para todas las ubicaciones. Utilizar esta técnica reducirá los requisitos de almacenamiento del modelo (Goodfellow *et ál*., 2016).

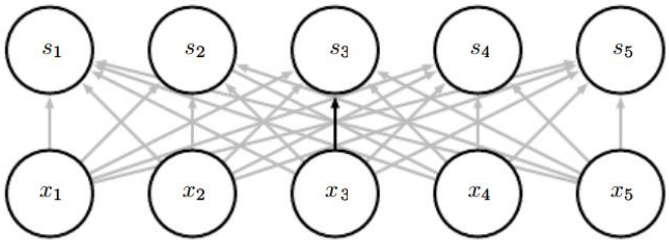

Figura 17: La flecha negra indica que el parámetro de la matriz de pesos correspondiente a esa neurona se utiliza una única vez. Figura de (Goodfellow *et ál*., 2016)

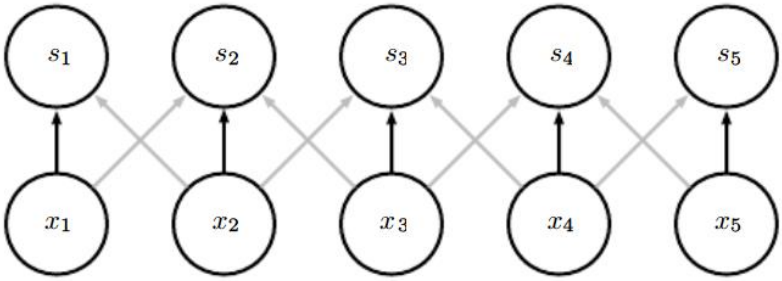

Figura 18: Las flechas negras implican que al aprender un único conjunto de parámetros para cada ubicación gracias al uso del *parameter sharing* todas las neuronas de la red utilizarán ese conjunto. Figura de (Goodfellow *et ál*., 2016).

Normalmente, la arquitectura de una capa de una CNN suele ser la mostrada en la Figura 19. Típicamente, estas capas suelen estar formadas por tres etapas: la etapa de convolución, la de activación y la de *pooling* o submuestreo. Primero, se introducen los datos a una capa que realiza un número determinado de convoluciones en paralelo para extraer un conjunto de *feature maps* o mapas de características. En la segunda etapa, cada *feature map* extraído en la etapa anterior atraviesa una función de activación no lineal, como por ejemplo la función de activación lineal rectificada (*Rectified Linear Unit,* ReLU). Al finalizar esta etapa de activación, se pasa a la etapa de submuestreo que utiliza funciones de *pooling* para modificar la salida de la capa.

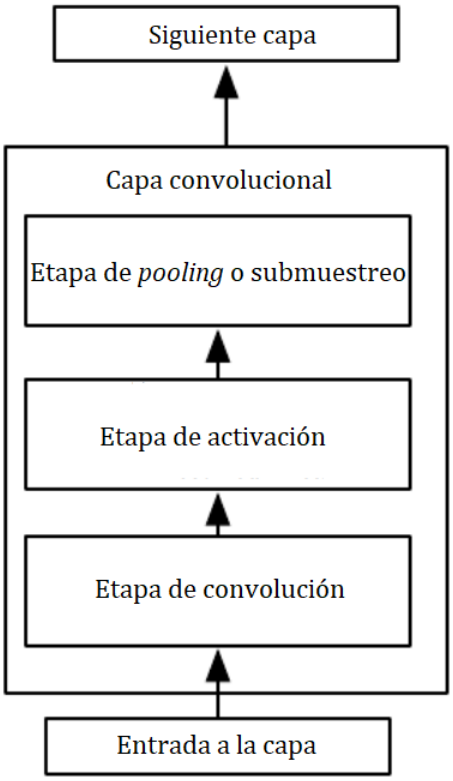

Figura 19: Arquitectura de una capa convolucional. Figura adaptada de (Goodfellow *et ál*., 2016).

4.2.1 Etapa convolucional

La etapa de convolución o convolucional es la encargada de la extracción automática de características. Obtiene los *feature maps* gracias a las distintas de convoluciones entre los datos de entrada a la capa y el filtro o *kernel*. Las CNNs utilizan convoluciones discretas cuya notación es $y = (x * w)$. Una convolución discreta entre dos vectores de una dimensión tiene la siguiente definición matemática (Raschka & Mirjalili, 2017):

$$y[i] = \sum_{k=-\infty}^{\infty} x[i-k] \cdot w[k]. \quad (1)$$

En este caso, la *x* será el vector de entrada a la capa convolucional y *w* será el *kernel*. El índice *i* representará cada elemento del vector de salida *y*, comúnmente denominado *feature map* (Goodfellow *et ál*., 2016). En problemas de DL se trabaja con vectores finitos; sin embargo, el índice *k* de la ecuación anterior es infinito y puede tomar valores negativos. Para que el sumatorio se calcule correctamente, se supone que tanto *x* como *w* contendrán ceros, lo que dará como resultado un vector *y* con un tamaño finito y con muchos ceros. Esto no es útil en situaciones prácticas, por lo que *x* se rellena solo con un número finito de ceros. Este proceso de relleno es conocido como *zero-padding* o *padding.* El número de ceros con el que se rellena *x*, tanto por la izquierda como por la derecha, se denota con *p*. Al añadir este proceso, si se desea realizar la convolución entre *x* y *w* con *n* y *m* elementos respectivamente, entonces la ecuación de la convolución discreta será la siguiente (Raschka & Mirjalili, 2017):

$$y[i] = \sum_{k=0}^{k=m-1} x^p[i+m-k] \cdot w[k]. \quad (2)$$

Existen varias opciones de *padding*: *full*, *same* y *valid*. La más utilizada es *same* para conseguir que el vector de salida tenga las mismas dimensiones que el vector de entrada. La menos utilizada es la opción *full,* ya que aumenta las dimensiones de la salida al utilizar una longitud de *padding* de *m-1*. La opción *valid* implica no rellenar con ceros (Raschka & Mirjalili, 2017).

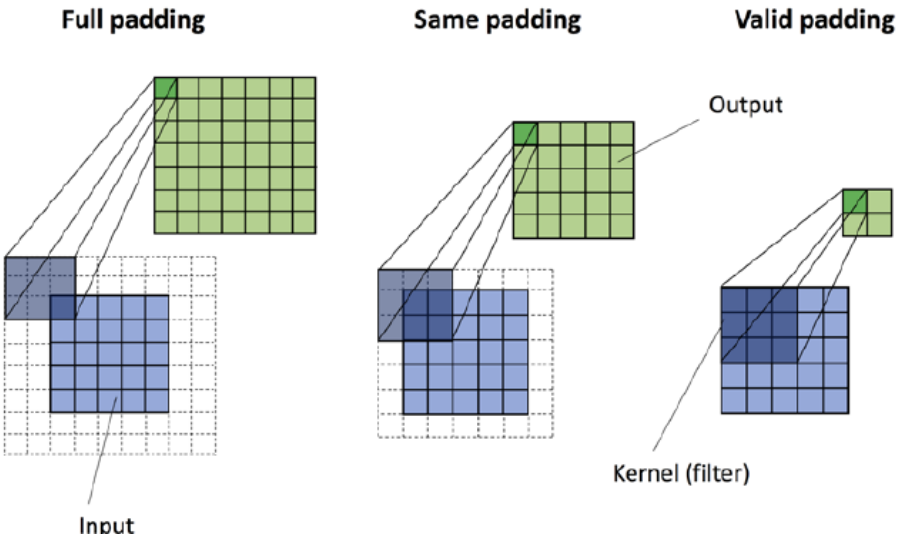

Figura 20: Tipos de *padding*. Figura adaptada de (Raschka & Mirjalili, 2017).

Calcular el tamaño del vector de salida es sencillo gracias a la siguiente fórmula:

$$o = \frac{n+2\cdot p-m}{s} + 1. \quad (3)$$

En este caso, n sería el tamaño del vector de entrada, p el factor de *padding*, m el tamaño del filtro y s, el *stride*, es decir, el número de celdas que debe moverse el filtro para realizar la convolución, siguiendo el ejemplo anterior: $o = \left(\frac{8-4}{2}\right) + 1 = 3$.

En el caso de convoluciones 2D, la fórmula sería la siguiente (Raschka & Mirjalili, 2017):

$$Y[i,j] = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} X[i-k_1, j-k_2] \cdot W[k_1, k_2]. \quad (4)$$

Para realizar este tipo de convoluciones es necesario rotar el filtro. La Figura 21 muestra un ejemplo de convolución discreta sin *padding*. Asimismo, un ejemplo de convolución 2D sin rotar el filtro se muestra en la Figura 22.

4.2.2 Etapa de activación

En esta etapa se umbralizan las características extraídas en la capa anterior y se pasan a la capa siguiente. La función de activación más utilizada en esta etapa es la ReLU. Esta función no cambia las dimensiones de la capa ,ya que lo que realiza es un mapeo uno a uno de los valores de activación de cada elemento de la capa convolucional (Aggarwal, 2018).

4.2.3 Etapa de submuestreo o *pooling*

El objetivo principal de esta capa es reducir las dimensiones de cara a la entrada de la siguiente capa. Esto es importante ya que, si se mantienen las dimensiones de una capa a otra, la carga computacional sería muy elevada. Para llevar a cabo esta reducción de las dimensiones, se realiza un muestreo preservando las características más importantes que detecta cada filtro (Raschka & Mirjalili, 2017).

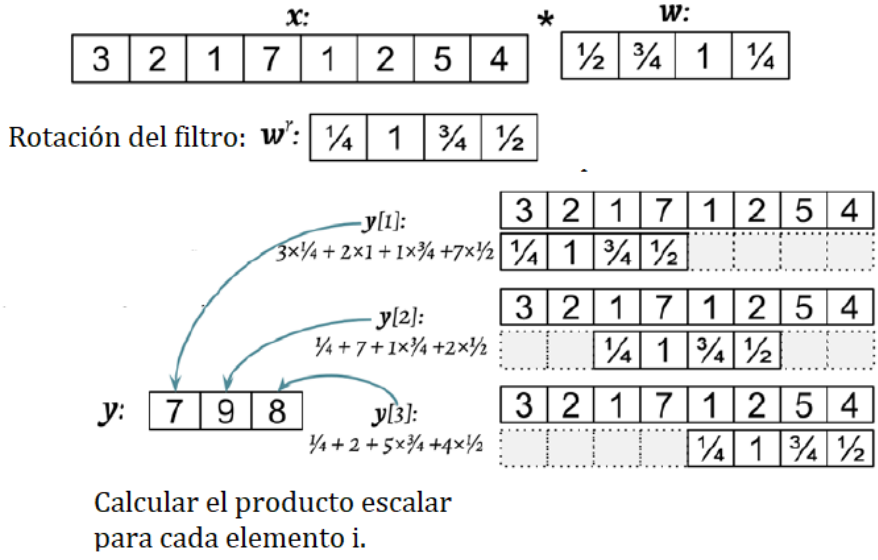

Figura 21: Convolución discreta 1D. Figura adaptada de (Raschka & Mirjalili, 2017).

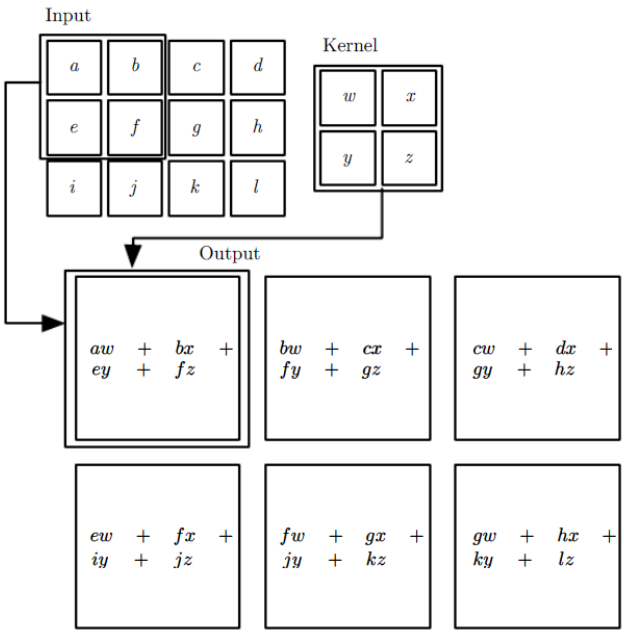

Figura 22: Ejemplo de convolución 2D. Figura de (Goodfellow *et ál.*, 2016).

Esta etapa se denota por P_{n1xn2} y determina el tamaño de bloques vecinos donde la función de *pooling* será aplicada. El tamaño de bloques vecinos implica el número de posiciones adyacentes utilizadas en cada dimensión. Esto se conoce como *pool size*. Por defecto, el tamaño que se aplica suele ser de dos vecinos. Una función de *pooling* reemplaza la salida de la red en un punto determinado, con una estadística resumida de las salidas cercanas. Existen dos funciones de submuestreo principales (ver Figura 23) (Aggarwal, 2018; Goodfellow *et ál.*, 2016; Raschka & Mirjalili, 2017):

- ***MaxPooling*.** Selecciona el valor máximo de cada área de vecinos. Esto implica que no haya varianza local, es decir, hace que los pequeños cambios en un área de vecinos no cambien el resultado de la agrupación total. Su uso ayuda a generar características más resistentes al ruido presente en los datos de entrada.
- ***MeanPooling* o *AveragePooling*.** Selecciona el valor medio de cada área de vecinos.

4.3 Arquitectura seleccionada y algoritmos utilizados

La programación de las redes neuronales utilizadas en este TFG se ha realizado utilizando la librería *Keras* de *Python* (Chollet, 2020) . Concretamente, se han utilizado CNNs, explicadas en el apartado anterior. Se han escogido las distintas arquitecturas a utilizar de manera experimental. Todas las arquitecturas utilizadas comparten algunas capas, por lo que, en este apartado, se explicará cada una de las capas utilizadas y, posteriormente, se profundizará en algunas de las arquitecturas utilizadas. Todas las arquitecturas comienzan con la definición del tipo de modelo que se desea utilizar, en el caso de este TFG, se ha utilizado el modelo secuencial de *keras*. Este tipo de modelo permite construir redes neuronales capa a capa (Chollet, 2020).

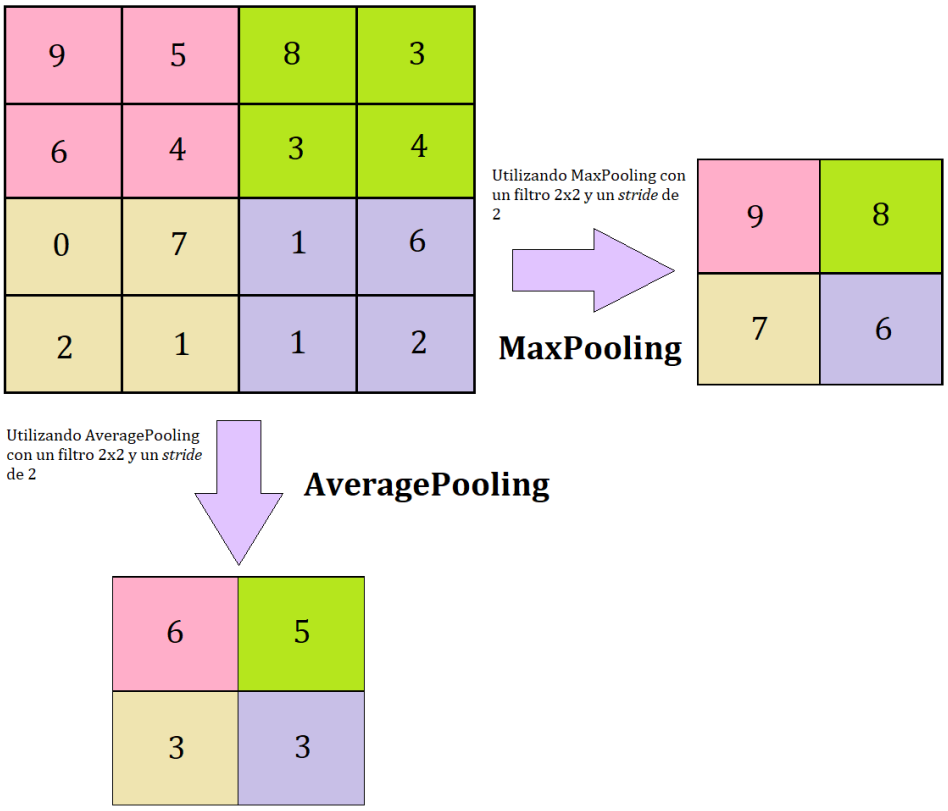

Figura 23: Ejemplo de *MaxPooling* y *AveragePooling*.

A continuación, se inserta la primera capa convolucional. Para ello, se utiliza una capa de tipo ***Conv1D***. Este tipo de capas son las utilizadas para realizar convoluciones en una dimensión. En esta capa deben especificarse ciertos parámetros; los más importantes son (Team keras, 2020c):

- ***Filters*:** debe ser un número entero que indicará el tamaño del espacio de salida, por ejemplo, el número de filtros de salida en la convolución.
- ***Kernel_size:*** será un número entero o una lista de números enteros que indican la longitud de la ventana deslizante utilizada para realizar la convolución.
- ***Strides:*** número entero que indica el número de espacios que debe moverse el filtro o *kernel*.
- ***Padding:*** se indica el tipo de *padding* que se desea realizar; por defecto, se utiliza el tipo *valid*.
- ***Activation:*** se especifica la función de activación que se desea utilizar después de la convolución; la más común es la ReLU, explicada en el apartado anterior.

En este caso, al tratarse de la primera capa que se inserta en el modelo, hay que especificar el tamaño de los datos de entrada. Es decir, las dimensiones de las señales que van a introducirse a la red. En el caso de este TFG, aunque se explicará en el Capítulo 5, se utilizarán 4 conjuntos de dimensiones distintas, una para cada caso de estudio: (2500,19), (1000,19), (200,19) y (500,19).

Se utilizarán tantas capas *Conv1D* como se desee. Como ya se ha explicado en el apartado anterior, ***4.2. Redes neuronales convolucionales***, la estructura de las capas convolucionales suele estar formada por la etapa convolucional, la etapa de detección y la etapa de *pooling*. En el caso de *keras*, la capa *Conv1D* incluye la etapa convolucional y la de detección; no obstante, esto puede separarse, por lo que normalmente después de utilizar una de estas capas, excepto en el caso de la capa inicial, debe utilizarse otra que realice el proceso de *pooling.* En este TFG, la capa elegida para este proceso ha sido la de *MaxPooling1D*. En esta capa, debe especificarse el *pool size*.

Otra de las capas utilizadas ha sido ***BatchNormalization***. Esta técnica normaliza las salidas de cada capa para cada *mini-batch* del entrenamiento, de forma que su valor medio es cercano a 0 y su desviación estándar cercana a 1. Puede utilizarse antes o después de la función de activación, es decir, antes o después de la etapa de detección (Team keras, 2020b; Vasilev *et ál*., 2019).

Una técnica de regularización común es el ***Dropout*** cuyo esquema se muestra en la Figura 24. Puede aplicarse a la salida de algunas capas de la red. Esta técnica elimina de forma aleatoria y periódica algunas de las conexiones entre elementos/neuronas de distintas capas de la red. Para ello, se define un factor de *dropout* en un rango de 0 a 1 que especifica la probabilidad de que se eliminen las conexiones. Esta regularización se utiliza para asegurar que ninguna neurona confíe demasiado en otras neuronas y aprenda algo útil para la red. Esta técnica puede aplicarse después de capas convolucionales, de *pooling* o capas completamente conectadas (Team keras, 2020d; Vasilev *et ál*., 2019).

La capa ***Flatten*** adapta las dimensiones de una capa a las dimensiones de la siguiente. Normalmente, se utiliza para conectar capas del tipo *Dropout* o *MaxPooling* con la capa de salida o con capas que estén totalmente conectadas (Team keras, 2020e; Vasilev *et ál*., 2019).

Un esquema de la configuración habitual de las distintas capas descritas anteriormente se muestra en la *Figura 25.*

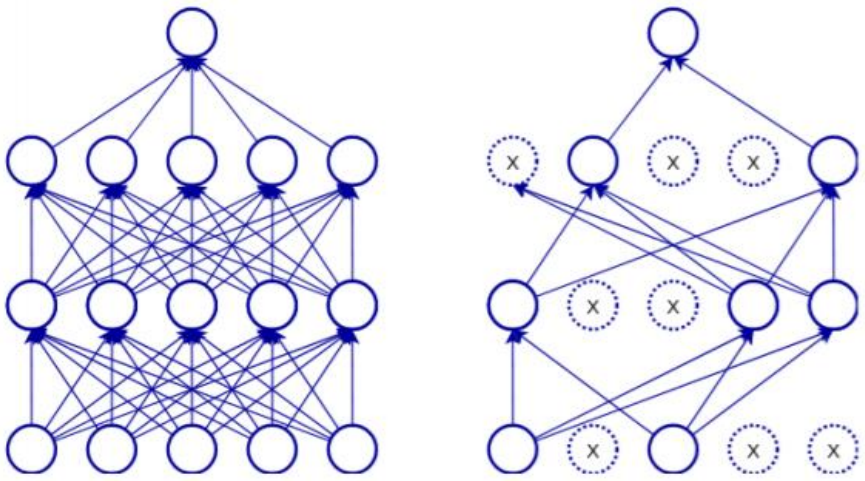

Figura 24: Ejemplo de red neuronal con capas totalmente conectadas que utiliza la técnica de *Dropout*. Figura de (Vasilev *et ál*., 2019).

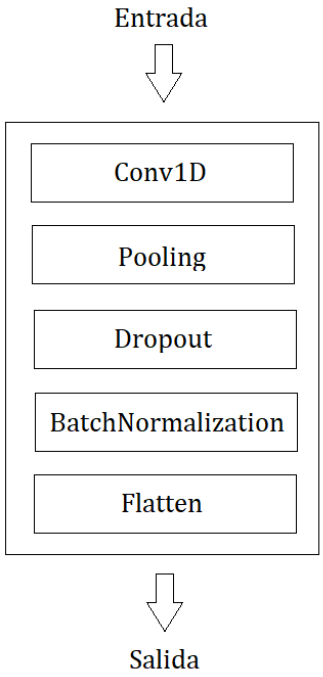

Figura 25: Esquema de la configuración habitual de las capas descritas.

Por último, se utiliza una capa de tipo ***Dense***, es decir, una capa totalmente conectada, como capa de salida (Vasilev *et ál.*, 2019). En esta capa, debe indicarse de forma obligatoria la dimensión del espacio de salida, por ejemplo, el número de clases que se desea clasificar. También debe indicarse la función de activación que se desea utilizar. En las arquitecturas utilizadas en este TFG, las **funciones** de **activación** que mejores resultados han obtenido han sido:

- ***Softmax.*** Esta función de activación convierte los valores de entrada a la capa en probabilidades cuya suma total es uno. Las salidas ofrecidas por una capa *Dense* que utiliza esta función de activación pueden interpretarse como una distribución de probabilidad normalizada de cada clase (Vasilev *et ál.*, 2019). Su representación matemática es la siguiente:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_i}}. \quad (5)$$

- ***Sigmoid.*** Suele utilizarse para casos de clasificación binaria (Vasilev *et ál.*, 2019). Esta función se caracteriza por estar definida entre 0 y 1 como se muestra en la Figura 26. Convierte los valores de entrada en valores entre 0 y 1 de forma que los valores que sean muy altos son convertidos en unos, mientras que los valores muy bajos son convertidos en ceros (Gulli & Pal, 2017; Vasilev *et ál.*, 2019). Su representación matemática es la siguiente:

$$\sigma(x) = \frac{1}{1+e^{-x}}. \quad (6)$$

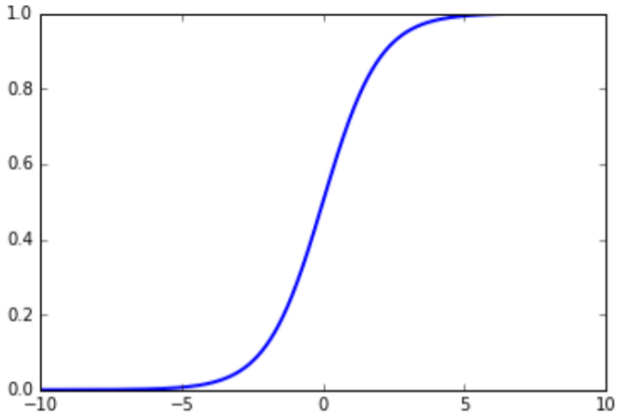

Figura 26: Representación gráfica de la función de activación *Sigmoid*.

4.3.1 Arquitecturas utilizadas

A lo largo de este TFG se han utilizado distintas arquitecturas de modelos de aprendizaje supervisado, pero dos de ellas han sido las que mejores resultados han obtenido. La primera, es una arquitectura que sigue las etapas explicadas para las capas convolucionales y que, al final de estas capas utiliza capas de *Dropout, BatchNormalization* y *Flatten* (ver Figura 26). Como se observa en la Figura 27, se comienza definiendo un modelo secuencial y se añade una capa *Conv1D* inicial indicando los parámetros ya explicados. A continuación, se sigue una estructura del tipo capa convolucional más una capa de *MaxPooling*, tantas *como* se desee y siempre especificando los parámetros deseados. Una vez añadidas las capas con la estructura explicada, se añade primero una capa de *Dropout,* a continuación, una de *BatchNormalization*, seguida por una capa *Flatten* y, por último, se añade la capa de salida de tipo *Dense.*

Los parámetros utilizados a lo largo del modelo han sido elegidos de manera experimental. Se ha probado con distintas funciones de activación, tanto en las capas intermedias como en la de salida. Las funciones de activación que mejores resultados han obtenido han sido: ReLU, para las capas intermedias; y *Sigmoid* y *Softmax*, para la capa de salida. La función de activación *Sigmoid* obtiene buenos resultados en los casos de clasificación binaria, mientras que *Softmax* obtiene mejores resultados para los casos de clasificación múltiple.

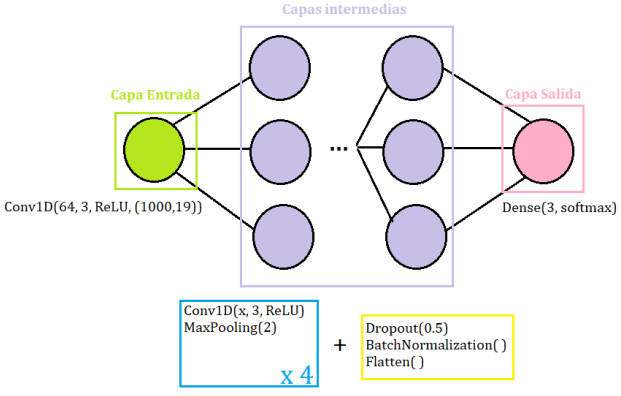

Figura 27: Ejemplo de la primera arquitectura seleccionada.

La segunda arquitectura elegida está formada por bloques de tipo: etapa convolucional, *BatchNormalization,* etapa de activación, etapa de *pooling* y *Dropout* (ver Figura 27). Tal y como ocurre con la arquitectura anterior, se comienza definiendo el modelo secuencial al que se añade la primera capa *Conv1D,* seguida de tantos bloques como se desee. Al finalizar la estructura por bloques se utiliza la capa *Flatten*, para adaptar las dimensiones entre la última capa de *Dropout*, y la siguiente capa que será la capa de salida, de tipo *Dense*.

4.3.2 Algoritmos de optimización, métodos de regularización para la prevención del *overfitting* y funciones de pérdida

A la hora de entrenar un modelo, se utilizan distintos algoritmos tanto de optimación como de regularización. Estos algoritmos ayudan en el aprendizaje de la red y evitan que se produzca *overfitting*, respectivamente. El término *overfitting* hace referencia a realizar predicciones que se ajustan perfectamente a los datos con los que se ha entrenado el modelo, pero que no se generalizan en conjuntos de datos más grandes y distintos a los utilizados en el entrenamiento (Vasilev *et ál.*, 2019). Los dos algoritmos de optimización elegidos han sido *Stochastic Gradient Descent* (SGD) y *Adam*.

El algoritmo **SGD** realiza una iteración con cada muestra de entrenamiento; es decir, calcula la pérdida y actualiza el peso para cada muestra del entrenamiento. Para que los pesos se actualicen con cada muestra del aprendizaje, el parámetro *batch*_size del entrenamiento debe ser 1. Al utilizar este algoritmo de optimización, la curva de pérdida en el entrenamiento será muy ruidosa, debido a que los pesos se actualizan con demasiada frecuencia. Sin embargo, aunque aparezca este ruido, la optimización es relativamente rápida comparada con la que ofrecen otros algoritmos (Moolayil, 2019). Los pesos se actualizan siguiendo la fórmula:

$$pesos = pesos - tasa\ de\ aprendizaje \cdot pérdidas. \tag{7}$$

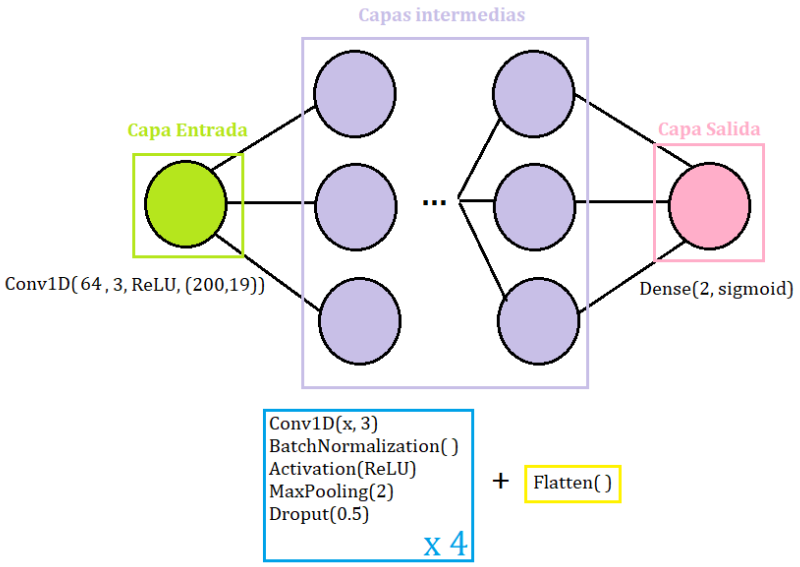

Figura 28: Ejemplo de la arquitectura por bloques.

Si se desea reducir las fluctuaciones en las optimizaciones conseguidas al utilizar este algoritmo, debe crearse un *mini-batch* en el entrenamiento. Esto permitirá promediar la pérdida de todas las muestras de este *mini-batch* y actualizar los pesos al final del mismo. Los *mini-batch* se crean con el parámetro *batch_size* en la función de entrenamiento y suelen ser potencias de dos (Moolayil, 2019).

El algoritmo **Adam** (*Adaptative Moment Estimation)* es el más popular y el más utilizado en DL (Moolayil, 2019). Este algoritmo calcula una tasa de aprendizaje (*learning rate*) adaptativa para cada parámetro. Define el momento y la varianza del gradiente de las pérdidas y aprovecha la combinación de ambas métricas para actualizar los pesos. Esta combinación ayuda a suavizar la curva de aprendizaje y mejora el proceso de aprendizaje de la red (Moolayil, 2019). Para intentar reducir el *overfitting* que presentaban los modelos, se han utilizado tanto el algoritmo de regularización *ReduceLROnPlateau* como la clase *EarlyStopping* de *keras*.

- ***ReduceLROnPlateau*** se utiliza para reducir la *learning rate* cuando una métrica específica no mejora. Para utilizar este algoritmo con *keras*, debe especificarse la métrica a monitorizar para cambiar la *learning rate* cuando ésta no mejore. También debe especificarse el factor por el que desea multiplicarse la *learning rate* y el número de épocas que debe esperar antes de hacer efectivo este cambio (Team keras, 2020g).
- En cuanto al ***EarlyStopping***, se utiliza para parar el entrenamiento cuando una métrica específica no mejora. Tal y como ocurre con el algoritmo de regularización, debe indicarse la métrica a monitorizar y el número de épocas que debe esperar antes de parar el entrenamiento. Esta clase tiene la opción de restaurar los pesos finales por pesos de otras épocas. Al parar el entrenamiento, es posible actualizar los pesos para cambiarlos por los pesos de la época donde la métrica monitorizada obtiene los mejores valores (Team Keras, 2020).

Las funciones de pérdida utilizadas cuando se compila el modelo son: *binary cross-entropy* para la clasificación binaria y *categorical cross-entropy* para el caso de clasificación múltiple. Se han probado otras como el *mean square error* pero se han obtenido peores resultados, ya que esta función de pérdida está diseñada para problemas de regresión. Las funciónes de pérdida ***binary cross-entropy*** y ***categorical cross-entropy*** se utilizan para calcular la entropía cruzada entre las etiquetas reales y las predichas. Sin embargo, como se ha indicado, la primera se utiliza para un caso de clasificación binaria, mientras que la segunda se utiliza en casos de clasificación múltiple (Team keras, 2020a).

Capítulo 5

5.Análisis de los resultados

5.1 Bases de datos utilizadas 58

5.1.1 Base de datos del Hospital Universitario Río Hortega 58

5.1.2 Base de datos POCTEP 59

5.2 Preprocesado de los datos 60

5.2.1 Base de datos del Hospital Universitario Río Hortega 60

5.2.2 Base de datos del proyecto POCTEP 60

5.2.3 Procesado común a las dos bases de datos 61

5.3 Análisis de precisión y análisis de clasificación de sujetos mediante matrices de confusión 62

5.3.1 Resultados obtenidos para el problema de clasificación binaria en la base de datos del Hospital Universitario Río Hortega 63

5.3.1.1 Resultados obtenidos para segmentos de cinco segundos 64

5.3.1.2 Resultados obtenidos para segmentos de un segundo 70

5.3.1.3 Resultados obtenidos para la aplicación de transfer learning 75

5.3.2 Resultados obtenidos para el problema de clasificación binaria en la base de datos del proyecto POCTEP 75

5.3.2.1 Resultados obtenidos para segmentos de cinco segundos 75

5.3.2.2 Resultados obtenidos para segmentos de un segundo 81

5.3.2.3 Resultados obtenidos para la aplicación de transfer learning 85

5.3.3 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del Hospital Universitario Río Hortega 86

5.3.3.1 Resultados obtenidos para segmentos de cinco segundos 86

5.3.3.2 Resultados obtenidos para segmentos de un segundo 93

5.3.3.3 Resultados obtenidos para la aplicación de transfer learning 99

5.3.4 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del proyecto POCTEP 100

5.3.4.1 Resultados obtenidos para segmentos de cinco segundos 100

5.3.4.2 Resultados obtenidos para segmentos de un segundo 105

5.3.4.3 Resultados obtenidos para la aplicación de transfer learning 109

5.1 Bases de datos utilizadas

5.1.1 Base de datos del Hospital Universitario Río Hortega

En la realización de este TFG se han utilizado dos bases de datos distintas. La base de datos proveniente del Hospital Universitario Río Hortega de Valladolid (HURH) y la base de datos recogida gracias al proyecto europeo 'Análisis y correlación entre el genoma completo y la actividad cerebral para la ayuda en el diagnóstico de la enfermedad de Alzheimer' ('Interreg V-A Spain-Portugal, POCTEP 2014-2020'). Las dos bases de datos incluyen sujetos diferentes, sin ninguna relación entre ellos. Sin embargo, comparten los grupos incluidos y las variables recogidas. Todas las personas incluidas dieron su consentimiento expreso para participar en el estudio. En el caso de los pacientes incapacitados, sus familiares fueron los responsables de consentir la participación en dicho estudio. El uso de los datos se ha realizado cumpliendo el Código Ético de la Asociación Médica Mundial (Declaración de Helsinki) y de acuerdo con los protocolos aprobados por los Comités Éticos del Hospital Universitario Río Hortega de Valladolid y de la Universidad de Oporto en Oporto, Portugal. **Base de datos del Hospital Universitario Río Hortega**

Esta base de datos está formada por registros EEG, datos sociodemográficos y variables clínicas recogidas en su totalidad en el HURH. Está formada por un total de 196 sujetos: 45 sujetos de control, 69 pacientes diagnosticados con DCL y 82 enfermos diagnosticados con demencia por EA. La clasificación de estos sujetos en su grupo correspondiente se realizó siguiendo los criterios NIA-AA (Jack *et ál.*, 2018). Para evaluar las capacidades cognitivas de los sujetos se utilizó el MMSE. La Tabla 2 muestra los datos sociodemográficos y clínicos más importantes de los sujetos que forman parte de esta base de datos.

Grupo	*N*	Edad (media ±SD)	Género (Mujer:Hombre)	MMSE (media ± SD)
Control	45	76,29 ± 3,97	31:14	28,82 ± 1,15
DCL	69	76,44 ± 6,45	40:29	29,96 ± 1,87
EA	82	80,11 ± 5,61	48:34	20,68 ± 3,89

Tabla 2: Datos sociodemográficos y clínicos de la base de datos del HURH. *N* indica el número de sujetos y SD la desviación estándar.

Los registros fueron adquiridos mediante el electroencefalógrafo XLTEK® de *Natus Medical*, que consta de 19 canales. La configuración de los electrodos es la correspondiente al Sistema Internacional 10-20, con referencia común (Maturana-Candelas *et ál.*, 2019), utilizando los electrodos: F3, F4, F7, F8, Fp1, Fp2, T3, T4, T5, T6, C3, C4, P3, P4, O1, O2, Fz, Cz, y Pz. La frecuencia de muestreo utilizada es de 200 Hz. Todos los registros fueron obtenidos de sujetos sentados en reposo y con los ojos cerrados. Los registros EEG contienen 5 min. de actividad cerebral espontánea.

Esta señales obtenidas fueron procesadas tal y como se indica a continuación (Maturana-Candelas *et ál*., 2019):

- Aplicación de un filtro digital paso banda de respuesta finita con una ventana de *Hamming* en el rango frecuencial de 1 a 70 Hz. Esto permite acotar la señal al intervalo de frecuencias de interés.
- Aplicación de un filtrado *notch* para eliminar el ruido introducido por la red eléctrica que produce en 50 Hz.
- Análisis de componentes indepndientes (*Independent Component Analysis*,ICA) para eliminar artefactos relacionados con ruido introducido por los músculos, el corazón o los ojos.
- Selección de épocas de 5 s libres de artefactos a través de una inspección visual.

5.1.2 Base de datos POCTEP

Esta base de datos está formada por registros EEG, datos sociodemográficos y variables clínicas recolectadas gracias a un proyecto europeo desarrollado por el Grupo de Ingeniería Biomédica de la Universidad de Valladolid. La base de datos está formada por un total de 253 sujetos: 51 sujetos de control, 51 sujetos diagnosticados con DCL y 151 sujetos diagnosticados con EA. La clasificación de estos sujetos en su grupo correspondiente se realizó siguiendo los criterios NIA-AA (Jack *et ál*., 2018). Para evaluar las capacidades cognitivas de los sujetos se ha utilizado el MMSE. La Tabla 3 muestra los datos sociodemográficos y clínicos de los sujetos que forman parte de esta base de datos.

Grupo	*N*	Edad (media ±SD)	Género (Mujer:Hombre)	MMSE (media ± SD)
Control	51	80,14 ± 7,09	25:26	28,82 ± 1,13
DCL	51	76,44 ± 6,45	36:15	23,33 ± 2,84
EA Leve	51	80,69 ± 7,05	30:21	22,49 ± 2,27
EA moderado	50	81,30 ± 8,04	43:7	13,60 ± 2,76
EA severo	50	79,98 ± 7,82	43:7	2,42 ± 3,70

Tabla 3: Datos sociodemográficos y clínicos de la base de datos del proyecto POCTEP. *N* indica el número de sujetos y SD, la desviación estándar.

Los registros fueron adquiridos mediante el sistema de EEG *Nihon Kohden Neurofax JE-921ª*, que consta de 19 canales. La configuración de los electrodos es la correspondiente al Sistema Internacional 10-20, con referencia común (Maturana-Candelas *et ál*., 2019), utilizando los electrodos: F3, F4, F7, F8, Fp1, Fp2, T3, T4, T5, T6, C3, C4, P3, P4, O1, O2, Fz, Cz, y Pz. La frecuencia de muestreo utilizada es de 500 Hz. Todos los registros fueron obtenidos de sujetos sentados en reposo y con los ojos cerrados en un entorno libre de ruidos para minimizar la presencia de artefactos. Los registros EEG contienen 5 minutos de actividad cerebral. Esta señales

obtenidas fueron procesadas tal y como se indica a continuación (Maturana-Candelas *et ál*., 2019):

- Aplicación de un filtro digital paso banda de respuesta finita con una ventana de *Hamming* en el rango frecuencial de 1 a 70 Hz. Esto permite acotar la señal al intervalo de frecuencias de interés.
- Aplicación de un filtrado *notch* para eliminar el ruido introducido por la red eléctrica que produce en 50 Hz.
- Aplicación de ICA para eliminar artefactos relacionados con ruido introducido por los músculos, el corazón o los ojos.
- Selección de épocas de 5 s libres de artefactos a través de una inspección visual.

5.2 Preprocesado de los datos

El preprocesado de los datos es necesario para preparar y adaptar los datos al formato utilizado en la entrada del modelo. Además del procesado descrito para cada base de datos, ha sido necesario aplicar otras técnicas de preprocesado como la codificación de etiquetas o la aplicación de técnicas de diezmado e interpolación. Las técnicas utilizadas han sido aplicadas a ambas bases de datos, adaptando siempre las dimensiones y frecuencias de muestreo a cada caso. Por esto, se explicará por una parte el procesamiento aplicable a ambas bases de datos y el específico para cada una.

5.2.1 Base de datos del Hospital Universitario Río Hortega

Esta base de datos ha sido obtenida utilizando una frecuencia de muestreo de 200 Hz. Se han realizado dos casos de estudio: el primero utilizando segmentos de 5 s y el segundo utilizando segmentos de 1 s. Para el primer caso, se han seleccionado segmentos de 1000 muestras y se han eliminado los artefactos de manera que finalmente se obtienen 8933 segmentos de 1000 muestras en 19 canales; esto se traduce en un vector con las siguientes dimensiones: (8933, 1000, 19). Para el segundo caso, se han seleccionado segmentos de 200 muestras y se han eliminado los artefactos de manera que se obtienen 44665 segmentos de 200 muestras en 19 canales; esto da lugar a un vector con dimensiones: (44665, 200, 19). Este caso se ha desarrollado para intentar mejorar los resultados, ya que los modelos entrenados con esta base de datos presentan *overfitting* y una de las formas de resolver este problema es aumentar el número de datos con los que se entrena el modelo.

Se han probado modelos entrenados con la base de datos del proyecto POCTEP sobre las señales EEG disponibles en la base de datos del HURH . Para ello, se han adaptado los segmentos de esta base de datos al formato utilizado en la entrada del modelo. Para realizar esta adaptación ha sido necesario realizar una interpolación, ya que los segmentos utilizados en la base de datos del proyecto POCTEP tendrán unas dimensiones de (x, 2500/500, 19) como se explica en el siguiente apartado.

5.2.2 Base de datos del proyecto POCTEP

Esta base de datos ha sido obtenida utilizando una frecuencia de muestreo de 500 Hz. Tal y como ocurre con la base de datos del HURH, se han realizado dos casos de estudio: el primero utilizando segmentos de 5 s y el segundo utilizando segmentos de 1 s para intentar evitar la aparición de *overfitting*. Para el primer caso, se han seleccionado segmentos de 2500 muestras y se han eliminado los artefactos de manera que finalmente se obtienen 9852 segmentos de 2500 muestras en 19 canales; esto se traduce en un vector con las siguientes dimensiones: (9852, 2500, 19). Para el segundo caso, se han seleccionado segmentos de 500 muestras y se han eliminado los artefactos de manera que se obtienen 49260 segmentos de 500 muestras en 19 canales; lo cual da lugar a un vector con dimensiones: (49260, 500, 19). En este caso, también se han evaluado modelos entrenados con la base de datos del HURH en segmentos de EEG disponibles en esta base de datos. Para ello, se ha realizado un diezmado para adaptar los datos de la base de datos al formato de datos de la entrada al modelo, que en este caso es de (x, 1000/200, 19).

5.2.3 Procesado común a las dos bases de datos

Tras obtener los segmentos libres de artefactos, el paso siguiente es la creación de un vector que contiene el número de segmentos asociado a cada sujeto. Esto es fundamental, ya que, gracias a un mapeo entre este vector y el vector que contiene los sujetos de test, puede obtenerse una clasificación por sujeto. También debe obtenerse el grupo al que pertenece cada uno de los sujetos. Se crea un vector que contiene el grupo al que pertenece cada sujeto y cada segmento. Esto es necesario para poder crear las etiquetas que necesita el modelo, ya que aprenderá utilizando el paradigma de aprendizaje supervisado. A continuación, se ha procedido a la codificación de estas etiquetas. Esta codificación ha sido realizada utilizando la función *LabelEncoder* de la librería *Scikit-learn.* Esta función permite codificar las etiquetas con valores entre cero y el número de clases existentes menos uno (Team Scikit-learn, 2020). Para realizar la codificación, primero se define el *LabelEncoder*. Una vez definido, se utiliza la función *fit* para adaptar este codificador a los datos que deben codificarse y, por último, se realiza la transformación a de los datos números en el rango explicado utilizando la función *transform.* Un ejemplo de codificación, en un caso de cinco clases (Control, DCL, DCL1, DCL2 y EA), es el presentado en la Figura 29, donde la variable *grupo_sujetos* representa el grupo al que pertenece cada segmento obtenido y *encoded_grupo_sujetos*, un vector con la codificación de cada segmento.

Para el modelo desarrollado esto no es suficiente, por lo que para finalizar la codificación debe utilizarse una función de *keras* llamada np_*utils* y el método to_*categorical* (Team keras, 2020f). Esta función convierte un vector de números enteros, siguiendo el ejemplo sería el vector *encoded_grupo_sujetos*, en una matriz binaria con dimensiones: (número segmentos, número clases). Con los datos del ejemplo expuesto en la Figura 30, se obtendría lo representado en la Figura 30, una matriz de 8933 segmentos por 5 clases. La posición en la que toma un valor 1 corresponde a la clase que tiene asignada ese segmento.

```
grupo_sujetos: ['DCL1', 'DCL1', 'DCL1', 'DCL1', 'EA', 'EA', 'EA', 'EA']
encoded_grupo_sujetos: [2 2 2 2 4 4 4 4]
Posibles valores de las etiquetas: [0 1 2 3 4]
```

Figura 29: Ejemplo de codificación utilizando *LabelEncoder*.

```
Dimensiones: (8933, 5)
[[0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 1. 0. 0.]
 ...
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]
 [1. 0. 0. 0. 0.]]
```

Figura 30: Aplicación de la función *np_utils* de la librería *keras* para finalizar la codificación de las etiquetas.

Una vez realizada la codificación, se dividen los segmentos obtenidos en conjuntos de entrenamiento, validación y test. En este caso, se dividen estos datos en un 70% para entrenamiento y un 30% para test. Del 70% utilizado para entrenamiento, se obtiene un porcentaje del 10% para validación. Por último, una vez realizado el procesado de los datos, éstos ya pueden introducirse al modelo. El 70% de los datos irá destinado al entrenamiento del modelo y el 30%, al test para comprobar el funcionamiento.

Para intentar mejorar los resultados de los distintos modelos, en algunos casos se ha probado a añadir otras dos técnicas de preprocesado: referenciado promedio o *Common Average Reference* (CAR) y cálculo del *z-score*. La primera técnica se lleva a cabo antes de realizar la limpieza de artefactos y elimina la media de las señales EEG de todos los electrodos en cada electrodo (Alhaddad, 2012). La segunda se emplea, tras la limpieza de artefactos, para estandarizar los datos por segmento. Para ello, se utiliza la función *z-score* de la librería *Scipy* (Team Scipy, 2020).

5.3 Análisis de precisión y análisis de clasificación de sujetos mediante matrices de confusión

A lo largo de este TFG se han realizado distintas pruebas sobre diferentes casos de estudio. Se han utilizado las dos bases de datos explicadas y para cada una de ellas, se han estudiado distintos escenarios. Se ha comenzado realizando pruebas para un caso de **clasificación binaria**, donde se clasificaban controles y enfermos con EA. Para ellos, en ambas todos los pacientes clasificados como DCL han sido etiquetados como EA mientras que los controles se han mantenido.

Tras realizar las pruebas y crear los modelos para el problema de clasificación binaria, se ha pasado a un problema de **clasificación múltiple** con **tres clases**: controles, pacientes con DCL y enfermos con EA. Por último, se ha realizado alguna prueba para un problema de clasificación múltiple con cinco clases en la base de datos POCTEP, subdividiendo los enfermos de Alzheimer en función del grado de severidad: leve, moderada o severa. En este caso, las pruebas realizadas presentaban una precisión muy baja, menor al 30%; por lo que se descartó continuar haciendo pruebas para este problema.

Para todos los escenarios descritos se han desarrollado las siguientes pruebas:

- Modelos para segmentos de 5 s.
- Modelos para segmentos de 5 s utilizando CAR y *z-score*.
- Modelos para segmentos de 1 s.
- Modelos para segmentos de 1 s utilizando CAR y *z-score*.
- Prueba de modelos entrenados en una base de datos para el problema de clasificación sobre los datos existentes en la otra base de datos, *transfer learning*.

Para realizar un análisis de clasificación mediante matrices de confusión, se expondrán las distintas matrices de confusión obtenidas para todos los casos de estudio y los modelos que no aplican técnicas de preprocesado CAR y *z-score* explicados en el apartado anterior. Los modelos obtienen una matriz de confusión donde se muestra la clasificación por segmentos, ya que lo que se introduce al modelo son segmentos de los registros EEG, de una duración de 1 o 5 s. Para conseguir la clasificación por sujetos, es necesario crear un vector que contenga el número de segmentos asociado a cada sujeto. Esto es fundamental, ya que gracias a un mapeo entre este vector y el vector que contiene los sujetos de test puede obtenerse dicha clasificación.

Se han probado dos opciones diferentes a la hora de realizar la clasificación por sujetos. La primera ha sido realizar el mapeo de los dos vectores explicados y contar, para cada usuario, cuántos segmentos han sido clasificados en cada clase. La clase que mayor número de segmentos obtiene es la clase en la que se clasifica al sujeto. La segunda opción ha sido realizar el mismo mapeo empleado en la primera opción y, a continuación, calcular la media de las probabilidades que predice el modelo para cada segmento. La clase que mayor media de probabilidad tenga, será la clase donde se clasifica al sujeto. En ambas opciones, en caso de empate entre las distintas clases, se ha decidido clasificar al sujeto como enfermos con EA.

En este análisis se expondrán únicamente los resultados de clasificación por sujeto y se omitirán las matrices de confusión de clasificación de segmentos. Esto se debe a que es más sencillo y directo entender los resultados de clasificación sobre los distintos sujetos.

Primero se expondrán los resultados de clasificación obtenidos para el problema de clasificación binaria sobre la base de datos del HURH utilizando tanto segmentos de 5 s como segmentos de 1 s. A continuación, se muestran estos mismos resultados para la base de datos del proyecto POCTEP. Tras realizar el análisis del problema de clasificación binaria, se pasa al problema de clasificación múltiple. Se expondrán primero las matrices de confusión obtenidas al aplicar los distintos modelos sobre la base de datos del HURH y, por último, se expondrán los resultados para este problema sobre la base de datos del proyecto POCTEP.

5.3.1 Resultados obtenidos para el problema de clasificación binaria en la base de datos del Hospital Universitario Río Hortega

El problema de clasificación binaria es el que mejores resultados ha obtenido en este TFG. Obtiene resultados de aproximadamente el 70% de precisión para los datos de test en todos los casos estudiados, a excepción de los casos que utilizan CAR y *z-score* donde esta métrica desciende hasta aproximadamente un 60%. En cuanto al uso de *transfer learning*, si no se utiliza CAR ni *z-score*, la precisión del modelo se mantiene. Sin embargo, si se utiliza CAR y *z-score*, la precisión del modelo disminuye de manera notable.

5.3.1.1. Resultados obtenidos para segmentos de cinco segundos

Los modelos utilizados en el caso de segmentos de 5 s serán los mismos que los utilizados en el caso de segmentos de 5 s aplicando las técnicas de preprocesado CAR y *z-score.* Las Figuras 31-34 muestran la estructura utilizada para la creación de los modelos que mejores resultados han obtenido para el problema binario en los casos de segmentos de 5 s. En los modelos donde el número de filtros de las capas de tipo *Conv1D* no sea el mismo en cada bloque, aparecerá una *x*.

Todos los modelos presentan unas dimensiones de entrada de (1000,19) y utilizan como función de activación la función *sigmoid*, ya que es un caso binario. Ocurre lo mismo con la función de pérdida, se utiliza *binary_crossentroy*. También se emplea tanto *earlyStopping* como *ReduceLROnPlateau* para reducir el *overfitting.* La diferencia principal será la estructura de cada modelo y los optimizadores utilizados: SGD o Adam.

El modelo 3, representado en la Figura 31, presenta la siguiente estructura. Comienza con una de capa de entrada de tipo *Conv1D* con dimensiones (1000,19), 64 filtros y un tamaño de *kernel* igual a 3. En este modelo, el tamaño del *kernel* se mantendrá para todas las capas de tipo *Conv1D.* Las capas intermedias siguen una estructura de dos capas convolucionales de tipo *Conv1D*, seguidas de una capa *MaxPooling* a excepción de la última etapa del modelo, que estará formada por dos capas *Conv1D* seguidas de una capa de *Dropout* con factor 0.5, *BatchNormalization, MaxPooling* con *pool_size=2* y *Flatten*. El modelo estará formado por un total de 2 bloques con la estructura de dos capas *Conv1D* y una de *MaxPooling*, a los que se añadirá la última etapa formada por las capas explicadas en este punto. Finalmente, la capa de salida será una capa tipo *Dense* con dos unidades, una por cada clase (controles y EA), que emplea la función de activación *sigmoid*. Tanto la capa inicial como las capas intermedias emplean la función de activación ReLU. El número de filtros empleados por cada capa *Conv1D* va desde 64 en la primera hasta 750 en la última. En cuanto a la parte del entrenamiento del modelo, se utilizan los dos algoritmos de regularización ya mencionados, la función de pérdida *binary_crossentropy* y el optimizador Adam, con una *learning rate* o tasa de aprendizaje de 0.001. Se define un número de épocas de entrenamiento igual 120 pero el entrenamiento termina en la época 67 debido al uso de *earlyStopping*.

El modelo 5, indicado en la Figura 32, sigue una estructura de tipo capa *Conv1D* de entrada con dimensiones (1000,19) y 64 filtros. Capas intermedias del tipo capa *Conv1D* más capa *MaxPooling* con *pool_size=2 y,* finalmente, una capa de salida, de tipo *Dense* con tantas unidades como clases a clasificar, en este caso, dos. Todas las

capas del tipo *Conv1D* utilizan la función de activación ReLU y un tamaño de *kernel* igual a 3, mientras que la capa de salida utiliza la función de activación *sigmoid*. Como ya se ha mencionado, la parte de entrenamiento es común para todos los modelos, cambiando algunos parámetros como el optimizador o el número de épocas. En este caso, el entrenamiento se realiza en 116 épocas, aunque se hayan definido 120, esto se debe al uso de *earlyStopping.* En cuanto al optimizador utilizado, se ha elegido SGD con un *learning rate* igual 0.01.

El modelo 6, cuya arquitectura se representa en la Figura 33, es similar al 3 en cuanto a la estructura utilizada en las capas intermedias. La primera capa es una capa de entrada de tipo *Conv1D* con 64 filtros, un tamaño de *kernel* igual a 3, función de activación ReLU y unas dimensiones (1000,19). A continuación, se añaden las capas intermedias, cuya estructura es del mismo tipo que la del modelo 3: dos capas *Conv1D* con función de activación ReLU seguidas de una capa de *MaxPooling* con *pool_size=2*. En este caso, cambia la estructura de los dos últimos bloques de las capas intermedias. El primer bloque tendrá una capa *Conv1D* y una de *MaxPolling* mientras que el segundo bloque estará formado por dos capas *Conv1D* seguidas de las capas *Dropout*, *BatchNormalization, MaxPooling* y *Flatten*. Por último, se añade una capa de salida de tipo Dense con una función de activación *sigmoid*. En el entrenamiento del modelo se ha fijado el número de épocas a 200, parando en la época 120 gracias al uso de *earlyStopping*. El optimizador utilizado en este modelo será SGD con un *learning rate* de 0.01.

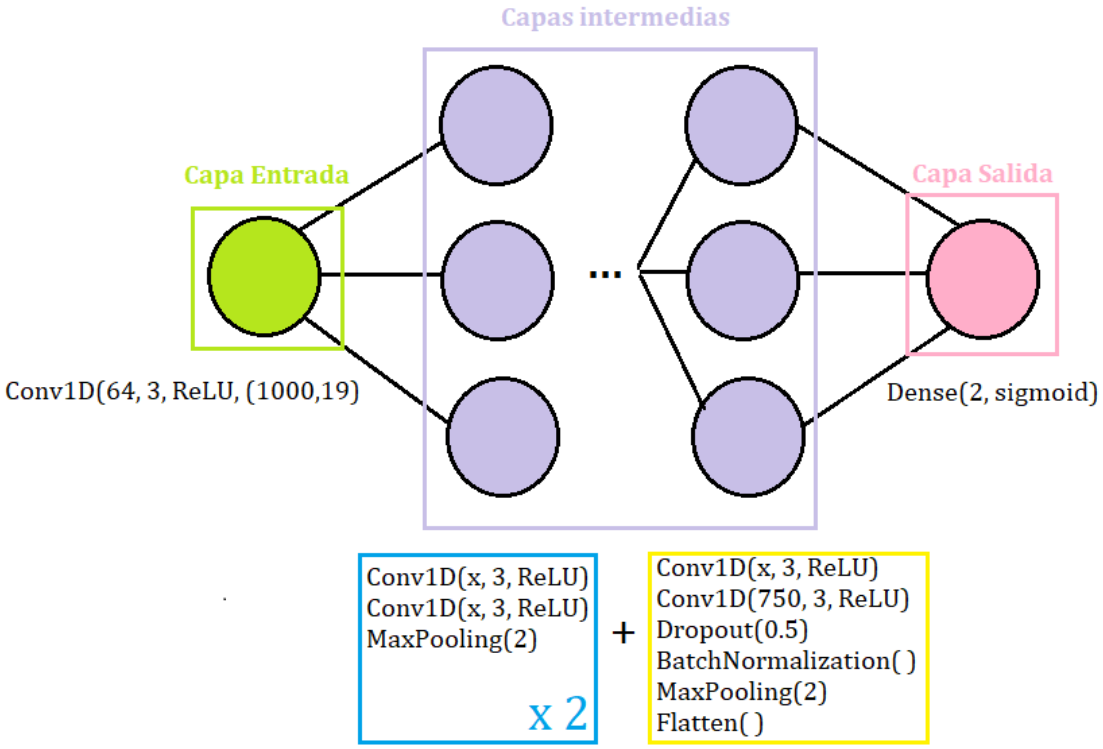

Figura 31: Arquitectura del modelo 3 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH.

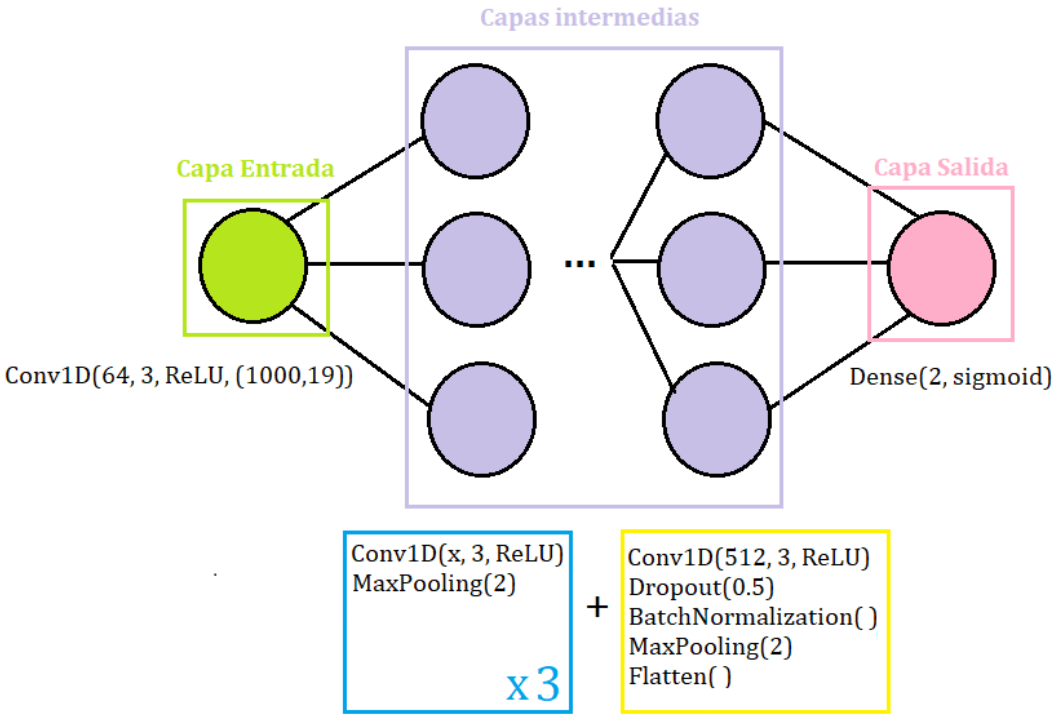

Figura 32: Arquitectura del modelo 5 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH.

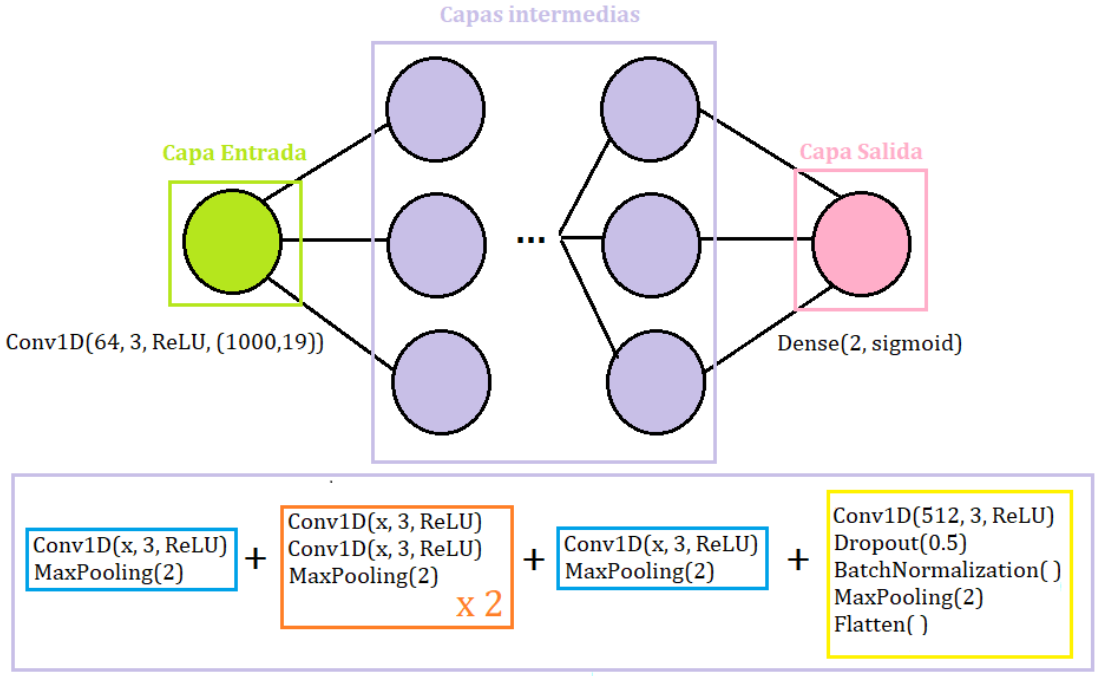

Figura 33: Arquitectura del modelo 6 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH.

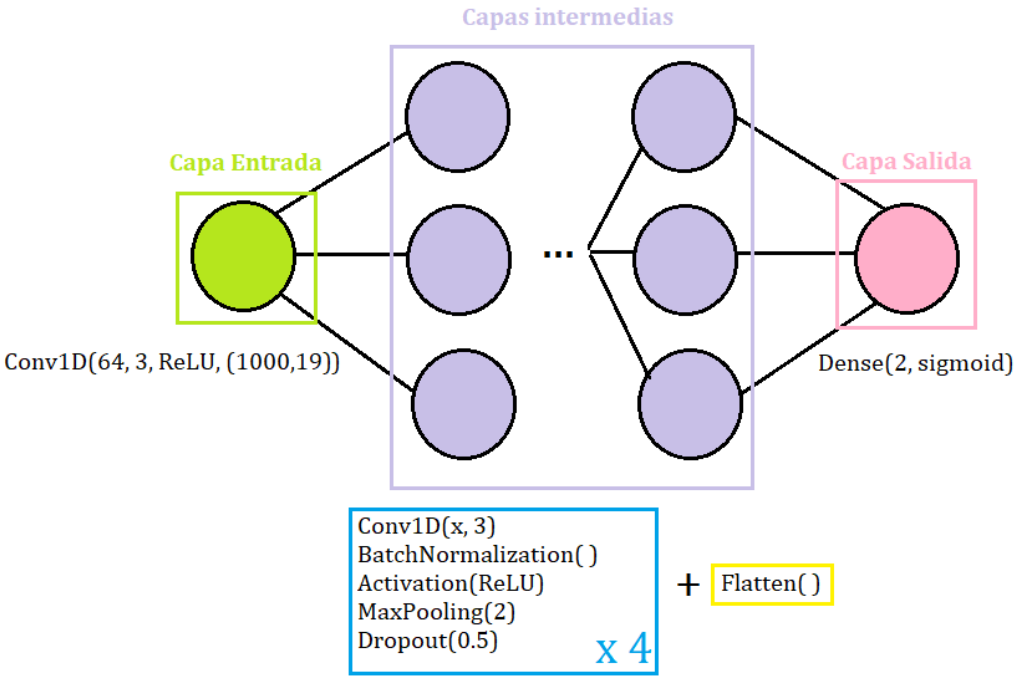

Figura 34: Arquitectura del modelo 7 utilizado en el problema de clasificación binaria para los casos de segmentos de 5 s en la base de datos del HURH.

Por último, el modelo 7, representado en la Figura 34, emplea la estructura de bloques explicada en ***4.3.1 Arquitecturas utilizadas***. Comienza como el resto de modelos, con una capa *Conv1D* que será la capa de entrada. A continuación, se añaden capas con estructura *Conv1D* activada con una ReLU, *BatchNormalization*, *MaxPooling* con *pool_size = 2*, y por último *Dropout* con un factor de 0.5. En el último de estos bloques se utilizará una capa de tipo *Flatten* para adaptar las dimensiones a la capa de salida de tipo *Dense*. En este caso se ha utilizado el optimizador SGD con una *learning rate* de 0.01 y, aunque el número de épocas estaba fijado en 120, ha parado en la época 70 debido al uso de *earlyStopping*.

La Tabla 4 muestra un resumen de los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa*.

	Entrenamiento	Validación	Test	Control	EA	kappa
Modelo 3	100	97,02	68,57	10	94,74	0,02
Modelo 5	100	99,99	63,14	12	96,96	0,11
Modelo 6	99,94	95,24	76,90	9,09	96,62	0,13
Modelo 7	99,76	98,75	61,27	84,61	100	0,13

Tabla 4: Resumen de resultados de precisión en tanto por ciento (%) y valor de kappa para modelos entrenados con segmentos de 5 s en la base de datos del HURH.

La Figura 35 presenta el porcentaje de precisión conseguido en los distintos modelos para los conjuntos de entrenamiento, test y validación. En color rosa, se incluye el resultado obtenido por estos modelos sobre el conjunto de test aplicando las técnicas CAR y *z-score.* Se han ordenado los modelos de manera ascendente según la precisión obtenida en el conjunto de test. Analizando esta figura, se observa que las diferencias existentes entre los conjuntos de validación y entrenamiento son pequeñas. Esto indica que el comportamiento del modelo es bueno, es decir, que está aprendiendo correctamente. También se observa que existe una diferencia considerable entre el comportamiento del modelo sobre los datos de validación y los de test. Esto puede ser el resultado de utilizar una división aleatoria a la hora de elegir los conjuntos de entrenamiento y validación. Al utilizar este tipo de división, se utilizan segmentos distintos de los mismos sujetos en la parte de entrenamiento y validación. Sin embargo, en la parte de test, se utilizan todos los segmentos de cada sujeto, es decir, no se utilizan segmentos distintos de los mismos sujetos. En cuanto al uso de las técnicas CAR y *z-score*, mejora ligeramente el comportamiento de algunos modelos mientras que empeora el de otros, como el Modelo 6. La máxima precisión es la alcanzada por el modelo 6, un 76,90% mientras la mínima será la conseguida por el modelo 7, de un 61,27%.

Las tablas 5-8 muestran las matrices de confusión obtenidas por los modelos desarrollados. Habrá un total de 58 sujetos: 4 sujetos de control y 54 enfermos.

La matriz de confusión obtenida por el modelo 7 se muestra en la . Es el resultado de un modelo con una precisión del 61,27%. Se consigue clasificar a todos los sujetos de control de forma correcta, pero 22 enfermos se clasifican de forma incorrecta confundiéndose con sujetos de control. En este caso, la clase que mejores resultados tiene es la de los controles, con el 100% (4/4) de los sujetos clasificados de manera correcta y la que peor la de los enfermos con un 59,25% (32/54) de los sujetos clasificados correctamente.

Real \ **Estimado**	**Control**	**Enfermo**
Control	4	0
Enfermo	22	32

Tabla 5: Matriz de confusión de clasificación de sujetos obtenida por el modelo 7 entrenado con segmentos de 5 s en la base de datos del HURH.

La Tabla 6 muestra la matriz de confusión con los resultados de clasificación por sujeto obtenidos por el modelo 5. Este modelo presenta una precisión del 63,15%. En este caso, los enfermos obtienen la misma clasificación que el modelo anterior, un 59,25% (32/54) de los sujetos pertenecientes a esta clase se clasifican correctamente. Sin embargo, en este caso, un sujeto de control se clasifica erróneamente como un enfermo pasando del 100% obtenido en el modelo anterior, al 75% (3/4) de sujetos clasificados de manera correcta.

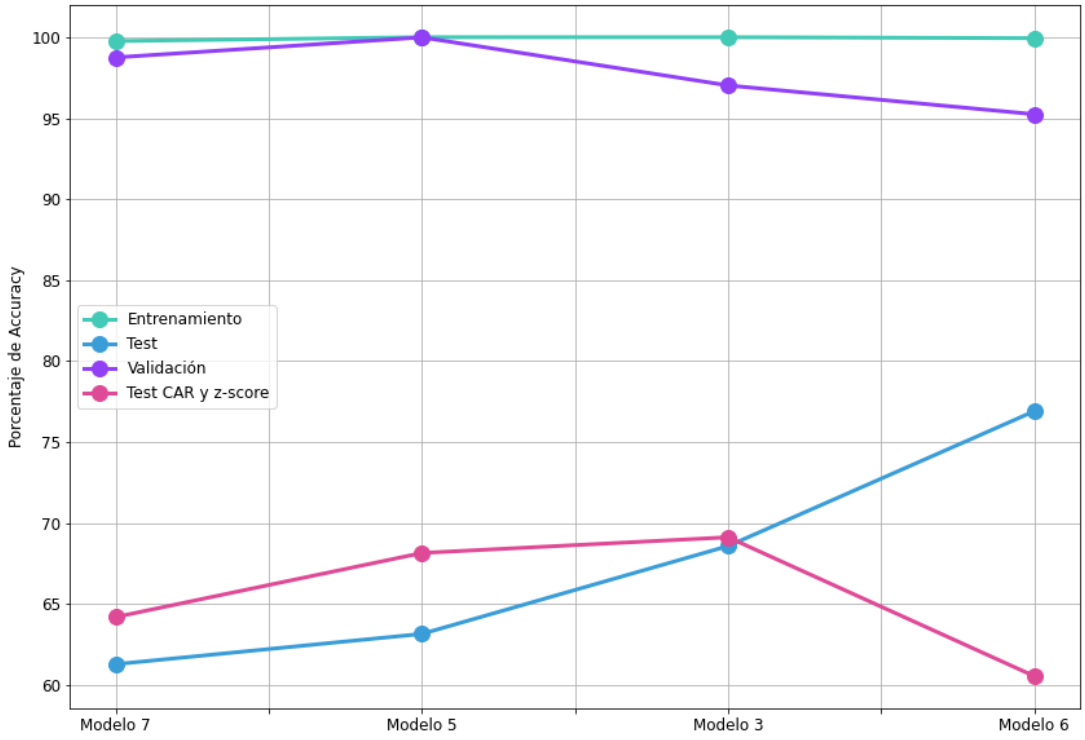

Figura 35: Porcentaje de *precisión* obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y *z-score* para el problema de segmentos de 5 s en la base de datos del HURH.

Real \ Estimado	Control	Enfermo
Control	3	1
Enfermo	22	32

Tabla 6: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s en la base de datos del HURH.

En el caso del modelo 3 (ver Tabla 7), la clasificación de enfermos aumenta en dos sujetos; es decir, se clasifican correctamente 36 en lugar de los 32 clasificados por los modelos anteriores, obteniéndose un 66,66% (36/54) de sujetos clasificados correctamente. El porcentaje de sujetos de control clasificados de manera correcta sigue descendiendo hasta llegar al 50% (2/2).

Real \ Estimado	Control	Enfermos
Control	2	2
Enfermos	18	36

Tabla 7: Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 5 s en la base de datos del HURH.

Por último, el modelo 6, que presenta un precisión del 76,90%, obtiene la matriz de confusión mostrada en la Tabla 8. Este es el modelo que mejores resultados de precisión obtiene por lo que debería ser también el que mejor matriz de confusión presente. La clasificación correcta de enfermos aumenta hasta llegar a los 44 sujetos, un 81,48%, frente los 32 y 36 conseguidos por el resto de modelos. En cuanto a los sujetos de control, es capaz de clasificar de manera correcta un único sujeto por lo que la tasa de sujetos clasificados correctamente desciende hasta alcanzar el 25%.

Real \ Estimado	Control	Enfermos
Control	1	3
Enfermos	10	44

Tabla 8: Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 5 s en la base de datos del HURH.

En líneas generales, se observa que, tal y como ocurre con el caso de segmentos de cinco segundos, la precisión obtenida por la clase enfermos es mucho mayor a la obtenida por la clase control. En cuanto a los coeficientes *kappa*, la mayoría de los modelos obtienen un valor cercano al 0.10 siendo el más elevado un valor de 0.13 perteneciente a los modelos 6 y 7.

5.3.1.2. Resultados obtenidos para segmentos de un segundo

Los modelos utilizados en el caso de segmentos de 1 s serán los mismos que los utilizados en el caso de segmentos de un segundo aplicando las técnicas CAR y *z-score.* Las Figura 36*,* Figura 37 y Figura 38, muestran la estructura de los cuatro modelos que mejores resultados de precisión han obtenido para segmentos de 1 s. Como ocurre en el caso de segmentos de 5 s, al tratarse de un problema de clasificación binaria, todos los modelos tendrán en común el uso de la función de activación *sigmoid* en la capa de salida, y la función de pérdida *binary_crossentropy* para el entrenamiento. También se utilizarán algoritmos de regularización como *ReduceLROnPlateau* y *earlyStopping* para reducir el *overfitting*. Las principales diferencias serán las mismas que en el caso anterior, el algoritmo de optimización, las épocas utilizadas en el entrenamiento del modelo, el valor de la *learning rate* o la estructura seguida por las capas intermedias.

En este caso, los modelos 3 y el 4 tienen la misma estructura y los mismos valores, lo único que cambia es el optimizar utilizado: en el modelo 3 se usa Adam con una *learning rate* de 0.001 y en el modelo 4 se emplea SGD con una *learning rate* de 0.01. Ambos siguen la estructura seguida por el modelo 3 del caso anterior, como se observa en la Figura 36: capa de entrada, dos capas *Conv1D* seguidas de *MaxPooling* a excepción de la última etapa en la que se utilizarán, además de *MaxPooling*, capas de *Dropout, BatchNormalization* y *Flatten.* En ambos casos, el tamaño del *kernel* de las capas *Conv1D* es de 3 o 6. Ambos se han entrenado fijando el número de épocas de entrenamiento a 120 pero el modelo 3 para en la época 50 y el 4, en la 39 debido al uso de *earlyStopping*. Los modelos 5 y 6 también presentan la misma arquitectura por bloques explicada anteriormente: capa *Conv1D* con

tamaño de kernel 3, capa *BatchaNormalization*, capa *Activation* con la función ReLU, capa *MaxPooling* con *pool_size=2* y capa *Dropout* con factor 0.5.

La diferencia principal entre estos dos modelos es el número de filtros de cada capa; en el modelo 6, se utiliza el mismo número de filtros para todas las capas intermedias mientras que en el modelo 5, el número de filtros es distinto en cada capa. Ambos utilizan SGD con una *learning rate* de 0.01 y 120 épocas para el entrenamiento, pero debido al uso de *earlyStopping,* el modelo 5 para en la época 56 y el 6, en la 57.

La Tabla 9 muestra un resumen de los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa*.

	Entrenamiento	**Validación**	**Test**	**Control**	**EA**	**kappa**
Modelo 3	99,53	93,08	70,14	47,62	86,49	0,19
Modelo 4	99,81	97,85	69,95	50	81,81	0,12
Modelo 5	92,57	68,95	62,26	37,93	86,21	0,18
Modelo 6	99,99	100	69,95	50	86,84	0,13

Tabla 9: Resumen de resultados de precisión en tanto por ciento (%) y valor de kappa para modelos entrenados con segmentos de 1 s en la base de datos del HURH.

La Figura 39 muestra el porcentaje de precisión conseguido por cada modelo en los distintos conjuntos de datos: entrenamiento, test y validación. En rosa, se muestra el resultado de cada modelo sobre el conjunto de datos de test utilizando CAR y *z-score*. En el modelo 5 se observa una gran diferencia entre el conjunto de validación y el de entrenamiento, pero existe una diferencia menor entre validación y test que la presente en el resto de modelos. En cuanto al uso de CAR y *z-score*, los resultados obtenidos en el conjunto de test sin utilizar estas técnicas son mejores por lo que se descarta su uso en el resto de problemas. En el caso del problema de clasificación múltiple con tres clases, se hará la prueba en un modelo para cada caso de estudio aunque, como se explicará más adelante, los resultados siguen siendo peores.

Los modelos utilizados en este caso, han sido testeados utilizando 58 sujetos: 15 sujetos de control y 43 enfermos. Los resultados para la clasificación de sujetos son los mostrados en las siguientes tablas.

El modelo 3 (ver) es el que mejores resultados de precisión obtiene sobre el conjunto de datos de test. Su coeficiente kappa es el mejor comparado con el resto de modelos, un 0,19. En este caso se clasifica correctamente un 66,66% (10/15) de sujetos de control. La clase enfermo tiene mejores resultados pues tiene una precisión del 86,49% que comparada con la obtenida por la clase control, un 47,62% es casi el doble.

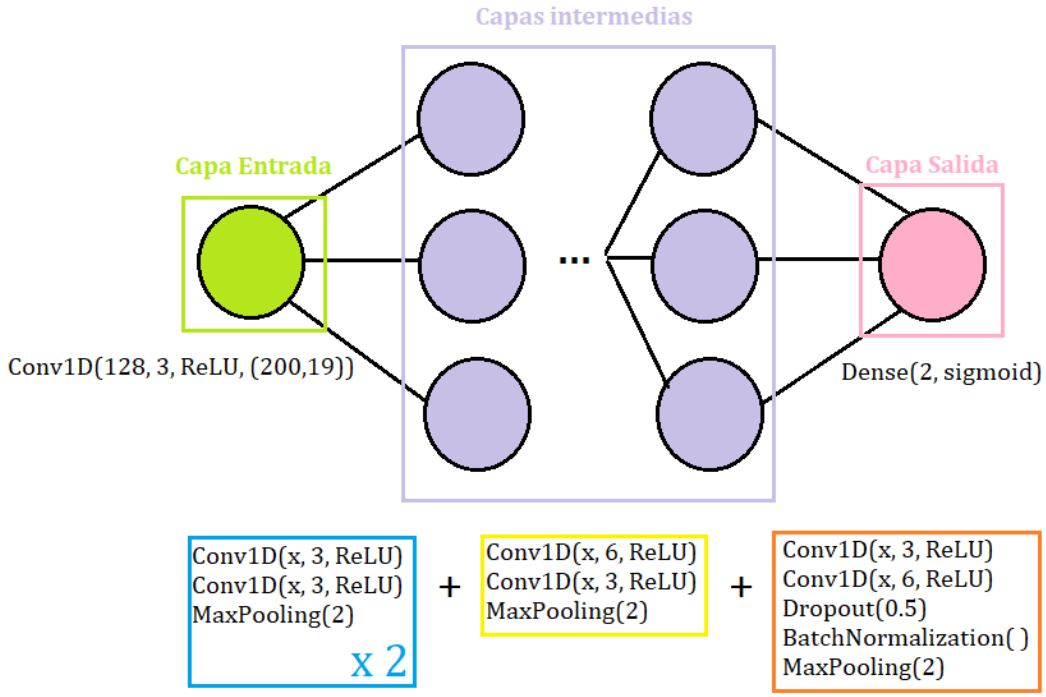

Figura 36: Arquitectura de los modelos 3 y 4 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH.

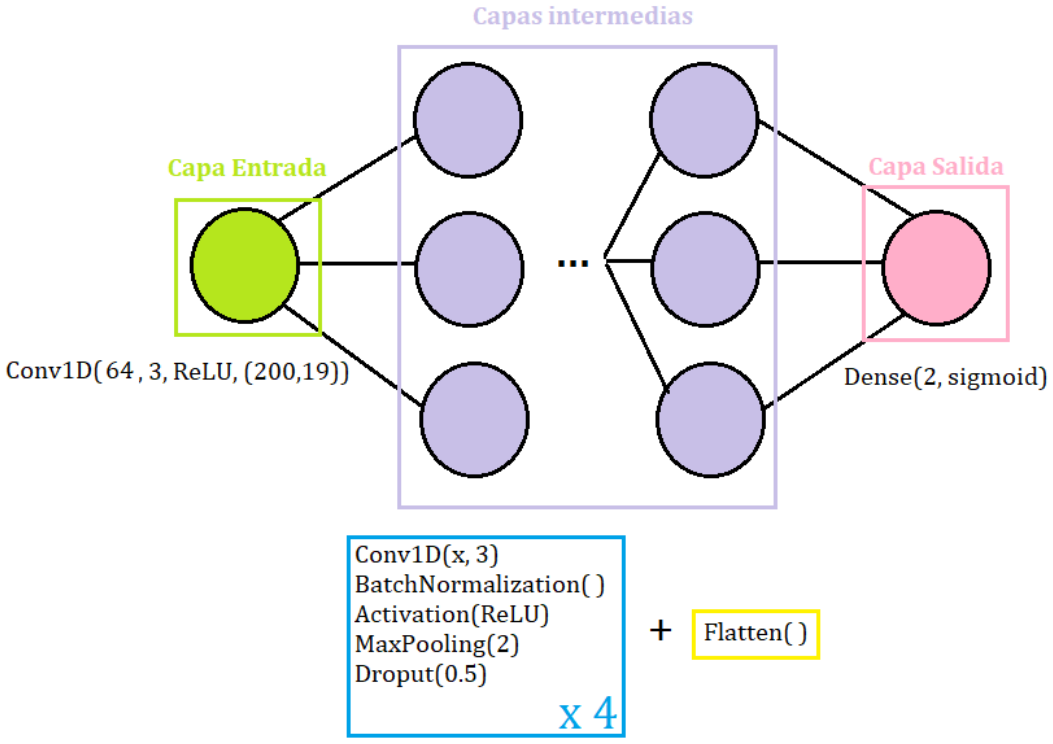

Figura 37: Arquitectura del modelo 5 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH.

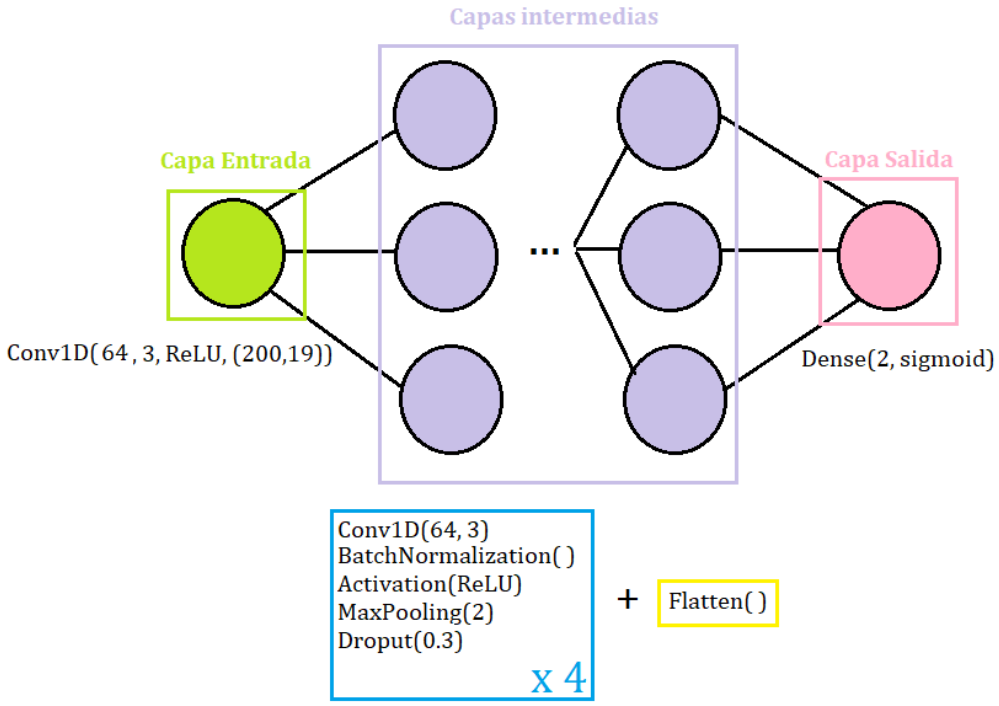

Figura 38: Arquitectura del modelo 6 utilizado en el problema de clasificación binaria para los casos de segmentos de 1 s en la base de datos del HURH.

Se clasifica correctamente un 74,42% (32/43) de los sujetos pertenecientes a esta clase.

Real \ Estimado	Control	Enfermos
Control	10	5
Enfermos	11	32

Tabla 10: Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 1 s en la base de datos del HURH.

El modelo 4 (ver), cuya precisión es del 69,94%, clasifica de forma correcta 7 sujetos de control y 36 enfermos. Si se compara con otros modelos, mejora su rendimiento en la clase de los enfermos a costa de bajar el de la clase de los controles, pues pasa de los 11 sujetos de control obtenidos en el modelo 5 a 7, un 46,66%, mientras que aumenta de 25 a 36 enfermos clasificados correctamente, es decir, alcanza el 83,72% de sujetos clasificados de forma correcta.

Real \ Estimado	Control	Enfermos
Control	7	8
Enfermos	7	36

Tabla 11: Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 1 s en la base de datos del HURH.

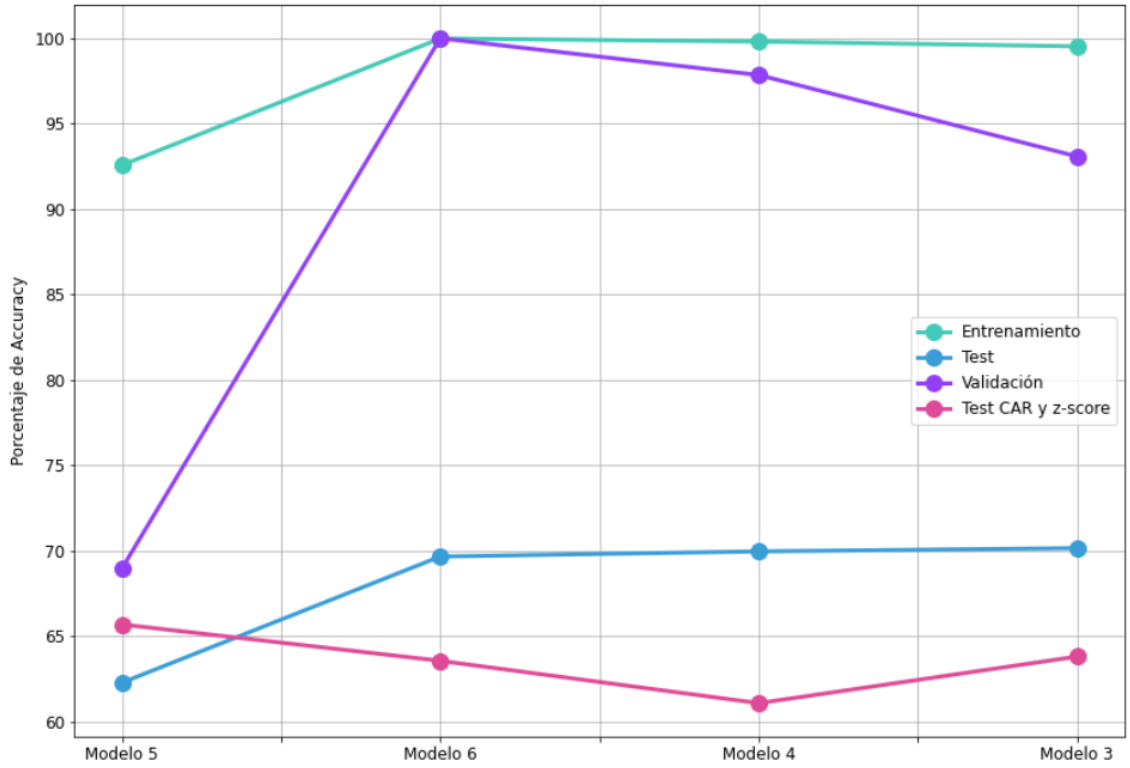

Figura 39: Porcentaje de *precisión* obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y *z-score*, para el problema de segmentos de 1 s en la base de datos del HURH.

El modelo 5, presenta la matriz de confusión representada en la Tabla 12. Este modelo es capaz de clasificar correctamente a 11 controles y 25 enfermos. Los porcentajes de clasificación correcta de sujetos pertenecientes a las dos clases son del 73,33% para los sujetos de control y del 58,13% para los enfermos.

Real \ Estimado	Control	Enfermos
Control	11	4
Enfermos	18	25

Tabla 12: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s en la base de datos del HURH.

El modelo 6, es capaz de clasificar más sujetos de EA de manera correcta que el modelo anterior, sube hasta alcanzar el 76,74%. En el caso de sujetos de control, comparado con los 11 que obtenía el modelo 5, este modelo pierde un sujeto en esta clasificación, clasificando de manera correcta 10 sujetos de control.

En líneas generales, se observa que, tal y como ocurre con el caso de segmentos de cinco segundos, la precisión obtenida por la clase enfermos es mucho mayor a la obtenida por la clase control. Los coeficientes *kappa* que presentan los distintos modelos tienen un rango de entre 0.12 del modelo 4 a 0.19 del modelo 3.

Real \ Estimado	Control	Enfermos
Control	10	5
Enfermos	10	33

Tabla 13: Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s en la base de datos del HURH.

5.3.1.3. Resultados obtenidos para la aplicación de *transfer learning*

La última prueba realizada con estos modelos ha sido probar uno de los modelos, tanto de segmentos de 5 s como de 1 s, sobre los datos de la base de datos del proyecto POCTEP. El modelo 6, que está entrenado con segmentos de 5 s sacados de los datos de la base de datos del HURH, se ha probado sobre los datos de la base de datos del proyecto POCTEP. En este caso, el modelo no se adapta bien a estos datos, se pasa de una precisión del 76,90% (HURH) a una del 55.86% (POCTEP). El modelo 3, que está entrenado con segmentos de 1 s sacados de los datos de la base de datos del HURH, se ha probado sobre los datos de la base de datos del proyecto POCTEP. En este caso, el modelo no se adapta bien a estos datos, se pasa de una precisión del 70,14% (HURH) a una del 42,74% (POCTEP).

Si se utilizan las técnicas CAR y *z-score*, los resultados obtenidos con *transfer learning* empeoran en ambas bases de datos. El modelo 5, entrenado con segmentos de 5 s de las señales de la base de datos del HURH, se ha probado sobre los datos de la base de datos del proyecto POCTEP. Esta prueba, presenta una *precisión* mucho menor que la obtenida en el mismo caso sin utilizar estas técnicas de preprocesado. La métrica precisión pasa de un 60,60% a un 27,63% tras la aplicación del referenciado promedio y el *z-score*. El modelo 4, ha sido entrenado con segmentos de 1 s obtenidos de la base de datos del HURH. Este modelo ha sido probado sobre los datos de la base de datos del proyecto POCTEP en dos escenarios distintos: sin aplicar las técnicas CAR y *z-score* y aplicándolas. En el primer escenario se ha obtenido una precisión de 61,01% mientras que, en el segundo, esta métrica baja a un 49,15%.

Tras realizar estas pruebas, se concluye que utilizar *transfer learning* no parece una técnica adecuada en este problema de clasificación.

5.3.2 Resultados obtenidos para el problema de clasificación binaria en la base de datos del proyecto POCTEP

5.3.2.1. Resultados obtenidos para segmentos de cinco segundos

Para el problema de clasificación binaria aplicado sobre la base de datos del proyecto POCTEP, los resultados de precisión son inferiores a los obtenidos por los modelos entrenados con segmentos de 5 s sobre los datos de la base de datos HURH. En este caso, la máxima precisión obtenida es del 65,89% frente al 76,90% alcanzado por el modelo 6 de la base de datos del HURH. Todos los modelos empleados en el caso de estudio de segmentos de 5 s utilizan algoritmos de

regularización como *ReduceLROnPlateau* y *earlyStopping* para intentar reducir el *overfitting* obtenido al entrenar los modelos inicialmente. También tienen en común el uso de la función de activación ReLU en las capas de tipo *Conv1D*, la función de activación *sigmoid* por tratarse de un problema de clasificación binaria y la función de pérdida *binary*_crossentropy por el mismo motivo. Las principales diferencias serán la estructura de las capas intermedias y los algoritmos de optimización utilizados: SGD o Adam, ambos con distintos valores de *learning rate*. A continuación, se expone la estructura de los modelos que mejores resultados en términos de precisión obtienen para este caso de estudio en las Figuras 40-42.

El modelo 1, cuya estructura se representa en la Figura 40, presenta una capa *Conv1D* formada por 64 filtros, con un tamaño de *kernel* de 3 y unas dimensiones para los datos de entrada de (2500,19). Las capas intermedias siguen la estructura: etapa de convolución, etapa de activación y etapa de *pooling*. Las capas de tipo *Conv1D* encargadas de las dos primeras etapas tendrán un tamaño de *kernel* igual a 3 y utilizarán la función de activación ReLU. En cuanto a la última etapa, se llevará a cabo gracias a una capa *MaxPooling* que, en este caso, tendrá un tamaño de *pool* igual a 2.

El modelo 2 tiene la misma estructura que los modelos 3 y 4, siendo la diferencia principal el número de bloques que forman sus capas intermedias. Los modelos 3 y 4 están formados por 4 bloques de dicha estructura mientras que el modelo 2, está formado por 3 bloques. Utiliza SGD con una *learning rate* de 0.01. En lo que se refiere al entrenamiento del modelo, se produce durante 72 épocas, aunque originalmente se definen 80, debido al uso de *earlyStopping*.

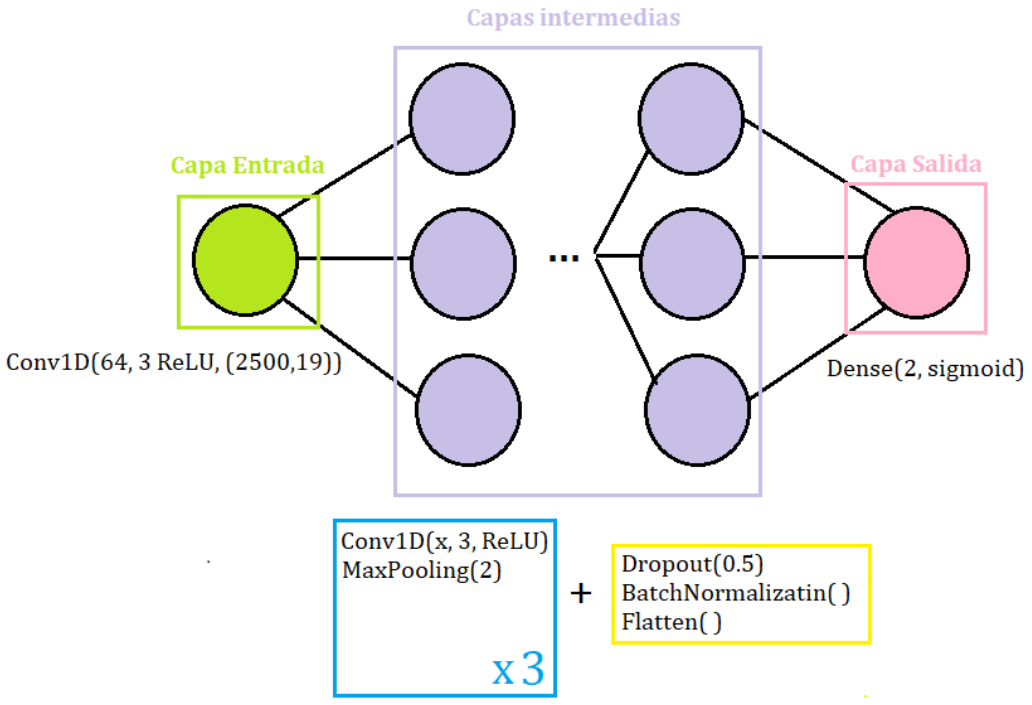

Figura 40: Arquitectura del modelo 1 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

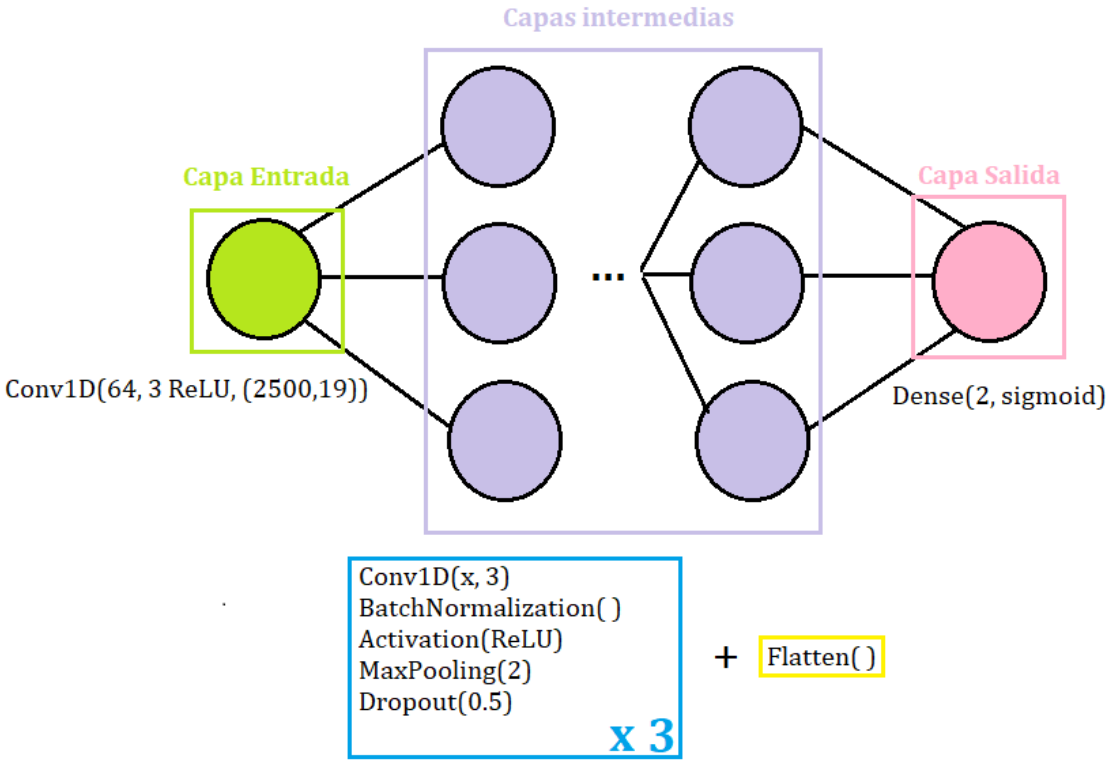

Figura 41: Arquitectura del modelo 2 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

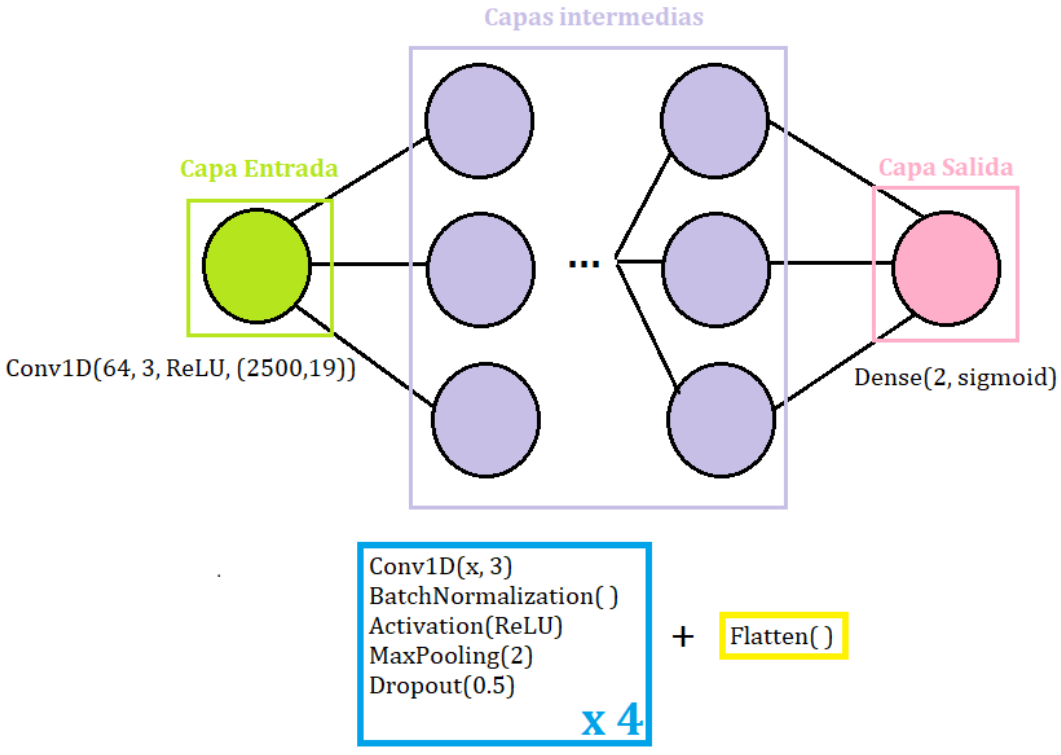

Figura 42: Arquitectura de los modelos 3 y 4 utilizados en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

El modelo 3 mostrado en la Figura 42 tiene como capa de entrada una capa *Conv1D* formada por 64 filtros, un tamaño de *kernel* igual a 3, una función de activación ReLU y unas dimensiones de datos de entrada de (2500,19). Las capas intermedias estarán formadas por cuatro bloques con una estructura del tipo: capa *Conv1D* con un tamaño de *kernel* igual a 3, capa *BatchNormalization*, *Activation*, *MaxPooling* con tamaño de *pool* igual a 2 y *Dropout* con factor igual a 0.5. En cuanto al algoritmo de optimización, utiliza Adam con una *learning rate* de 0.001 y aunque el valor de épocas necesarias para el entrenamiento se fija en 80, para en la época 66 debido al uso de los algoritmos de regularización.

Por último, el modelo 4, también representado en la Figura 43, sigue la misma estructura que el modelo 3. La diferencia entre estos dos modelos es el algoritmo de optimización utilizado. El modelo 4 utiliza SGD con una *learning rate* de 0.01 mientras que, como ya se ha explicado, el modelo 3 utiliza Adam. El entrenamiento de este modelo se lleva a cabo durante 72 épocas, aunque originalmente se fijan 80. Esto se debe al uso de los distintos algoritmos de regulación empleados en este proceso.

La Tabla 14 muestra un resumen de los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa*.

	Entrenamiento	**Validación**	**Test**	**Control**	**EA**	**kappa**
Modelo 3	99,27	98,58	41	47,62	86,49	0,19
Modelo 2	100	100	37,73	50	81,81	0,12
Modelo 3	92,57	68,95	62,26	37,93	86,21	0,18
Modelo 4	99,99	100	69,95	50	86,84	0,13

Tabla 14: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del proyecto POCEP.

Los modelos empleados para el caso de segmentos de 5 s utilizando las técnicas de preprocesado CAR y *z-score* serán los mismos que los utilizados en el caso de segmentos de 5 s.

Los resultados de precisión obtenidos sobre los conjuntos de datos de entrenamiento, validación, test y test utilizando CAR y *z-score* son los que se muestran en la Figura 43. En este caso, los resultados de entrenamiento y validación tienen valores menores a los obtenidos con la base de datos del HURH, además no existen diferencias significativas entre estos dos conjuntos de datos. Sin embargo, sí existen diferencias significativas entre los conjuntos de validación y test, tal y como ocurría en el caso de la base de datos del HURH. La máxima precisión obtenida sobre los datos de test sin utilizar técnicas de preprocesado es del 65,89%.

Este valor máximo lo obtiene el modelo 3. Si comparamos estos resultados con los obtenidos por el mismo conjunto de datos de test al que se aplican las técnicas CAR y *z-score*, la precisión disminuye aproximadamente un 8% hasta llegar al 57,73%. En algunos casos, utilizar estas técnicas aumenta el valor de la precisión, pero normalmente esta métrica suele bajar al utilizar dichas técnicas. El objetivo principal de estas técnicas es aumentar la precisión cuando se utiliza *transfer learning* pero como se explicará más adelante en este mismo apartado, este objetivo no se consigue. Por este motivo, se descarta el uso de las técnicas de preprocesado en el caso homólogo para el problema de clasificación múltiple.

Los datos para testear los modelos pertenecientes a esta base de datos están formados por 75 sujetos. Las matrices de confusión obtenidas para el caso de estudio de segmentos de 5 s son las mostradas en las tablas 15-18. Los 75 sujetos utilizados en este caso de estudio se dividen en: 26 controles y 49 enfermos. El modelo que mejores resultados obtiene es el 3, que presenta una precisión del 65,89% sobre los datos de test.

El primer modelo analizado es el modelo 1, que presenta un porcentaje de clasificación correcta del 53,85% (14/26) para los sujetos pertenecientes a la clase de control.

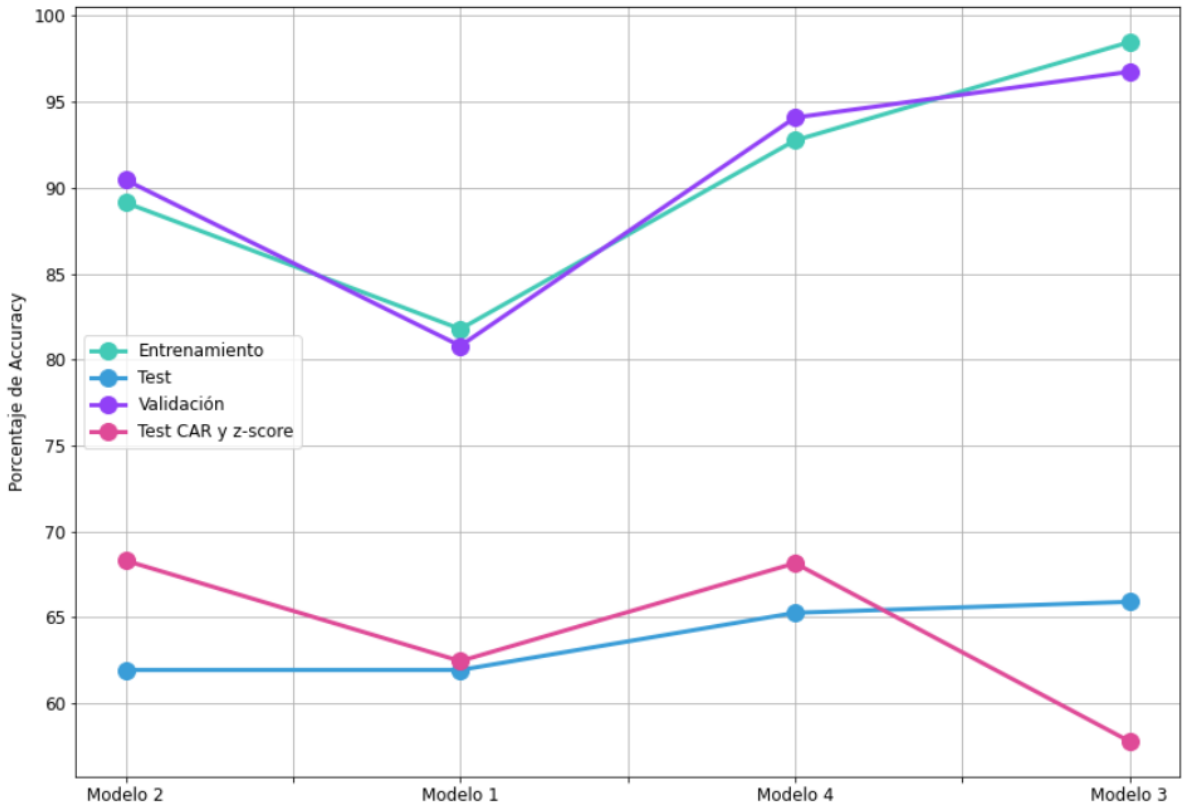

Figura 43: Porcentaje de *precisión* obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y *z-score* para el problema de clasificación binaria con segmentos de 5 s en la base de datos del proyecto POCTEP.

La clase de los enfermos presenta un porcentaje del 69,39% (34/49) de sujetos clasificados correctamente.

Real \ Estimado	Control	Enfermo
Control	14	12
Enfermo	15	34

Tabla 15: Matriz de confusión obtenida por el modelo 1 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP.

En el caso del modelo 2, es el que peores resultados obtiene pues clasifica el 0% de los sujetos de control de forma correcta. Por otra parte, es capaz de clasificar un total de 48 enfermos de forma correcta consiguiendo una tasa de clasificación correcta del 97,96%. Los resultados de clasificación son malos, ya que, aunque alcance casi un 100% de aciertos para una clase obtiene un 0% en la otra.

Real \ Estimado	Control	Enfermos
Control	0	26
Enfermos	1	48

Tabla 16: Matriz de confusión obtenida por el modelo 2 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP.

El modelo 3 (ver) es el que mejor precisión consigue: un 65,89%. Los resultados de clasificación por sujetos muestran que este modelo tiende a clasificar a todos los sujetos como enfermos.

Real \ Estimado	Control	Enfermos
Control	2	24
Enfermos	7	48

Tabla 17: Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP.

Por último, el modelo 4 (ver Tabla 18) consigue clasificar a 14/26 (53.85%) sujetos de control correctamente. Por otra parte, la clase de los enfermos obtiene un 75,51% (37/49) de sujetos clasificados correctamente.

De forma general, la precisión obtenida para cada clase resulta mejor para la clase enfermos. Suele haber bastante diferencia entre estas precisiones, aproximadamente un 38% de media, llegando a obtener la clase control una precisión del 0% en uno de los casos.

Real \ Estimado	Control	Enfermos
Control	14	12
Enfermos	12	37

Tabla 18: Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 5 s en la base de datos del proyecto POCTEP.

5.3.2.2. Resultados obtenidos para segmentos de un segundo

En cuanto al caso de segmentos de 1 s, la máxima precisión obtenida será de 65,41%. En esta base de datos, los resultados obtenidos para los casos de estudio de segmentos de 5 s y segmentos de 1 s, se mantienen. Tal y como ocurre en el caso de estudio anterior, los modelos empleados para este caso utilizando las técnicas CAR y *z-score* serán los mismos que los empleados en el caso de segmentos de 1 s sin utilizar ninguna de estas técnicas de preprocesado. Todos los modelos utilizarán ReLU como función de activación en las capas de tipo *Conv1D*. También emplearán la función de activación *sigmoid* en la capa de salida y la función de pérdida *binary_crossentropy* al compilar el modelo para su posterior entrenamiento. Las dos opciones utilizadas para la optimización en los distintos modelos son: Adam o SGD con distintas tasas de aprendizaje.

Por último, igual que ocurre con todos los modelos explicados hasta ahora, se utilizarán algoritmos y técnicas de regulación como *earlyStopping* y *reduceLROnPlateau* para intentar evitar o reducir el *overfitting* en los modelos. Las arquitecturas correspondientes se presentan en las Figuras 44 y 45.

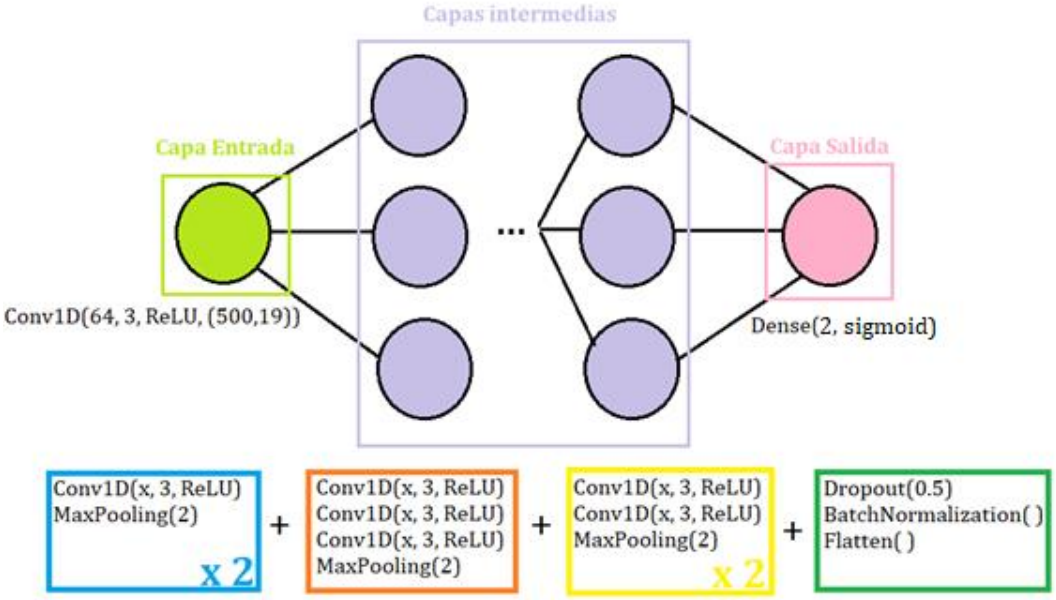

Figura 44: Arquitectura de los modelos 4 y 5 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP.

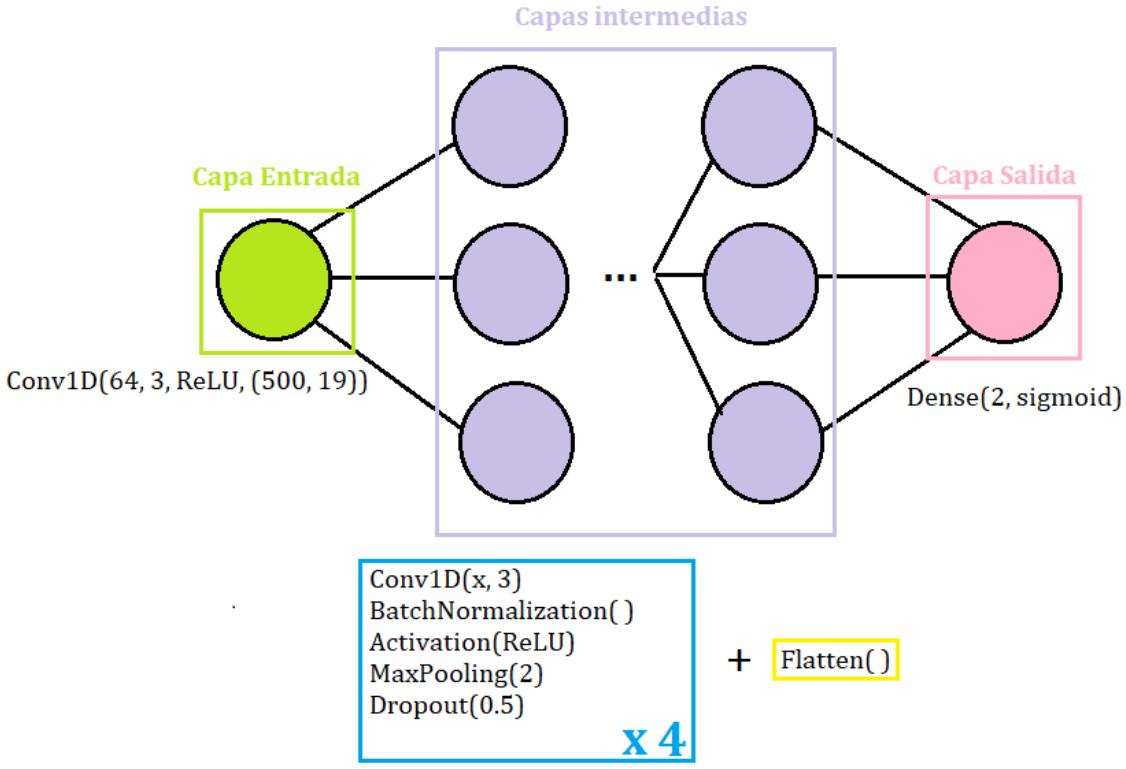

Figura 45: Arquitectura del modelo 6 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP.

El modelo 5 representado en la Figura 45 tiene una capa *Conv1D* como capa de entrada. Esta capa estará formada por 64 filtros, con un tamaño de *kernel* de 3, una función de activación ReLU y unas dimensiones de datos de entrada al modelo de (500, 19). Las capas intermedias siguen la estructura: capa *Conv1D*, capa *BatchNormalization,* capa *Activation,* capa *MaxPooling* y, por último, capa *Dropout*. Tras el último bloque de capas intermedias se coloca una capa *Flatten* para adaptar las dimensiones de estas capas a la capa de salida. Las capas de tipo *Conv1D* estarán formadas por distinto número de filtros en cada bloque, pero tendrán el mismo tamaño de *kernel* en todas las capas, que será 3. El factor de *dropout* será de 0.5 para este tipo de capas, y el tamaño de *pool* para la capa *MaxPooling* será 2. Tras las capas intermedias se añade la capa de salida, una capa tipo *Dense* con dos unidades, una por cada clase existente en el problema y la función de activación *sigmoid* ya mencionada.

Los modelos 4 y 5, siguen la misma estructura, la mostrada en la Figura 44. La diferencia entre estos dos modelos es el algoritmo de optimización, el modelo 4 emplea el algoritmo Adam con una tasa de aprendizaje de 0.001 mientras que el modelo 5 utiliza SGD con una tasa de aprendizaje de 0.01. La estructura de estos modelos está formada por una capa de entrada de tipo *Conv1D* con 64 filtros, un tamaño de *kernel* igual a 3 y unas dimensiones de entrada de datos al modelo de (500,19). Tras esta capa, se añaden las capas intermedias, formadas por los distintos tipos de bloques mostrados en la Figura 45. El primer tipo es un bloque formado por una capa *Conv1D* y otra *MaxPooling*. El segundo tipo es un bloque formado por dos

capas de tipo *Conv1D* seguidas por una capa *MaxPooling*. El tercer tipo de bloque sigue una estructura de tres capas *Conv1D* seguidas de otra *MaxPooling*. El cuarto y último tipo de bloque está formado por las capas *Dropout* con un factor 0.5, *BatchNormalization* y *Flatten*. En total se utilizan dos bloques del primer tipo, tres del segundo y uno del tercero. Las capas *Conv1D* de todos los bloques tienen un tamaño de *kernel* igual a 3 y una función de activación ReLU. El tamaño de *pool* utilizado en todas las capas *MaxPooling* es de 2.

La Tabla 19 muestra un resumen de los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa*.

	Entrenamiento	**Validación**	**Test**	**Control**	**EA**	**kappa**
Modelo 4	81,77	80,79	61,92	76,19	61,11	0,33
Modelo 5	89,13	90,45	61,92	70	58,19	0,23
Modelo 6	98,48	96,75	65,89	80	52,86	0,06

Tabla 19: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del proyecto POCTEP.

Los resultados para este caso de estudio son los mostrados en la Figura 46. En este caso se observa que aplicar técnicas de preprocesado hace que, en la mayoría de los casos, los resultados de precisión desciendan.

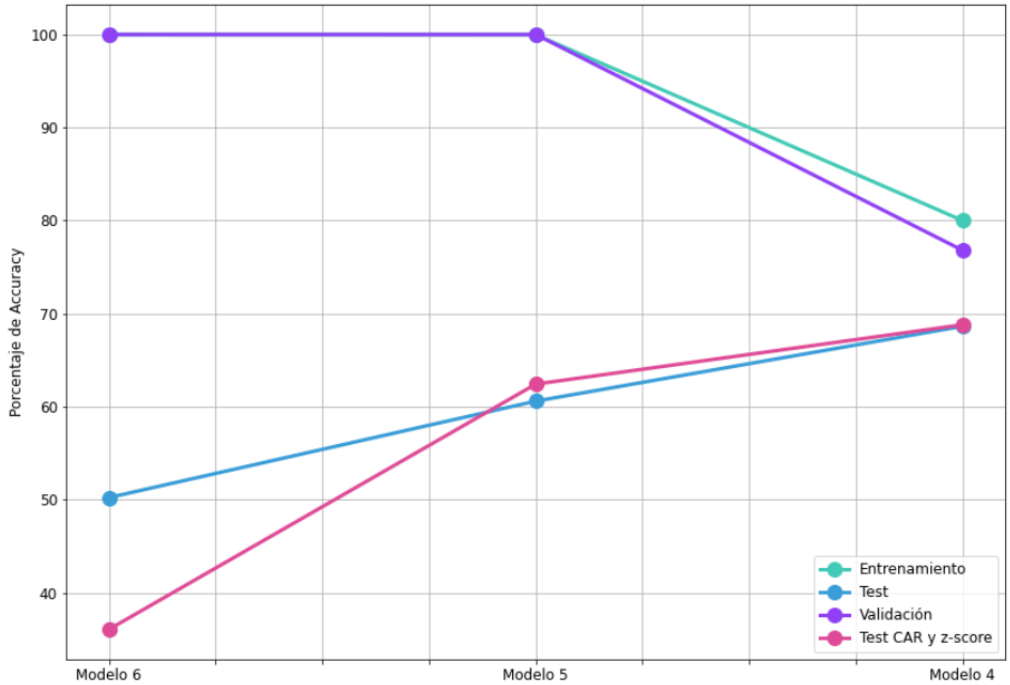

Figura 46: Porcentaje de *precisión* obtenido en los distintos modelos sobre los conjuntos de datos de entrenamiento, test, validación y test utilizando las técnicas CAR y *z-score* para el problema de clasificación binaria con segmentos de 1 s en la base de datos del proyecto POCTEP.

Por este motivo se descarta su uso en el mismo caso de estudio para el problema de clasificación múltiple. El modelo que presenta los mejores resultados en términos de precisión es el modelo 4, con un 68,65%. El que peores resultados muestra es el modelo 6, con un 50,23%. Siguen existiendo diferencias significativas entre los conjuntos de datos de test y validación, excepto en el modelo 4, donde la diferencia no es tan grande. Esto indica que es un modelo que o no presenta *overfitting* o si lo presenta no es muy elevado, lo que hace que los resultados de este modelo sean fiables. En los otros casos, aunque se hayan utilizado algoritmos de regularización como *reduceLROnPlateau* o *earlyStopping* parece que este problema sigue existiendo. Si se comparan estos resultados con los obtenidos en el caso de segmentos de 5 s, se observa que el máximo valor de precisión obtenido se mantendrá. Para este caso de estudio, esta métrica toma un valor máximo de 68,65 % mientras que en el caso de segmentos de cinco segundos tomará un valor del 65,89%.

Los resultados conseguidos para el caso de estudio de segmentos de 1 s se muestran en las tablas Tabla 20, Tabla 21 y Tabla 22. Los 75 sujetos utilizados en este caso de estudio se dividen en: 37 sujetos de control y 38 enfermos.

El modelo 4 es el que mejores resultados de clasificación consigue. Clasifica correctamente a 16 de los 37 sujetos de control y a 33 de los 48 enfermos. Comparado con el modelo anterior, los porcentajes de clasificación de la clase de los controles aumentan un poco, pasando del 37,84 % al 43,24%. Sin embargo, para la clase de los enfermos los resultados disminuyen, pasando del 97,37% al 86,84%. Este modelo es el que obtiene el coeficiente *kappa* más elevado de todos los modelos desarrollados por lo que será el que presente los resultados el más fiables.

Real \ Estimado	Control	Enfermo
Control	16	21
Enfermo	5	33

Tabla 20: Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP.

El modelo 5 (ver Tabla 21) mejora los resultados si se compara con el modelo 6. Consigue clasificar correctamente 14/37 (37.84%) sujetos de control y 37/38 (97.37%) enfermos. Este modelo es el que obtiene un valor de *kappa* más elevado, 0,33, por lo que, aunque no sea el que mejor precisión obtiene en el conjunto de test, es el que presenta unos resultados más fiables.

Real \ Estimado	Control	Enfermo
Control	14	23
Enfermo	6	32

Tabla 21: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP.

El modelo 6, cuya matriz de confusión se muestra en la Tabla 22, clasifica correctamente 4/37 (12.12%) sujetos de control y 37/38 (97.37%) enfermos. No se trata de un buen modelo, ya que, aunque sea capaz de clasificar aproximadamente el 100% de sujetos de una clase de forma correcta, el porcentaje de sujetos clasificados de forma correcta que pertenece a la otra clase es muy pequeño.

Real \ Estimado	Control	Enfermo
Control	4	33
Enfermo	1	37

Tabla 22: Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s en la base de datos del proyecto POCTEP.

5.3.2.3. Resultados obtenidos para la aplicación de *transfer learning*

La aplicación de *transfer learning* para estos casos de estudios ha obtenido resultados poco favorables. Se ha decido probar un modelo por cada caso de estudio explicado. Los modelos utilizados han sido entrenados de dos formas distintas: (*i*) utilizando técnicas de preprocesado como CAR y *z-score*; (*ii*) sin utilizar estas técnicas. Para el caso de estudio de segmentos de 5 s, se ha decidido aplicar *transfer learning* con el modelo 1. Como ya se ha explicado, ha sido necesario hacer un interpolado en los datos de la base de datos del HURH para adaptar los datos a las dimensiones de entrada del modelo. Se estudian los resultados para los dos escenarios descritos:

- En el escenario (*i*), donde se aplican las técnicas de preprocesado explicadas, el valor de *precisión* que presenta el modelo 1 sobre los datos de test es del 62,44%. Tras aplicar *transfer learning*, los resultados disminuyen de manera drástica hasta alcanzar el 8,45%.
- En el escenario (*ii*), donde no se aplican técnicas como CAR y *z-score*, el valor de precisión que presenta el modelo 1 sobre los datos de test es del 61,92%. Si se aplica *transfer learning* a este modelo, el resultado de precisión conseguido sobre los datos de test es del 18,07%.

Para el caso de estudio de segmentos de 1 s, el modelo elegido ha sido el 4. Los resultados para los dos escenarios descritos son los siguientes:

- En el escenario (i), el valor de precisión que presenta este modelo sobre los datos de test es del 68,81%. Tras la aplicación del paradigma *transfer learning*, el resultado obtenido es peor que el original, del 30,53% por lo que el rendimiento del modelo baja en este caso.
- En el escenario (ii), el valor de precisión obtenido por el modelo 4 es del 68,65%. Si se aplica *transfer learning* el resultado obtenido es del 31,31% reduciéndose la precisión hasta aproximadamente la mitad del valor original.

Como puede observarse, los resultados obtenidos tras la aplicación de este paradigma no son buenos, los modelos no son capaces de adaptarse a los datos de

la base de datos del HURH, por lo que, para este caso, no se recomienda utilizar *transfer learning* sino crear un modelo para cada base de datos.

5.3.3 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del Hospital Universitario Río Hortega

5.3.3.1. Resultados obtenidos para segmentos de cinco segundos

Los resultados obtenidos para este problema de clasificación múltiple con tres clases: controles, pacientes con DCL y enfermos con EA, empeoran al compararlos con el problema de clasificación binaria. En este caso, se obtiene una precisión de aproximadamente el 40% para los datos de test en todos los casos estudiados. La aplicación de las técnicas CAR y *z-score* se ha descartado en este problema, ya que se han realizado pruebas con algunos modelos y los resultados no mejoraban. En cuanto a la aplicación de *transfer learning*, los resultados de precisión obtenidos al utilizar esta técnica hacen que esta métrica descienda aproximadamente en un 10%. Esto implica que, los modelos entrenados con la base de datos del HURH en los que no se aplican las técnicas CAR y z-score, presentan unos valores de precisión mayores a los obtenidos al aplicar *transfer learning*.

A continuación, se muestra la estructura de los modelos que mejores resultados de precisión han obtenido para este problema (Figuras 47-51).

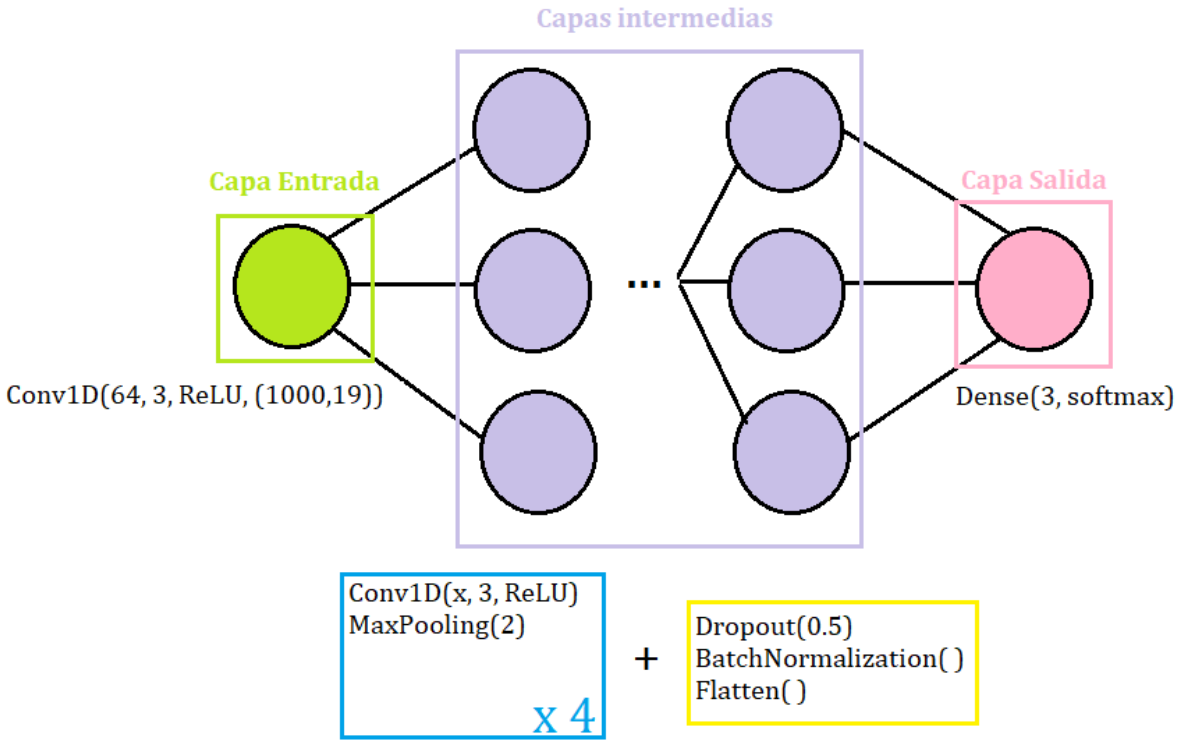

Figura 47: Arquitectura del modelo 5 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH.

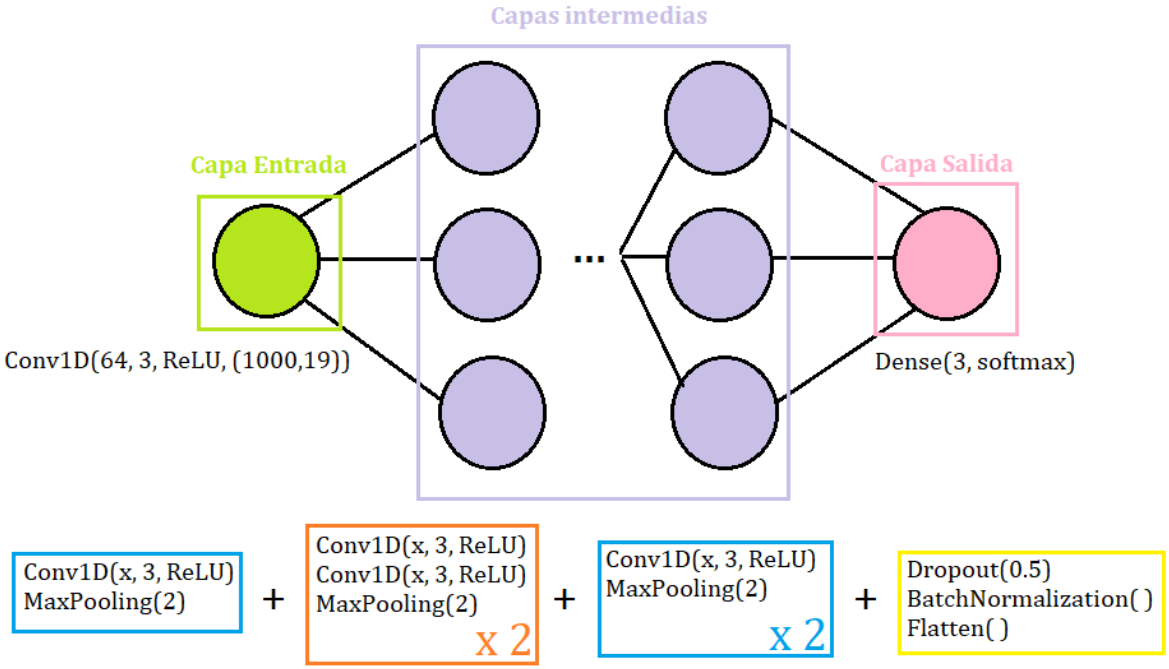

Figura 48: Arquitectura del modelo 6 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH.

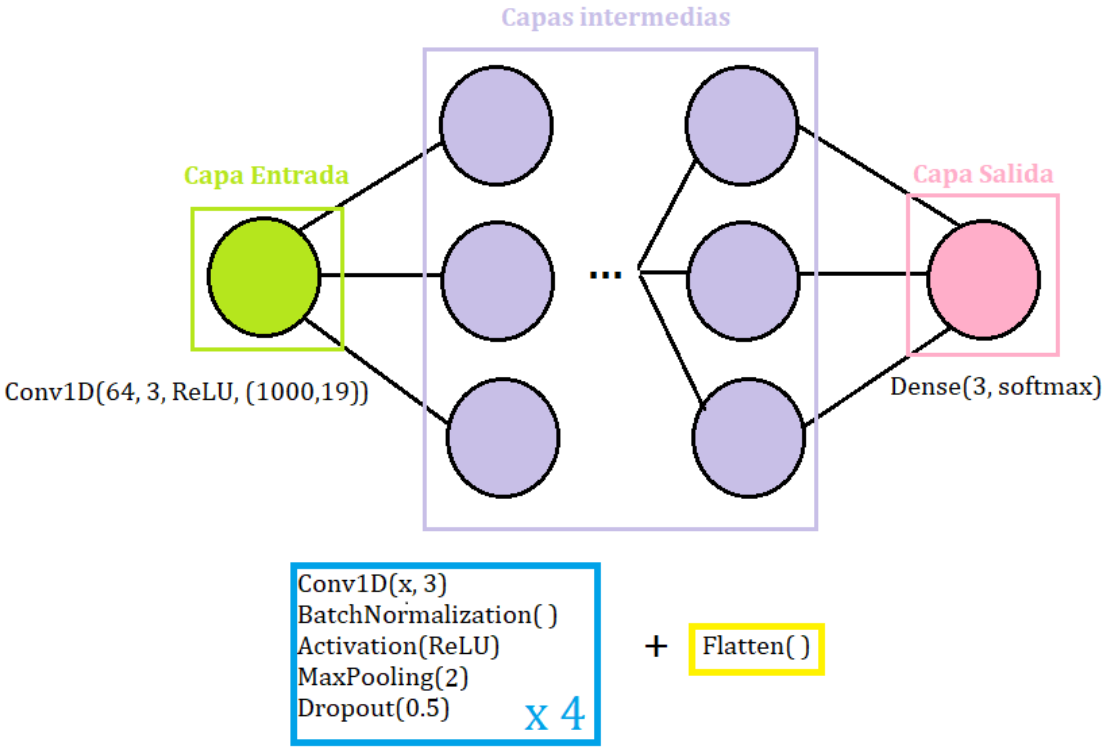

Figura 49: Arquitectura del modelo 8 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH.

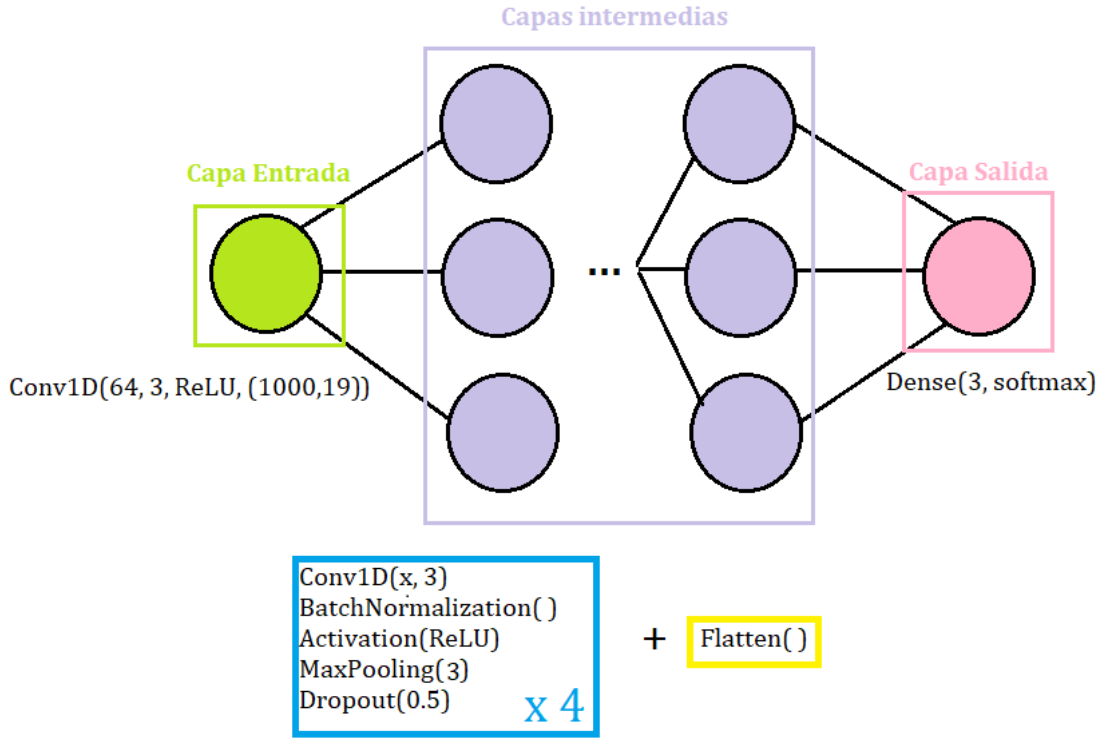

Figura 50: Arquitectura del modelo 9 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH.

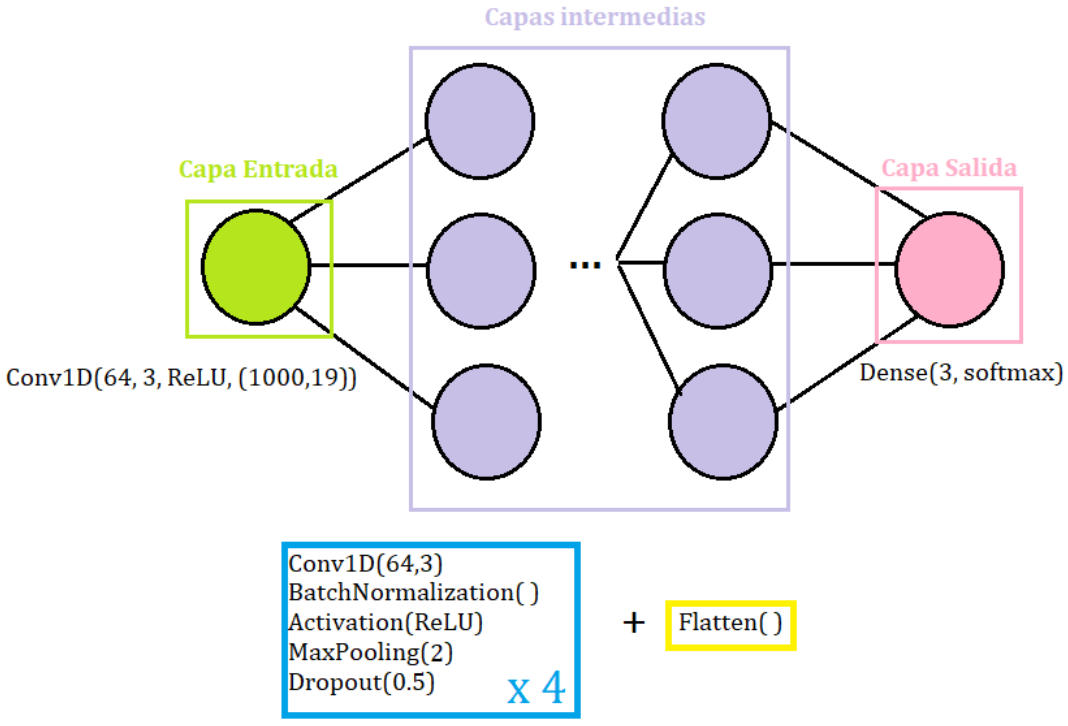

Figura 51: Arquitectura del modelo 10 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del HURH.

Todos los modelos seleccionados utilizan la función de activación *Softmax* por tratarse de un problema de clasificación múltiple.

Como función de pérdida utilizarán *categorical_crossentropy*, por el mismo motivo por el que se utiliza *Softmax* como función de activación. Tal y como ocurría en el problema de clasificación binaria, la principal diferencia será la estructura de los modelos y los optimizadores utilizados por cada uno de ellos.

El modelo 5, representado en la Figura 47, comienza con una capa de entrada formada por una capa *Conv1D* con 64 filtros, tamaño de *kernel* igual a 3, función de activación ReLU y unas dimensiones (1000,19). Las capas intermedias siguen la estructura: etapa convolucional, etapa de activación y etapa de *pooling* explicada anteriormente. Estas capas están formadas por cuatro bloques de capas convolucionales y utiliza como optimizador el algoritmo SGD con una *learning rate* de 0.01. Cada capa *Conv1D* utiliza ReLU como función de activación y tiene un tamaño de *kernel* igual a 3. Las capas *MaxPooling* tienen un tamaño de *pool* igual a 2. El entrenamiento de este modelo se lleva a cabo durante 110 épocas y utiliza SGD con una *learning rate* de 0.01.

El modelo 6, cuya estructura se representa en la Figura 48, tendrá como primera capa una capa *Conv1D* con 64 filtros, un tamaño de *kernel* igual a 3 y unas dimensiones (1000,19). Esta capa utilizará la función de activación ReLU. Siguiendo a esta capa, aparecen cuatro bloques, dos del tipo capa *Conv1D* y capa *MaxPooling*, uno con dos capas *Conv1D* y una *MaxPooling* y por último una con las capas *Dropout, BatchNormalization* y *Flatten* para adaptar las dimensiones a la capa de salida. Tras esta estructura de bloques se sitúa la capa final, de tipo *Dense* con tres unidades, una por clase y una función de activación *Softmax.* El tamaño de *pool* será 2 en todas las capas de tipo *MaxPooling,* se utilizará la función de activación ReLU en todas las capas *Conv1D* y el factor de *dropout* será 0.5.

Los modelos 8, 9 y 10 tendrá exactamente la misma estructura. Comienzan con una capa *Conv1D* con 64 unidades, tamaño de *kernel* 3, función de activación ReLU y dimensiones (1000,19). A continuación, aparecen 4 bloques de tipo *Conv1D*, *BatchNormalization, Activation, MaxPooling* y *Dropout*. Por último, tendrán una capa *Flatten* seguida de una de tipo *Dense* con 3 unidades y la función *Softmax.* La diferencia entre los tres modelos está en los bloques utilizados en las capas intermedias, ya que, aunque siguen la misma estructura, los valores de las unidades de la capa *Conv1D,* representados con una x, y el valor de *pool_size* de la capa *MaxPooling,* cambian de un modelo a otro. Los modelos 8 y 10 tendrán como única diferencia el tamaño de filtros de cada capa *Conv1D*. En el modelo 10, todas las capas de este tipo tendrán el mismo número de filtros, mientras que, en el modelo 8, cada capa tendrá un número de filtros diferente al anterior. La diferencia entre los modelos 8 y 9 es únicamente el valor del *pool_size* de la capa *MaxPooling*. En el modelo 8, el valor de este parámetro será 2 mientras que, en el 9, este valor será 3. El resto es exactamente igual, tienen el mismo número en cada capa *Conv1D* y utilizan las mismas funciones de activación.

Todos los modelos emplean el optimizador Adam con un *learning rate* de 0.001, excepto el modelo 5 que utiliza SGD con una tasa de aprendizaje de 0.01. En cuanto a las épocas utilizadas para entrenar el modelo, todos los modelos tienen este parámetro fijado en 120 épocas para el entrenamiento, excepto el modelo 9, donde se fija a 300. Como se ha explicado a lo largo de este capítulo, aunque se fije un número determinado de épocas, cada modelo parará en un número de épocas menor al fijado debido al uso de *earlyStopping*. Para los modelos donde el número de épocas se fija en 120, el que mayor número de épocas emplea en el entrenamiento es el modelo 5, que para después de 108 épocas. El modelo 10 le sigue con 97 y, por último, los modelos 6 y 8, que paran tras 85 y 87 épocas respectivamente. El modelo 9, con un número de épocas fijado en 300, parará tras 136 épocas obteniendo los resultados de validación y de entrenamiento representados en la Figura 52. La Figura 52 muestra el porcentaje de precisión obtenido por cada uno de estos cinco modelos en los distintos conjuntos de datos: entrenamiento, validación y test. Los modelos han sido ordenados de forma ascendente para el porcentaje de precisión obtenido sobre el conjunto de datos de test. Al analizar la Figura 52, se observa que las diferencias existentes entre el conjunto de datos de entrenamiento y el de validación son pequeñas. Destaca la diferencia entre estos dos conjuntos de datos en el modelo 6, donde es un poco mayor que la obtenida en el resto. En este caso también existen diferencias significativas entre el conjunto de test y el de validación, por lo que de nuevo puede ser una señal de *overfitting* que no se soluciona aplicando algoritmos y capas de regularización. La máxima *precisión* alcanzada en este caso, es la conseguida por el modelo 9, un 40%.

La Tabla 23 muestra un resumen de los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa*.

	Entrenamiento	**Validación**	**Test**	**Control**	**DCL**	**EA**	**kappa**
Modelo 5	100	95,76	35,09	5,55	33,33	51,14	0,05
Modelo 6	97,14	87,13	39,11	10,53	44,44	63,33	0,05
Modelo 8	100	99,01	39	0	60	55	0,05
Modelo 9	100	99,50	40,02	8,33	55,55	62,16	0,02
Modelo 10	99,97	98,75	37,35	5,55	30	56,67	0,09

Tabla 23: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del HURH.

Los resultados obtenidos para este problema de clasificación múltiple en el caso de segmentos de 1 s, se muestran en la Figura 56. En este caso, se muestran los cinco modelos que mejores resultados, en términos de precisión, han obtenido. Como ocurre en el caso de segmentos de cinco segundos, los modelos utilizados emplean como función de activación *Softmax* y como función de pérdida

categorical_crossentropy por tratarse de un caso de clasificación múltiple con tres clases. Todos utilizan *earlyStropping* y *ReduceLROnplateau* para reducir el *overfitting* en el entrenamiento. En cuanto a los algoritmos de optimización, se utilizan SGD y Adam con distintas tasas de aprendizaje.

En este caso, al tratarse de un problema de clasificación múltiple con tres clases, tendremos matrices de confusión de tamaño 3x3. Para esta base de datos, se testea el modelo con un total de 58 sujetos: 4 sujetos de control, 23 pacientes con DCL y 31 enfermos con EA. A continuación, se muestran los resultados correspondientes al caso de segmentos de 5 s. La clasificación en este problema obtiene resultados malos, pues la mayor precisión lograda para esta base de datos es de un 40%, conseguida por el modelo 9.

Los resultados de clasificación por sujetos para el modelo 5 se muestran en la . La clase que peores resultados obtiene es la clase de los controles, donde se clasifica correctamente un único sujeto de los 4 existentes. Le sigue la clase de los pacientes con DCL, con una clasificación correcta de 4 sujetos de un total de 23. Por último, la que mejor comportamiento obtiene es la clase de los enfermos con EA, donde se clasifican correctamente la mitad de los sujetos, un total de 16 sujetos de los 31 existentes.

Real \ Estimado	Control	DCL	EA
Control	1	2	1
DCL	8	4	11
EA	9	6	16

Tabla 24: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH.

Si se compara con el modelo anterior, el modelo 6 mejora los resultados de clasificación para las clases de los controles y enfermos con EA, y mantiene los sujetos clasificados correctamente de la clase de los pacientes con DCL (ver). Este modelo clasifica correctamente la mitad de los sujetos de control, 2 sujetos. En el caso de los pacientes con DCL, mantiene los 4 sujetos obtenidos por el modelo anterior. Por último, para la clase de los enfermos con EA, clasifica 19 sujetos de forma correcta, por lo que esta es la clase que mejores resultados obtiene.

Real \ Estimado	Control	DCL	EA
Control	2	0	2
DCL	10	4	9
EA	7	5	19

Tabla 25: Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH.

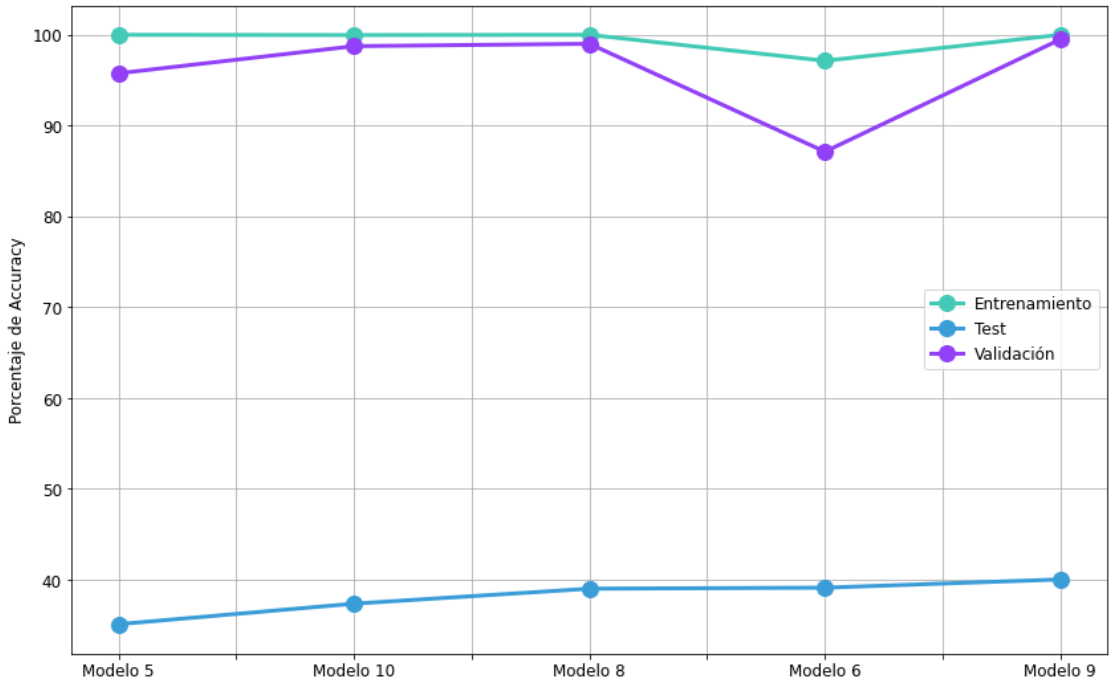

Figura 52: Porcentaje de precisión obtenido por los modelos para el caso de segmentos de 5 s en el problema de clasificación múltiple en la base de datos del HURH.

El modelo 8 (ver Tabla 26) obtiene un 0% de clasificación correcta para la clase control, seguido de un 13,04% de pacientes con DCL y finalmente, un 70,97% de los enfermos con EA. Esta última clase será la que mejores resultados de clasificación obtenga.

Real \ Estimado	Control	DCL	EA
Control	0	0	4
DCL	6	3	14
EA	7	2	22

Tabla 26: Matriz de confusión obtenida por el modelo 8 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH.

El modelo 9 (ver Tabla 27) es el que mejores resultados de clasificación de sujetos presenta. Clasifica de manera correcta el 25% de sujetos de control, el 21,74% de pacientes con DCL y el 74,19% de enfermos con EA. La clase para la que mejor funciona el modelo es la de los enfermos con EA.

Real \ Estimado	Control	DCL	EA
Control	1	1	2
DCL	6	5	12
EA	5	3	23

Tabla 27: Matriz de confusión obtenida por el modelo 9 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH.

Por último, el modelo 10 (ver Tabla 28) consigue clasificar correctamente el 54,84% de los sujetos pertenecientes a la clase de enfermos con EA, siendo ésta la que mejores resultados consigue. Para la clase de los controles consigue un 25% y para los pacientes con DCL, un 13,04%.

En líneas generales, la precisión para la clase EA suele ser mucho mayor que la obtenida para la clase control, que alguna ocasión es del 0%. En cuanto a la clase DCL la precisión también es menor que la que obtiene la clase EA pero es elevada comparada con la de la clase control. En cuanto a los coeficientes *kappa*, todos los modelos obtienen coeficientes cercanos a ceros lo que implica que muchos de los resultados presentados son prácticamente aleatorios, excepto el del modelo 10 cuya *kappa* es 0.09.

Real \ Estimado	Control	DCL	EA
Control	1	0	3
DCL	10	3	10
EA	7	7	17

Tabla 28: Matriz de confusión obtenida por el modelo 10 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del HURH.

5.3.3.2. Resultados obtenidos para segmentos de un segundo

La estructura de los cinco modelos que mejores resultados de precisión han obtenido en el conjunto de datos de test, son los mostrados en las Figura 53, Figura 54 y Figura 55.

El modelo 3 presenta una capa inicial de tipo *Conv1D* con 128 filtros, un tamaño de *kernel* de 3, función de activación ReLU y unas dimensiones de (200,19). Las capas intermedias estarán formadas por cuatro bloques que siguen la estructura: dos capas *Conv1D* con un número determinado de filtros *x*, un tamaño de *kernel* 3 y función de activación ReLU seguidas de una capa *MaxPooling* con un *pool_size* igual a 2. Tras estos cuatro bloques, se introduce otro bloque con capas *Dropout* con un factor 0.5, *BatchNormalization* y *Flatten*. La capa de salida es una capa *Dense* con 3 unidades, una por clase existente, y función de activación *Softmax*. Los resultados de entrenamiento y validación de este modelo, mostrados en la Figura 56, se han obtenido tras 54 épocas de entrenamiento aunque, originalmente, se fijó el número de épocas a 120.

Los modelos 8 y 9 tienen exactamente la misma estructura. La única diferencia entre ambos es el algoritmo utilizado en la optimización. En el primero, se utiliza Adam, con una *learning rate* de 0.001 y en el segundo, SGD, con una *learning rate* de 0.01. Ambos modelos siguen la estructura mostrada en la Figura 54. Comienzan con una capa *Conv1D* con 64 filtros, un tamaño de filtro de 3, función de activación ReLU y dimensiones de los datos de entrada al modelo de (200,19). Tras la capa de entrada, las capas intermedias estarán formadas por cinco bloques de tipo: etapa convolucional, etapa de activación y etapa de *pooling*, es decir, por una capa *Conv1D*

con función de activación ReLU seguida de una capa de *MaxPooling* con *pool_size* igual a 2. Tras estos cinco bloques, se inserta otro bloque con las siguientes capas: *Dropout* con factor 0.4, *BatchNormalization* y *Flatten*, que adaptará las dimensiones de este bloque a la capa de salida. Esta última capa será del tipo *Dense* y estará formada por 3 unidades con una función de activación *Softmax*. El entrenamiento de estos modelos se lleva a cabo tras 55 épocas en el caso del modelo 8 y 121 en el del modelo 9. En ambos casos, se ha fijado el número de épocas a 150 pero debido al uso de *earlyStopping* y *ReduceLROnPlateau* el modelo finaliza el entrenamiento sin completar las épocas fijadas, obteniendo los resultados expuestos en la Figura 56.

Ocurre lo mismo con los modelos 10 y 11, que siguen la misma estructura teniendo como única diferencia el algoritmo utilizado para la optimización. El modelo 10 utiliza Adam con una *learning rate* de 0.001 y el 11, SGD, con una *learning rate* de 0.01. La estructura de estos modelos se muestra en la Figura 55. Comienzan con una capa *Conv1D* con 64 filtros, un tamaño de filtro de 4, función de activación ReLU y dimensiones de los datos de entrada al modelo de (200,19). Tras la capa de entrada, las capas intermedias estarán formadas por cuatro bloques de tipo: etapa convolucional, etapa de activación y etapa de *pooling*, es decir, por una capa *Conv1D* con un tamaño de filtro de 4 y función de activación ReLU, seguida de una capa de *MaxPooling* con *pool_size* igual a 2. Tras estos cuatro bloques, se inserta otro bloque con la misma estructura, pero diferentes valores: una capa *Conv1D* con 1024 filtros, un tamaño de filtro de 3 y función de activación ReLU, seguida de una capa de *MaxPooling* con *pool_size* igual a 2.

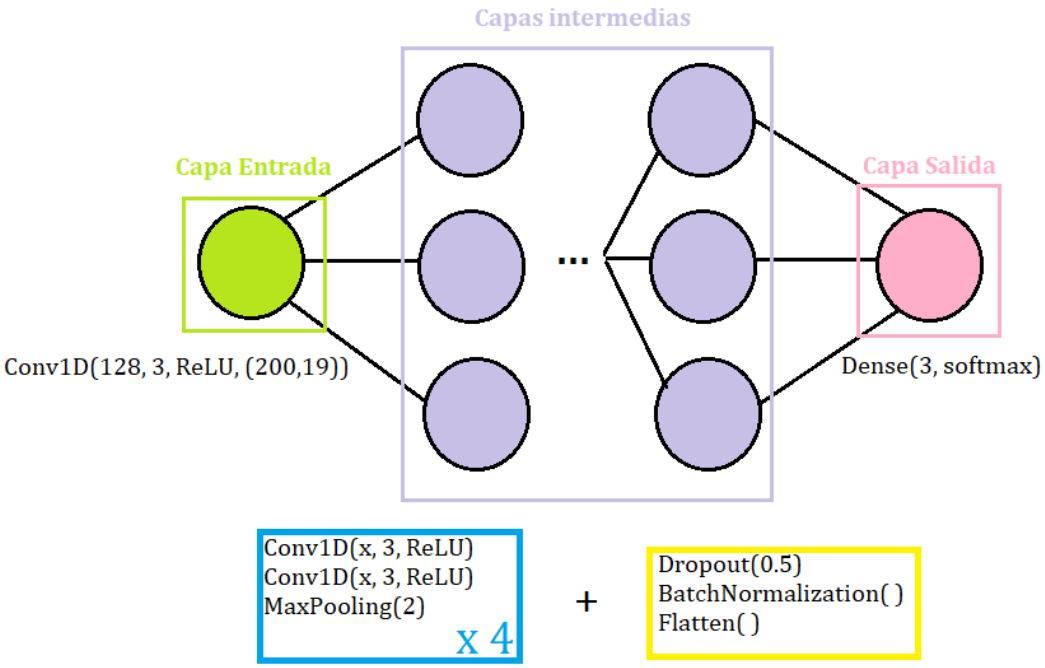

Figura 53: Arquitectura del modelo 3 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH.

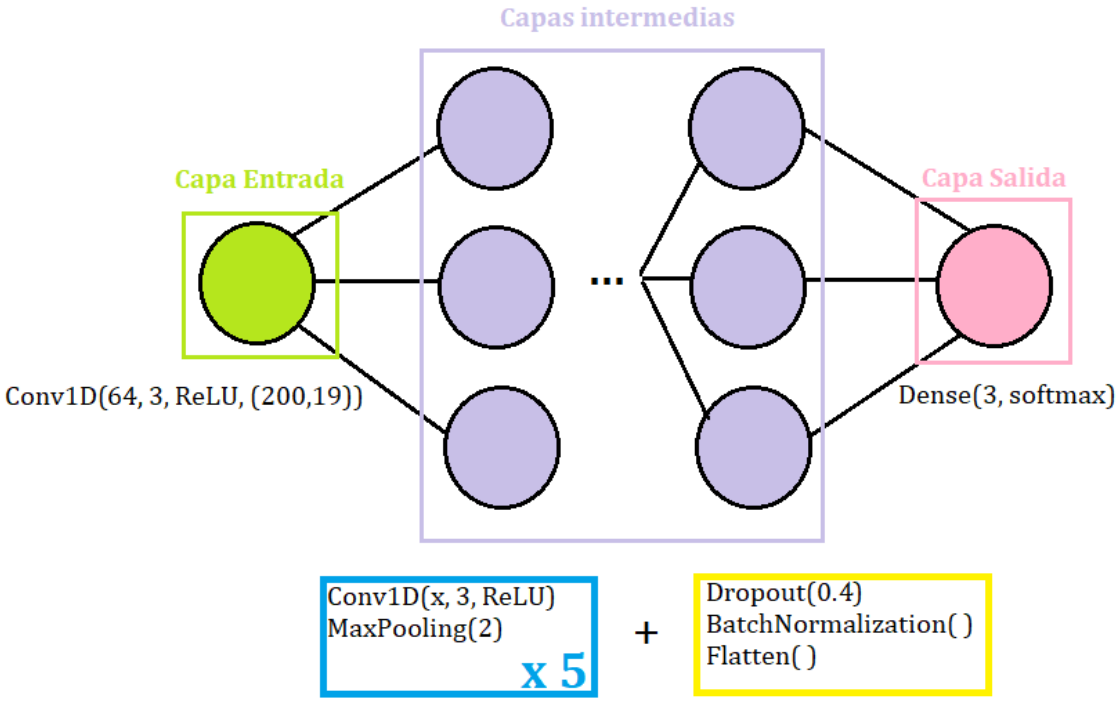

Figura 54: Arquitectura de los modelos 8 y 9 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH.

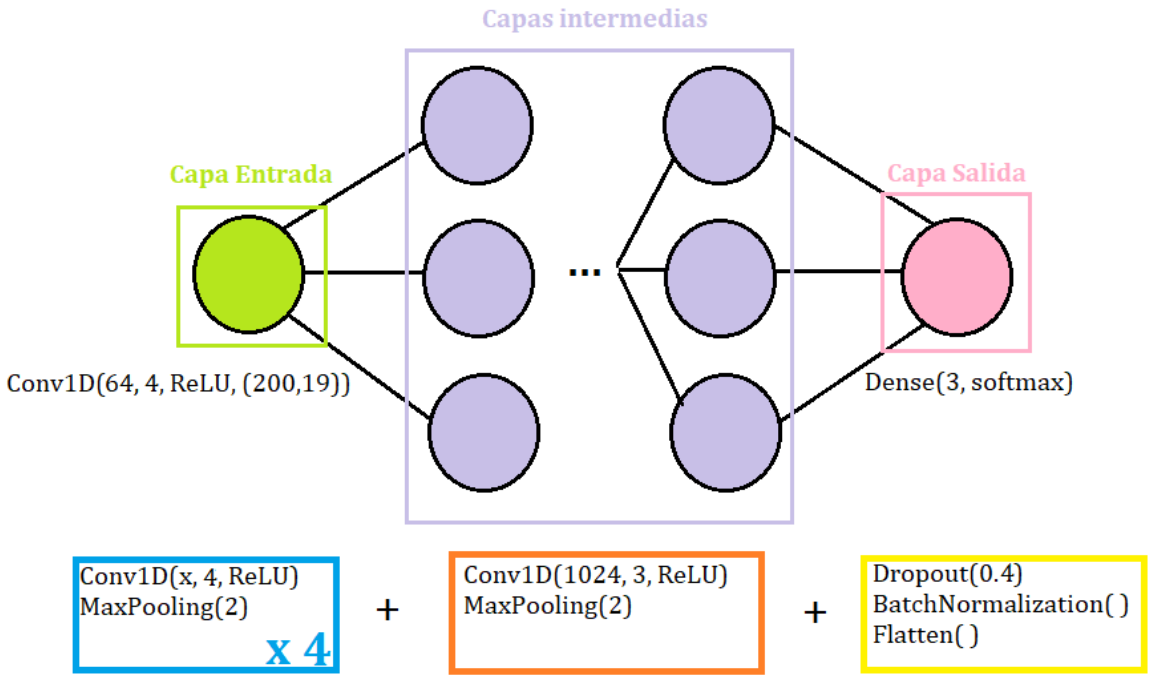

Figura 55: Arquitectura de los modelos 10 y 11 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del HURH.

Por último, se añade un último bloque formado por las siguientes capas: *Dropout* con factor 0.4, *BatchNormalization* y *Flatten*, que adaptará las dimensiones de este bloque a la capa de salida. Esta última capa, será del tipo *Dense* y estará formada por 3 unidades con una función de activación *Softmax*. El entrenamiento de estos modelos se lleva a cabo tras 86 épocas en el caso del modelo 10 y 73 en el del modelo 11. En ambos casos, se ha fijado el número de épocas a 150 pero debido al uso de *earlyStopping* y *ReduceLROnPlateau* el modelo finaliza el entrenamiento sin completar las épocas fijadas, obteniendo los resultados expuestos en la Figura 56.

A continuación, la Tabla 29 muestra un resumen con los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase (Control, DCL y EA) y el coeficiente *kappa* obtenido para cada uno de los modelos desarrollados. Puede observarse que todos los valores del coeficiente *kappa* están comprendidos alrededor de 0,10, siendo el valor más elevado 0,12 obtenido por el modelo 3.

	Entrenamiento	**Validación**	**Test**	**Control**	**DCL**	**EA**	**kappa**
Modelo 3	99,27	98,58	41	47,83	71,43	42,86	0,12
Modelo 8	100	100	37,73	54,54	50	31,71	0,10
Modelo 9	100	100	39,16	50	53,85	30,30	0,10
Modelo 10	100	100	38,69	43,48	50	32	0,11
Modelo 11	99,99	100	38	40,90	54,54	32	0,10

Tabla 29: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del HURH.

En lo referente a la precisión por clases, en este caso ocurre lo opuesto al caso de cinco segmentos, la clase que peor precisión obtiene es EA, seguida de DCL y, por último, son los controles los que se clasifican con mayor precisión.

En este caso de estudio, el porcentaje de precisión para el conjunto de datos de entrenamiento y el de validación, coincide en la mayoría de los casos con un valor del 100%. En la gráfica de líneas no se observa correctamente por lo que, en la Figura 56, se ha añadido una gráfica de barras con el porcentaje de precisión correspondiente a cada uno de los conjuntos de datos utilizados: entrenamiento, validación y test. En la Figura 56 se observa que la diferencia entre los conjuntos de entrenamiento y validación es prácticamente nula en la mayoría de los casos. Sin embargo, la diferencia entre estos dos conjuntos y el de test es elevada. La mayor diferencia de un 62,3% en el modelo 8, y la menor, de un 57,6% en el modelo 3. Puede observarse una pequeña evolución en los resultados de precisión. El mínimo valor de esta métrica en el conjunto de datos de test es el que presenta el modelo 8, un 37,7%. El máximo valor de precisión es el alcanzado por el modelo 3, un 41%. La diferencia entre estos dos valores no es significativa, ya que todos los modelos

tienen aproximadamente la misma precisión en los datos de test. Si se compara con el caso de segmentos de 5 s, el valor de esta métrica se mantiene. Por lo tanto, utilizar segmentos de 1 s para aumentar los datos de entrenamiento y poder así evitar la aparición de *overfitting* no mejora los resultados obtenidos y no elimina este problema.

Al no obtenerse mejores resultados en el problema de clasificación binaria, se descarta aplicar CAR y *z-score* a todos los modelos. Se ha probado a utilizar estas técnicas en dos modelos, uno para cada caso de estudio. El modelo elegido para el caso de segmentos de 5 s ha sido el modelo 9 cuya precisión desciende del 40% obtenido sin estas técnicas al 39,85%. En este caso, el comportamiento del modelo se mantiene, ya que la precisión apenas desciende un 0,15%. Para el caso de segmentos de 1 s, se ha elegido el modelo 3 que presentaba una precisión del 41% sin aplicar CAR ni *z-score*. Esta métrica disminuye hasta un 34,63%, un descenso considerable por lo que en este caso se descarta utilizar estas técnicas de preprocesado.

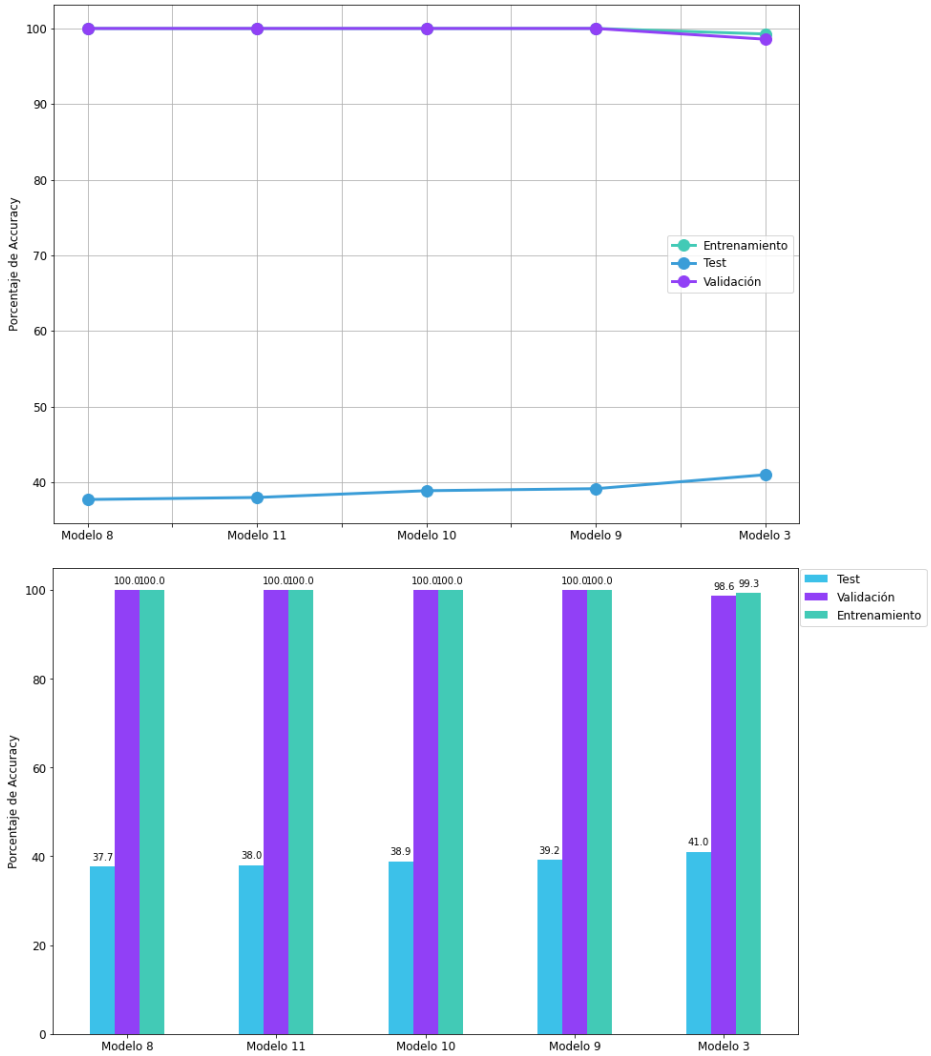

Figura 56: Porcentaje de precisión obtenido por los modelos para el caso de segmentos de1 s en el problema de clasificación múltiple en la base de datos del HURH.

Para el caso de segmentos de 1 s, el modelo que presenta una mayor precisión y, por lo tanto, el que mejores resultados de clasificación obtiene es el modelo 3 con un 41%. Para este caso, se utilizan el mismo número de sujetos que para el caso anterior, 58 sujetos: 15 sujetos de control, 26 pacientes con DCL y 17 enfermos con EA. Los resultados de clasificación por sujetos obtenidos por los distintos modelos son los presentados a continuación.

La Tabla 30 muestra la matriz de confusión con la clasificación por sujetos para el modelo 3, el modelo que presenta los mejores resultados de clasificación. Este modelo clasifica correctamente 11 sujetos de control, 10 pacientes con DCL y 9 enfermos con EA. La clase que mejor clasifica es la de los controles, clasificando el 73,33% de los sujetos correctamente; mientras que la clase que peores resultados obtiene es la de los pacientes con DCL, que clasifica correctamente el 38,46% de estos sujetos.

Real \ Predicho	Control	DCL	EA
Control	11	1	3
DCL	7	10	9
EA	5	3	9

Tabla 30: Matriz de confusión obtenida por el modelo 3 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH.

El modelo 8, clasifica correctamente el 76,47% (13/17) de los enfermos con EA siendo esta la clase que mejores resultados obtiene. Tal y como ocurría en el modelo anterior, la clase que peores resultados obtiene es la de los pacientes con DCL, con un 11,53% (3/26) de los sujetos clasificados correctamente. Comparado con el modelo 3, los resultados para la clase de los enfermos con EA son un poco mejores pero el rendimiento que obtiene el modelo para la clase de los pacientes con DCL disminuye de manera considerable.

Real \ Predicho	Control	DCL	EA
Control	6	1	8
DCL	3	3	20
EA	2	2	13

Tabla 31: Matriz de confusión obtenida por el modelo 8 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH.

En el caso del modelo 9, la clase de los enfermos con EA es la que mejores resultados presenta pues clasifica correctamente el 58,82% (10/17) de los sujetos pertenecientes a esta clase. El 40% (6/15) de los sujetos de la clase de los controles es clasificado correctamente, por lo que esta será la clase que mejores resultados obtenga tras la clase de los enfermos con EA. Por último, y siguiendo la línea de los modelos anteriores, la clase que peores resultados obtiene es la de los pacientes con DCL, con un porcentaje de clasificación correcta del 11,54% (7/26).

Real \ Predicho	Control	DCL	EA
Control	6	1	8
DCL	4	7	15
EA	2	5	10

Tabla 32: Matriz de confusión obtenida por el modelo 9 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH.

Los modelos 10 y 11, cuyos resultados de clasificación por sujetos se muestran en las Tabla 33 y Tabla 34, tienen unos resultados de clasificación correcta similares: 10/9 sujetos de control, 5/6 pacientes con DCL y 8 enfermos con EA. La clase que mejores resultados obtiene en ambos casos es la clase de los controles y la que peor, siguiendo el comportamiento general de los modelos de este caso de estudio, la clase de los pacientes con DCL.

Real \ Predicho	Control	DCL	EA
Control	10	2	3
DCL	7	5	14
EA	6	3	8

Tabla 33: Matriz de confusión obtenida por el modelo 10 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH.

Real \ Predicho	Control	DCL	EA
Control	9	1	5
DCL	8	6	12
EA	5	4	8

Tabla 34: Matriz de confusión obtenida por el modelo 11 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del HURH.

En líneas generales, para este problema de clasificación múltiple, los modelos tienden a clasificar a los sujetos como enfermos con EA. Esto puede verse en las matrices de confusión representadas en las tablas anteriores. En la mayoría de los modelos, cuando un sujeto de control o un paciente con DCL se clasifica de forma errónea, tiende a clasificarse como un enfermo con EA en la mayoría de los casos. Por otra parte, cuando los enfermos con EA se clasifican de manera incorrecta, la mayoría de los modelos tienden a clasificar a estos sujetos como sujetos de control.

5.3.3.3. Resultados obtenidos para la aplicación de *transfer learning*

Tal y como se ha realizado en el resto de escenarios, en este también se han realizado pruebas empleando el paradigma de *transfer learning*. Para modelos entrenados con segmentos de 5 s se ha elegido el modelo 9. Este modelo, cuyos resultados se muestran en la Figura 52, obtiene una *precisión* del 40%. Al aplicar

este modelo sobre los datos de la base de datos POCTEP, la *precisión* se reduce hasta el 31,31% por lo que, en este caso, utilizar *transfer learning* se descartaría. En cuanto a los modelos entrenados con segmentos de 1 s, el modelo elegido ha sido el 3, que es el que mejor *precisión* obtiene para este caso, un 41%. Tras aplicar *transfer learning* y cargar los datos de la base de datos del proyecto POCTEP en este modelo, se obtiene una *precisión* muy inferior a la obtenida inicialmente, un 30,18%. Este descenso hace que se descarte la aplicación de *transfer learning* también para este caso.

5.3.4 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del proyecto POCTEP

5.3.4.1. Resultados obtenidos para segmentos de cinco segundos

Para el caso de segmentos de 5 s los resultados son ligeramente mejores que para el caso de segmentos de 1 s. Si se compara con los resultados obtenidos para la otra base de datos, los resultados se mantienen; sin embargo, si se compara con el problema de clasificación binaria, los resultados empeoran aproximadamente en un 30%, pasando de, aproximadamente, el 70% al 40%.

Para este caso, se exponen los resultados de los cuatro modelos que mejor precisión obtienen sobre el conjunto de datos de test. La estructura de estos cuatro modelos es la mostrada en las Figuras 57-60. Todos los modelos utilizados emplean la función de activación *Softmax* y la función de pérdida *categorical_crossentropy* por tratarse de un caso de clasificación múltiple. Se emplean algoritmos de regularización como *ReduceLROnPlateau* y *earlyStopping* para intentar reducir el *overfitting* presente en el entrenamiento.

A continuación, la Tabla 35 muestra un resumen con los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase (Control, DCL y EA) y el coeficiente *kappa* obtenido para cada uno de los modelos desarrollados. Puede observarse que los valores del coeficiente *kappa* están comprendidos en un rango entre 0,07 (modelo 3) y 0,18 (modelo 4), lo que indica que los resultados obtenidos por los distintos modelos son pobres.

	Entrenamiento	Validación	Test	Control	DCL	EA	kappa
Modelo 1	100	95,46	41,43	69,23	28,57	41,81	0,13
Modelo 2	94,97	83,10	38,79	48,28	16,67	37,5	0,08
Modelo 3	94,97	83,10	38,79	48,28	16,67	37,5	0,07
Modelo 4	99,95	99,50	45,39	54,54	50	44,68	0,18
Modelo 5	100	99,89	41,52	47,06	0	40	0,17

Tabla 35: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 5 s en la base de datos del proyecto POCTEP.

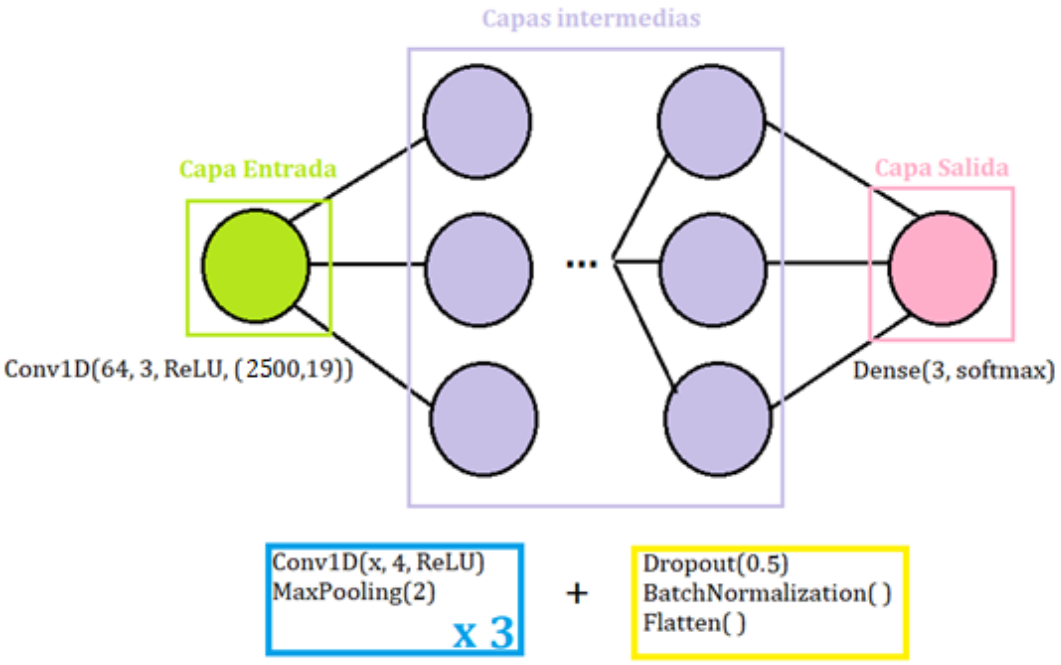

Figura 57: Arquitectura del modelo 1 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

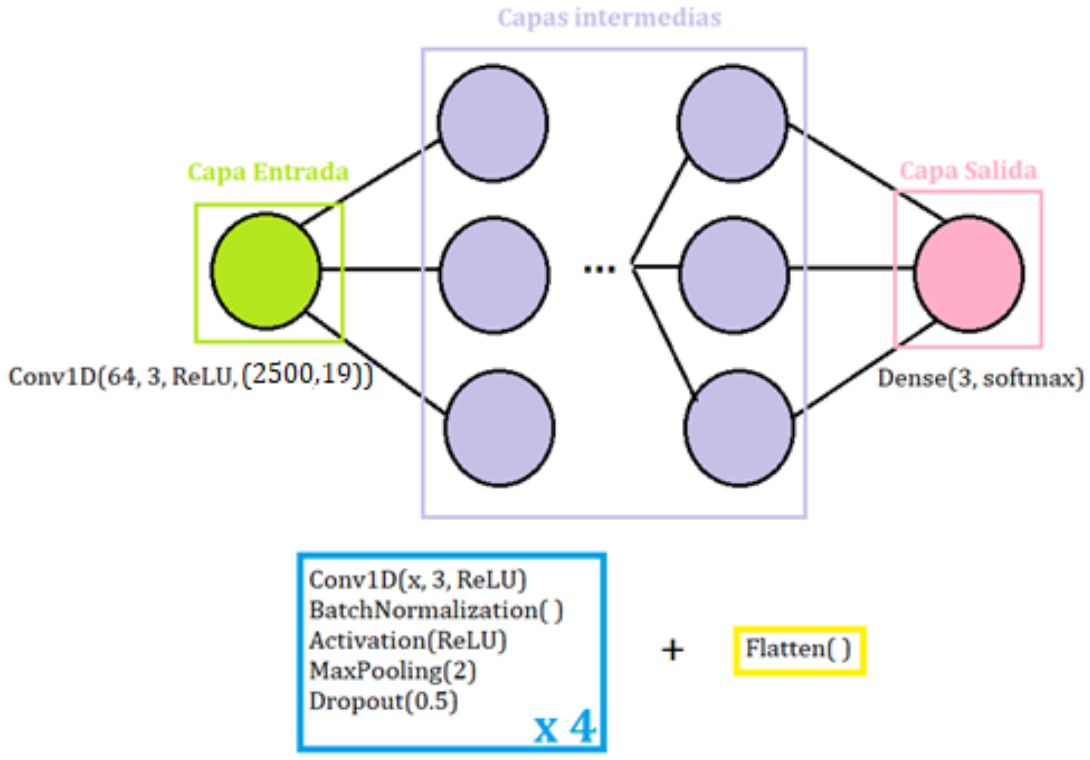

Figura 58: Arquitectura del modelo 2 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

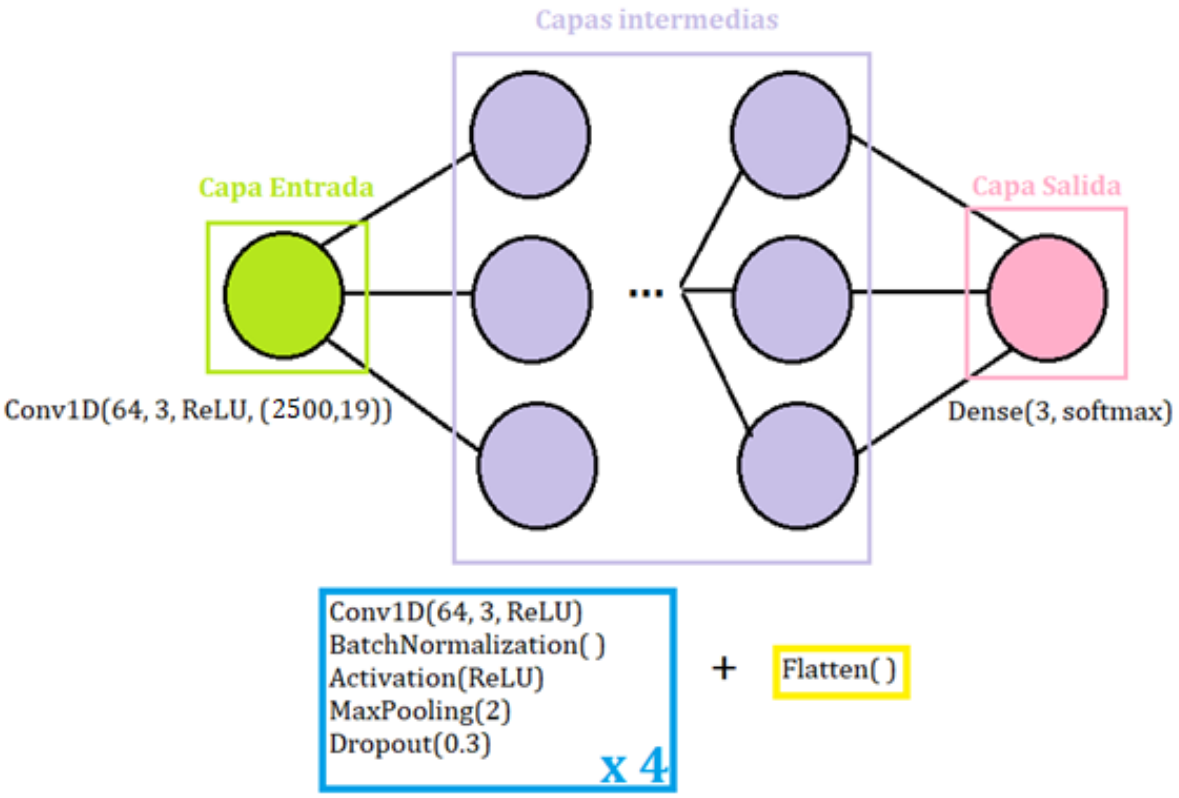

Figura 59: Arquitectura del modelo 3 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

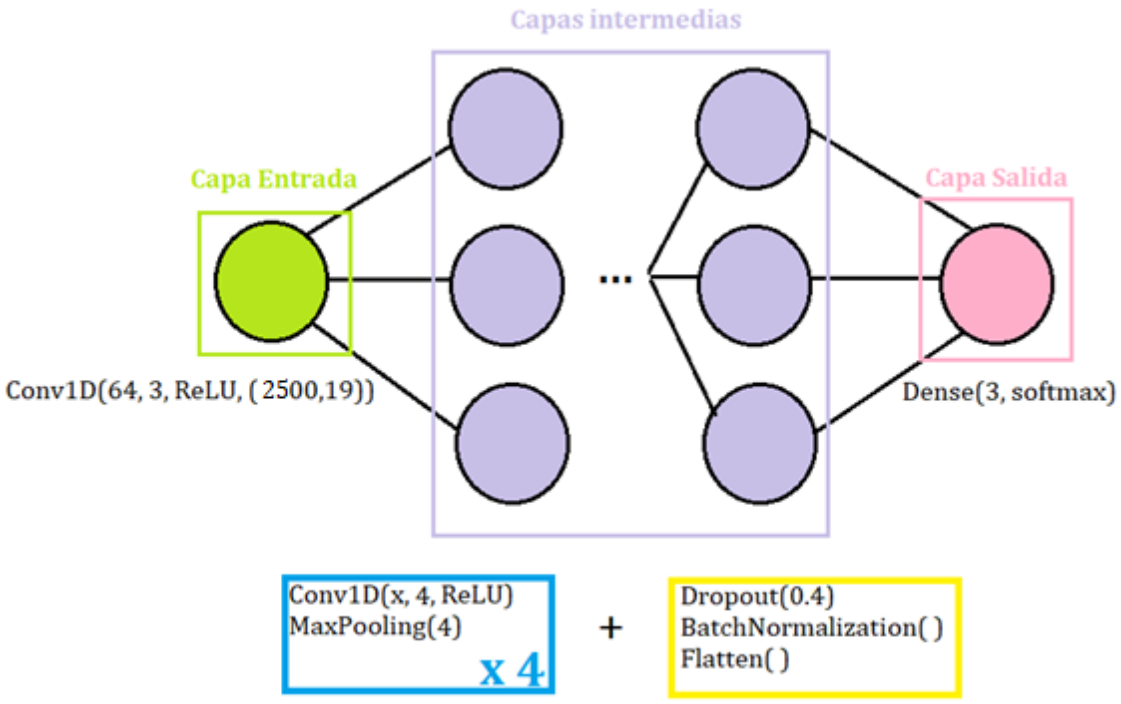

Figura 60: Arquitectura del modelo 4 utilizado en el problema de clasificación múltiple para los casos de segmentos de 5 s en la base de datos del proyecto POCTEP.

En lo referente a la precisión por clases, en este caso la clase DCL es la que peores resultados obtiene, llegando a obtener un 0% de precisión en el modelo 5. La clase con mayor precisión es control y le sigue la clase EA. La precisión de esta última clase disminuye debido a la clasificación errónea de sujetos de las otras dos clases que, de manera general, tienden a clasificar a los sujetos como sujetos EA. Aunque la precisión de la clase EA sea menor que la de control, esta clase es la que mejores resultados de clasificación presenta.

En este caso, en la se observa que existen diferencias entre el conjunto de entrenamiento y el de validación para los modelos 2, 3 y 1. También existen diferencias entre los conjuntos de validación y test. La máxima precisión obtenida es del 45,39%, alcanzada por el modelo 4.

Para testear los distintos modelos utilizados en este problema de clasificación múltiple para esta base de datos se han utilizado 75 sujetos. Estos 75 sujetos se dividen en: 26 sujetos de control, 25 pacientes con DCL y 24 enfermos con EA.

El modelo 1 consigue clasificar correctamente 9 sujetos de control, 2 pacientes con DCL y 23 enfermos con EA (ver). La clase que mejores resultados obtiene y que destaca sobre el resto, es la clase de los enfermos con EA, con un 95,83% de los sujetos clasificados correctamente. La clase que peores resultados obtiene es la de los pacientes con DCL, ya que el modelo clasifica correctamente tan sólo el 8% de los sujetos pertenecientes a esta clase.

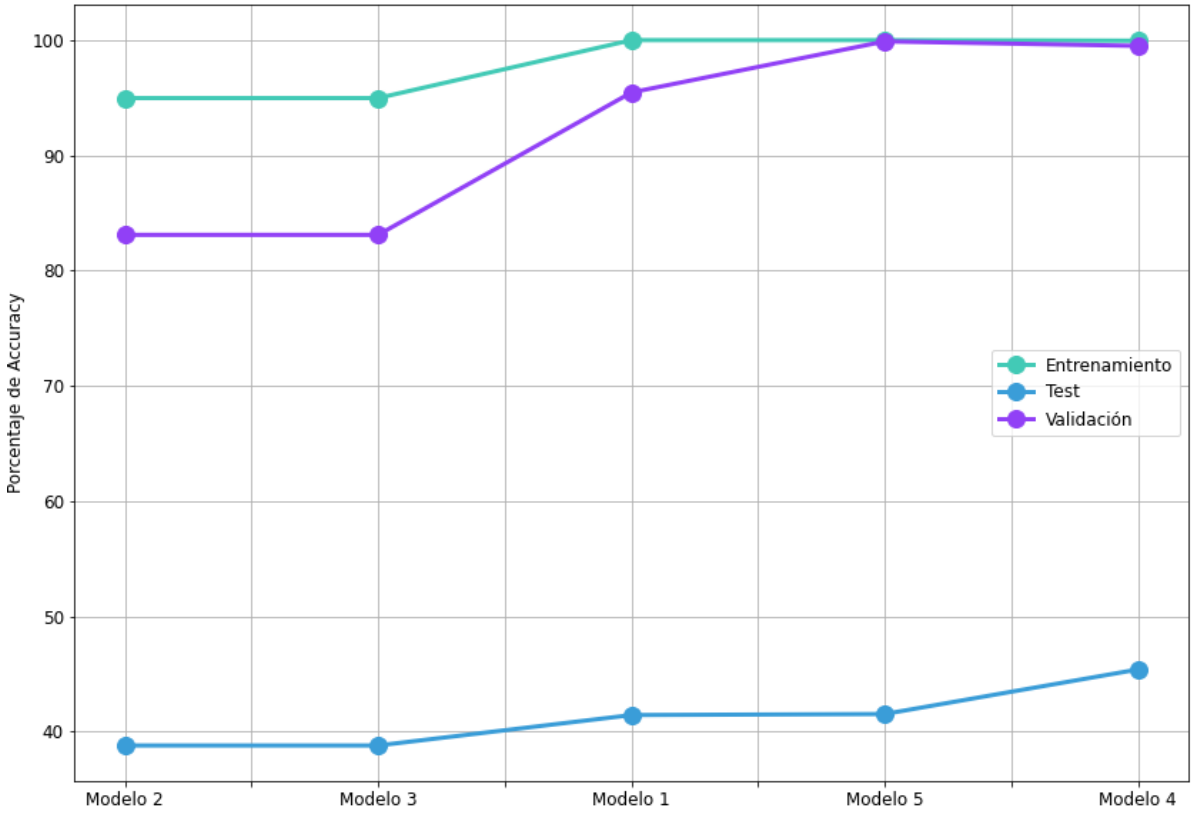

Figura 61: Porcentaje de *precisión* obtenido por los modelos para el caso de segmentos de 5 s en el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

Real \ Predicho	Control	DCL	EA
Control	9	4	13
DCL	4	2	19
EA	0	1	23

Tabla 36: Matriz de confusión obtenida por el modelo 1 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

Los resultados del modelo 4 se muestran en la. Los modelos 2 y 3 (ver Tabla 37) presentan exactamente los mismos resultados de clasificación de sujetos, pues como ya se ha explicado en este apartado, se trata del mismo modelo pero se emplean diferentes algoritmos de optimización que consiguen el mismo resultado. Estos modelos clasificarán correctamente el 62,5% de los enfermos con EA, seguido de un 53,84% de los sujetos de control y únicamente un 4% de los sujetos pertenecientes a la clase de pacientes con DCL.

Real \ Predicho	Control	DCL	EA
Control	14	3	9
DCL	8	1	16
EA	7	2	15

Tabla 37: Matriz de confusión obtenida por los modelos 2 y 3 entrenados con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

El modelo 4 (ver Tabla 38) es el que mejor *precisión* obtiene y, por ello, será el que mejores resultados de clasificación por sujetos presente. La clase que mejores resultados obtiene es la de los enfermos con EA, con un 87,50% de los sujetos clasificados correctamente. La que peores resultados obtiene es la de los pacientes con DCL, que sigue la tendencia de los modelos anteriores clasificando correctamente un 12% de los sujetos pertenecientes a esta clase.

Real \ Predicho	Control	DCL	EA
Control	12	3	11
DCL	7	3	15
EA	3	0	21

Tabla 38: Matriz de confusión obtenida por el modelo 4 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

Por último, el modelo 5 (ver Tabla 39) presenta unos resultados bastante buenos para la clase de enfermos con EA, donde un 91,66% de los sujetos pertenecientes a esta clase son clasificados de manera correcta. Sin embargo, los resultados para la clase de pacientes con DCL son malos, pues se consigue clasificar correctamente un 0% de los sujetos. En cuanto a la clase de los controles, tampoco son buenos ya que sólo el 30,76% de los sujetos se clasifican correctamente.

Real \ Predicho	Control	DCL	EA
Control	8	3	15
DCL	7	0	18
EA	2	0	22

Tabla 39: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 5 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

5.3.4.2. Resultados obtenidos para segmentos de un segundo

En el caso de segmentos de 1 s, se muestran los resultados para los tres modelos que mejor precisión presentan en el conjunto de datos de test. La estructura de estos modelos es la que se presenta en las Figuras 62 y 63. Todos los modelos utilizan las mismas funciones tanto de pérdida como de activación que emplean los expuestos en el caso de segmentos de 5 s, *Softmax* y *categorical_crossentropy*. También emplean los mismos algoritmos de regulación: *earlyStopping* y *ReduceLROnPlateau*.

El modelo 5 y el 6 siguen la misma estructura. Tienen como capa de entrada una *Conv1D* con 64 filtros, un tamaño de filtro de 3 y una función de activación ReLU. Las dimensiones de los datos de entrada al modelo son (1000,19). En cuanto a las capas intermedias, están formadas por bloques de varios tipos. El primer tipo, es un bloque formado por una capa *Conv1D* y otra *MaxPooling*. El segundo tipo es un bloque formado por dos capas de tipo *Conv1D* seguidas por una capa *MaxPooling*. El tercer tipo de bloque sigue una estructura de tres capas *Conv1D* seguidas de otra *MaxPooling*. El cuarto y último tipo de bloque, está formado por las capas *Dropout* con un factor 0.5, *BatchNormalization* y *Flatten*. En total se utilizan dos bloques del primer tipo, tres del segundo y uno del tercero. Las capas *Conv1D* de todos los bloques tienen un tamaño de *kernel* igual a 3 y una función de activación ReLU. El factor de *MaxPooling* utilizado en todas las capas de este tipo es de 2.

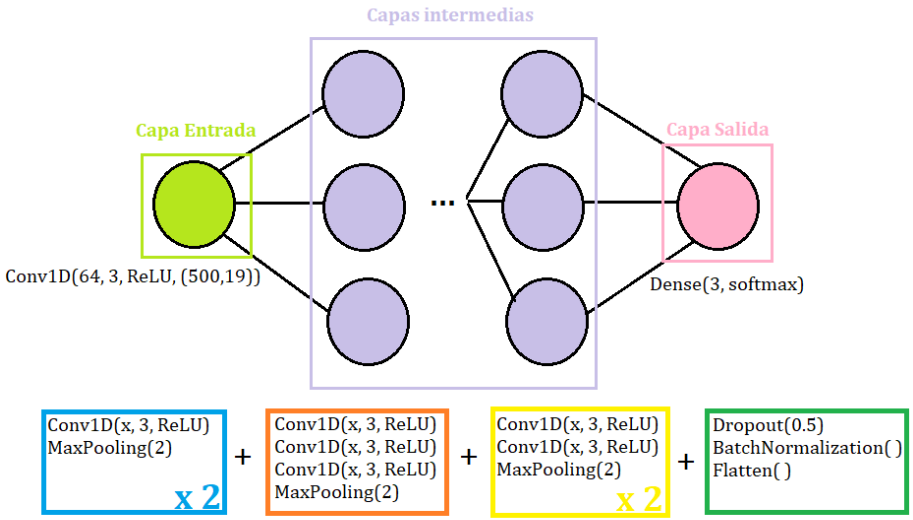

Figura 62: Arquitectura de los modelos 5 y 6 utilizados en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP.

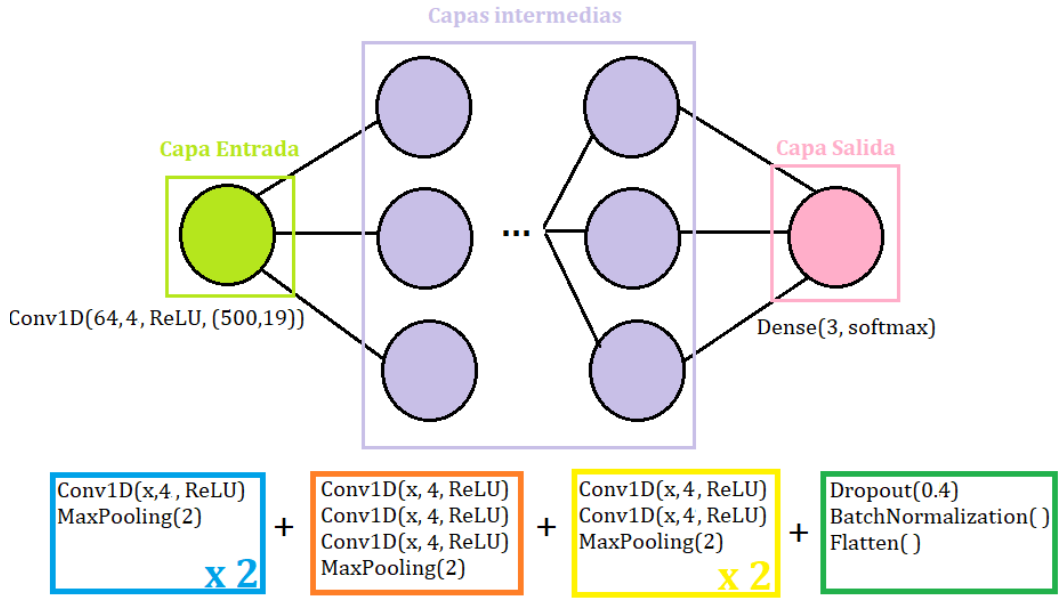

Figura 63: Arquitectura del modelo 7 utilizado en el problema de clasificación múltiple para los casos de segmentos de 1 s en la base de datos del proyecto POCTEP.

El modelo 5 utiliza el optimizador SGD con una *learning rate* igual a 0.01. En cuanto al entrenamiento, se fijan las épocas a 160 pero para el entrenamiento para a las 130 debido al uso de *earlyStopping*. El modelo 6, utiliza el optimizador Adam con una *learning rate* de 0.001. El entrenamiento de este modelo se lleva a cabo durante 130 épocas, aunque el valor fijado es de 250.

El modelo 7, sigue la estructura que siguen los modelos 5 y 6. La diferencia principal con estos dos modelos es que el tamaño de *kernel* de todas las capas *Conv1D* es de 4 en lugar de 3. También cambia el factor de *dropout* que baja de 0.5 a 0.4. El entrenamiento del modelo se lleva a cabo en 135 épocas, aunque el valor fijado originalmente es de 250 épocas.

A continuación, la muestra un resumen con los resultados obtenidos para cada modelo. En ella se incluye, además de la precisión para conjunto, la precisión por clase y el coeficiente *kappa* obtenido para cada uno de los modelos desarrollados. Puede observarse que los valores de *kappa* son muy próximos a cero lo que implica que los resultados son pobres y que pueden deberse al azar.

	Entrenamiento	Validación	Test	Control	DCL	EA	kappa
Modelo 5	99,96	100	36,55	69,23	28,57	41,81	0,06
Modelo 6	100	100	35,55	48,28	16,67	37,5	0,04
Modelo 7	100	100	37,07	48,28	16,67	37,5	0,07

Tabla 40: Resumen de resultados en tanto por ciento (%) para modelos entrenados con segmentos de 1 s en la base de datos del proyecto POCTEP.

Los resultados mostrados en la Figura 64 no muestran diferencias entre los conjuntos de validación y entrenamiento, ambos tienen una precisión del 100%. En el caso del conjunto de datos de test, la máxima precisión obtenida es del 37,10% y es alcanzada por el modelo 7. La mínima es de 31,6%, obtenida por el modelo 6. Si se comparan estos resultados con los obtenidos empleando segmentos de 5 s, se pasa de una precisión del 45,39% a una del 37,10% por lo que utilizar segmentos de 1 s no hace que los resultados mejoren, en este caso, empeoran. Si se hace una comparación entre los resultados obtenidos por esta base de datos y los resultados obtenidos por la base de datos del HURH, los resultados mejoran ligeramente en el caso de segmentos de 5 s.

En cuanto al empleo de las técnicas CAR y *z-score*, no se ha empleado en todos los modelos, ya que, como ocurre en la otra base de datos, los resultados no mejoran.

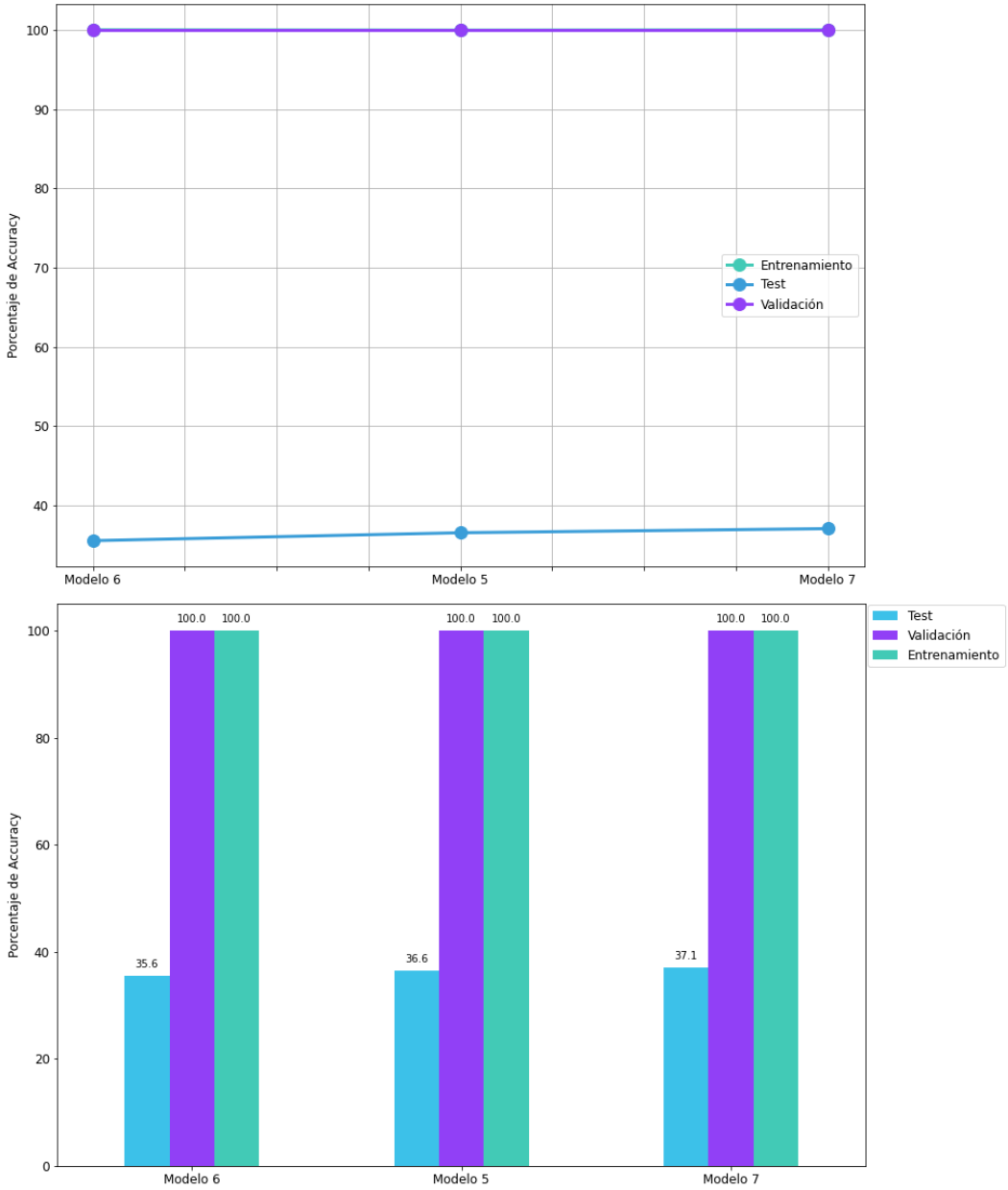

Figura 64: Porcentaje de *precisión* obtenido por los modelos para el caso de segmentos de 1 s en el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

En esta ocasión, se han reentrenado dos modelos utilizando estas dos técnicas: modelo 4, empleado en el caso de segmentos de 5 s; y modelo 5, empleado en el caso de segmentos de 1 s. El modelo 4 reentrenado utilizando CAR y *z-score* obtiene una precisión del 40,54%. Este resultado es menor que el obtenido por el mismo modelo sin estas técnicas de preprocesado, por lo que pasará de un 45,39% a un 40,54%. En el caso del modelo 5, la precisión obtenida es ligeramente superior a la obtenida por el mismo modelo si emplear las técnicas de preprocesado. El modelo con estas técnicas obtiene un 37,27% frente al 36,60% obtenido sin ellas. El objetivo principal de estas técnicas era comprobar si el resultado de aplicar *transfer learning* mejoraba los resultados obtenidos por los modelos. Al no conseguirse aumentar el rendimiento del modelo obteniendo resultados peores a los obtenidos por los modelos que no utilizan estas técnicas, se decide no continuar haciendo este tipo de pruebas.

Para el caso de segmentos de 1 s (ver Tablas 41-43), los resultados de las clases de los pacientes con DCL y de los controles empeoran de manera considerable obteniendo la primera un 0% de sujetos clasificados correctamente en todos los modelos utilizados. En el caso de la clase de los controles, el porcentaje de sujetos clasificados de manera correcta es del 7,69% para los modelos 5 y 6 y del 11,53% en el modelo 7. La clase de los enfermos con EA, sin embargo, obtiene unos resultados de clasificación excelentes; los modelos 5 y 7 son capaces de clasificar correctamente al 100% de los sujetos pertenecientes a esta clase y el modelo 6 alcanza el 95,83%.

De manera general, estos tres modelos obtendrán malos resultados, aunque clasifiquen correctamente en torno al 100% de los sujetos de la clase EA. En líneas generales, y más concretamente en el caso de los segmentos de 1 s, todos los modelos tienden a clasificar a los sujetos como enfermos con EA. Esto ya ocurría en la base de datos del HURH, pero se produce de manera más notable en los resultados obtenidos para esta base de datos.

Real \ Predicho	Control	DCL	EA
Control	2	1	23
DCL	1	0	24
EA	0	0	24

Tabla 41: Matriz de confusión obtenida por el modelo 5 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

Real \ Predicho	Control	DCL	EA
Control	2	2	22
DCL	2	0	23
EA	1	0	23

Tabla 42: Matriz de confusión obtenida por el modelo 6 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

Real \ Predicho	Control	DCL	EA
Control	3	1	22
DCL	1	0	24
EA	0	0	24

Tabla 43: Matriz de confusión obtenida por el modelo 7 entrenado con segmentos de 1 s para el problema de clasificación múltiple en la base de datos del proyecto POCTEP.

5.3.4.3. Resultados obtenidos para la aplicación de *transfer learning*

Tras descartar seguir aplicando CAR y *z-score*, se aplica *transfer learning* con varios modelos: el modelo 4 para el caso de segmentos de 5 s, y el 7, para el caso de segmentos de 1 s.

El modelo 4, entrenado con segmentos de cinco segundos, se aplica sobre los datos de la base de datos del Hospital Universitario Río Hortega. Los resultados de *precisión* obtenidos son del 44,25%. Para este modelo, esta métrica desciende ligeramente de un 45,39% al 44,25% mencionado por lo que los resultados obtenidos aplicando *transfer learning* no son malos, son los esperados pues el modelo mantiene su funcionamiento.

Ocurre lo contrario con el modelo 7, en este caso la *precisión* obtenida tras aplicar este paradigma desciende de un 37,1% a un 33,37%. En este caso, desciende casi aproximadamente un 4% y en estos niveles de *precisión*, esta diferencia puede ser considerable.

En este caso de estudio, aplicar *transfer learning* sería posible en el caso de modelos entrenados con segmentos de cinco segundos. Se descarta en el caso de modelos entrenados con segmentos de un segundo por la disminución de *precisión* obtenida.

Capítulo 6

6. Discusión

6.1 Introducción 111
6.2 Problema de clasificación binaria 112
6.3 Problema de clasificación múltiple 114
6.4 Comparativa con otros estudios de la literatura 116
6.5 Limitaciones 118

6.1. Introducción

La clasificación automática de sujetos pertenecientes a las clases de sujetos de control, pacientes con DCL y enfermos con EA se llevó a cabo mediante la evaluación de distintas arquitecturas CNN. Primero, se consideraron sólo dos clases: controles y enfermos; es decir, se planteó un problema de clasificación binaria. Tras implementar y entrenar distintos modelos, se consiguió como máxima precisión posible para este problema, aproximadamente el 70-75% para las dos bases de datos analizadas en el TFG. Al alcanzar este valor máximo, se decidió incrementar el número de clases pasando así a un problema de clasificación múltiple con tres clases: controles, pacientes con DCL y enfermos con EA. Para este problema se implementaron diferentes modelos y se consiguió una máxima precisión de en torno al 40%; esto confirma que, al incrementar el número de clases, esta métrica desciende de manera considerable.

Para cada problema de clasificación, se comienza entrenando los modelos sin utilizar algoritmos de regularización. Al aparecer *overfitting*, se decide introducir tanto de capas de tipo *Dropout* como algoritmos de regularización como *reduceLROnPlateau* para intentar eliminar o reducir este problema. En algunos modelos, este problema se reduce, mientras que en otros persiste; esto hace que el modelo no generalice bien sobre el conjunto de datos de test, pero encajando casi perfectamente en los conjuntos de validación y test. También se ha utilizado *earlyStopping* para intentar que el entrenamiento finalizase antes de llegar al número de épocas fijado.

Utilizar algoritmos como *reduceLROnPlateau* y *earlyStopping* para reducir el *overfitting* hacen que el entrenamiento de los modelos se lleve a cabo durante menos épocas de las fijadas en el modelo. Ambos algoritmos monitorizan el valor de la pérdida en el conjunto de datos de validación. El primero hará que el entrenamiento pare cuando se supere el mínimo valor de *learning rate* fijado. El segundo, parará cuando el valor de la pérdida en el conjunto de validación pase de un mínimo fijado. Este algoritmo tendrá activada la opción de restaurar los pesos de la época con mejores resultados en todos los modelos implementados.

Al implementar estos algoritmos de regularización, se consigue obtener unos resultados de aproximadamente el 100% de precisión para los conjuntos de datos de entrenamiento y validación. Esto implica que los diferentes modelos son capaces de aprender a clasificar a los sujetos de manera casi perfecta en número de épocas reducido. Al aprender las características de cada clase de forma casi perfecta, generaliza mal al introducir unos datos diferentes a los existentes en los conjuntos de entrenamiento y validación.

Con el fin de intentar reducir el *overfitting* y mejorar los resultados, se probó a entrenar el modelo utilizando como conjuntos de datos de entrenamiento y validación segmentos pertenecientes a usuarios únicos. Esto implica que los datos no se dividen de forma aleatoria, sino que se escogen los segmentos de forma que todos los segmentos que se utilicen pertenezcan a un único sujeto y no a usuarios distintos. El resultado obtenido no fue bueno, los modelos no fueron capaces de

entrenar de forma correcta y obtuvieron unas precisiones muy bajas tanto para validación como para test.

6.2. Problema de clasificación binaria

Los resultados para este problema de clasificación binaria se resumen en cuatro tablas: Tabla 4, Tabla 9, Tabla 14 y Tabla 19. Las dos primeras tablas muestran los resultados pertenecientes a la base de datos HURH, las últimas, los obtenidos para la base de datos de proyecto POCETP. En general, en estas tablas se observa la diferencia de precisión existentes entre validación y test que implican la existencia de *overfitting*. Este problema ha intentado solucionarse utilizando, además de los algoritmos de regularización, segmentos de 1 s con la intención de tener más datos con los que entrenar y testear el modelo. Los resultados obtenidos para este caso en general no han mejorado, pues sólo se ha conseguido evitar el *overfitting* en los modelos 5 y 6 de la base de datos del proyecto POCTEP.

El hecho de que no se reduzca el *overfitting* empleando técnicas de regularización y aumentando la cantidad de datos en el entrenamiento, refleja que será una característica intrínseca de los datos la que lo produzca. El *overfitting* presente en estos modelos se deberá entonces a la complejidad de los propios datos. Una señal EEG en crudo recogido para cada una de las clases es una señal muy compleja lo que hace más difícil la extracción de características por parte de la CNN. Como se verá en el apartado comparativa, aplicar transformaciones a esta señal ayuda a que se reduzca la complejidad mejorando los resultados.

El modelo que mejores resultados de precisión obtiene en este problema de clasificación binaria es el modelo 6. Este modelo está entrenado con segmentos de 5 s extraídos de la base de datos del HURH y presenta un valor de precisión del 76,90%. Comparado con los modelos utilizados para el mismo problema y la misma base de datos, su arquitectura (Figura 33) es lo que hace que tenga un mejor rendimiento. Para construir este modelo, se combinan las arquitecturas de los modelos 3 y 5 probados para este caso, sin cambiar el valor de parámetros como el tamaño de *kernel,* el factor de *dropout* o el *pool_size*. Mantener el valor de estos parámetros hace evidente que es la propia arquitectura híbrida, más compleja que las utilizadas en los otros modelos, junto con la profundidad de la misma la que hace que mejore la precisión. En general, las arquitecturas con mayor precisión en test son en las que el modelo ha tenido menor *overfitting* en entrenamiento.

Si se comparan los resultados de los casos de segmentos de 5 s, la precisión conseguida por los modelos entrenados con la base de datos del HURH es, de forma general, más alta que la conseguida por los entrenados con la base de datos del proyecto POCTEP. Ocurre lo mismo en el caso de segmentos de 1 s. No hay una gran diferencia entre ellos, pues en ambos casos la precisión del conjunto de datos de test está situada por encima del 60% y por debajo del 77%.

En cuanto a la clasificación por sujetos, de manera general se observa que los modelos utilizados para este problema tienden a aprender a clasificar a los sujetos como enfermos con EA. Esto puede deberse a que existe una gran diferencia entre

el número de sujetos de control y el número de enfermos con EA contenidos en las distintas bases de datos. Si el conjunto de entrenamiento presenta una mayoría de enfermos con EA, entonces va a aprender las características para este grupo mayoritario y no será capaz de aprender las características de la clase de los controles. A continuación, se analizan dos de los modelos desarrollados donde se exponen dos casos: número de sujetos de las distintas clases balanceado en el conjunto de test y no balanceado. Se analizan los modelos 4 y 6, que además son los que mejores resultados obtienen para cada base de datos.

- Caso de número de sujetos no balanceado**.** La Tabla 8contiene la matriz de confusión del modelo 6 entrenado con segmentos de 5 s de la base de datos del HURH. En este caso, existen únicamente 4 sujetos de control en este conjunto de datos de test. Se ha probado a entrenar distintos modelos equilibrando el número de sujetos de control presentes en los datos de entrenamiento y en los de test, pero los resultados obtenidos han sido similares. Este modelo clasifica de manera correcta a un 25% de los sujetos de control, clasificando al 75% restante como enfermos con EA. Sin embargo, clasifica correctamente un 81,48% de los enfermos con EA, 44 de los 54 existentes.
- Caso de número de sujetos balanceado. La Tabla 20 contiene la matriz de confusión del modelo 4 entrenado con segmentos de 1 s de la base de datos del proyecto POCTEP. Clasifica correctamente a 16 de los 37 sujetos de control, un 43,24%, y 33 de los 38 enfermos con EA, un 86,84%. En este caso, el número de sujetos de control y el de enfermos con EA está balanceado en el conjunto de test, pero el problema es que, de los 51 sujetos de control existentes en la base de datos del proyecto POCTEP, 37 se han utilizado para test y 14 se han divido entre entrenamiento y validación. Esto implica que el modelo no será capaz de aprender bien las características de la clase de los controles, pues habrá una mayoría de sujetos pertenecientes a la clase de los enfermos con EA, 164 que se dividen entre validación y test.

Este análisis confirma que, aunque los datos del conjunto de test estén balanceados, al existir un número de sujetos de control y EA tan diferente en las bases de datos el problema comentado anteriormente persiste. Persiste porque el equilibrio en el conjunto de test hace que exista un desequilibrio en validación y entrenamiento.

En cuanto a las técnicas de preprocesado CAR y *z-score*, algunos modelos obtienen iguales o mejores resultados tras su aplicación. Esto no sucede para todos modelos como se observa en las Figuras 34 y 42. Es posible que los resultados tras la aplicación de la normalización no sean buenos debido a las componentes aleatorias de los modelos que influyen en el entrenamiento, como el *dropout* o la inicialización aleatoria de los pesos en la primera época. En general, aplicar CAR y *z-score* no mejora, pero tampoco empeora los resultados de manera considerable. Puede que no mejoren los resultados porque la CNN ya realiza una normalización interna de los datos dentro de la red, y volver a normalizar no aporta nada nuevo.

Por último, en lo referente a la aplicación de *transfer learning*, como ya se ha explicado para cada caso en el Capítulo 5, los resultados obtenidos son malos. Los modelos elegidos para aplicar este paradigma presentan un descenso de la precisión. Este descenso indica que un modelo entrenado con la base de datos del HURH no se adaptará bien a los datos de la base de datos del proyecto POCTEP y viceversa. Si este paradigma se aplica sobre los modelos que han sido entrenados utilizando CAR y *z-score*, los resultados son mucho peores que los obtenidos si no se utilizan dichas técnicas. El peor resultado aparece en el modelo 1 entrenado con segmentos de cinco segundos con la base de datos del proyecto POCTEP, donde se pasa de un 62,44% de precisión a un 8,45%. El hecho de que los resultados empeoren se debe a que el entrenamiento del modelo también es peor, se parte de unos resultados de precisión menores y se obtiene una precisión aún menor al utilizar este paradigma. En general, el paradigma de *transfer learning* funciona peor cuando el modelo entrenado con la base de datos del proyecto POCTEP se valida sobre los datos de la base de datos HURH. Esto puede deberse a la interpolación necesaria para adaptar los datos al formato de los datos de entrada de la red neuronal.

6.3. Problema de clasificación múltiple

Los resultados para este problema son los presentados en las Figura 52, Figura 56, y Figura 64. Las dos primeras figuras muestran los resultados obtenidos para la base de datos del HURH, mientras que las dos últimas representan los resultados obtenidos para la base de datos del proyecto POCTEP. Al aumentar el número de clases, se observa una reducción en la precisión obtenida por los modelos. En este caso, se sigue observando la diferencia entre los datos de validación y test, ya comentada en el problema de clasificación binaria. Los algoritmos y capas de regularización y el uso de segmentos de 1 s no eliminan el *overfitting* por lo que los resultados obtenidos no son buenos. Los resultados de precisión para los conjuntos de datos de test y validación son de, aproximadamente, el 100% en la mayoría de los casos. Esto implica que el modelo será capaz de ajustarse de manera perfecta a los datos de entrenamiento y validación apareciendo así el *overfitting*. Por otra parte, con el conjunto de datos de test ocurrirá lo contrario. Al haberse utilizado en entrenamiento y validación segmentos distintos de los mismos sujetos, el modelo no será capaz de generalizar bien sobre los datos de test. No generaliza correctamente sobre este conjunto de datos porque para crearlo se ha calculado el número de segmentos correspondiente a cada sujeto de manera que no se utilizan segmentos distintos de los mismos sujetos. Tal y como ocurre en el caso anterior, se probó a realizar pruebas creando los conjuntos de entrenamiento y validación de la misma forma que se construyó el de test. Los resultados obtenidos eran malos, la red neuronal no era capaz de entrenar adecuadamente obteniendo una precisión de validación y test muy baja.

En este caso, el modelo que mejor precisión alcanza es el modelo 4, entrenado con segmentos de 5 s con los datos de la base de datos del proyecto POCTEP, con un 45,39%. Es el modelo que mejores resultados obtiene debido a su arquitectura y al valor de algunos parámetros como el factor de *dropout* y el tamaño de *kernel.* En este

caso, utilizar una arquitectura simple y menos profunda ofrece mejores resultados posiblemente debido a que presenta menos *overfitting*. A esta arquitectura deben añadirse los valores de *dropout* y de tamaño de *kernel.* Se aumenta el tamaño de *kernel* hasta llegar a 4 y se disminuye el factor de *dropout* fijado en el resto de modelos un 0,1 bajando así a 0,4 y consiguiendo un aumento de la precisión. *Si* se compara con el mejor resultado obtenido para el problema de clasificación binaria, la precisión se reduce notablemente, baja un 31,51% debido, principalmente, al aumento del número de clases y al desbalanceo de los sujetos existentes en cada una de ellas.

Si se comparan los resultados de los casos de segmentos de 5 s, de manera opuesta al problema de clasificación binaria, la precisión conseguida por los modelos entrenados con la base de datos del proyecto POCTEP es, de forma general, más alta que la conseguida por los entrenados con la base de datos del HURH. No hay una gran diferencia entre ellos pues en ambos casos la precisión del conjunto de datos de test está situada en torno al 40%. Sin embargo, si se comparan los resultados de los casos de segmentos de 1 s, ocurre lo mismo que para el problema de clasificación binaria. Los modelos que mejores resultados presentan en este caso de estudio son los modelos 3 y 7. El modelo 3 entrenado con la base de datos del HURH, y el modelo 4, entrenado con la base de datos del proyecto POCTEP. Alcanzan, respectivamente, un 41% y un 37,07%. En el caso de la base de datos HURH, se observa que una arquitectura simple funciona mejor que una más compleja en la que además se disminuye el valor de *dropout* y se aumenta el número de filtros. En el caso de la base de datos del proyecto POCTEP, se obtendrán mejores resultados aumentando el número de filtros y disminuyendo el factor de *dropout*.

En cuanto a las técnicas de preprocesado CAR y *z-score*, para la base de datos del HURH, se han probado en dos modelos, uno por caso de estudio. Para el caso de estudio de segmentos de 5 s, se ha elegido el modelo 9; mientras que, para el caso de segmentos de 1 s, se elige el modelo 3. En ambos casos la precisión obtenida desciende, muestra una variación pequeña, de entre el 1 y el 7% con respecto a la precisión obtenida por los modelos sin utilizar mencionadas técnicas. Para la base de datos de POCTEP, también se prueban dos modelos, uno por caso de estudio. Los modelos elegidos han sido el 4 y el 5, entrenados con segmentos de 5 y 1 s respectivamente. El resultado obtenido es similar al obtenido para la otra base de datos, la precisión obtenida disminuye.

En cuanto a la clasificación por sujetos, las matrices de confusión expuestas en ***5.3.3 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del Hospital Universitario Río Hortega*** y ***5.3.4 Resultados obtenidos para el problema de clasificación múltiple en la base de datos del proyecto POCTEP*** se observa que los modelos utilizados para este problema, tal y como ocurría en el problema de clasificación binaria, tienden a aprender a clasificar a los sujetos como enfermos con EA. En los modelos entrenados con la base de datos del proyecto POCTEP esto es más evidente debido a que el número de enfermos con EA es mucho mayor que el número de sujetos de control y pacientes con DCL. Como se explica para el problema anterior, si el conjunto de entrenamiento presenta una mayoría de enfermos con EA,

entonces va a aprender las características para este grupo mayoritario y no será capaz de aprender las características de las clases controles y pacientes con DCL. En este caso la clase que peores resultados obtiene para todos los modelos es la clase de los pacientes con DCL.

Por último, en lo referente a la aplicación de *transfer learning*, como ya se ha explicado para cada caso en el Capítulo 5, los resultados obtenidos en este problema no son buenos, pero son mejores que los obtenidos en el problema de clasificación binaria. Los modelos elegidos para aplicar este paradigma presentan un descenso de la precisión de entre el 1 y el 5% con respecto a la precisión obtenida por los modelos aplicados sobre los datos de la base de datos con la que fueron entrenados. En este caso, al tratarse de un descenso relativamente pequeño, se podría confirmar que algunos de los modelos entrenados con la base de datos del HURH se podrán adaptar a los datos de la base de datos del proyecto POCTEP y viceversa. En este caso ocurre lo opuesto al caso de modelos entrenado con la base de datos del HURH, y una posible causa es el uso de diferentes frecuencias de muestreo en las bases de datos. Para adaptar los datos de la base de datos de POCTEP a las dimensiones de entrada a la red neuronal se utiliza un diezmado y no una interpolación como sucedía en el caso de HURH. Con el diezmado el *transfer learning* funciona mejor que si se realiza un interpolado. Si este paradigma se aplica sobre los modelos que han sido entrenados utilizando CAR y *z-score*, los resultados son parecidos a los obtenidos sin utilizarlas: la precisión se reduce aproximadamente el mismo porcentaje en ambos casos.

6.4. Comparativa con otros estudios de la literatura

Algunos estudios han conseguido mejores resultados de precisión para el mismo problema de clasificación binaria.

El estudio realizado por (Kulkarni *et ál.*, 2020) utiliza una base de datos formada por 200 sujetos cuya división por clases no consta. Utiliza registros EEG con una duración de 5 minutos y es capaz de obtener una precisión del 96,5% gracias al preprocesado aplicado a las señales EEG. En este caso, en lugar de introducir el EEG 'en crudo' a la CNN como se ha hecho en este TFG, se aplica la transformada *wavelet* continua (CWT, *Continuous Wavelet Transform*) que ayuda a reducir la complejidad de las señales EEG. Esta transformada hace que se consiga una representación bidimensional de las señales, es decir, se consigue tener señales EEG en el dominio tiempo-frecuencia. Estas señales han sido creadas utilizando segmentos de EEG de 20 segundos de duración. La entrada a la CNN es una 'imagen' y este tipo de redes neuronales funciona bien para este tipo de entrada (LeCun *et ál.*, 2015). La arquitectura la red CNN utilizada es similar a una de las arquitecturas utilizadas en este TFG. Está formada por dos capas convolucionales completas, con etapa convolucional, etapa de activación empleando ReLU y etapa de *pooling* utilizando un *pool_size* igual a 2. La principal diferencia entre este estudio y el realizado en este TFG es la dimensión de la red y el tipo de entrada ya que, la arquitectura es similar por lo que, en este caso, se puede afirmar que obtiene mejores resultados debido al procesamiento del EEG y el uso de CNN 2-D.

El estudio realizado por (Ieracitano *et ál*., 2019) obtiene una precisión menor que el realizado por (Kulkarni *et ál*., 2020) un 91,88%. En este caso, se utiliza una base de datos formada por 189 sujetos, 63 EA, 63 DCL y 63 controles que representan cada una de las clases de forma balanceada. Tal y como se realiza en este TFG, las señales EEG serán divididas en segmentos de 5 segundos. Tras esta división se obtiene la densidad espectral de potencia (PSD, *Power Spectral Density*) de cada canal obteniendo, lo que denominan una imagen PSD, que será lo que se introduce a la red CNN. La arquitectura de esta red es similar a la utilizada por (Kulkarni *et ál*., 2020), una red CNN 2-D que utiliza ReLU como función de activación y *maxpooling* en la etapa de *pooling*. De nuevo, este caso presenta similitudes con este TFG, pero existen dos diferencias principales; la primera es el procesado de la señal EEG, que será clave para la obtención de unos resultados mejores y la segunda que el número de sujetos de cada clase está balanceado.

Por último, en el estudio realizado por (Morabito *et ál*., 2016) se emplea una base de datos formada por 142 sujetos: 23 sujetos de control, 56 DCL y 63 EA. Las señales EEG son divididas en segmentos de 5 s, tal y como se realiza en este TFG; a continuación, se realiza la misma transformación utilizada por (Kulkarni *et ál*., 2020), una CWT para obtener una representación 2D del EEG. En este caso, la precisión obtenida es un 11,5% menor a la obtenida por (Kulkarni *et ál*., 2020), un 85% utilizando la misma técnica de procesado pero distintos parámetros en la CNN.

Tal y como ocurre en el caso del problema de clasificación binaria, algunos estudios consiguen mejores resultados utilizando técnicas de procesado sobre la señal de EEG antes de introducirla a la CNN. Morabito *et ál*. (2016), cuyo estudio para un problema de clasificación binaria se describe anteriormente, sigue el mismo proceso para la clasificación múltiple. Con este proceso, consigue una precisión del 82%, prácticamente el doble de lo que se consigue en este TFG.

Otro de los estudios a tener en cuenta, es el realizado por Ieracitano *et ál*. (2019), que además de estudiar el problema de clasificación binaria estudia también el de clasificación múltiple aplicando la misma metodología. Con esta metodología consiguen una precisión menor a la conseguida utilizando CWT, un 78,49% que dista bastante del 45% obtenido al utilizar las señales EEG en crudo.

La mayoría de los estudios para el diagnóstico de EA utilizando técnicas de *deep learning* aplican redes CNN junto con imágenes diagnósticas como MRI, fMRI o PET. Es por ello que, a continuación, se analizan algunos de estos estudios para poder comparar la metodología y los resultados obtenidos con los de este TFG.

El estudio realizado por Korolev *et ál*., (2017) utiliza dos arquitecturas o tipos distintos de redes neuronales convolucionales. Con un total de 231 imágenes MRI dividas en 50 EA, 120 DCL y 61 sujetos de control consigue una precisión para el problema de clasificación binaria de un 79% para la red *VoxCNN* y un 80% para la red *ResNet*. Estos resultados de clasificación son similares a los obtenidos en este TFG, donde se consigue un 76,90%. La primera red utilizada es una CNN 3-D que utiliza una arquitectura similar a la utilizada en este trabajo. La arquitectura utilizada es un híbrido de las arquitecturas representadas en la Figura 36 y la Figura

44 pero más profunda. La segunda, la red *ResNet* tiene una arquitectura compleja donde se suman las salidas de varias capas.

Otro estudio es el realizado por Liu *et ál*., (2014) donde se utilizan dos CNN con la misma arquitectura, pero distinta entrada. En una de ellas se utiliza como entrada imágenes PET y en la otra, imágenes MRI. Ambas redes tienen una arquitectura similar a la red *VoxCNN* utilizada por Korolev *et ál*., (2017). A la salida de esta red realizan un análisis de correlación y en función del resultado se realiza la clasificación. En este caso se utiliza un total de 392 sujetos: 101 controles, 200 DCL y 91 EA y se obtiene una precisión del 98,47%, una precisión bastante elevada. El resultado de este estudio apoya el uso de las redes CNN en el diagnóstico de la EA.

En el caso del estudio llevado a cabo por Farooq *et ál*., (2017) se realiza una clasificación múltiple con cuatro clases: control, DCL, DCL severo y EA. Utiliza imágenes MRI de un total de 149 sujetos: 45 controles, 49 DCL, 22 DCL severo y 33 EA. Utiliza la red utilizada por Korolev *et ál*., (2017), *ResNet* modificada de dos formas distintas y *GoogLeNet*, ambas redes son redes ya creadas y entrenadas que se reentrenan o se adaptan utilizando *fine-tuning* para utilizar *transfer learning* en un ámbito distinto al ámbito para el que fue creado el modelo. En este caso, se alcanzan unas precisiones muy elevadas del 98,88% para la red *GoogLeNet* y del 98,01% y 98,14% para las otras dos redes. En este caso, se observa que los resultados obtenidos son realmente buenos ya que se trata de un problema de clasificación múltiple con cuatro clases. Este tipo de redes, son redes con arquitecturas complejas que suelen estar preentrenadas con millones de datos que normalmente se utilizan en el ámbito para el que fueron desarrollados, normalmente imágenes, pero que pueden adaptarse a otros ámbitos fácilmente (Szegedy *et ál*., 2015).

Finalmente, (Jo *et ál*., 2019) realiza una revisión de algunos de los principales estudios realizados para el problema de clasificación binaria. En esta revisión se observa que los estudios que emplean imágenes como MRI o PET junto con CNN consiguen unos resultados de precisión de en torno al 80-90%.

En general, los resultados obtenidos en este TFG no son tan buenos como los obtenidos por los estudios mencionados por varias razones: (*i*) en la mayoría de los estudios, las clases están representadas de manera equitativa; (*ii*) se utiliza un procesado de la señal EEG para aprovechar las características espacio-temporales del mismo, eliminando complejidad que presenta la señal EEG y facilitando la extracción de características y el aprendizaje a la red CNN; (*iii*) al aprovechar las características espacio-temporales del EEG se obtiene una 'imagen' de dos dimensiones por lo que se utilizan redes CNN 2-D cuya arquitectura es similar a la utilizada en este TFG.

6.5. Limitaciones

Los resultados expuestos en el Capítulo 5 y discutidos en los apartados previos de este mismo capítulo están sujetos a ciertas limitaciones. Estas limitaciones deben tenerse en cuanta a la hora de interpretar estos resultados.

Los datos utilizados en el problema de clasificación binaria no están formados únicamente de sujetos de control y pacientes diagnosticados de EA. A esta última clase, se le han añadido los sujetos diagnosticados con DCL, haciendo que el desbalanceo entre sujetos de las distintas clases sea aún mayor para las dos bases de datos con las que se trabaja.

Para cada base de datos, se realiza una separación en entrenamiento, validación y test del 70-30%, donde el 70% es utilizado para entrenamiento y test. Al hacerse de manera automática, se obtiene distinto número de sujetos de test para cada caso de estudio (segmentos de 5 y 1 s) y para cada base de datos. Esto hace que las comparaciones entre modelos sean más complejas dificultando la comprensión de los resultados.

En algunos casos, los modelos han sido testeados con un número de sujetos de test muy desigual. Esto puede observarse en el problema de clasificación binaria con segmentos de 5 s utilizando la base de datos HURH, donde el número de sujetos de las clases control y enfermos está muy desbalanceado (4 controles versus 54 enfermos). Como ya se ha explicado anteriormente, este desequilibrio hace que el modelo tienda a clasificar a los sujetos dentro de la clase mayoritaria.

Capítulo 7

7. Conclusiones y líneas futuras

7.1 Cumplimiento de los objetivos del trabajo fin de grado 121

7.2 Conclusiones 121

7.3 Líneas futuras 122

7.1. Cumplimiento de los objetivos del trabajo fin de grado

En este TFG se han implementado diversas arquitecturas de DL basadas en señales EEG para tratar de identificar diferentes fases de la EA. En el apartado ***1.5. Objetivos*** del Capítulo 1 se enumeran una serie de objetivos a cumplir para la realización de este TFG. A continuación, se evalúa su grado de consecución:

1. El primer objetivo se ha cumplido pues se han consultado diversos artículos y libros relacionados con las enfermedades estudiadas (DCL y demencia por EA), las señales EEG y los diferentes métodos de ML y DL.
2. El segundo objetivo también ha sido llevado a cabo, pues se ha aprendido a programar redes neurales utilizando *Python* y la librería *Keras*.
3. Las dos bases de datos utilizadas en este TFG han sido estudiadas y procesadas para conseguir los datos necesarios para la implementación de las redes neuronales.
4. Se han diseñado distintas arquitecturas CNN para abordar problemas de clasificación binarios y multiclase de la EA, utilizando como datos de entrada señales EEG.
5. En cuanto a las métricas utilizadas para conocer el rendimiento del modelo, se ha utilizado la precisión, el error y las matrices de confusión que contienen la clasificación de sujetos.
6. Se han expuesto y analizado los resultados obtenidos en todos los casos de estudio y se han comparado con distintos estudios existentes en la literatura.
7. Se han extraído las conclusiones de los resultados y se han identificado las líneas futuras de investigación.

7.2. Conclusiones

Tras analizar los resultados obtenidos, a continuación, se describen las conclusiones extraídas:

1. No se han obtenido resultados satisfactorios al clasificar automáticamente las señales EEG de sujetos con diferentes grados de la EA.
2. La señal EEG es muy compleja lo que dificulta la extracción de características y el aprendizaje por parte de la CNN. Utilizar transformaciones como CWT reduce la complejidad de la señal y ayuda a obtener mejores resultados.
3. Utilizar segmentos de 1 s con el objetivo de conseguir aumentar los datos de entrenamiento del modelo para reducir el *overfitting* no permite mejorar los resultados de clasificación en relación a los modelos entrenados utilizando segmentos de 5 s.
4. Para el problema de clasificación binaria existe una gran diferencia entre el número de sujetos de control y el de enfermos. Esto implica que el

modelo no aprende bien las características de la clase minoritaria, tendiendo a clasificar a la mayoría de sujetos como enfermos.

5. El número de sujetos presente en estas bases de datos es bajo lo que provoca que la CNN no aprenda características que le permitan generalizar bien a sujetos independientes.
6. A pesar de utilizar capas y algoritmos de regularización sigue existiendo *overfitting*, por lo que se concluye que, para este problema y con este tipo de señales, es necesario aumentar el número de registros EEG para conseguir mejores resultados.
7. El paradigma *transfer learning* no ha funcionado razonablemente en el contexto de análisis planteado en el TFG, pues los resultados de clasificación que se consiguen son menores que los obtenidos por los modelos utilizados para un problema específico.
8. Las técnicas de preprocesado CAR y *z-score* utilizadas con el objetivo de mejorar los resultados al aplicar *transfer learning* hacen que los resultados sean incluso peor que los obtenidos al aplicar este paradigma sin utilizar dichas técnicas. Esto implica que la precisión obtenida tras entrenar nuevamente los modelos utilizando estas técnicas es menor. Es posible que los resultados tras la aplicación de la normalización no sean buenos debido a las componentes aleatorias de los modelos que influyen en el entrenamiento, como el *dropout* o la inicialización aleatoria de los pesos en la primera época. Otra posibilidad es que no mejoren los porque la CNN ya realiza una normalización interna de los datos dentro de la red, y volver a normalizar no aporta nada nuevo.

7.3. Líneas futuras

Este TFG no ha obtenido buenos resultados por los motivos expuestos previamente. Por tanto, para intentar mejorar estos resultados se recomienda, en primer lugar, aumentar el tamaño de las bases de datos utilizadas. Esto es importante para prevenir el *overfitting*, pues se ha demostrado que, además de las técnicas de regularización, un método para prevenir este problema es aumentar el tamaño del conjunto de datos utilizado (Vasilev *et ál.*, 2019). Además de aumentar el número de registros de base de datos, se recomienda que el número de sujetos pertenecientes a las distintas clases sea parecido, es decir, que exista un equilibro entre clases. Otra mejora podría ser aumentar el tamaño o duración de los segmentos una vez se haya aumentado el tamaño de las bases de datos utilizadas.

Para intentar mejorar el comportamiento de los datos actuales pueden utilizarse otras técnicas de procesado además de las utilizadas en este TFG. En este sentido, se ha observado que se produce una mejora significativa de la precisión de los modelos si en lugar de utilizar directamente la señal de EEG a la arquitectura DL, se introduce su versión transformada mediante la PSD o la CWT (Kulkarni *et ál.*, 2020). Sería por lo tanto interesante que en futuros estudios se evaluara si utilizar una representación la señal EEG mediante algún tipo de transformada puede mejorar los resultados de clasificación de la EA.

Otra línea de investigación que podría mejorar los resultados consistiría en emplear redes convolucionales de 2-D y 3-D en lugar de utilizar las 1-D (Jo *et ál.*, 2019). Estas arquitecturas podrían utilizarse para aprender características que ocurren en el mismo instante temporal en varios canales de EEG. Asimismo, sería interesante evaluar la aplicación de otro tipo de redes neuronales como los *Autoencoders* o *Recurrent Neural Networks* (Petrosian *et ál.*, 2001), que son útiles a la hora de aprender dependencias en series temporales como, por ejemplo, el EEG.

Glosario de siglas y acrónimos

Aβ:	Proteína Beta-Amiloide
Adam:	*Adaptative Moment Estimation*
APOE:	Apoelipoproteína E
CAR:	*Common Average Reference*
CNN:	*Convolutional Neural Network*
CWT:	*Continuous Wavelet Transform*
DCL:	Deterioro Cognitivo Leve
DL:	*Deep Learning*
EA:	Enfermedad de Alzheimer
EEG:	Electroencefalografía
ERP:	*Event Related Potencials*
IA:	Inteligencia Artificial
ICA:	*Independent Component Analysis*
MEG:	Magnetoencefalografía
ML:	*Machine Learning*
MMSE:	*Mini-Mental State Examination*
MRI:	*Magnetic Resonance Imaging*
NIA-AAA:	*National Institute on Aging – Alzheimer's Association*
NN:	*Neural Network*
PET:	*Positron Emission Tomography*
POCTEP:	Programa Operativo de Cooperación Transfronteriza España - Portugal
PSD:	*Power Spectral Density*
ReLU:	Rectified Linear Unit
SGD:	*Stochastic Gradient Descent*
TFG:	Trabajo Fin de Grado

Bibliografía

Abásolo, D., Simons, S. (2014). *Distance-Based Lempel–Ziv Complexity for the Analysis of Electroencephalograms in Patients with Alzheimer's Disease*.

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing. https://doi.org/10.1007/978-3-319-94463-0

Alhaddad, M. J. (2012). Common Average Reference (CAR) Improves P300 Speller. *International Journal of Engineering and Technology*, *2*(3), 14.

Allegri, R. F., Arizaga, R. L., Bavec, C. V., Colli, L. P., Demey, I., Fernández, M. C., Frontera, S. A., Garau, M. L., Jiménez, J. J., Golimstok, Á., Kremer, J., Labos, E., Mangone, C. A., Ollari, J. A., Rojas, G., Salmini, O., Ure, J. A., & Zuin, D. R. (2011). Enfermedad de Alzheimer. Guía de práctica clínica. *Neurología Argentina*, *3*(2), 120-137. https://doi.org/10.1016/S1853-0028(11)70026-X

Alom, Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., & Nasrin, M. S. (2018). *The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches*. 39.

Alzheimer's Association. (2018). 2018 Alzheimer's disease facts and figures. *Alzheimer's & Dementia*, *14*(3), 367-429. https://doi.org/10.1016/j.jalz.2018.02.001

Ashford, J. W. (2019). Treatment of Alzheimer's Disease: Trazodone, Sleep, Serotonin, Norepinephrine, and Future Directions. *Journal of Alzheimer's Disease*, *67*(3), 923-930. https://doi.org/10.3233/JAD-181106

Babiloni, C., Pizzella, V., Gratta, C. D., Ferretti, A., & Romani, G. L. (2009). Chapter 5 Fundamentals of Electroencefalography, Magnetoencefalography, and Functional Magnetic Resonance Imaging. *International Review of Neurobiology*, *86*, 67-80. https://doi.org/10.1016/S0074-7742(09)86005-4

Bachiller, A. (2012). *Análisis de la señal de electroencefalograma mediante distancias espectrales para la ayuda en el diagnóstico de la enfermedad de Alzheimer*. http://uvadoc.uva.es/handle/10324/2654

Barthélemy, R., Li, Y., Gordon, A. (2020). *A soluble phosphorylated tau signature links tau, amyloid and the evolution of stages of dominantly inherited Alzheimer's disease | Nature Medicine*. https://www.nature.com/articles/s41591-020-0781-z

Blennow, K., Leon, M. J. de, & Zetterberg, H. (2006). Alzheimer's disease. *The Lancet*, *368*(9533), 387-403. https://doi.org/10.1016/S0140-6736(06)69113-7

Braak, H., Braak, E. (1991). Neuropathological stageing of Alzheimer-related changes. *Acta Neuropathologica*, *82*(4), 239-259. https://doi.org/10.1007/BF00308809

Bronzino, J. D. (2006a). *The biomedical engineering handbook* (3rd ed).

Bronzino, J. D. (2006b). *Medical Devices and Systems*.

Castellanos-Aguilar, L., Martínez-Zuñiga, N., Gutierrez-Murcia, J. L., Viramontes-Pintos, A., Garcés-Ramírez, L., de-la-Cruz, F., Guadarrama-Ortíz, P., Luna-Muñoz, J. (2017). La tinción con el colorante rojo tiazina es un método de diagnóstico pos-mortem rápido y confiable para la enfermedad de Alzheimer. *Archivos de Neurociencias*, *22*(4), 33-43.

Chassagnon, G., Vakalopolou, M., Paragios, N., & Revel, M.-P. (2020). Deep learning: Definition and perspectives for thoracic imaging. *European Radiology*, *30*(4), 2021-2030. https://doi.org/10.1007/s00330-019-06564-3

Chengxuan, Q., Kivipelto, M., von Straus, Eva. (2009). *Epidemiology of Alzheimer's disease: Occurrence, determinants, and strategies toward intervention*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3181909/

Chollet, F. (2020). *Keras documentation: The Sequential model*. https://keras.io/guides/sequential_model/

Cohen, M. X. (2017). Where Does EEG Come From and What Does It Mean? *Trends in Neurosciences*, *40*(4), 208-218. https://doi.org/10.1016/j.tins.2017.02.004

Cummings, J. (2004). *Alzheimer's Disease | NEJM*. https://www.nejm.org/doi/full/10.1056/NEJMra040223

Dauwels, J., Srinivasan, K., Ramasubba Reddy, M., Musha, T., Vialatte, F.-B., Latchoumane, C., Jeong, J., & Cichocki, A. (2011). Slowing and Loss of Complexity in Alzheimer's EEG: Two Sides of the Same Coin? *International Journal of Alzheimer's Disease*, *2011*. https://doi.org/10.4061/2011/539621

DeTure, M. A., Dickson, D. W. (2019). *The neuropathological diagnosis of Alzheimer's disease*. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6679484/

Ding, W., Ho Sohn, J., Kawczynski, M. G . (2018). *A Deep Learning Model to Predict a Diagnosis of Alzheimer Disease by Using 18F-FDG PET of the Brain | Radiology*. https://pubs.rsna.org/doi/full/10.1148/radiol.2018180958

Dubois, B., Hampel, H., Feldman, H. H., Scheltens, P., Aisen, P., Andrieu, S., Bakardjian, H., Benali, H., Bertram, L., Blennow, K., Broich, K., Cavedo, E., Crutch, S., Dartigues, J.-F., Duyckaerts, C., Epelbaum, S., Frisoni, G. B., Gauthier, S., Genthon, R., … Jack, C. R. (2016). Preclinical Alzheimer's disease: Definition, natural history, and diagnostic criteria. *Alzheimer's & dementia : the journal of the Alzheimer's Association*, *12*(3), 292-323. https://doi.org/10.1016/j.jalz.2016.02.002

Folstein, M. F., Folstein, S. E., & McHugh, P. R. (1975). «Mini-mental state». A practical method for grading the cognitive state of patients for the clinician. *Journal of Psychiatric Research*, *12*(3), 189-198. https://doi.org/10.1016/0022-3956(75)90026-6

Gil-Nagel, A. (2001). *Manual de electroencefalografía*. McGraw-Hill Interamericana.

Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. MIT Press.

Guevara, Hernández, & Sanz. (2010). *Programas para análisis de señales y evaluación cognoscitiva—Guevara | Densidad espectral | Corteza cerebral*. Scribd. https://es.scribd.com/document/78804586/Programas-para-analisis-de-senales-y-evaluacion-cognoscitiva-Guevara

Gulli, A., Pal, S. P. (Software. (2017). *Deep Learning with Keras: Implementing Deep Learning Models and Neural Networks with the Power of Python*. Packt Publishing.

Hu, J., Sasakawa, T., Hirasawa, K., & Zheng, H. (2007). A Hierarchical Learning System Incorporating with Supervised, Unsupervised and Reinforcement Learning. En D. Liu,

S. Fei, Z.-G. Hou, H. Zhang, & C. Sun (Eds.), *Advances in Neural Networks – ISNN 2007* (pp. 403-412). Springer. https://doi.org/10.1007/978-3-540-72383-7_48

Ieracitano, C., Mammone, N., Bramanti, A., Hussain, A., & Morabito, F. C. (2019). A Convolutional Neural Network approach for classification of dementia stages based on 2D-spectral representation of EEG recordings. *Neurocomputing*, *323*, 96-107. https://doi.org/10.1016/j.neucom.2018.09.071

Jack, C. R., Bennett, D. A., Blennow, K., Carrillo, M. C., Dunn, B., Haeberlein, S. B., Holtzman, D. M., Jagust, W., Jessen, F., Karlawish, J., Liu, E., Molinuevo, J. L., Montine, T., Phelps, C., Rankin, K. P., Rowe, C. C., Scheltens, P., Siemers, E., Snyder, H. M., & Sperling, R. (2018). NIA-AA Research Framework: Toward a biological definition of Alzheimer's disease. *Alzheimer's & dementia : the journal of the Alzheimer's Association*, *14*(4), 535-562. https://doi.org/10.1016/j.jalz.2018.02.018

Jeong, J. (2004). EEG dynamics in patients with Alzheimer's disease. *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology*, *115*(7), 1490-1505. https://doi.org/10.1016/j.clinph.2004.01.001

Jo, T., Nho, K., Saykin. J. (2019). *Frontiers | Deep Learning in Alzheimer's Disease: Diagnostic Classification and Prognostic Prediction Using Neuroimaging Data | Frontiers in Aging Neuroscience*.

Kok, J. N. (2009). Artificial Intelligence: Definition, Trends, Techniques and Cases. *ARTIFICIAL INTELLIGENCE*, 5.

Kulkarni, N., Salvi, A., & Parhad, S. (2020). Convolutional Neural Network-Based Diagnosis of Alzheimer's Disease Using Time–Frequency Features. En R. R. Chillarige, S. Distefano, & S. S. Rawat (Eds.), *Advances in Computational Intelligence and Informatics* (pp. 331-339). Springer. https://doi.org/10.1007/978-981-15-3338-9_38

LeCun, Y., Bengio Y., Hinton G. (2015). *Deep learning | Nature*. https://www.nature.com/articles/nature14539

Leutcher, A., Newton, T., Cook, I. (1992). *Changes in brain functional connectivity in Alzheimer's-type and infarct dementia*. https://www.researchgate.net/publication/21732519_Changes_in_brain_functional_connectivity_in_Alzheimer's-type_and_infarct_dementia

Liu, S., Liu, S., Cai, W., Pujol, S., Kikinis, R., & Feng, D. (2014). Early diagnosis of Alzheimer's disease with deep learning. *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*, 1015-1018. https://doi.org/10.1109/ISBI.2014.6868045

Livingston, Huntley, & Sommerlad. (2020). *Dementia prevention, intervention, and care: 2020 report of the Lancet Commission—The Lancet*.

Maturana-Candelas, A., Gómez, C., Poza, J., Pinto, N., & Hornero, R. (2019). EEG Characterization of the Alzheimer's Disease Continuum by Means of Multiscale Entropies. *Entropy*, *21*(6), 544. https://doi.org/10.3390/e21060544

Millett, D. (2001). Hans Berger: From Psychic Energy to the EEG. *Perspectives in Biology and Medicine*, *44*(4), 522-542. https://doi.org/10.1353/pbm.2001.0070

Möller, H. J., Graeber, M. B. (1998). The case described by Alois Alzheimer in 1911. Historical and conceptual perspectives based on the clinical record and

neurohistological sections. *European Archives of Psychiatry and Clinical Neuroscience*, *248*(3), 111-122. https://doi.org/10.1007/s004060050027

Moolayil. (2019). *Learn Keras for Deep Neural Networks*. Springer

Morabito, F. C., Campolo, M., Ieracitano, C., Ebadi, J. M., Bonanno, L., Bramanti, A., Desalvo, S., Mammone, N., & Bramanti, P. (2016). Deep convolutional neural networks for classification of mild cognitive impaired and Alzheimer's disease patients from scalp EEG recordings. *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, 1-6. https://doi.org/10.1109/RTSI.2016.7740576

Nouri. (2012). *Supercharge Your Data Science Career! Follow me of LinkedIn for more: Steve Nouri https://www.linkedin.com/in/stevenouri | Ali Abedi—Academia.edu*. https://www.academia.edu/40112713/Supercharge_Your_Data_Science_Career_Follow_me_of_LinkedIn_for_more_Steve_Nouri_https_www_linkedin_com_in_stevenouri

Pan, S. J., Yang, Q. (2010). A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345-1359. https://doi.org/10.1109/TKDE.2009.191

Park, S. A., Han, S. M., Kim, C. E. (2020). New fluid biomarkers tracking non-amyloid-β and non-tau pathology in Alzheimer's disease. *Experimental & Molecular Medicine*, *52*(4), 556-568. https://doi.org/10.1038/s12276-020-0418-9

Petersen, R. C. (2004). Mild cognitive impairment as a diagnostic entity. *Journal of Internal Medicine*, *256*(3), 183-194. https://doi.org/10.1111/j.1365-2796.2004.01388.x

Petersen, Ronald C. (2014). *Mayo Clinic on Alzheimer's Disease*. RosettaBooks.

Petrosian, A. A., Prokhorov, D. V., Lajara-Nanson, W., & Schiffer, R. B. (2001). Recurrent neural network-based approach for early recognition of Alzheimer's disease in EEG. *Clinical Neurophysiology*, *112*(8), 1378-1387. https://doi.org/10.1016/S1388-2457(01)00579-X

Poza, J. (2008). *Análisis tiempo-frecuencia de la actividad magnetoencefalográfica espontánea en la enfermedad de Alzheimer* [Universidad de Valladolid]. https://doi.org/10.35376/10324/136

Purves, A. G. J., Fitzpatrick. (2001). *Neuroscience* (2nd ed.). Sinauer Associates.

Raschka, S., Mirjalili, V. (2017). *Python Machine Learning*. Packt Publishing Ltd.

Rodrigo, J., Martínez, A., Patricia Fernández, A., Serrano, J., Luisa Bentura, M., Moreno, E., Aparicio, M., Martínez-Murillo, R., & Regidor, J. (2007). Características neuropatológicas y moleculares de la enfermedad de Alzheimer. *Revista Española de Geriatría y Gerontología*, *42*(2), 103-110. https://doi.org/10.1016/S0211-139X(07)73533-3

Rossini, P. M., Di Iorio, R., Vecchio, F., Anfossi, M., Babiloni, C., Bozzali, M., Bruni, A. C., Cappa, S. F., Escudero, J., Fraga, F. J., Giannakopoulos, P., Guntekin, B., Logroscino, G., Marra, C., Miraglia, F., Panza, F., Tecchio, F., Pascual-Leone, A., & Dubois, B. (2020). Early diagnosis of Alzheimer's disease: The role of biomarkers including advanced EEG signal analysis. Report from the IFCN-sponsored panel of experts. *Clinical Neurophysiology: Official Journal of the International Federation of Clinical Neurophysiology*, *131*(6), 1287-1310. https://doi.org/10.1016/j.clinph.2020.03.003

Rubinov, M., Sporns, O. (2010). Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*, *52*(3), 1059-1069. https://doi.org/10.1016/j.neuroimage.2009.10.003

Sabbagh, M. N., Lue, L.-F., Fayard, D., & Shi, J. (2017). Increasing Precision of Clinical Diagnosis of Alzheimer's Disease Using a Combined Algorithm Incorporating Clinical and Novel Biomarker Data. *Neurology and Therapy*, *6*(Suppl 1), 83-95. https://doi.org/10.1007/s40120-017-0069-5

Si, Wang, & Liu. (2019). *Brain Network Modeling Based on Mutual Information and Graph Theory for Predicting the Connection Mechanism in the Development of Alzheimer's Disease*. ResearchGate. https://doi.org/10.20944/preprints201901.0208.v1

Song, J., Davey, C., Poulsen, C., Luu, P., Turovets, S., Anderson, E., Li, K., & Tucker, D. (2015). EEG source localization: Sensor density and head surface coverage. *Journal of Neuroscience Methods*, *256*, 9-21. https://doi.org/10.1016/j.jneumeth.2015.08.015

Sperling, R. A., Aisen, P. S., Beckett, L. A., Bennett, D. A., Craft, S., Fagan, A. M., Iwatsubo, T., Jack, C. R., Kaye, J., Montine, T. J., Park, D. C., Reiman, E. M., Rowe, C. C., Siemers, E., Stern, Y., Yaffe, K., Carrillo, M. C., Thies, B., Morrison-Bogorad, M., … Phelps, C. H. (2011). Toward defining the preclinical stages of Alzheimer's disease: Recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease. *Alzheimer's & Dementia*, *7*(3), 280-292. https://doi.org/10.1016/j.jalz.2011.03.003

Szegedy, C., Wei Liu, Yangqing Jia, Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1-9. https://doi.org/10.1109/CVPR.2015.7298594

Team keras. (2020a). *Keras documentation: Probabilistic losses*. https://keras.io/api/losses/probabilistic_losses/#categorical_crossentropy-function

Team keras, K. (2020b). *Keras documentation: BatchNormalization layer*. https://keras.io/api/layers/normalization_layers/batch_normalization/

Team keras, K. (2020c). *Keras documentation: Conv1D layer*. https://keras.io/api/layers/convolution_layers/convolution1d/

Team keras, K. (2020d). *Keras documentation: Dropout layer*. https://keras.io/api/layers/regularization_layers/dropout/

Team Keras, K. (2020). *Keras documentation: EarlyStopping*. https://keras.io/api/callbacks/early_stopping/

Team keras, K. (2020e). *Keras documentation: Flatten layer*. https://keras.io/api/layers/reshaping_layers/flatten/

Team keras, K. (2020f). *Keras documentation: Python & NumPy utilities*. https://keras.io/api/utils/python_utils/#to_categorical-function

Team keras, K. (2020g). *Keras documentation: ReduceLROnPlateau*. https://keras.io/api/callbacks/reduce_lr_on_plateau/

Team Scikit-learn. (2020). *sklearn.preprocessing.LabelEncoder—Scikit-learn 0.23.2 documentation*. https://scikit-

learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder.fit

Team Scipy. (2020). *scipy.stats.zscore—SciPy v1.5.2 Reference Guide*. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.zscore.html

Vasilev, I., Slater, D., Spacagna, G., Roelants, P., Zocca, V. (2019). *Python Deep Learning: Exploring deep learning techniques and neural network architectures with PyTorch, Keras, and TensorFlow, 2nd Edition*. Packt Publishing Ltd.