Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

MENCIÓN EN COMPUTACIÓN

Tecnología Deep Learning Aplicada a Señales Respiratorias para la ayuda al Diagnóstico de la Apnea del Sueño

---

Autora
Doña Marta Fernández Poyatos

Tutores:

Don Benjamín Sahelices Fernández
Don Gonzalo César Gutiérrez Tobal

Agradecimientos

A mis tutores Benjamín y Gonzalo, por su constante dedicación y ayuda para desarrollar este proyecto. Gracias por todo el conocimiento que me habéis transmitido estos meses. También al Grupo de Ingeniería Biomédica, por permitirme participar en vuestros proyectos y proporcionarme los medios necesarios para completar el trabajo con éxito.

A los profesores del Grado en Ingeniería Informática que me han acompañado estos últimos cinco años. De todos y cada uno de ellos me llevo un gran aprendizaje.

A mis amigos y compañeros de clase por hacer de estos años una etapa preciosa de la que me llevo grandes recuerdos.

Por último y más importante, a mis padres y a mi pareja. A mis padres por su apoyo incondicional y todos los valores que me han inculcado desde pequeña. Y a mi pareja por creer en mi e inspirarme siempre a seguir adelante.

Resumen

La apnea del sueño es una enfermedad con una alta prevalencia en la población adulta, llegando a afectar hasta al 38 % de la misma. Las personas que la padecen sufren pausas respiratorias recurrentes durante el sueño que acarrean una ventilación inadecuada y sueño fragmentado y no reparador. Además, la apnea del sueño ha sido asociada con múltiples e importantes problemas de salud, incluyendo enfermedades cardiovasculares, diabetes y un riesgo más elevado de padecer cáncer. A pesar de ello, se trata de una enfermedad muy infradiagnosticada.

La prueba diagnóstica estándar es la polisomnografía nocturna (PSG). La PSG se lleva a cabo en una unidad del sueño especializada y durante la misma se monitorizan y adquieren hasta 32 señales fisiológicas de cada paciente. A pesar de la eficacia de la PSG, esta presenta importantes limitaciones como su complejidad, su alto coste, y el tiempo consumido por los especialistas en la inspección de todas las señales nocturnas adquiridas. Debido a estas limitaciones y la alta prevalencia de la enfermedad, las unidades del sueño no son capaces de abordar todas las pruebas necesarias, produciéndose largas listas de espera y retrasos en el diagnóstico y acceso al tratamiento. Es por ello que se hace necesario encontrar alternativas diagnósticas más sencillas.

El objetivo de este trabajo es evaluar si, usando solamente 2 (señal de respiración torácica y señal de respiración abdominal) de las 32 señales adquiridas durante la PSG, se puede ayudar en la simplificación del diagnóstico de la apnea del sueño. Para ello, se han aplicado tecnologías *deep learning* sobre registros del sueño de 8.257 pacientes obtenidos de la base de datos del Sleep Heart Health Study (SHHS).

Las señales torácicas y abdominales fueron separadas en tres grupos (entrenamiento, validación y test). Estos tres grupos fueron utilizados para entrenar arquitecturas *deep learning* basadas en redes neuronales convolucionales, obtener un subconjunto óptimo de hiperparámetros y establecer un rendimiento diagnóstico generalizable de los modelos entrenados, respectivamente. El problema se abordó desde el punto de vista de la regresión del índice de apnea-hipopnea (IAH), que es la variable clínica utilizada para establecer la presencia y severidad de la enfermedad. También se realizaron pruebas para determinar la precisión de los modelos a la hora de estimar diferentes tipos de eventos apneicos.

El mayor rendimiento diagnóstico fue obtenido por un modelo con capas convolucionales de una dimensión, cuya estimación del IAH alcanzó un coeficiente kappa de 4 clases de 0.406 en test y 0.439 en validación y una sensibilidad y especificidad de 0.164-0.991, 0.664-0.871, y 0.894-0.639, respectivamente, en cada uno de los 3 umbrales de IAH que determinan la severidad de la enfermedad (5 eventos/hora, 15 e/h, y 30 e/h). No obstante, se alcanzaron rendimientos aun mayores (índice kappa de 0.536 en test y 0.873 en validación) al considerar solamente eventos apneicos de carácter central.

Estos resultados sugieren que la información contenida en la respiración torácica y abdominal utilizada para entrenar modelos de *deep learning* podrían resultar de utilidad en la simplificación del diagnóstico de la apnea del sueño, especialmente en el caso de eventos apneicos centrales.

Palabras clave: Apnea del sueño, señal torácica, señal abdominal, SHHS, *deep learning*.

Abstract

Sleep apnea is a disease with a high prevalence in the adult population, affecting up to 38 % of them. People who suffer from this desease experience recurrent breathing pauses during sleep leading to inadequate ventilation and fragmented, unrefreshing sleep. In addition, sleep apnea has been associated with multiple major health problems, including cardiovascular disease, diabetes and an increased risk of cancer. Despite this, it is a highly under-diagnosed disease.

The standard diagnostic test is the overnight polysomnography (PSG). The PSG is performed in a specialised sleep unit and up to 32 physiological signals are monitored and acquired from each patient. Despite the efficacy of the PSG, it has important limitations such as its complexity, its high cost, and the time consumed by specialists in inspecting all the acquired nocturnal signals. Due to these limitations and the high prevalence of the disease, sleep units are not able to deal with all the necessary tests, resulting in long waiting lists and delays in diagnosis and access to treatment. This is why it is necessary to find simpler diagnostic alternatives.

The aim of this work is to evaluate whether using only 2 (thoracic breathing signal and abdominal breathing signal) out of the 32 signals acquired during the PSG can help in simplifying the diagnosis of sleep apnea. To this end, deep learning technologies have been applied to sleep records of 8.257 patients obtained from the Sleep Heart Health Study (SHHS) database.

The thoracic and abdominal signals were separated into three groups (training, validation and test). These three groups were used to train convolutional neural network-based deep learning architectures, obtain an optimal subset of hyperparameters, and establish a generalisable diagnostic performance of the trained models, respectively. The problem was approached from the point of view of the regression of the apnea-hypopnea index (AHI), which is the clinical variable used to establish the presence and severity of the disease. Tests were also performed to determine the accuracy of the models in estimating different types of apneic events.

The highest diagnostic performance was obtained by a model with one-dimensional convolutional layers, whose estimate of the AHI reached a 4-class kappa coefficient of 0.406 in test and 0. 439 in validation, and a sensitivity and specificity of 0.164-0.991, 0.664-0.871, and 0.894-0.639, respectively, at each of the 3 AHI thresholds determining disease severity (5 events/hour, 15 e/h, and

30 e/h). However, even higher performances (kappa index of 0.536 in test and 0.873 in validation) were achieved when considering only central apneic events.

These results suggest that the information contained in thoracic and abdominal respiration used to train deep learning models could be useful in simplifying the diagnosis of sleep apnea, especially in the case of central apneic events.

Key words: sleep apnea, thoracic signal, abdominal signal, SHHS, deep learning.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

La apnea del sueño es una enfermedad caracterizada por episodios de ausencia completa (apnea) o parcial (hipopnea) del flujo aéreo en las vías respiratorias durante el sueño, con una duración mínima de 10 segundos [1]. Existen tres tipos de eventos apneicos: obstructivos (provocado por la obstrucción de las vías respiratorias), centrales (provocado por un fallo en la señal enviada por el cerebro a los pulmones) y mixtos (una mezcla de los anteriores) [2]. La apnea del sueño está relacionada con múltiples patologías de tipo cardiovascular y puede llegar a aumentar la mortalidad en pacientes con enfermedades pulmonares o complicaciones en procesos postoperatorios [3].

La técnica utilizada para el diagnóstico de la apnea es la polisomnografía nocturna (PSG). Se trata de una prueba llevada a cabo en una unidad especializada del sueño en el hospital, en la que se recogen múltiples señales biomédicas del paciente, que posteriormente deben ser analizadas minuciosamente por los médicos. En este trabajo se van a estudiar tan solo dos de las señales, las que reflejan los movimientos respiratorios del paciente mediante una banda torácica y otra abdominal.

El objetivo de este trabajo es el estudio de las señales abdominal y torácica para la ayuda al diagnóstico de la apnea del sueño mediante técnicas de *deep learning*, aplicado a un problema de regresión para la predicción del número de eventos de apnea. Para ello, se desarrollarán diferentes modelos de redes convolucionales.

En el capítulo 2 se introducen los conceptos médicos básicos acerca de la apnea del sueño, los diferentes tipos de apnea existentes y la PSG, para la recopilación de parámetros fisiológicos durante el sueño.

En el capítulo 3 se introducen todos los conceptos tecnológicos necesarios para este trabajo. Se introduce la base de datos que se va a utilizar, junto con las herramientas más relevantes que han sido necesarias para el desarrollo del proyecto y una introducción a las redes neuronales y a

la arquitectura más relevante en este trabajo: las redes neuronales convolucionales.

En el capítulo 4 se elabora una planificación de todas las tareas que se deben realizar a lo largo del proyecto, junto con un análisis de los posibles riesgos que pueden surgir en este tiempo. Se hace un seguimiento del cumplimiento de plan de proyecto.

En el capítulo 5 se presenta todo el preprocesado de las señales desde que son extraídas del estudio del sueño del paciente y hasta que pueden ser analizadas mediante *deep learning*, junto con la transformación de las señales a imágenes mediante espectrogramas. En este capítulo, además, se definen las diferentes arquitecturas que emplean redes convolucionales que se van a estudiar.

En el capítulo 6 se muestran los resultados de la experimentación con los diferentes modelos definidos en el capítulo anterior. También se presentan los resultados sobre test de los modelos que han obtenido rendimientos diagnósticos más elevados en validación.

El último capítulo de este proyecto está dedicado a la extracción de conclusiones, al estudio de las limitaciones del proyecto y a la presentación de las posibles líneas futuras de investigación.

Capítulo 2

Contexto conceptual

En este capítulo se van a introducir los conceptos básicos de apnea del sueño y de la polisomnografía (PSG), el método de recopilación de parámetros fisiológicos durante el sueño, que es el estándar para el diagnóstico.

2.1. Apnea del sueño

La apnea del sueño es un trastorno del sueño en el que la respiración se detiene durante 10 segundos, en repetidas ocasiones, a lo largo de la noche. Existen tres tipos fundamentales de eventos apneicos: obstructivos, centrales y mixtos.

2.1.1. Apnea obstructiva del sueño (AOS)

Es un tipo de alteración respiratoria que ocurre durante el sueño, caracterizada por repetidos episodios de colapso de las vías respiratorias superiores [3], producido habitualmente cuando los músculos de la garganta se relajan. Esto hace que se reduzca (hipopnea) o que se dentenga por completo (apnea) el flujo de aire hacia el sistema respiratorio, produciendo *hipoxemia* e *hipercapnia*. La respuesta del cuerpo ante estos eventos es activar el sistema simpático para reactivar los músculos de la garganta y recuperar el flujo aéreo, lo que lleva a la aparición de microdespertares y a un sueño no reparador.

La prevalencia de la apnea obstructiva del sueño es de entre el 9 % y el 38 % en la población adulta, siendo mayor en hombres que en mujeres, y aumentando con la edad [4]. También es mayor en hombres y mujeres con obesidad, en comparación con los que sufren sobrepeso. La prevalencia

en la población de edad avanzada es muy elevada, llegando a un 88 % en hombres de entre 65 y 69 años y a un 90 % en hombres de 60 a 85 años. En el caso de las mujeres, esta cifra es algo más baja, de un 66 % en el primer caso y de un 78 % en el segundo.

A pesar de estas cifras de prevalencia tan elevadas, la apnea obstructiva del sueño es una enfermedad muy infradiagnosticada. Se estima que alrededor del 82 % de hombres y 93 % de mujeres en Estados Unidos que padecen esta enfermedad, no han sido diagnosticados [5].

Está demostrado que existe una cierta asociación entre la apnea obstructiva del sueño y otras múltiples patologías de tipo cardiovascular como insuficiencia cardíaca o infarto de miocardio [3]. También aumenta el riesgo de accidente cerebrovascular y la fibrilación auricular. En personas con hipertensión, puede producir hipertensión resistente hasta al 40 % incluso hasta el 90 % de personas. Puede aumentar la mortalidad en pacientes con enfermedades pulmonares obstructivas y provocar complicaciones cardiopulmonares en procesos postoperatorios, con una mayor probabilidad de desaturación e ingreso en cuidados intensivos. También puede asociarse a desordenes metabólicos, afectando principalmente a hombres, en los que un incremento de 10 kg de peso, eleva 5 veces la probabilidad de incrementar el índice de apnea-hipopnea (IAH).

2.1.1.1. Síntomas

La AOS presenta síntomas característicos como: somnolencia, fatiga, insomnio, ronquidos o apnea observada, además de trastornos médicos o psiquiátricos asociados, como pueden ser: hipertensión, cardiopatía coronaria, diabetes, trastorno del estado de ánimo, fibrilación auricular, accidente cerebrovascular, insuficiencia cardíaca congestiva o disfunción cognitiva, junto con al menos cinco eventos respiratorios (con pausas respiratorias de al menos 10 segundos) mayoritariamente obstructivos durante una hora de sueño.

El índice de apnea-hipopnea (IAH) es obtenido por los especialistas médicos tras la inspección de todas las señales adquiridas durante la PSG. Se trata de la suma de eventos de apnea e hipopnea por hora de sueño del paciente. Este índice sirve para establecer la severidad de la AOS como sigue [3]:

- **No apnea**: IAH menor de 5 e/h.
- **Leve**: IAH entre 5 e/h y 15 e/h.
- **Moderada**: IAH entre 16 e/h y 30 e/h.
- **Grave**: IAH mayor de 30 e/h.

2.1.1.2. Causas

La apnea obstructiva del sueño se produce cuando los músculos localizados en la parte posterior de la garganta se relajan. Estos músculos son los encargados de sujetar el paladar blando, la lengua, la *úvula*, las amígdalas y las paredes laterales de la garganta [2].

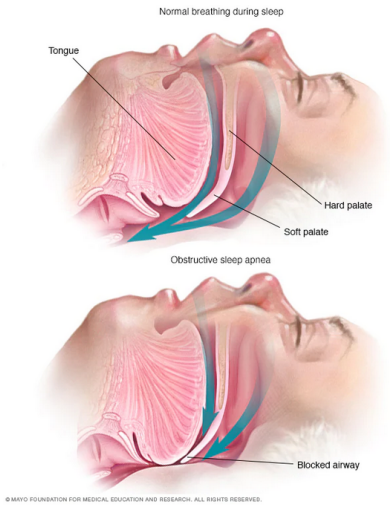

Figura 2.1: Apnea obstructiva del sueño: causas (extraída de [2])

En la figura 2.1 podemos ver en la imagen superior la situación normal del sueño, con un flujo de aire regular entrando por las vías respiratorias. En segundo lugar se muestra la situación anormal previamente descrita: la relajación de los músculos en la parte posterior de la garganta. Esto provoca el cierre o estrechamiento de las vías aéreas. No se recibe suficiente oxígeno, por lo que se reduce el nivel de este en sangre. El cerebro, al detectar esta falta de aire, hace que la persona se despierte por unos instantes para poder abrir de nuevo las vías aéreas [2].

2.1.1.3. Factores de riesgo

Existen ciertos aspectos que aumentan el riesgo de padecer apnea obstructiva del sueño [2], entre los que podemos encontrar: sexo (hombre), antecedentes familiares, obesidad, circunferencia del cuello, vías respiratorias estrechas, consumo de alcohol y tabaco, congestión nasal y ciertas afecciones (insuficiencia cardíaca congestiva, hipertensión arterial, diabetes del tipo 2, Parkinson, etc).

2.1.2. Apnea central del sueño (ACS)

Es un tipo de alteración del sueño en el que la respiración se detiene durante al menos 10 segundos. Se produce debido a que el cerebro no envía la señal necesaria a los músculos que controlan la respiración [6]. La diferencia con la apnea obstructiva del sueño es que en este caso, se observa que el cuerpo realiza un esfuerzo por respirar, pero en el caso de la apnea central del sueño, este esfuerzo no ocurre.

Este tipo de apnea es menos común que la apnea obstructiva del sueño, aunque existe un importante solapamiento entre la patogénesis y la fisiopatología de la apnea obstructiva y la central, por lo que hace que sea difícil distinguir una de otra [7]. Normalmente se considera que la apnea es central en vez de obstructiva cuando más del 50 % de los eventos se pueden clasificar como de tipo central.

La apnea central del sueño, al igual que la apnea obstructiva, puede provocar ciertos problemas como pueden ser: frecuentes despertares nocturnos, somnolencia diurna excesiva y mayor riesgo de efectos cardiovasculares adversos.

La prevalencia de la apnea central del sueño varía mucho entre sus distintos tipos. Los individuos que la sufren, en comparación con los que sufren apnea obstructiva del sueño, son frecuentemente más mayores, con un índice de masa corporal más bajo, puntuaciones más bajas en la *escala de somnolencia de Epworth* y tiene más probabilidades de ser hombres. Entre personas con insuficiencia cardíaca, AOS es mucho más frecuente (alrededor de un 55.1 %) que ACS (alrededor de un 4.1 %) [8].

2.1.2.1. Tipos de ACS

Existen diferentes tipos de apnea central del sueño, entre los que se encuentran los siguientes [6]:

- **Respiración de Cheyne-Stokes**. Está asociado generalmente a la insuficiencia cardíaca congestiva o con accidentes cerebrovasculares. Se caracteriza por un aumento y disminución gradual del esfuerzo para respirar. Puede llegar a producirse una falta total de aire en casos en los que la respiración es más débil.
- **Apnea causada por medicamentos**. Ciertos medicamentos analgésicos de tipo opioide (morfina, oxicodona, codeína, etc.) pueden hacer que la respiración se vuelva irregular, e incluso puede hacer que se detenga por completo temporalmente.
- **Respiración periódica de gran altitud**. Al encontrarse el paciente a una gran altitud, puede producirse un patrón de respiración de Cheyne-Shokes. El cambio del nivel de oxígeno

en el aire respirado a gran altitud provoca hiperventilación alternada con una frecuencia de respiración baja.

- **Apnea central del sueño emergente por tratamiento**. Es una combinación de apneas del sueño obstructivas y centrales. Las personas que padecen AOS pueden desarrollar ACS al usar una máscara de presión positiva continua sobre las vías respiratorias, utilizada para tratar la apnea del sueño.
- **Apnea central del sueño causada por una afección médica**. Algunas afecciones médicas pueden provocar apnea central del sueño del tipo no Cheyne-Stokes. Entre estas se encuentran la enfermedad renal en etapa terminal y el accidente cerebrovascular, entre otras.
- **Apnea central del sueño idiopática**. Como su nombre indica, se desconoce la causa de este tipo de apnea, es poco común.

2.1.2.2. Síntomas

La apnea central del sueño presenta los siguientes síntomas [6]: observación de episodios de pausa en la respiración o patrones de respiración irregular durante el sueño, dificultad para permanecer dormido, dificultad para concentrarse, despertar brusco y con falta de aliento, somnolencia diurna excesiva, cambios de humor, dolores de cabeza por la mañana, ronquidos, etc.

Aunque los ronquidos indican cierta obstrucción en las vías respiratorias, también pueden ser síntoma de ciertos casos de apnea central del sueño. Sin embargo, en este último caso, los ronquidos no son tan evidentes como en el caso de la apnea obstructiva del sueño.

2.1.2.3. Causas

Una de las causas de la apnea central sueño se encuentra en enfermedades que afectan a la capacidad del tronco encefálico para controlar la respiración. El tronco encefálico es el encargado de conectar al cerebro con la columna vertebral. También es el responsable de controlar ciertas funciones como el pulso y la respiración [6].

Según el tipo de apnea central del sueño, la causa es diferente. Los diferentes tipos de ACS han sido comentados en el apartado anterior.

2.1.2.4. Factores de riesgo

Existen ciertos aspectos que aumentan el riesgo de padecer apnea central del sueño, entre los que podemos encontrar: sexo (hombres), edad, uso de opioides, trastornos cardíacos, presión

positiva sobre las vías respiratorias y accidentes cerebrovascular, tumores cerebrales o lesiones estructurales en el tronco encefálico [6].

2.1.3. Síndrome de apnea del sueño compleja

Es un tipo de alteración respiratoria que ocurre durante el sueño. Es una combinación de la apnea obstructiva y la apnea central, por eso también puede llamarse: **apnea del sueño mixta**. Al combinar características de los dos tipos de apnea, aumenta la probabilidad de que las paradas respiratorias sean más frecuentes o más prolongadas [9].

Los episodios de apnea mixta suelen comenzar como apnea obstructiva, por lo que se administra al paciente el tratamiento de este tipo de apnea. Sin embargo, tras el correcto funcionamiento de este tratamiento, desaparece la apnea obstructiva, pero surgen o persisten los eventos de apnea central [9].

La prevalencia oscila entre el 0.56 % y el 18 % [9], sin existir unas características predictivas claras para compararlo con la apnea obstructiva del sueño. El pronóstico de lo pacientes que padecen apnea del sueño mixta es similar al de la apnea obstructiva del sueño.

2.2. Polisomnografía (PSG)

Es un método sistemático de recogida de parámetros de carácter fisiológico durante una noche de sueño en una unidad especializada en el hospital. Un polisomnograma (PSG) es una herramienta que evalúa, mediante múltiples señales fisiológicas, como un *electroencefalograma* (EEG), un *electrooculograma* (EOG), un *electrocardiograma* (ECG), una oximetría de pulso o la respiración torácica y abdominal, las causas subyacentes de los trastornos del sueño [10].

La polisomnografía se utiliza para el diagnóstico de la apnea obstructiva del sueño (AOS), la apnea central del sueño (ACS) y la *hipoxia* relacionada con el sueño. También se puede emplear para evaluar otros trastornos del sueño como: las convulsiones nocturnas, la *narcolepsia*, el trastorno de los movimientos oculares rápidos y el trastorno del movimiento periódico de las extremidades [10].

Gracias a los grandes avances tecnológicos de hoy en día es posible, en determinadas circunstancias, realizar este tipo de pruebas de diagnóstico de apnea del sueño en casa de los pacientes. Un ejemplo de posible uso de este tipo de pruebas domésticas es el conjunto de datos que se va a utilizar en este trabajo.

2.2.1. Procedimiento

Una polisomnografía (PSG) requiere un sistema de monitorización completo para registrar las fases del sueño, el flujo de aire, el esfuerzo respiratorio, los movimientos de las extremidades, la frecuencia cardíaca, la saturación de oxígeno y la posición del cuerpo. Este tipo de estudio se realiza en un laboratorio del sueño con un técnico presente durante toda la realización de la prueba [10].

2.2.1.1. Fases del sueño

Las fases del sueño se determinan utilizando información procedente de un *electroencefalograma*, un *electrooculograma* y un *electromiograma*. La actividad cerebral de las regiones frontal, central y occipital del cerebro, el movimiento de los ojos y un electromiograma submentoniano se utilizan para determinar las cinco etapas de sueño y vigilia posibles: despierto (W), etapa 1 (N1), etapa 2 (N2), etapa 3 (N3), sueño REM (*Rapid Eye Movement*).

En el caso del EEG, los electrodos se colocan en la cabeza de acuerdo con el Sistema 10-20, como se puede ver en la imagen 2.2.

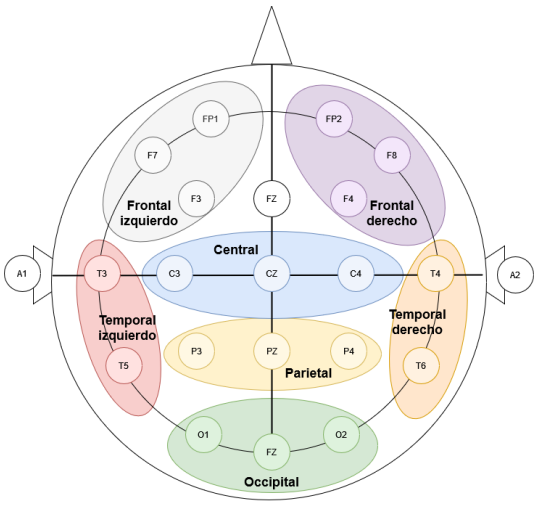

Figura 2.2: Sistema 10-20 de posicionamiento de electrodos en EEG

El EMG del mentón se emplea para diferenciar la fase REM de sueño de la vigilia y de otras fases del sueño. Se sitúa un electrodo por encima del borde inferior de la mandíbula y otros dos por debajo de la misma [10].

Un cambio brusco en la frecuencia del EEG o frecuencias más altas de 16 Hz durante al menos tres segundos, nos indican que el paciente se ha despertado durante las fases de sueño N1, N2 o N3. Estos despertares se evalúan durante el sueño y pueden utilizarse para calificar algunos eventos respiratorios, como las hipopneas.

2.2.1.2. Flujo de aire y esfuerzos respiratorios

Se emplean dos sensores cualitativos del flujo de aire. El primero de ellos es un termistor nasal, que descubre los cambios de temperatura del flujo de aire inspirado frente al espirado para detectar los eventos de apnea. El segundo sensor es un transductor de presión nasal. Detecta los cambios de presión durante la inspiración y la espiración para monitorizar el flujo de aire. Con ello se detectan las hipopneas [10].

También, se utilizan dos sensores o cinturones de inductancia respiratoria (RIP) que detectan cambios en el esfuerzo respiratorio del torax y el abdomen. Estos cinturones emiten cambios de voltaje correspondientes al movimiento del tórax y del abdomen durante la respiración.

Todos estos sensores permiten determinar los diferentes tipos de eventos respiratorios: apneas obstructivas, apneas centrales e hipopneas.

2.2.1.3. Saturación de oxígeno

La saturación de oxígeno se obtiene a través de un oxímetro de pulso. Esta métrica se utiliza para ayudar a evaluar las hipopneas, cuando la saturación de oxígeno es de al menos un 3 % o un 4 %. Además, se utiliza para determinar si se necesita un suplemento nocturno de oxígeno.

2.2.1.4. Electrocardiograma

Un electrocardiograma con electrodos situados en el tronco se utiliza para monitorizar la frecuencia y el ritmo cardíaco. Con un único electrocardiograma, las arritmias complejas no siempre pueden quedar definidas, mientras otras enfermedades cardíacas como las taquicardias o la fibrilación auricular son fácilmente detectables.

2.2.1.5. EMG en extremidades

Para detectar el movimiento de las piernas, se colocan electrodos en los músculos tibiales izquierdo y derecho. Se evalúan los movimientos de las extremidades en función de un aumento de la amplitud de 8 mV sobre el EMG en reposo con una duración de 0.5 a 10 segundos.

2.2.1.6. Posición del cuerpo

Un técnico documenta la posición del cuerpo durante el sueño mediante un vídeo o un instrumento de monitorización de la posición pegado al paciente. Grabar la posición de pacientes que padecen apnea obstructiva es muy importante ya que los eventos respiratorios suelen ser más frecuentes al dormir hacia arriba.

2.2.2. Tipos de polisomnografías

Existen diferentes tipos de polisomnografías, entre los que se encuentran: polisonmografía base, estudio de la presión positiva en las vías respiratorias y estudio nocturno dividido.

2.2.2.1. Polisomnografía base

Una polisomnografía base incluye las mediciones de EEG, EOG, EMG de barbilla y extremidades, señales de flujo aéreo, señales de esfuerzo respiratorio, saturación de oxígeno, ECG y posición del cuerpo para evaluar las interrupciones del sueño.

2.2.2.2. Estudio de la presión positiva en las vías respiratorias (PAP)

El estudio de la presión positiva de las vías respiratorias se utiliza para evaluar las terapias PAP para controlar la distrofia simpático-refleja. Las terapias PAP utilizan una máquina para bombear aire bajo presión dentro de las vías respiratorias [10].

Las mediciones que deben tomarse son muy similares a las de la polisomnografía base. La principal excepción es que se utiliza un cable de PAP para medir el flujo aéreo de aire, en vez de utilizar un transductor de presión nasal y un termistor.

2.2.2.3. Estudio nocturno dividido

Combina los dos tipos anteriores de polisomnografía, permitiendo el diagnóstico y evaluación del tratamiento de los trastornos respiratorios relacionados con el sueño. Se recomienda este estudio para casos graves de apnea obstructiva con un índice de apnea-hipopnea mayor de 40.

2.3. Pruebas de apnea del sueño en casa (HSAT)

Es una alternativa a la polisomnografía para el diagnóstico de la apnea obstructiva del sueño. Recoge un menor número de señales que la PSG en el hospital. Entre las señales que se recogen destacan: flujo de aire, esfuerzo respiratorio, nivel de oxígeno en sangre, y ritmo cardíaco. Existen otras métricas secundarias que también pueden registrare: frecuencia de ronquido, volumen y movimientos del cuerpo. A diferencia de la PSG en el hospital, no se recoge información de la actividad cerebral [11].

Únicamente puede utilizarse en casos de AOS en pacientes sin complicaciones que tienen riesgo medio o alto de padecerlo y sin otro tipo de trastornos del sueño. No está recomendado para pacientes de más de 65 años ya que aún no se ha estudiado suficientemente en estos grupos de edad. Tampoco para pacientes con una masa corporal $> 40kg/cm^2$ debido a que, como no se monitorizan los niveles de CO_2, puede confundirse con el diagnóstico de padecer síndrome de hiperventilación por obesidad. Existen otros criterios de exclusión para este estudio de la apnea del sueño en casa:

- Enfermedad cardiorrespiratoria significativa.
- Debilidad de los músculos respiratorios debido a enfermedad neuromuscular.
- Consumo crónico de opiáceos.
- Despertar o alto riesgo de hipoventilación relacionada con el sueño.
- Reciente accidente cerebrovascular.
- Insomnio severo.
- Síntomas de otros trastornos del sueño.
- Razones personales o ambientales que puedan llevar a una mala interpretación de los resultados.

Las principales ventajas de HSAT sobre la polisomnografía son un gran ahorro de costes y un mejor acceso a la atención del paciente. En un entorno clínico apropiado, HSAT ha demostrado que no es peor técnica que la polisomnografía [10]. Además, es mucho más cómodo para el paciente, debido a que puede dormir en su casa no debe estar conectado a tantos sensores.

Capítulo 3

Contexto tecnológico

En este capítulo se va a introducir la base de datos estudiada, las herramientas utilizadas para el desarrollo de este trabajo y una introducción teórica a las redes neuronales que van a desarrollarse en capítulos posteriores.

3.1. SHHS

SHHS (Sleep Heart Health Study) [12] es un estudio diseñado para investigar la apnea obstructiva del sueño (AOS) y otros trastornos respiratorios del sueño (SDB) como factores de riesgo para el desarrollo de enfermedades cardiovasculares. Es un estudio multicentro (llevado a cabo en diferentes hospitales) desarrollado por el National Heart Lung y Blood Institute, cuyo principal objetivo es determinar las consecuencias cardiovasculares y de otros tipos producidos por trastornos respiratorios del sueño.

La principal hipótesis de este estudio [13] es probar que los trastornos respiratorios en el sueño están asociados a un aumento del riesgo de padecer enfermedades coronarias, ictus, accidentes cerebrovasculares, aumento longitudinal de la presión arterial, así como la mortalidad por todas las anteriores causas.

En este estudio participaron 6441 personas, tanto hombres como mujeres, con más de 40 años, entre los años 1995 y 1998. A este estudio se le denominó SHHS1. Posteriormente, se llevó a cabo una segunda etapa entre los años 2001 y 2003 en la que se repitió el estudio a 2651 de los individuos que ya participaron en la primera etapa. A este estudio se le llamó SHHS2. Todos los sujetos de estudio fueron sometidos a una polisomnografía, generalmente en sus hogares, para evaluar la presencia de apnea obstructiva del sueño (AOS) u otro tipo de trastornos respiratorios (SDB). En las polisomnografías se llevaron a cabo las siguientes mediciones:

- Electroencefalografías (EEG) de C3/A2 y C4/A1 EEGs, muestreadas a 125 Hz.
- Electrooculogramas (EOG) tanto para el ojo derecho como para el izquierdo, muestreados a 50 Hz.
- Electromiograma submentoniano bipolar (EMG) muestreado a 125 Hz.
- Movimiento torácico y abdominal (THOR y ABDO), registrado por bandas de pletismografía inductiva y muestreado a 10 Hz.
- Flujo de aire (*airflow*) detectado por un termopar nasal-oral muestreado a 10 Hz.
- Oximetría de pulso en la yema del dedo, muestreado a 1 Hz.
- Electrocardiograma (ECG) de una derivación bipolar, muestreado a 125 Hz y a 250 Hz, según el estudio.
- Frecuencia cardíaca (PR) derivada del ECG y muestreada a 1 Hz.
- Posición del cuerpo, utilizando un sensor de calibre de mercurio.
- Luz ambiental, encendida o apagada por un sensor de luz

3.1.1. Diseño del estudio

Como se comentó antes, en este estudio participaron alrededor de 6600 personas, de los que aproximadamente un tercio padecía una enfermedad cerebrovascular o cardiovascular en el momento de inscribirse para participar. Por tanto, alrededor de unas 4000 personas estaban a riesgo de sufrir estas enfermedades.

El criterio para participar en este estudio, además de tener más de 40 años, es no haber recibido ningún tratamiento para la apnea del sueño, ni haberle sido realizada una traqueotomía ni estar recibiendo una oxigenoterapia en sus domicilios. Estos datos se recogieron previamente a su inscripción a la participación en el estudio. Además, se llevó a cabo un seguimiento periódico de la mortalidad cardiovascular y cerebrovascular. Se prefirió escoger aproximadamente el mismo número de hombres que de mujeres para la muestra. De esta forma, el estudio puede aplicarse a los dos sexos. Se prefirió sobremuestrear a las personas que roncan y que tuvieran menos de 65 años, así se consigue enriquecer la muestra de jóvenes, en los que roncar multiplica entre tres y diez veces el riesgo de padecer apnea obstructiva del sueño. Por último, no se excluyeron a personas con hipertensión ni enfermedades cardiovasculares, aunque se reduzca el riesgo de que ocurran incidentes, ya que el estudio SHHS permitirá determinar la diferencia de riesgo que hay entre personas con prevalencia de la enfermedad y las que no padecen la enfermedad.

Todos estos individuos seleccionados para participar en el estudio fueron elegidos por estar participando en otros estudios, principalmente dirigidos a estudiar enfermedades cardiovasculares. Simplemente se añadió al estudio ya realizado una evaluación del sueño y de la respiración alterada por el sueño. Estos estudios son:

- **Artheriosclerosis Risk in Communities Study (ARIC)** → 1750 participantes.
  El objetivo principal de este estudio es investigar sobre el origen y la naturaleza de la arteroesclerosis y la arteroesclerosis clínica y subclínica en cuatro comunidades americanas.
- **Cardiovascular Health Study (CHS)** → 1000 participantes.
  El objetivo de este estudio es identificar los factores de riesgo de enfermedades coronarias e ictus en adultos.
- **Framingham Heart Study (FHS)** → 1000 participantes.
  Esta investigación aglomeró tanto a hombres como a mujeres con problemas cardíacos y a sus parejas. También se invitó a participar en el estudio a los hijos de estas parejas.
- **Strong Heart Study** → 600 participantes.
  Tiene como objetivo principal la estimación de la mortalidad y la morbilidad a causa de las enfermedades cardiovasculares. Además, compara los distintos factores de riesgo de cardiovascular entre los nativos americanos.
- **New York Hypertension Cohorts** → 1000 participantes.
  En este estudio participaron tres poblaciones pertenecientes a un proyecto denominado Psychosocial Factors and Cardiovascular Disease. La primera población estudia individuos con hipertensión, la segunda, estudia la relación entre el estrés laboral y y la presión sanguínea y la tercera realiza una evaluación cardiovascular completa a los individuos de estudio.
- **Tucson Epidemiologic Study of Airways Obstructive Diseases y el Health and Environment Study** → 900 participantes.
  Consiste en dos estudios de poblaciones que participaron en un programa especial de un centro especializado de investigación, dedicados respectivamente a estudiar la etilogía y la historia de las enfermedades obstructivas de las vías respiratorias.

3.1.2. Obtención de los datos

Los datos pertenecientes a la base de datos de SHHS se han obtenido mediante tres vías:

- Comparativa de información de las bases de datos de las que proceden.
- Cuestionario de hábitos de sueño.
- Polisomnografía en los hogares.

3.1.2.1. Comparativa de información de las bases de datos de las que procede

Una de las características más importantes de SHHS es el uso de datos existentes procedentes de otras bases de datos, por lo que resulta imprescindible comparar los métodos de recolección de datos de éstas. De esta forma, se han recogido tres grupos de variables:

- **Variables críticas para el principal objetivo del estudio**: enfermedad cardiovascular prevalente, hipertensión, diabetes, raza, sexo, edad, cigarrillos por día, consumo de alcohol y de cafeína, altura, peso, colesterol, triglicéridos, medicación, electrocardiograma, etc.
- **Variables para el análisis de subconjuntos específicos**: historial médico no cardiopulmonar, padres, hermanos, acceso a atención sanitaria, actividad física, nivel económico familiar, etc.
- **Variables para el estudio específico de cada base de datos o estudios auxiliares**: fumador pasivo, dieta, consumo de calorías, presión sanguínea durante 24 horas, etc.

3.1.2.2. Cuestionario de hábitos de sueño

Todos los participantes en este proyecto SHHS deben rellenar este formulario, además de todas las personas que participaron en los otros estudios de los que se han obtenido datos. Este cuestionario tiene cuatro propósitos fundamentales:

- Determinar el nivel de ronquidos de potenciales participantes para sobremuestrear a las personas que roncan habitualmente en los estudios con participantes jóvenes.
- Identificar participantes que deberían ser excluidos del estudio.
- Obtener datos pertenecientes a hábitos de sueño y síntomas de los participantes del SHHS que no están disponibles en otros estudios.
- Recoger información perteneciente a los hábitos de sueño de una muestra muy grande.

3.1.2.3. Polisomnografía en los hogares

Previamente se recogieron los siguientes datos en los hogares de los participantes: presión sanguínea en estado de reposo, peso, circunferencia del cuello, preguntas sobre enfermedades cardiovasculares y respiratorias y consumo de tabaco, medicación actual y calidad de vida.

La polisomnografía se llevó a cabo mediante un polisomnógrafo de la marca *Compumedics P-Series Sleep Monitoring System*. Esta herramienta consiste en un *Patient Interface Box* (PIB), que contiene amplificadores y filtros a los que están conectados unos electrodos. El PIB está unido mediante un cable a un dispositivo de almacenamiento en estádo sólido (PCMCIA *Memory Card*), una batería recargable de 15 horas y un oxímetro. El PIB y los electrodos están sujetos al cuerpo del paciente mediante una malla *bib*. Los sensores se colocan en los pacientes y los equipos se calibran durante la visita médica a casa del paciente. Los datos se almacenan en tiempo real en la memoria de PCMIA y se envían directamente a un centro de lectura de polisomnografía, para obtener una evaluación de calidad de todos los datos.

Tras la noche en la que el paciente se somete a la polisomnografía, se realiza un cuestionario para evaluar la calidad del sueño.

3.2. Herramientas utilizadas

En esta sección se van a comentar las tres herramientas fundamentales que se han utilizado para la elaboración de este trabajo de fin de grado: PyTorch, MATLAB y Comet ML.

3.2.1. PyTorch

PyTorch [14] es una biblioteca de aprendizaje automático *open source* utilizada ampliamente en aplicaciones de visión artificial y procesamiento del lenguaje natural. Ha sido desarrollado por el AI Research Lab de Facebook. Algunas de sus características más importantes son: el aprovechamiento de las tarjetas gráficas como *hardware* para llevar a cabo cálculos, la obtención de gradientes de forma automática y la construcción dinámica de los grafos de computación, permitiendo modificar la estructura interna del mismo en tiempo de ejecución.

A pesar de la existencia de una extenso abanico de bibliotecas de aprendizaje automático, tal y como pueden ser Tensorflow, Keras, Apache MXNet, Theano o Microsoft CNTK, la flexibilidad y grandes posibilidades de desarrollo que ofrece la biblioteca de Facebook, la fuerte comunidad que la respalda y su creciente tendencia como software predominante en la comunidad científica y académica de inteligencia artificial ha decantado la balanza para que todos y cada uno de los modelos de aprendizaje automático que se han implementado en este trabajo se hayan construido con dicho *framework*. Concretamente, de entre las dos APIs (*Application Programming Interface*) que ofrece PyTorch, la de Python y la de C++, se ha elegido la primera de ellas.

Una de las estructuras básicas que ofrece PyTorch son los tensores, los cuales pueden ser tratados y operados de manera muy similar a un `np.array` de la biblioteca NumPy, con la particularidad de que los tensores pueden ser alojados en la GPU para acelerar los cálculos en los

que se vean envueltos. Un tensor, como el que podemos ver en la figura 3.1, es una estructura matemática perteneciente a un espacio vectorial tensorial que es invariante ante un cambio de base [15].

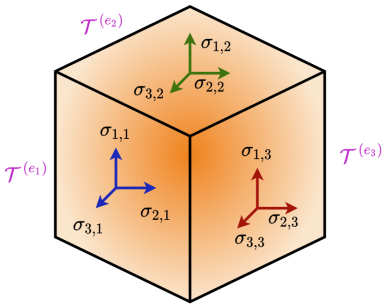

Figura 3.1: Tensor en $\mathbb{R}^3$.

En PyTorch, existen cinco elementos fundamentales en la elaboración de proyectos de *machine learning* [16]:

- **Clase *Dataset***: Es una clase abstracta que nos permite crear un *dataset* propio. Al menos debe tener dos funciones: una para acceder al elemento iésimo del conjunto de datos (`__getitem__()`) y otra para mostrar el número de instancias (`__len__()`). A continuación se muestra un ejemplo de clase *Dataset* en la que se toma como parámetros un conjunto de atributos de un individuo (`x`) y el valor de la clase para este (`y`).

```
class MyDataset(Dataset):
    def __init__(self, x, y) -> None:
        self.x = x
        self.y = y

    def __getitem__(self, index):
        x_ind = self.x[index]
        y_ind = self.y[index]
        x = torch.from_numpy(x_ind).float()
        y = torch.Tensor([y_ind]).float()
        return x.reshape(1, x.shape[0], x.shape[1]), y

    def __len__(self):
        return len(self.x)

dataset = MyDataset(x, y)
```

- **Clase *Dataloader***: Es una clase que permite iterar sobre el *dataset*. Además, permite personalizar el orden de las instancias, cargar datos en paralelo, cargar los datos por *batches* o no, y hacer copias de los datos desde el *host* a la GPU. A continuación se muestra un ejemplo de esta clase:

```
dataloader = DataLoader(dataset, batch_size = 32, shuffle = True)
```

- **Modelo**: Es una clase en la que definimos el modelo que queremos entrenar. Podemos utilizar modelos ya entrenados (mediante *transfer learning*) o podemos crear nuestro propio modelo. En este segundo caso, la clase modelo debería tener al menos dos métodos: el constructor (`__init__(self)`), en el que se definen las capas del modelo, y otro método en el que se defina el orden del flujo de los datos por la red (`forward()`). A continuación se muestra un ejemplo de cómo definir un modelo. En este caso solo se añade la primera capa, la definición del resto de capas se haría de forma análoga.

```
class model(nn.Module):
    def __init__(self):
        super().__init__()

        self.layer1 = nn.Sequential(
            nn.Conv2d(in_channels = 1, out_channels = 4, kernel_size = ...
   (1,128), stride=(1,2)) ,
            nn.BatchNorm2d(num_features = 4),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1)     )

        # Define more layers here...
        self.lin = nn.Linear(in_features = 128*2*61, out_features = 1)

def forward(self,x):
    out = self.layer1(x)

    # ...

    out = out.reshape(out.shape[0], out.shape[1]*out.shape[2]*out.shape...
   [3])  # Flatten
    out = self.lin(out)
    return out
```

- **Función de pérdida y optimizador**: A partir de la librería `torch.nn` podemos definir la función de pérdida y el optimizador que se desean utilizar. En el siguiente bloque de código se muestra un ejemplo.

```
criterion = torch.nn.HuberLoss(reduction= "mean")
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
```

- **Bucle de entrenamiento**: En PyTorch, no existe una instrucción sencilla para entrenar el modelo, sino que hay que programarlo paso a paso. En primer lugar se debe definir un bucle con el número de épocas que se desee entrenar el modelo y cargar por *batches* los datos a través del *Dataloader*. Debemos cargar todos los datos en la GPU. En el siguiente bloque de código podemos verlo:

```
for i,data in enumerate(dataloader, 0):
    inputs, labels = data[0].to('cuda'), data[1].to('cuda')
```

A continuación, se debe dar valor 0 a todos los gradientes, antes de empezar la etapa de retropropagación del error:

```
optimizer.zero_grad()
```

Posteriormente se introduce en el modelo el bloque de datos cargado en un *batch*, obteniéndose un resultado que se compara con los valores de las etiquetas y se evaluará mediante la función de pérdida que hemos definido previamente:

```
outputs = model(inputs)
loss = criterion(outputs, labels)
```

Por último, se optimiza el *loss* en la retropropagación del error y se actualiza el valor de los pesos del modelo a partir de los gradientes calculados:

```
loss.backward() # hago derivadas (backpropagation)
optimizer.step() # actualizo los pesos
```

3.2.2. MATLAB

MATLAB (MATrix LABoratory) [17] es una plataforma de programación y cómputo numérico muy utilizada para el análisis de datos, desarrollo de algoritmos y creación de modelos. Cuenta con un lenguaje de programación propio: M y cuenta con un formato contenedor de datos binarios propio, la extensión `.mat`. El lenguaje M es interpretado y se puede ejecutar tanto en un entorno interactivo como mediante *scripts* con la extensión `.m`.

En este proyecto se ha utilizado MATLAB para todo el procesado de las señales, generación de etiquetas y generación de espectrogramas a partir de las señales, debido a que posee unas funciones muy potentes y sencillas de utilizar para el diseño de filtros y en general, todo el tratamiento de señales.

3.2.3. Comet ML

En una plataforma de experimentación que permite monitorizar proyectos de *machine learning* y visualizar cómodamente en *dashboards* los resultados de los experimentos [18]. Puede integrarse fácilmente con múltiples librerías de *machine learning*, como: PyTorch, Fastai, Tensorflow, Keras, etc.

Al principio de cada experimento debemos crearlo también en Comet ML añadiendo este pequeño fragmento de código:

```
experiment = Experiment(
    api_key="API KEY",
    project_name="nombre_del_proyecto",
    workspace="nombre_de_usuario",
)
experiment.set_name("Nombre del experimento")
```

A continuación guardamos los hiperparámetros del modelo que vamos a entrenar:

```
hiperparametros = {
    'epocas' : 200,
    'lr' : 0.001,
    'batch_size' : 32,
    'seed' : 56389856,
    'dropout' : 0.1
}
```

Para monitorizar el desempeño del modelo en cada época, almacenamos el valor de las métricas de interés en cada época. Al hacer esto, luego Comet ML elaborará una serie de gráficos con los resultados obtenidos. Por ejemplo, para el *accuracy* en validación:

```
experiment.log_metric('Accuracy validation',accuracy)
```

También nos permite almacenar gráficos generados con la librería *matplotlib* de la siguiente forma:

```
experiment.log_figure(figure_name='Accuracy_plot', figure = plt)
```

Existen otras múltiples métricas que pueden almacenarse en Comet ML [19], pero estas son las que se han empleado en este proyecto.

3.3. Redes neuronales

En esta sección se introducirán las redes neuronales desde sus orígenes, sus conceptos fundamentales, un tipo concreto: las redes neuronales convolucionales y sus problemas más habituales.

3.3.1. Contextualización

Se conoce la **inteligencia artificial** como un conjunto de algoritmos planteados con el propósito de imitar la inteligencia humana. Es capaz de aprender y solucionar problemas. Un ejemplo de esto son los sistemas expertos basados en reglas.

La IA es un campo muy amplio, dentro del cual se engloba otras tecnologías, se puede ver en el diagrama de Venn de la figura 3.2.

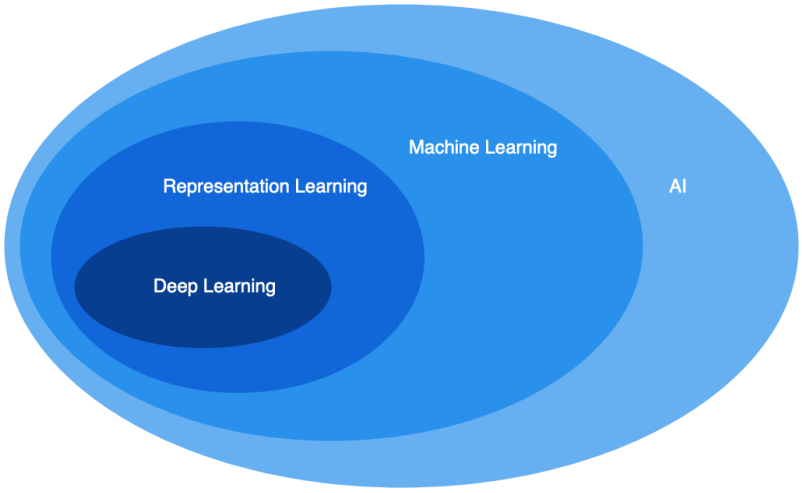

Figura 3.2: Diagrama de Venn de IA

El ***machine learning*** es una parte la inteligencia artificial, al que también se puede llamar aprendizaje automático, debido a su capacidad para aprender de forma automática y mejorar

a partir de la experiencia sin ser programado de forma explícita. En el *machine learning*, una máquina está entrenada para alcanzar un grado de automatización que resultaría imposible para un ser humano. El resultado de este proceso de aprendizaje permite realizar predicciones. Ejemplos de esto pueden ser: *k-means*, *k-nearest neighbours* (KNN), *support vector machine* (SVM) o árboles de decisión.

El ***representation learning*** es un aspecto del *machine learning* que automáticamente descubre patrones de características en los datos. Funciona reduciendo la alta dimensionalidad de los datos a otros de baja dimensión, facilitando el descubrimiento de patrones y anomalías y proporcionando una mejor comprensión del comportamiento general de los datos. Un ejemplo de esto pueden ser los *shallow autoencoders* [20].

En los métodos de ***deep learning***, los modelos son mucho más profundos que los métodos de *machine learning*, es decir, tienen un número de capas muy superior. Además, no es necesario extraer información de las señales o las imágenes previo a la etapa de aprendizaje, porque lo hace el propio método de *deep learning*. Algunos ejemplos son: *convolutional neural network* (CNN), *recurrent neural network* (RNN) y *generative adversarial networks* (GAN).

3.3.2. Orígenes

En el año 1943, el neurólogo Warren McCulloch y el matemático Walter Pitts presentaron por primera vez un modelo matemático de una neurona, que intenta simular el comportamiento de una neurona cerebral. Esta neurona es la unidad esencial con la que se construye una Red Neuronal Artificial. Este modelo era capaz de obtener una respuesta binaria a partir de una o varias entradas binarias, como se puede ver en la figura 3.3, donde $x_1, x_2, ..., x_n$ son el conjunto de variables binarias de entrada, g es la función que toma la entrada, y f es una función que realiza una agregación. En base al resultado de esta agregación, elige cuál será la salida de la neurona (y).

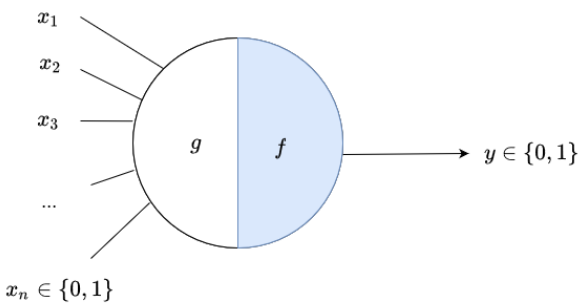

Figura 3.3: Primer modelo de neurona

La función de activación original era una activación binaria, intentando simular la excitación de las neuronas biológicas y el intercambio de información.

$$F(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.1}$$

Basado en este trabajo, en el año 1958, Frank Rosenblantt [24] ideó lo que hoy se conoce como perceptrón simple. Este modelo podemos verlo en la figura 3.4 y consta de los siguientes elementos:

- **Entradas**: X_i, es el valor de la entrada i-ésima .
- **Pesos**: W_{ki}, son los pesos para la entrada iésima.
- ***Bias***: b_k, es el valor añadido a la suma del producto de las entradas por sus correspondientes pesos. También se le conoce como *threeshold* o como valor umbral.
- **Función de activación:** φ, calcula un nuevo valor a partir de las entradas: el producto del *bias* por la suma del producto de las entradas por sus correspondientes pesos.
- **Salida**: Y_k, salida obtenida tras el procesamiento de los datos a través de las neuronas.

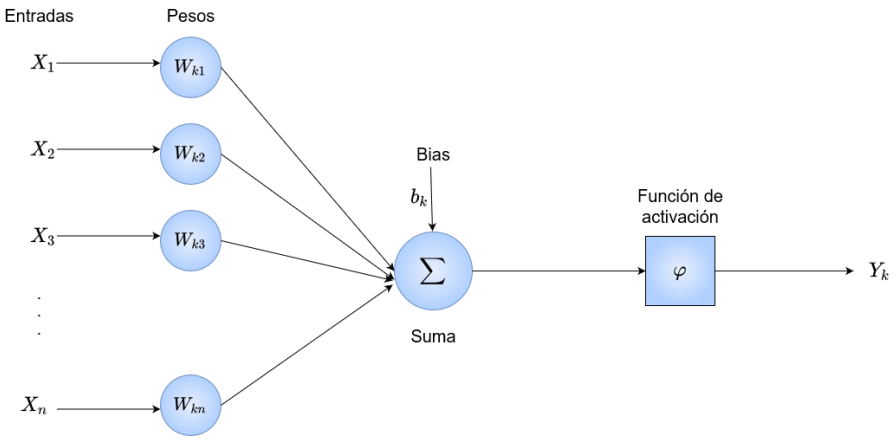

Figura 3.4: Modelo de Perceptrón Simple

Podemos ver la similitud de este modelo de neuronas con la neurona biológica si observamos la figura 3.5. Las dendritas son las encargadas de recibir la señal de otras neuronas, la entrada de nuestro modelo (X_n). El cuerpo celular (también llamado soma) es el encargado de procesar

la información, se corresponde con los nodos de la red. El axón es el encargado de transmitir la salida de la neurona, haciendo las veces salida de la red neuronal.

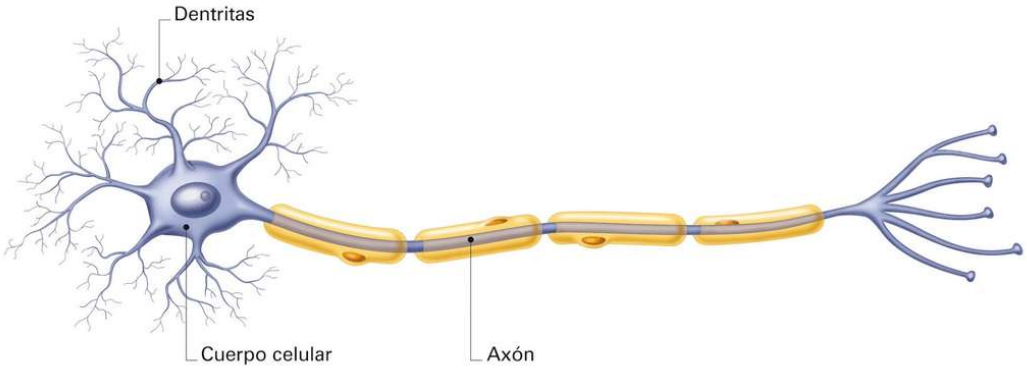

Figura 3.5: Neurona biológica (extraída de [25])

3.3.3. Generalidades de las redes neuronales

Toda red neuronal está compuesta por unos elementos básicos que se proceden a comentar a continuación. La información contenida en este apartado ha sido extraída de [21] y [35].

3.3.3.1. Neurona

Es el elemento básico de cómputo en las redes neuronales. Consta de un conjunto de entradas y unos pesos asociados a cada entrada. Para un conjunto de `n` entradas, la neurona realiza una suma ponderada de estas entradas, multiplicadas por los pesos de la siguiente forma:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} w_i \cdot x_i \tag{3.2}$$

Cada neurona tiene asociado un vector de pesos de dimensión `n` $(w_1, w_2, ..., w_n)$. Este vector de pesos va variando durante el entrenamiento de la red neuronal, de forma que se obtenga un resultado final lo más óptimo posible. Tras la estimación de los pesos w_i, la neurona representa un hiperplano en $\mathbb{R}^n$.

A la suma ponderada de las entradas, ocasionalmente, se le puede añadir un *bias* (`b`), que suele tener un valor 1. Este término permite mover de izquierda a derecha el hiperplano que calcula la neurona. El cálculo que realiza la neurona añadiendo el término *bias* es:

$$f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n}(w_i \cdot x_i) + w_{n+1} \cdot b \tag{3.3}$$

3.3.3.2. Función de activación

Las funciones de activación son funciones matemáticas que se aplican a la salida de la neurona para calcular un nuevo valor a partir del resultado de la suma ponderada de las entradas.

La elección de la función de activación tiene una gran impacto en la capacidad y el rendimiento de la red. Pueden utilizarse diferentes funciones de activación en distintas partes del modelo: puede ser distinta para todas las neuronas del modelo, puede ser la misma para todas las neuronas de una capa o puede ser la misma para toda la red. El primer caso es poco común, ya que normalmente las redes están diseñadas para utilizar la misma función de activación para todos los nodos de una capa [26].

Las funciones de activación más interesantes tienen un conjunto de propiedades deseables entre las que se encuentran las siguientes [27]:

- **No linealidad**: introduce transformaciones no lineales a la entrada, permitiendo a la red que aprenda y que haga tareas más complejas.
- **Diferenciable**: permite métodos de optimización basados en el descenso del gradiente.
- **Rango**: la salida debe estar acotada.
- **Monótona**: función siempre creciente. Permite garantizar que la superficie de error asociada a un modelo de una sola capa sea convexa y, por tanto, haya un único máximo local, que también es global.
- **Se aproxima a la identidad en el origen**: evita la influencia de la aleatoriedad de los pesos iniciales.

3.3.3.2.1. Función lineal

La ecuación para la función de activación lineal es la siguiente:

$$f(x) = a \cdot x \tag{3.4}$$

Existe un caso especial cuando $a = 1$ y $f(x) = x$, al que denominamos "**identidad**".

Esta función tiene rango $(-\infty, +\infty)$. Proporciona una superficie de error convexa para que la optimización sea más rápida. Además, su derivada $df(x)/dx = a$ es constante.

En la figura 3.6 podemos ver la función $f(x) = x$ representada.

3.3.3.2.2. Función signo

La función signo transforma la entrada en valores -1 si ésta es negativa, o 1 si es positiva.

$$f(x) = \begin{cases} +1 & \text{si x} \geq 0 \\ -1 & \text{resto} \end{cases} \tag{3.5}$$

Es una función antisimétrica, monótona, discontinua y no se aproxima a la identidad en el origen. No es derivable para $x = 0$. Podemos ver su función de densidad en la figura 3.7.

3.3.3.2.3. Función sigmoide

La función sigmoide (también llamada función logística) introduce el concepto de probabilidad en las redes neuronales. Su salida se encuentra en el rango $(0, 1)$.

$$f(x) = \frac{1}{1 + e^{-x}} = \frac{e^x}{1 + e^x} \tag{3.6}$$

Es una función no lineal, continua, diferenciable, monótona, asimétrica y tiene un rango de salida fijo, pero no está centrado en el cero.

Últimamente está siendo sustituida por la función tangente hiperbólica como función de activación para tareas de clasificación, ya que esta última es antisimétrica y está centrada en el origen. En la figura 3.8 podemos ver su función de densidad.

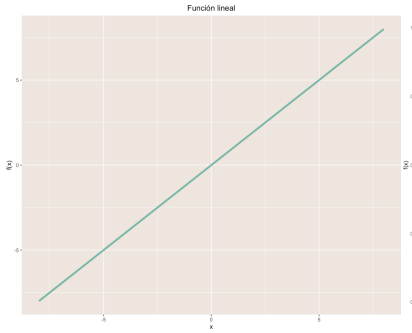

Figura 3.6: Función lineal

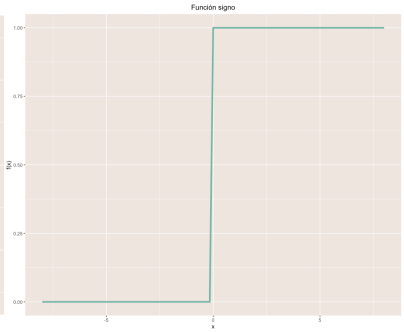

Figura 3.7: Función signo

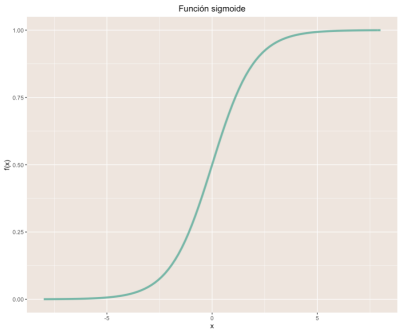

Figura 3.8: Función sigmoide

3.3.3.2.4. Función softmax

También se le denomina: "regresión logística multiclase". Es una generalización de la regresión logística que puede utilizarse para la clasificación multiclase. Su fórmula se parece mucho a la función sigmoide. Puede utilizarse para un clasificador en el que las clases son mutuamente exclusivas. Produce una salida en el intervalo $(0, 1)$. La suma de las salidas es igual a 1.

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \tag{3.7}$$

Al ser una función multiclase, es difícil representarla. Podemos hacer una representación 3d para el caso de dos componentes y $K = 2$. Podemos verlo en la figura 3.9.

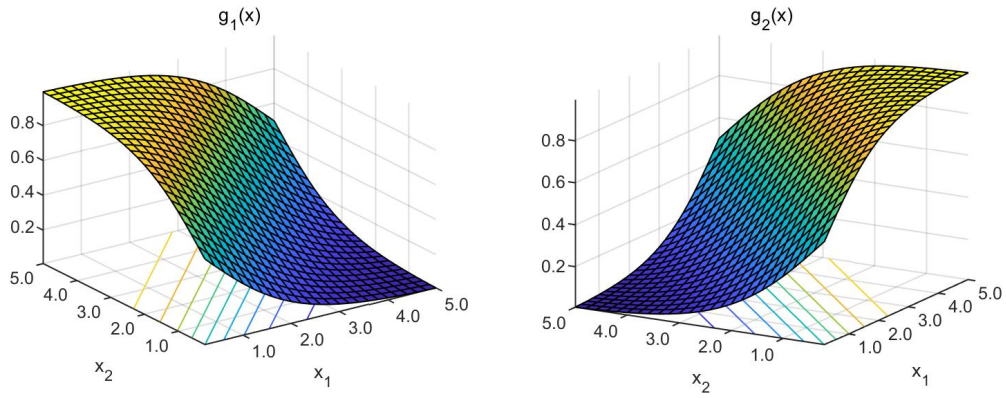

Figura 3.9: Función softmax (extraída de [29])

3.3.3.2.5. Función tangente hiperbólica

Como se ha comentado anteriormente, esta función está empezando a sustituir a la función sigmoide debido a que es antisimétrica y está centrada en el origen.Además es una función continua, monótona y diferenciable.

La función produce salidas en el rango $[-1, 1]$. Su densidad es:

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.8}$$

En la figura 3.10 podemos ver la representación de la función de densidad de la tangente hiperbólica.

3.3.3.2.6. Función ReLU (Rectificador lineal)

Es la función de activación más común en los métodos de *deep learning*. La función de densidad es la siguiente:

$$ReLU(x) = max(0, x) \tag{3.9}$$

Si la entrada es negativa, la salida de ReLU es 0; y si es positiva, la salida es x. Conserva la motivación biológica de la función escalonada (la neurona sólo se dispara si las entradas superan un determinado umbral).

Es una función no lineal, muy sencilla de calcular, su derivada es discontinua en el origen y permite *backpropagation*. Es una función no restringida, lo que significa que no hay un valor máximo. En la figura 3.11 podemos ver su función de densidad.

3.3.3.2.7. Función LeakyReLU

Existe un problema con la función ReLU en el caso de que la mayoría de los valores de entrada sean negativos o 0, ReLU produce una salida de 0, por lo que, en este caso concreto, no se podría hacer *backpropagation*, haciendo además que la tasa de aprendizaje sea demasiado alta y haya un *bias* negativo muy grande. A este problema se le llama *Dying* ReLU. La función de densidad es la siguiente:

$$f(x) = \begin{cases} 0{,}01x & \text{si x} < 0 \\ x & \text{resto} \end{cases} \tag{3.10}$$

Es una función en el rango $(-\infty, \infty)$, monótona, continua pero no diferenciable en 0. Es asimétrica y se aproxima a la identidad en el origen. Podemos ver su función de densidad en la figura 3.12.

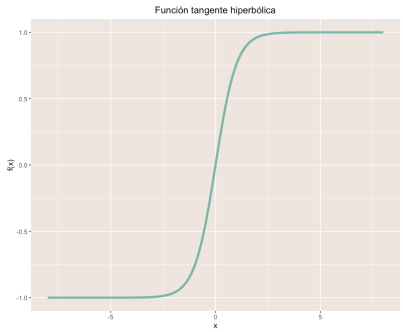

Figura 3.10: Función tangente hiperbólica

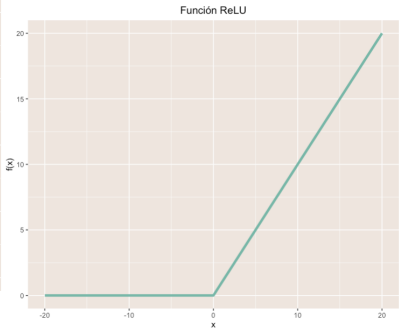

Figura 3.11: Función ReLU

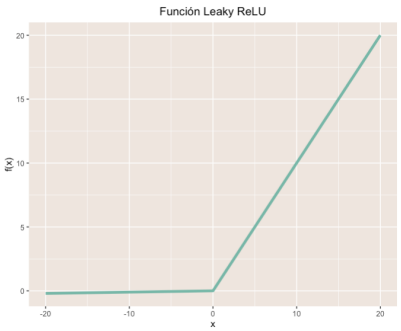

Figura 3.12: Función *leaky* ReLU

3.3.3.3. Entrenamiento

El entrenamiento de una red neuronal consta de tres etapas que se suceden de forma secuencial: alimentación hacia delante (*feedforward*), cálculo de la pérdida (*loss*) y retropropagación (*backpropagation*).

La etapa inicial de *feedforward* consiste en el paso de los datos hacia delante en la red, desde la entrada de la misma (*inputs*), en la primera capa de la red, a través de las capas intermedias y hasta el final de la red. La salida de una capa (`i`) es la entrada de la capa siguiente (`i + 1`).

La siguiente etapa, la llamada de cálculo de pérdida, ocurre tras la finalización de la etapa de *feedforward*, en la que se compara las predicciones con la salida deseada de la red, es decir, las etiquetas. Para hacer una comparación objetiva de cómo de diferentes son, se utilizan las denominadas funciones de pérdida (*loss functions*). Estas funciones de pérdida serán diferentes según el tipo de problema al que nos enfrentamos: regresión o clasificación.

Para el caso de **regresión**, podemos utilizar las siguientes funciones de pérdida [31]:

- **Mean Squared Error (MSE)**: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$

- **Mean Squared Logaritmic Error (MSLE)**: $MSLE = log(y_i + 1) - log(\hat{y}_i + 1) = log\left(\frac{y_i+1}{\hat{y}_i+1}\right)$
- **Mean Absolute Error (MAE)**: $MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2$
- **Hubber Loss**: se utiliza para regresiones robustas, es menos sensible a los *outliers* que el MSE. Se calcula de la siguiente forma:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{si } |a| \leq \delta \\ \delta \cdot \left(|a| - \frac{1}{2}\right)\delta & \text{resto} \end{cases} \tag{3.11}$$

En el caso de **clasificación**, podemos utilizar las siguientes funciones de pérdida:

- **Cross-Entropy Loss Function**:
  - Cuando el número de clases es igual a 2: $-(y_i \cdot \log(\hat{y}_i) + (1 - y) \cdot \log(1 - \hat{y}_i))$
  - Cuando el número de clases es > 2: $-\sum_{c=1}^{M} y_{0,c} \cdot \log(\hat{y_{0,c}})$
- **Hinge Loss**: $\sum_{y \neq t} \max(0, 1 + w_y x - w_t x)$ siendo w_t y w_y los parámetros del modelo.

El objetivo del entrenamiento de una red neuronal es minimizar la función de pérdida. Esto se realiza en la etapa de retropropagación o *backpropagation*, en la que se modifican los pesos de las neuronas para conseguir asignarles un valor correcto que minimice esta función de pérdida.

El método más común para la minimización de la función de pérdida es el método del descenso del gradiente. Es un método numérico de optimización que busca la porción de función en la que se alcanzan los menores valores.

El gradiente es un cálculo numérico que nos permite saber cómo ajustar los parámetros de una red, de forma que el error de la salida de la red sea mínimo. Marca la dirección del camino más rápido para maximizar una función. Como en las redes neuronales lo que deseamos es minimizar una función, deberemos realizar una modificación en los pesos en la dirección opuesta, es decir, desde las capas finales de la red y hasta las capas iniciales. Este proceso es al que debe su nombre la etapa en la que nos encontramos: *backpropagation*. Los gradientes calculados se acumulan desde las capas finales y hasta las iniciales mediante la regla de la cadena.

En la figura 3.13 podemos ver un ejemplo de una red neuronal sencilla, con una capa de entrada, dos capas ocultas y una capa de salida. A partir de esta pequeña red, vamos a explicar

las diferentes etapas que se acaban de mencionar. Para el desarrollo del ejemplo han sido utilizadas las referencias [32] y [33].

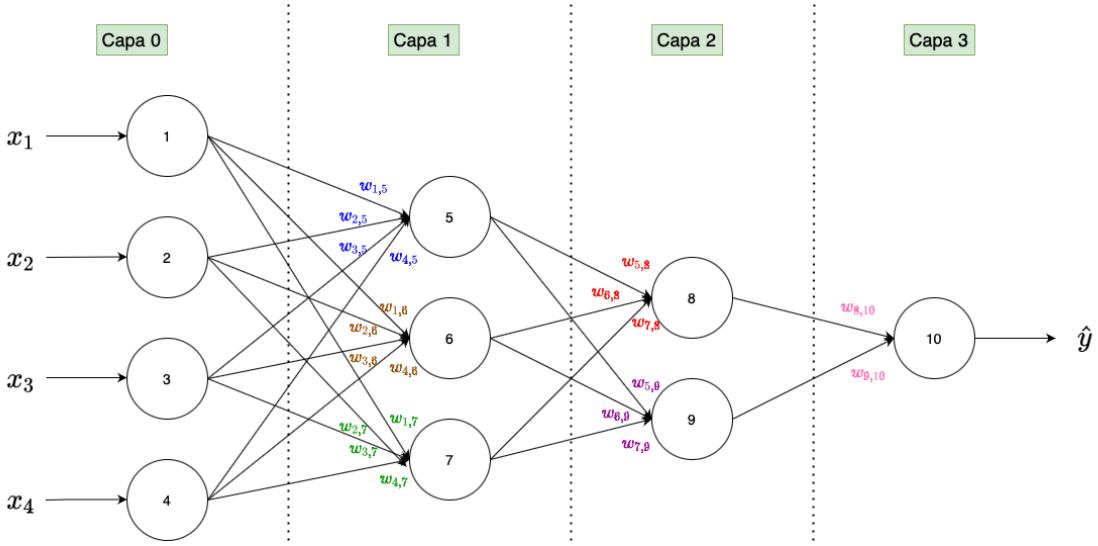

Figura 3.13: Ejemplo de red neuronal

Los pesos de cada una de las conexiones entre dos neuronas de dos capas diferentes, son los siguientes:

$$w^1 = \begin{bmatrix} w^1_{1,5} & w^1_{1,6} & w^1_{1,7} \\ w^1_{2,5} & w^1_{2,6} & w^1_{2,7} \\ w^1_{3,5} & w^1_{3,6} & w^1_{3,7} \\ w^1_{4,5} & w^1_{4,6} & w^1_{4,7} \end{bmatrix} \quad w^2 = \begin{bmatrix} w^2_{5,8} & w^2_{6,9} \\ w^2_{6,8} & w^2_{7,9} \\ w^2_{7,8} & w^2_{8,9} \end{bmatrix} \quad w^3 = \begin{bmatrix} w^3_{8,10} \\ w^3_{9,10} \end{bmatrix}$$

Estos, además, son los pesos que se deben estimar para calcular la salida de la red. Este cálculo podemos verlo en la siguiente ecuación:

$$\begin{aligned} \hat{y} = \varphi(w^3_{8,10} \cdot \varphi(w^2_{5,8} \cdot \varphi(w^1_{1,5}x_1 + w^1_{2,5}x_2 + w^1_{3,5}x_3 + w^1_{1,5}x_4) + \\ w^2_{6,8} \cdot \varphi(w^1_{1,6}x_1 + w^1_{2,6}x_2 + w^1_{3,6}x_3 + w^1_{1,6}x_4) + \\ w^2_{7,8} \cdot \varphi(w^1_{1,7}x_1 + w^1_{2,7}x_2 + w^1_{3,7}x_3 + w^1_{1,7}x_4)) + \\ w^3_{9,10} \cdot \varphi(w^2_{5,9} \cdot \varphi(w^1_{1,5}x_1 + w^1_{2,5}x_2 + w^1_{3,5}x_3 + w^1_{1,5}x_4) + \\ w^2_{6,9} \cdot \varphi(w^1_{1,6}x_1 + w^1_{2,6}x_2 + w^1_{3,6}x_3 + w^1_{1,6}x_4) + \\ w^2_{7,9} \cdot \varphi(w^1_{1,7}x_1 + w^1_{2,7}x_2 + w^1_{3,7}x_3 + w^1_{1,7}x_4))) \end{aligned} \tag{3.12}$$

El objetivo es minimizar una función de pérdida. Para este ejemplo vamos a utilizar la métrica de MSE y vamos a llamarlo C, ya que esta función de pérdida representa el **coste** de equivocarnos en la estimación de la salida de la red.

$$MSE = C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.13}$$

En la etapa de *feedforward*, siendo a^L la función de activación, w^L los pesos, y b^L el *bias* (todos para la capa L, en este caso, la capa 3), podemos calcular la suma ponderada de los pesos para cada capa de la siguiente forma:

De forma general:

$$\boxed{\begin{array}{lcl} z^{(L)} & = & w^{(L-1)}a^{(L-1)} + b^{(L-1)} \\ a^{(L)} & = & \sigma(z^{(L)}) \end{array}} \tag{3.14}$$

En el caso de nuestro problema:

- Primera capa: $z^1 = \begin{bmatrix} w^1_{1,5} & w^1_{2,5} & w^1_{3,5} & w^1_{4,5} \\ w^1_{1,6} & w^1_{2,6} & w^1_{3,6} & w^1_{4,6} \\ w^1_{1,7} & w^1_{2,7} & w^1_{3,7} & w^1_{4,7} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_1 \\ b_1 \end{bmatrix}$

  $a^1 = \sigma(z^1)$

- Segunda capa: $z^2 = \begin{bmatrix} w^2_{5,8} & w^2_{6,8} & w^2_{7,8} \\ w^2_{6,9} & w^2_{7,9} & w^2_{8,9} \end{bmatrix} \cdot \begin{bmatrix} a^1_1 \\ a^1_2 \\ a^1_3 \end{bmatrix} + \begin{bmatrix} b_2 \\ b_2 \end{bmatrix}$

  $a^2 = \sigma(z^2)$

- Tercera capa: $z^3 = \begin{bmatrix} w^3_{8,10} & w^3_{9,10} \end{bmatrix} \cdot \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix} + \begin{bmatrix} b_3 \end{bmatrix}$

  $a^3 = \sigma(z^3)$

En la etapa de *backpropagation*, vamos a analizar como cambia la función de coste si se produce un cambio en los pesos. Este estudio lo realizaremos capa a capa:

Capa L-1:

De forma general:

$$\boxed{\begin{aligned}&\frac{\partial C}{\partial w^{(L-1)}} = \frac{\partial Z^{(L)}}{\partial w^{(L-1)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C}{\partial a^{(L)}} \\ &\left.\begin{aligned}\frac{\partial C}{\partial a^{(L)}} &= 2(a^{(L)} - y) \\ \frac{\partial a^{(L)}}{\partial z^{(L)}} &= \sigma'(z^{(L)}) \\ \frac{\partial Z^{(L)}}{\partial w^{(L-1)}} &= a^{(L-1)}\end{aligned}\right\} = \frac{\partial C}{\partial w^{(L-1)}} = a^{(L-1)} \cdot \sigma'(z^{(L)}) \cdot 2(a^{(L)} - y)\end{aligned}} \tag{3.15}$$

Para un caso concreto, por ejemplo: $w^2_{5,8}$:

$$\frac{\partial C}{\partial w^2_{5,8}} = \frac{\partial Z^3}{\partial w^2_{5,8}} \cdot \frac{\partial a^3}{\partial z^3} \cdot \frac{\partial C}{\partial a^3} = a^3 \cdot \sigma'(z^3) \cdot 2(a^3 - y) \tag{3.16}$$

Capa L-2

De forma general:

$$\boxed{\begin{aligned}&\frac{\partial C}{\partial w^{(L-2)}} = \frac{\partial Z^{(L-1)}}{\partial w^{(L-2)}} \cdot \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \cdot \frac{\partial Z^{(L)}}{\partial a^{(L-1)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial C}{\partial a^{(L)}} = \frac{\partial Z^{(L-1)}}{\partial w^{(L-2)}} \cdot \frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} \cdot \frac{\partial C}{\partial a^{(L-1)}} \\ &\left.\begin{aligned}\frac{\partial a^{(L-1)}}{\partial z^{(L-1)}} &= \sigma'(z^{(L-1)}) \\ \frac{\partial Z^{(L-1)}}{\partial w^{(L-2)}} &= a^{(L-2)}\end{aligned}\right\} = \frac{\partial C}{\partial w^{(L-2)}} = a^{(L-2)} \cdot \sigma'(z^{(L-1)}) \cdot \frac{\partial C}{\partial a^{(L-1)}}\end{aligned}} \tag{3.17}$$

Para un caso concreto, por ejemplo: $w^1_{1,8}$:

$$\frac{\partial C}{\partial w^1_{1,8}} = \frac{\partial Z^2}{\partial w^1_{1,8}} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial Z^3}{\partial a^2} \cdot \frac{\partial a^3}{\partial z^3} \cdot \frac{\partial C}{\partial a^3} = \frac{\partial Z^2}{\partial w^1_{1,8}} \cdot \frac{\partial a^2}{\partial z^2} \cdot \frac{\partial C}{\partial a^2} \tag{3.18}$$

3.3.3.4. Optimizadores

Los optimizadores son algoritmos que sirven para cambiar los pesos y las tasas de aprendizaje para reducir el valor de la función de pérdida. La información extraída para esta sección ha sido adaptada de [34].

Los optimizadores más utilizados son:

Algoritmo de descenso de gradiente

En el apartado anterior hemos comentado cómo funciona el algoritmo de descenso del gradiente utilizado en la etapa de *backpropagation*. Es un algoritmo de primer orden que depende de la derivada de primer orden de la función de pérdida. Calcula de qué forma deben modificarse los pesos para que la función alcance el mínimo y los modifica en la dirección opuesta a la dirección de máximo crecimiento de la función, de la siguiente forma:

$$\Theta := \Theta - a \cdot \nabla J(\Theta) = \Theta - a \cdot \frac{\partial J}{\partial \Theta} \tag{3.19}$$

Este algoritmo es el más sencillo de entender e implementar, así como el más sencillo de computar. Es muy utilizado en los algoritmos de regresión y clasificación.

La desventaja de este optimizador es la posibilidad de quedarse atrapado en mínimos locales en el caso de que la función a optimizar tenga varios mínimos, y puede ocurrir que nunca se alcance el mínimo global. Como los pesos cambian después de calcular el gradiente en todo el conjunto de datos, si este es muy grande, puede llegar a tardar mucho tiempo en alcanzar el valor mínimo. Por este mismo motivo, requiere de una gran cantidad de memoria para calcular el gradiente en todo el conjunto de datos.

Algoritmo de descenso de gradiente estocástico

Es una variante del algoritmo de descenso de gradiente, pero actualizando los pesos de forma más frecuente, se modifican tras el cálculo de la función de pérdida en cada ejemplo de entrenamiento, es decir, si tenemos un conjunto de datos con N instancias, el algoritmo de descenso de gradiente estocástico actualizará los parámetros N veces en cada época, en vez de una sola vez, como ocurre con el algoritmo de descenso de gradiente básico.

Podemos definir el algoritmo de descenso de gradiente estocástico de la siguiente forma:

$$\Theta := \Theta - a \cdot \nabla J(\Theta; x^{(i)}; y^{(i)}) = \Theta - a \cdot \frac{\partial J(\Theta, x^{(i)}; y^{(i)})}{\partial \Theta} \tag{3.20}$$

Entre las ventajas de este algoritmo se encuentran: una velocidad de convergencia menor, debido a las actualizaciones frecuentes de los pesos del modelo, y un menor uso de memoria, debido a no ser necesario el almacenamiento de la función de pérdida.

Este algoritmo también cuenta con ciertas desventajas, por ejemplo: una alta varianza de los pesos del modelo, la posibilidad de seguir disparándose a pesar de haber alcanzado el mínimo, o un descenso de la tasa de aprendizaje si se quiere obtener la misma convergencia que el algoritmo de descenso de gradiente.

Algoritmo de descenso de gradiente *mini batch*

Este algoritmo es una mejora respecto a los dos algoritmos mencionados previamente. El conjunto de datos se divide en varias épocas y tras cada época se actualizan los pesos. Podemos definir este algoritmo de la siguiente forma:

$$\Theta := \Theta - a \cdot \nabla J(\Theta; B^{(i)}) = \Theta - a \cdot \frac{\partial J(\Theta, B^{(i)})}{\partial \Theta} \tag{3.21}$$

Siendo $B^{(i)}$ un *batch* introducido en las muestras de entrenamiento.

Este algoritmo de descenso de gradiente *mini batch* requiere menos cantidad de memoria que los algoritmos anteriores. Además, tiene menos varianza, gracias a las frecuentes actualizaciones de los pesos del modelo.

Momentum

Este algoritmo fue desarrollado para reducir la alta varianza de los algoritmos de descenso de gradiente y suavizar la convergencia. Acelera la convergencia hacia la dirección correcta y reduce la fluctuación hacia direcciones irrelevantes. Se añade un nuevo hiperparámetro: γ, denominado *momentum*, que habitualmente se establece con un valor aproximado de 0.9.

$$V(t) := \gamma V(t-1) + a \cdot \nabla J(\Theta) = \gamma V(t-1) + a \cdot \frac{\partial J(\Theta)}{\partial \Theta} \tag{3.22}$$

Este modelo reduce las oscilaciones y la alta varianza de los pesos, además, converge más rápido que el descenso de gradiente. Su mayor desventaja es el nuevo hiperparámetro que se introduce en él, incrementando la complejidad y siendo necesario seleccionarlo manualmente.

ADAM (Adaptative Moment Estimation)

Este algoritmo trabaja con momentos de primer y segundo orden, siendo estos su media ($M(t)$) y su varianza ($V(t)$). Disminuye la velocidad de búsqueda del mínimo, para evitar pasar por él sin detectarlo.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{3.23}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{3.24}$$

Siendo β_1^t y β_2^t los parámetros de deterioro exponencial para las estimaciones de los momentos de primer y segundo orden respectivamente.

Este modelo destaca por su rápida velocidad de convergencia y corrige el desvanecimiento de la tasa de aprendizaje. Por contra, tiene un elevado coste computacional.

3.3.4. Redes Neuronales Convolucionales (CNN)

Las redes neuronales convolucionales son un tipo especializado de red neuronal dedicada al procesamiento de datos con topología de rejilla, por ejemplo series temporales (1D) o imágenes (2D). Son una variación del perceptrón multicapa, que reciben entradas de matrices multidimensionales en vez de vectores de una dimensión. Este tipo de red tiene la capacidad de aprender características, por lo que los datos que se utilicen para su entrenamiento requieren de un preprocesado mucho menor.

Como su propio nombre indica, esta red utiliza una operación matemática llamada "convolución". Una convolución es un tipo de operación lineal.

La arquitectura de este tipo de redes es similar a las conexiones entre neuronas en el cerebro humano y se inspira en la organización de la corteza visual. Esto permite distinguir características significativas en los datos, permitiendo hacer una clasificación correcta de estos. Una red convolucional está formada por un conjunto de capas convolucionales, además de otros elementos. Una capa convolucional completa está formada por una operación de convolución y otra de *pooling*. A continuación se explican estas dos operaciones.

El material mostrado en esta sección ha sido adaptado de [35] y [36].

3.3.4.1. Operación de convolución

Una convolución es una operación en la que se aplica un filtro o *kernel* a unos datos de entrada, obteniendo como resultado una característica. Es una operación matemática que transforma dos funciones (f y g) en una tercera, que representa una superposición de f con una versión trasladada e invertida de g.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a) \cdot w(t-a) \tag{3.25}$$

Siendo:

- x: la matriz de entrada (suele ser un array multidimensional).
- w: la función *kernel* o núcleo (también suele ser un array multidimensional).
- $s(t)$: salida obtenida, también llamada *feature map*.

Esta operación convolución podemos verla perfectamente representada en la imagen 3.14:

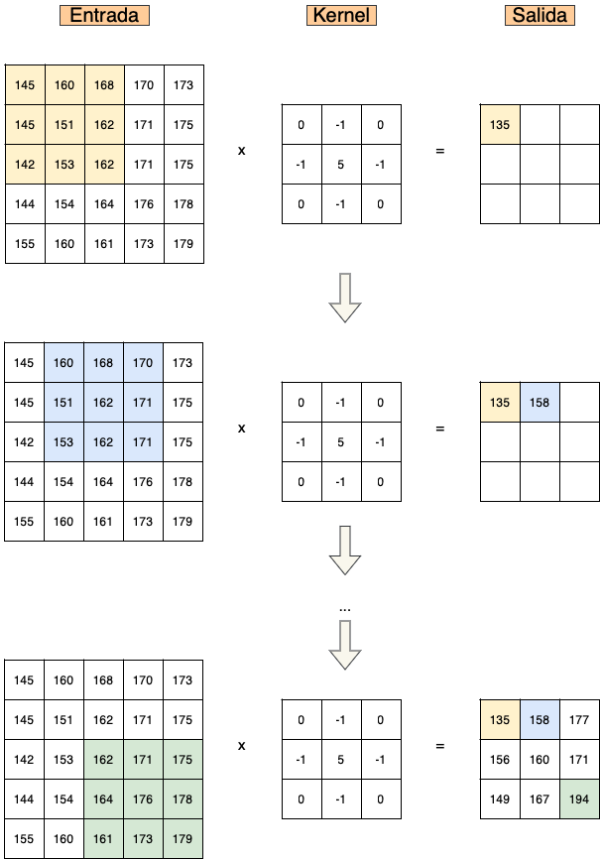

Figura 3.14: Operación de convolución

La operación de convolución está definida por tres parámetros: *kernel*, *padding* y *stride*.

El parámetro ***kernel*** es una matriz con un tamaño que nosotros definimos, que permite aplicar filtros en los datos para la extracción de características. En el ejemplo de la figura 3.14 podemos ver que se aplica un *kernel* de dimensión 3x3 a una matriz de entrada de dimensión 5x5, y dando como resultado una matriz de dimensión 3x3.

El orden de aplicación del *kernel* sobre la matriz de entrada es de izquierda a derecha y de arriba a abajo, podemos verlo en la figura 3.15:

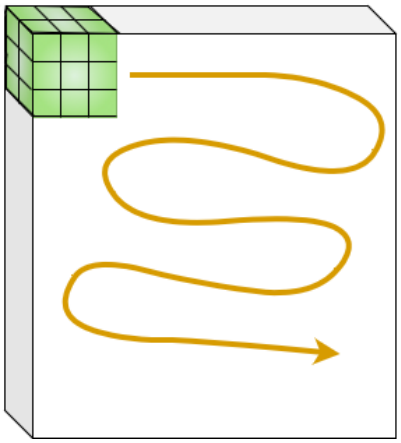

Figura 3.15: Movimiento del *kernel*

Podemos diseñar nosotros mismos el *kernel*, aunque existen ciertos tipos de kernel predefinidos [37]:

- ***Blur***: quita el énfasis de las diferencias en los valores de elementos adyacentes.
- ***Bottom sobel***, ***left sobel***, ***right sobel***, ***top sobel***: muestra únicamente las diferencias entre valores adyacentes en una dirección determinada.
- ***Emboss***: en imágenes, da la sensación de profundidad, enfatiza las diferencias de valores en una dirección dada, de arriba a la izquierda hasta abajo a la derecha de la matriz de datos.
- ***Identity***: no modifica la matriz de entrada.
- ***Outline***: se utiliza para resaltar diferencias grandes. Si dos valores tienen valores cercanos, tendrán el mismo valor, y los valores más cercanos a éstos que difieran bastante, aparecerán con un valor opuesto.
- ***Sharpen***: enfatiza diferencias entre píxeles adyacentes.

Podemos escribir estos *kernels* en forma de matriz:

$$\text{blur} = \begin{bmatrix} 0{,}0625 & 0{,}125 & 0{,}0625 \\ 0{,}125 & 0{,}25 & 0{,}125 \\ 0{,}0625 & 0{,}125 & 0{,}0625 \end{bmatrix} \quad \text{emboss} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix} \quad \text{identity} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\text{outline} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{sharpen} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{bottom sobel} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\text{left sobel} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{right sobel} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{top sobel} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

El parámetro ***padding*** permite añadir unos bordes artificiales a la matriz de datos. Habitualmente se aplica a imágenes, de forma que el *kernel* cubra una mayor parte de la imagen, haciendo que se opere más veces sobre los píxeles de los bordes de la imagen. En la figura 3.16 podemos ver cómo se ha aplicado un *padding* de ceros a una matriz de dimensión 3x3, obteniendo una matriz de tamaño 5x5.

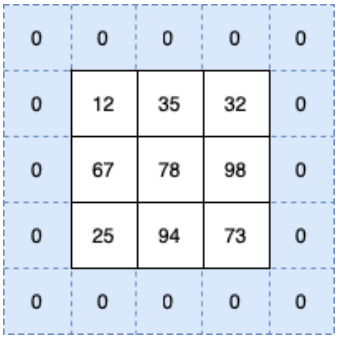

Figura 3.16: Aplicación del parámetro padding

El parámetro ***stride*** es el que controla el movimiento del *kernel* sobre la matriz de datos de entrada. Podemos fijarnos en el ejemplo de la figura 3.14, en la que el parámetro vale (1,1), esto significa que avanza un dato a la derecha y otro hacia abajo, siguiendo el movimiento que se indica en la figura 3.15. Si por ejemplo el *stride* fuera de (3,1), entonces el *kernel* se desplazaría de tres en tres hacia hacia la derecha y de uno en uno hacia abajo.

Podemos calcular la dimensión de la salida tras la operación de convolución de la siguiente forma [38]:

$$O = \frac{I - K + 2P}{S} + 1 \tag{3.26}$$

Siendo:

- **O**:Dimensión de la matriz de salida de la operación de convolución.
- **I**: Dimensión de la matriz de entrada de la operación de convolución.
- **K**: Tamaño del *kernel*.
- **S**: *Stride* de la operación de convolución.
- **P**: *Padding*.

3.3.4.2. Operación de *pooling*

La operación de ***pooling*** sigue habitualmente a una convolución en una red convolucional. Esta operación se encarga de reducir el tamaño del mapa de características, resumiendo las propiedades más importantes de éste.

Permite reducir los requisitos computacionales necesarios para entrenar la red. Además, sirve para extraer características dominantes invariantes frente a pequeñas rotaciones o traslaciones.

Ayuda a reducir el sobreajuste que cometen todas las redes convolucionales debido a su construcción, ya que las convoluciones recuerdan la posición exacta en las que se encuentran las características más importantes que ha seleccionado, pero pequeños cambios en la posición de estas características hace que la red haga de nuevo los cálculos.

Podemos distinguir diferentes operaciones de pooling, entre las que se encuentran:

- ***MaxPooling***: realiza un submuestreo de la entrada, tomando como salida el valor máximo de los datos de entrada.
- ***AveragePooling***: realiza un submuestreo de la entrada, tomando como salida el valor medio de los datos de entrada.
- ***MinPooling***: realiza un submuestreo de la entrada, tomando como salida el valor mínimo de los datos de entrada.

En la figura 3.17 podemos ver un ejemplo de aplicación de cada una de las tres operaciones de *pooling* mencionadas previamente, de tamaño 2x2, sobre unos datos de entrada de tamaño 4x4, reduciendo su dimensión a 2x2.

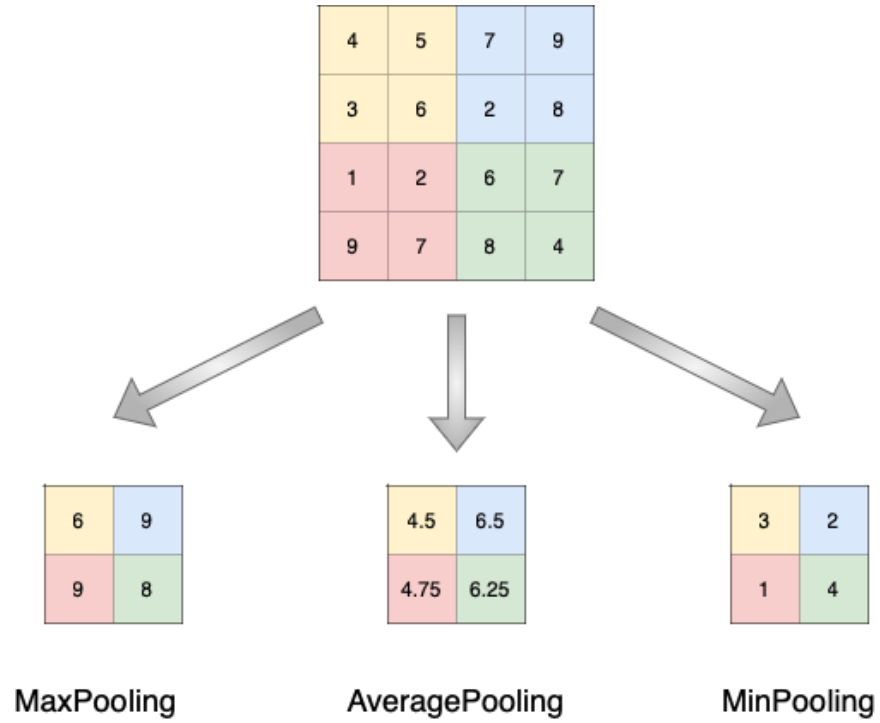

Figura 3.17: *Pooling* 2x2

Para calcular la dimensión de salida de la operación de *pooling*, debemos realizar la siguiente operación:

$$O = \frac{I - P_s}{S} + 1 \tag{3.27}$$

Siendo:

- **O**:Dimensión del volumen de salida de la operación de convolución.
- **I**: Dimensión del volumen de entrada de la operación de convolución.
- **S**: *Stride* de la operación de convolución.
- P_s: *Padding*.

3.3.4.3. *Flattening* o aplanado

Este es un sencillo paso que se realiza tras las capas convolucionales. Consiste en transformar la matriz de datos en un vector columna. El motivo de realizar este aplanado es ser introducido posteriormente en los datos en unas capas *fully-connected*.

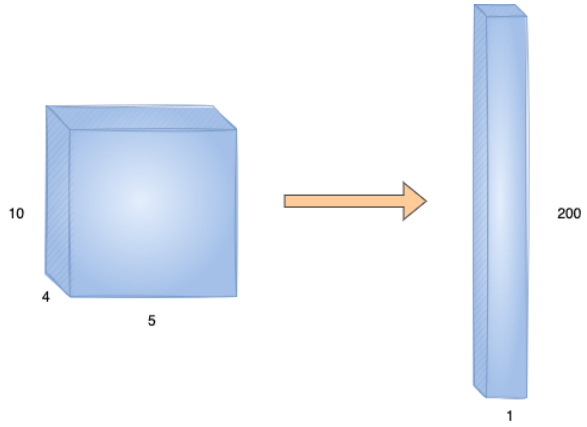

Figura 3.18: Aplicación del aplanado

En la figura 3.18 podemos ver una operación de aplanado. En este ejemplo vemos que los datos en tres dimensiones ($10 \times 5 \times 4$) se transforman en un vector de una dimensión cuyo valor es el producto de las tres dimensiones anteriores (200).

3.3.4.4. Capas *fully-connected*

Tras las capas convolucionales de la red y el aplanado de los datos, pasamos por unas capas *fully-connected* (FC), que se corresponden con un perceptrón multicapa clásico. Todas las neuronas de una capa están conectadas con todas las neuronas de la siguiente capa, pero no se conecta con ninguna de su misma capa ni se devuelven datos a neuronas de capas anteriores. La última capa *fully-connected* es la capa de salida de la red convolucional, y debe tener tantas neuronas como categorías tenga el problema que se desea resolver. En el caso de clasificación binaria, con una única neurona es suficiente, esta neurona nos indica la probabilidad de pertenencia a una clase o a otra.

En la figura 3.19 podemos ver un ejemplo de este tipo de capas *fully-connected* de un problema de clasificación binario.

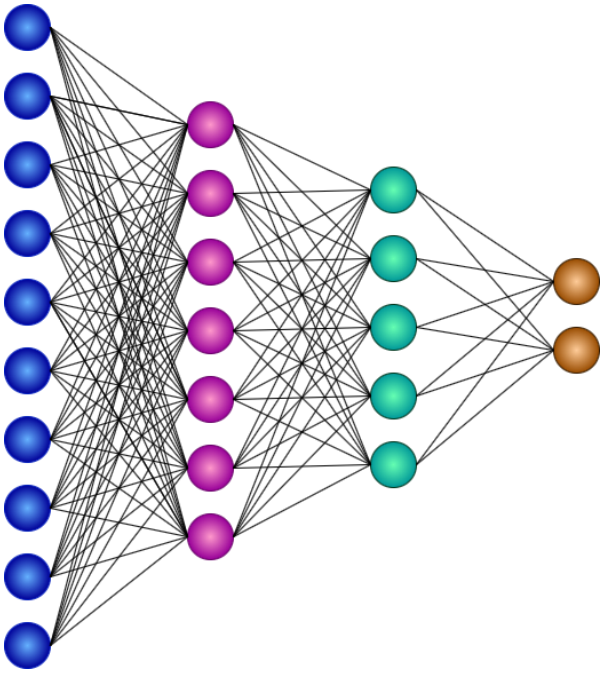

Figura 3.19: Capas *fully-connected* de un problema de clasificación binario

3.3.5. Problemas de las redes neuronales

Las redes neuronales son estructuras muy complejas que, a pesar de sus múltiples aplicaciones y de su gran utilidad, también tienen problemas, habitualmente relacionados con el desvanecimiento o explosión del gradiente y con el sobreajuste o infraajuste. En esta sección hablaremos de estos problemas y de cómo podemos eliminarlos de nuestras redes neuronales [39] [40].

3.3.5.1. Evanescencia del gradiente

Este problema se da cuando tenemos una red neuronal muy profunda con un aprendizaje basado en *backpropagation*. Cuando se realiza el producto de derivadas, si éstas tienen un valor pequeño, entonces el gradiente disminuirá de forma exponencial, haciendo que el valor de los pesos de la red en las primeras capas sea muy pequeño, cercano a 0. En el caso extremo de que el gradiente llegue a 0, la red dejará de aprender.

El principal síntoma de que nos enfrentamos a un problema de evanescencia del gradiente es que el modelo mejorará muy lentamente durante el entrenamiento, incluso llegando a parar, lo que significa que más entrenamientos no mejorarán más el modelo.

Las posibles soluciones a este problema son las siguientes:

- Reducción del número de capas.
- Una mejor elección de la inicialización de los pesos.
- Usar funciones de activación como ReLU, que hace que las derivadas no sean tan pequeñas.

3.3.5.2. Explosión del gradiente

Este problema, al igual que el de evanescencia del gradiente, se produce en redes neuronales muy profundas, con muchas capas. El motivo por el que se produce la explosión del gradiente es justo el contrario al de la evanescencia del gradiente: en este caso, cuando el producto de las derivadas tiene un valor grande, el gradiente crecerá de forma exponencial hasta llegar a "explotar". Esto hace que el modelo sea inestable e incapaz de aprender.

El principal síntoma de que nos enfrentamos a un problema de explosión del gradiente es que el modelo no aprende con los datos de entrenamiento, obteniendo un *loss* muy pobre y con valores muy variables en las sucesivas épocas de entrenamiento. El valor del *loss* puede llegar a ser `NaN` en algunas etapas del entrenamiento.

Las posibles soluciones a este problema son las siguientes.

- Reducción del número de capas.
- Una mejor elección de la inicialización de los pesos.
- Comprobar y limitar el tamaño de los gradientes del modelo.

3.3.5.3. Sobreajuste u *Overfitting*

Este problema se produce porque, si bien el modelo aprende muy bien con los datos de entrenamiento, tiene un desempeño muy pobre con los datos de test y de validación. Esto significa que la red puede llegar a memorizar los datos de entrenamiento pero tiene una capacidad muy baja de generalización. Es decir, el error durante el entrenamiento es pequeño, pero en validación y test es grande.

El principal motivo por el que se produce este problema es que los modelos son demasiado complicados para los datos.

Para reducir el sobreajuste, existen diferentes técnicas a utilizar:

- **Modelos más simples**: modelos menos complejos, con un menor número de parámetros, menor número de capas y de neuronas por capa.
- **Regularización**: es una forma indirecta de simplificar el modelo. Existen diferentes métodos de regularización que pueden utilizarse:
  - ***Dropout***: consiste en eliminar de forma aleatoria durante el entrenamiento de la red una cierta cantidad de neuronas. Esto hace que el entrenamiento tenga más ruido, rompiendo con situaciones en las que se de demasiado peso a ciertas entradas de capas anteriores. Este método de regularización también permite construir modelos más robustos. En la figura 3.20 podemos ver un ejemplo de cómo funciona en unas capas fully-connected.

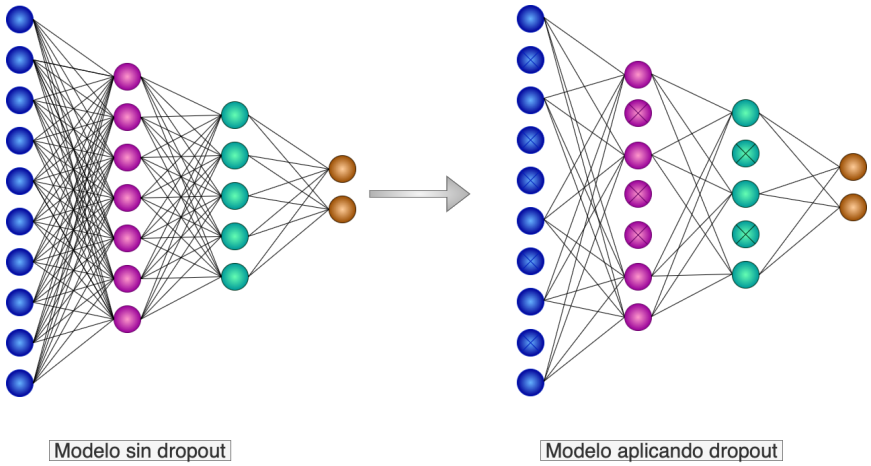

Figura 3.20: Aplicación de dropout sobre el modelo de la figura 3.19

  - **Regularización L1/L2**: consiste en añadir una cierta penalización a la función de coste, con el fin de obtener modelos menos complejos y que generalicen mejor. Los tres tipos de regularización son [41]:
    - **Lasso (L1)**: La penalización para este tipo de regularización es: $\lambda \cdot \sum_{i=1}^{n} |\Theta_i|$. Lasso sirve para casos en los que se sospecha que hay ciertos atributos de entrada irrelevantes y en los que los atributos no están muy correlacionados entre ellos, ya que nos permite conseguir que algunos coeficientes tengan un valor 0.
    - **Ridge (L2)** : La penalización para este tipo de regularización es: $\lambda \cdot \sum_{i=1}^{n} |\Theta_i^2|$. Ridge sirve para casos en los que se sospecha que existen varios atributos correlacionados, minimizando el efecto de esta correlación y haciendo que el modelo generalice mejor.
    - **Redes elásticas (L1 y L2)**: Es una combinación de las regularizaciones Lasso y Ridge, de la siguiente forma: $p \cdot Lasso + (1 - p) \cdot Ridge$ (siendo p un parámetro

que indica la importancia relativa de Lasso y Ridge). Se utiliza en casos en los que tengamos un gran número de atributos, algunos muy correlacionados entre ellos y otros irrelevantes.

- ***Early stopping***: Consiste en parar el entrenamiento cuando el error de validación no mejore durante un número establecido de épocas. De esta forma se evita que se produzca un sobreentrenamiento.

- ***Data augmentation***: Consiste en aumentar el número de datos de entrenamiento de la red de forma artificial, haciendo pequeñas transformaciones en los datos. Por ejemplo: en imágenes, podemos difuminarlas o rotarlas. En el caso de series temporales, podemos añadir una pequeña cantidad de ruido.

- **Aprendizaje por *ensembles***: los *ensembles* son mecanismos que permiten combinar diferentes modelos. Existen principalmente tres tipos de métodos de *ensemble learning*: *bagging*, *boosting* y *stacking*. La combinación de modelos puede provocar que se consigan modelos con un mejor rendimiento y más generalizables que los modelos base con los que se construyen.

3.3.5.4. Infraajuste o *Underfitting*

Este problema es el contrario al sobreajuste, se produce cuando el modelo es demasiado sencillo para los datos. Hace predicciones precisas pero incorrectas inicialmente. El error es grande tanto en entrenamiento como en validación y en test.

Existen diferentes técnicas para eliminar el infraajuste:

- **Modelos más complejos**: modelos más eficaces, con mayor número de parámetros, capas y neuronas por capa.

- **Menos regularización**: ya se ha explicado previamente en qué consiste la regularización y las diferentes técnicas que pueden utilizarse. La regularización busca simplificar el modelo, pero cuando hay infraajuste se busca todo lo contrario, por lo que debemos evitar en estos casos el uso de regularización.

Si nos fijamos en la figura 3.21 podemos ver, como se acaba de comentar, la relación directa que existe entre la complejidad y el error sobre los modelos de redes convolucionales.

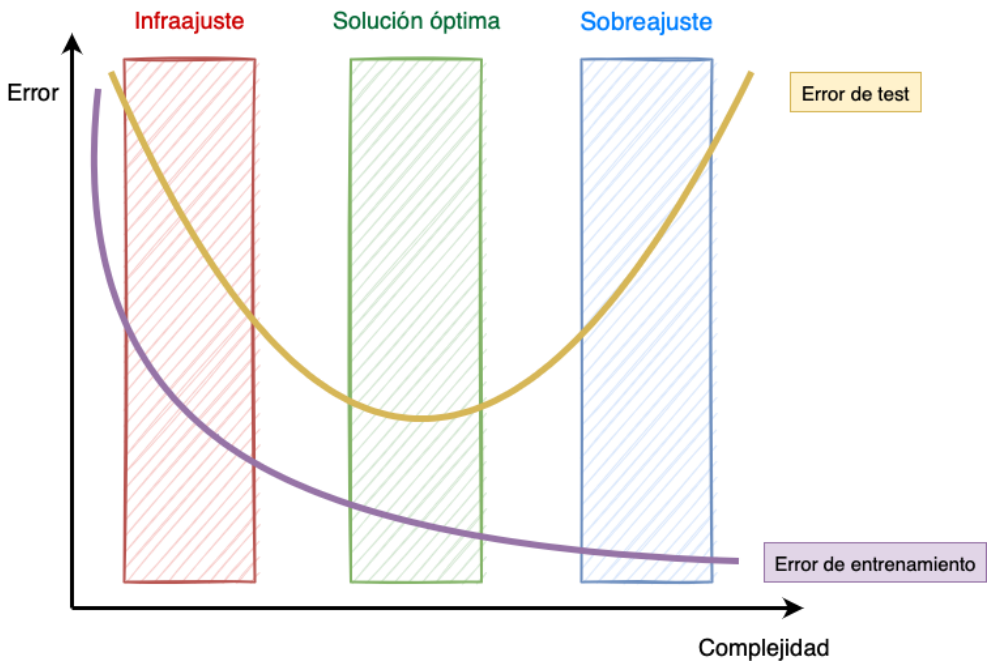

Figura 3.21: Relación entre error y complejidad en los modelos

Modelos con un error de entrenamiento y de test altos están cometiendo infraajuste, mientras que modelos con un bajo error de entrenamiento pero un alto error sobre test, están cometiendo sobreajuste. La situación ideal se produce cuando tanto el error de test como de entrenamiento son bajos. Este es el tipo de modelos que debemos buscar para obtener las mejores predicciones.

Capítulo 4

Planificación

4.1. Metodología de trabajo

La metodología que se ha seguido para el desarrollo de este proyecto es de tipo ágil. Esta metodología nos da un enfoque iterativo de la gestión de proyectos y del desarrollo de software. El proyecto se divide en pequeñas tareas y se evalúan constantemente los requisitos, la planificación y los resultados de forma que se pueda responder rápidamente a cambios. Existen múltiples metodologías ágiles, entre las que destacan Scrum, Kanban o *eXtreme Programming* (XP). Sin embargo, este proyecto es un trabajo de investigación, por lo que se trabajará con otra metodología, también ágil, llamada SCORE (*SCrum fOr REsearch*).

4.1.1. Metodología SCORE

La metodología SCORE (*SCrum fOr REsearch*) [43] es un tipo de metodología Scrum diseñado para actividades de investigación. En esta última, las tareas se dividen por equipos. Cada uno se dedica al desarrollo de estas tareas en *sprints* de cuatro a seis semanas. Al final de cada *sprint* se produce una reunión llamada *sprint backlog*. Durante cada *sprint* se realiza una reunión rápida diaria del equipo de trabajo en la que se habla sobre las tareas realizadas desde la anterior reunión, problemas que se han encontrado y qué es lo próximo que se va a hacer.

La metodología SCORE tiene dos elementos fundamentales en los que se diferencia con la metodología SCRUM:

- ***Status meeting***: En Scrum se mantenían breves reuniones diarias, sin embargo, para proyectos de investigación, esta frecuencia es demasiado elevada. En SCORE se mantienen tan

solo dos o tres reuniones por semana. En este tipo de reuniones se tratan los mismos temas que se han mencionado en las reuniones de Scrum, pero siempre teniendo presente el objetivo a largo plazo, habitualmente, la obtención de un resultado de investigación y su publicación.

- ***On-demand Technical Meetings***: Son un tipo de reuniones que se planean bajo demanda debido a ciertos temas que surgen en las *status meetings* y que deben tratarse más en profundidad. En estas reuniones también se pueden planificar futuras tareas. La frecuencia de estas reuniones se adapta a las necesidades del momento, pudiendo ser cada dos semanas o casi a diario, en caso de *deadlines* próximos.

4.1.2. Adaptación de SCORE al proyecto

Para la planificación de este proyecto se ha adaptado la metodología SCORE a las características propias de un Trabajo de Fin de Grado y a las únicas tres personas participantes: los dos tutores y la alumna.

Al tener que compatibilizar este proyecto con una asignatura y el trabajo de fin de grado de Estadística, las *status meetings* se han desarrollado vía *email* y con una menor frecuencia, aproximadamente una o dos por semana, en la que se ha informado a los tutores del desarrollo del proyecto. En caso de necesidad, se han solicitado reuniones presenciales o virtuales en las que se han resuelto las dudas relativas al proyecto, se han debatido los resultados obtenidos y se han determinado los pasos a seguir.

4.2. Planificación

Este trabajo de fin de grado está reconocido en el Grado en Ingeniería Informática de la Universidad de Valladolid con 12 créditos ECTS. Cada crédito equivale a 25 horas de trabajo, por lo tanto el proyecto supondrá 300 horas de trabajo, que se repartirán durante todo el segundo cuatrimestre del curso 2021-2022.

En la figura 4.1 podemos ver el diagrama de descomposición de trabajo, en el que se muestra un esquema de las diferentes actividades que se va a desarrollar a lo largo del proyecto: una estudio del contexto previo para el correcto entendimiento del problema a resolver y de las herramientas que se van a utilizar, el procesamiento de los datos que se van a analizar y la experimentación de diferentes modelos para conseguir las mejores predicciones posibles. Este diagrama nos permite ver la estructura del proyecto, dividiendo el trabajo en tareas más manejables, lo que nos permitirá elaborar una estimación de tiempos adecuada.

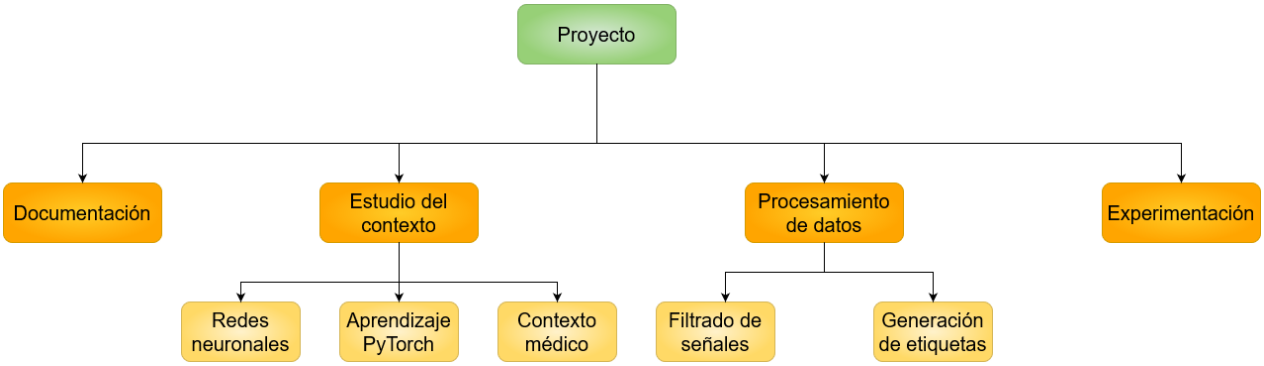

Figura 4.1: Diagrama de *Work Breakdown Structure*(WBS)

4.2.1. Plan de actividades calendarizado

La planificación de las tareas que van a desarrollarse, así como los hitos que deben cumplirse para la finalización a tiempo del proyecto, se muestran a partir del diagrama de Gantt de la figura 4.2

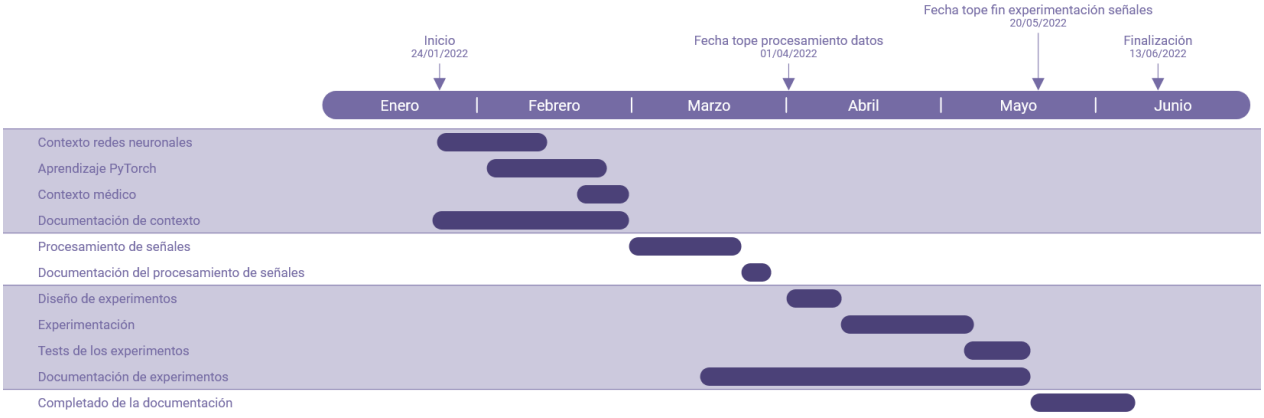

Figura 4.2: Diagrama de Gantt

El proyecto comenzará el día 24 de enero de 2022, tras el fin de los exámenes ordinarios del primer cuatrimestre. A lo largo del desarrollo del proyecto se han fijado tres hitos en los que se deben haber terminado tres tareas fundamentales para lograr terminar a tiempo el proyecto:

- **1 de abril de 2022**: Fecha límite para haber terminado el filtrado de las señales y la generación de las etiquetas del conjunto de datos.

- **20 de mayo de 2022**: Fecha límite para haber terminado la experimentación para la optimización de hiperparámetros de los modelos que se diseñen.
- **13 de junio de 2022**: Fecha límite para terminar la memoria. De esta forma se asegura tener el tiempo suficiente para repasarla antes de la fecha límite de entrega.

Durante los últimos días del mes de enero y prácticamente todo el mes de febrero, se va a estudiar el contexto de redes neuronales y médico, así como la herramienta PyTorch que se utilizará más adelante. Durante el mes de marzo se llevará a cabo el procesamiento de las señales: la aplicación de un filtro cada una las señales y la generación de sus etiquetas. El mes de abril y parte del mes de mayo se dedicará al diseño y ejecución de los experimentos, así como la elaboración de los tests de los modelos que nos ofrezcan un mejor resultado. Por último, el final del mes de mayo y el principio del mes de junio se dedicará a terminar de escribir la memoria. Hay que destacar que entre los hitos fijados y la calendarización de las tareas previas a la fecha de los hitos en el diagrama de Gantt, se han dejado unos días "flotador", es decir, unos días de margen, por si surgen imprevistos.

4.2.2. Identificación de riesgos

A continuación se plantean los riesgos que pueden materializarse a lo largo del desarrollo del proyecto, así como la probabilidad de ocurrencia que estos tengan, el impacto que pueda tener su materialización, la exposición a cada uno de ellos, su clasificación (amenaza u oportunidad) y las posibles estrategias a seguir para evitar que se materialicen.

A la probabilidad de ocurrencia y al impacto de la materialización del riesgo se le asigna un valor de 1 a 10, siendo 1 nada probable o con ningún impacto y 10, algo que ocurrirá muy probablemente o que muy probablemente tendrá graves consecuencias. La exposición al riesgo la podemos calcular como el producto de estas dos métricas.

Riesgo 1 (R01)	**Mala comprensión inicial del proyecto**
Descripción	Este trabajo de fin de grado es un proyecto complejo, que aplica conocimientos de redes neuronales a un campo completamente diferente a la informática. Puede que no se comprenda correctamente y que dificulte la culminación con éxito del proyecto.
Probabilidad	3
Impacto	9
Riesgo	27
Clasificación del riesgo	Amenaza

Estrategia	Estudio previo del estado del arte del contexto médico y físico.

Tabla 4.1: Riesgo 1

Riesgo 2 (R02)	**Mala estimación de tiempos y problemas de organización**
Descripción	Este proyecto es largo y complejo, en el que además, puede dedicarse mucho tiempo a la experimentación con muy diversos modelos, por lo que una mala planificación puede ocasionar retrasos en su entrega.
Probabilidad	4
Impacto	5
Riesgo	20
Clasificación del riesgo	Amenaza
Estrategia	Definición clara de todas las partes del proyecto, así como una estimación correcta del tiempo necesario para cada tarea; definiendo además una serie de hitos que cumplir de manera estricta.

Tabla 4.2: Riesgo 2

Riesgo 3 (R03)	**Desarrollo técnico demasiado complicado**
Descripción	Puede que el proyecto se sea demasiado ambicioso, haciendo que quede inconcluso o que se deban hacer cambios de última hora para hacer más sencilla la idea inicialmente planificada y diseñada.
Probabilidad	3
Impacto	8
Riesgo	24
Clasificación del riesgo	Amenaza
Estrategia	Estudio de viabilidad de las ideas para el proyecto, aplicadas a las tecnologías que se van a utilizar.

Tabla 4.3: Riesgo 3

Riesgo 4 (R04)	**Flexibilidad horaria para su dedicación al proyecto**

Descripción	Durante los meses de desarrollo del proyecto, la única carga lectiva son 12 ECTS: una asignatura y el TFG del grado en Estadística, por lo que se podrán dedicar las horas necesarias para su desarrollo con éxito.
Probabilidad	9
Impacto	7
Riesgo	63
Clasificación del riesgo	Oportunidad
Estrategia	Al no tener que cursar las usuales cinco o seis asignaturas de cada cuatrimestre, hay una mayor flexibilidad de horario para dedicar al proyecto, además de un mayor número de horas.

Tabla 4.4: Riesgo 4

Riesgo 5 (R05)	**Escasez de recursos bibliográficos**
Descripción	El desarrollo de este proyecto requiere un amplio conocimiento del estado del arte de la apnea del sueño, así como del tratamiento de señales y de *deep learning*.
Probabilidad	1
Impacto	9
Riesgo	9
Clasificación del riesgo	Amenaza
Estrategia	Búsqueda de referencias en libros especializados en las áreas a estudiar.

Tabla 4.5: Riesgo 5

Riesgo 6 (R06)	**Recursos hardware insuficientes**
Descripción	El entrenamiento de los modelos de Deep Learning, debido a la gran cantidad de datos existentes, requiere de una GPU potente para hacer los cálculos.
Probabilidad	9
Impacto	5
Riesgo	45
Clasificación del riesgo	Amenaza

Estrategia	Préstamo por parte de la UVa de una GPU potente para el entrenamiento de los modelos.

Tabla 4.6: Riesgo 6

Riesgo 7 (R07)	**Dificultad en el tratamiento de los datos**
Descripción	Las señales de apnea, debido a la tecnología que se emplea para la recogida de los datos, tienen mucho ruido que deberá ser eliminado ya que, sino, empeoraría mucho el funcionamiento de los modelos de *deep learning*.
Probabilidad	5
Impacto	8
Riesgo	40
Clasificación del riesgo	Amenaza
Estrategia	Aprendizaje de los métodos de filtrado de señal para poder seleccionar el óptimo, que nos de la señal con la menor cantidad de ruido posible.

Tabla 4.7: Riesgo 7

Riesgo 8 (R8)	**Finalización tardía del proyecto**
Descripción	Debido al retraso en las tareas programadas o la posibilidad de un menor tiempo de dedicación al proyecto ocasionado por el surgimiento de imprevistos, puede que no se llegue a entregar a tiempo.
Probabilidad	2
Impacto	9
Riesgo	18
Clasificación del riesgo	Amenaza
Estrategia	Se requiere una buena planificación, con hitos fijados y una predisposición al cumplimiento a tiempo de estos, además de unos días flotador con margen para imprevistos.

Tabla 4.8: Riesgo 8

4.3. Cumplimiento del plan

La planificación se cumplió con éxito, incluso permitiendo el desarrollo de nuevas tareas cuyas ideas surgieron y se propusieron a los tutores a lo largo del proyecto. En el diagrama de Gantt de la figura 4.4 podemos ver el resultado final de la calendarización de las tareas realizadas.

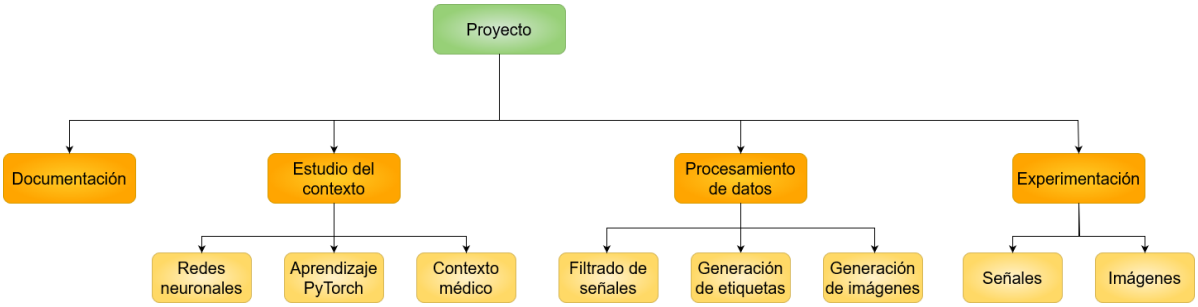

Figura 4.3: Diagrama de *Work Breakdown Structure* (WBS) tras el desarrollo del proyecto

Además de las tareas ya previstas, se ha iniciado otra vía de estudio de las señales de apnea del sueño: el procesado de imágenes, por lo que se han debido transformar todas las señales en imágenes y se ha realizado una pequeña experimentación mediante *transfer learning*. El motivo del número de experimentos tan reducidos de este tipo es el largo tiempo de entrenamiento que requiere. Estas nuevas tareas se han añadido al diagrama de descomposición de actividades (WBS), como podemos observar en la figura 4.4.

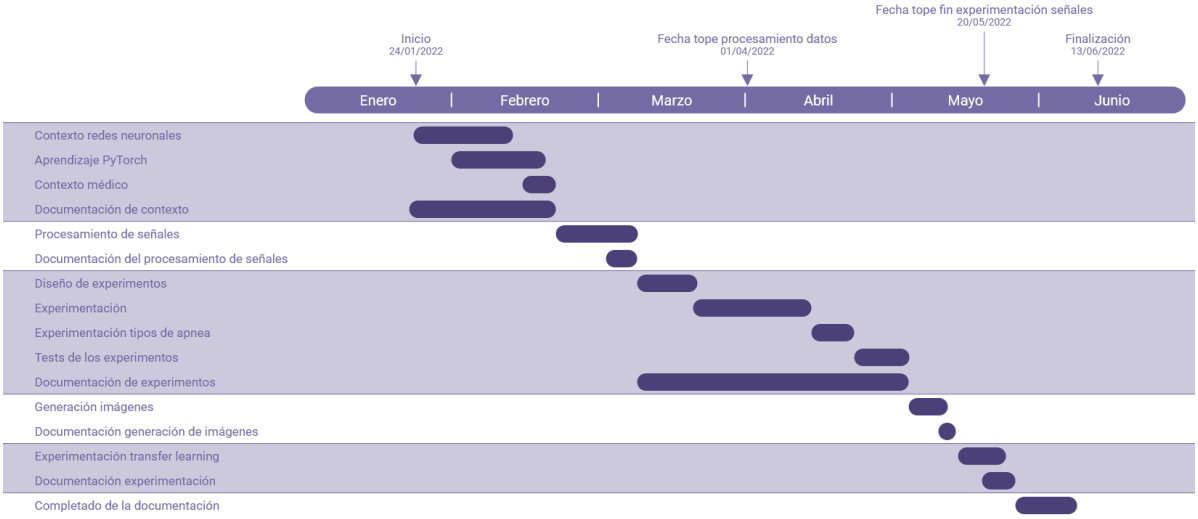

Figura 4.4: Diagrama de Gantt tras el desarrollo del proyecto

Capítulo 5

Métodos

5.1. Tratamiento de datos

En este apartado se va a hablar sobre todo el tratamiento de los datos que se ha realizado, desde su extracción y hasta la obtención del formato necesario para ser introducidos en una red neuronal. En la figura 5.1 podemos ver las distintas etapas de este proceso que se van a explicar a continuación.

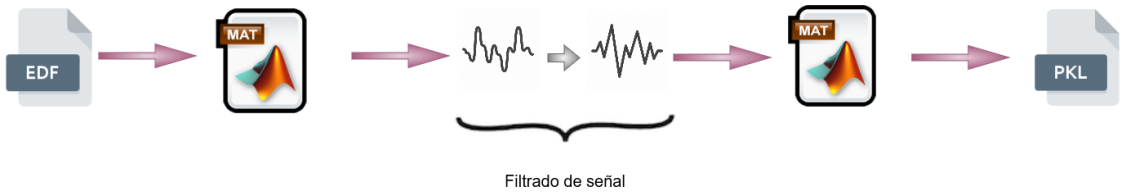

Figura 5.1: Esquema del procesamiento de los datos

5.1.1. Procesamiento inicial de las señales

Esta primera etapa ya ha sido realizada en el Grupo de Ingeniería Biomédica (GIB) de la Universidad de Valladolid. Los datos extraídos de los pacientes se encuentran en formato `.edf` (European Data Format), muy habitual para la recogida de datos de electroencefalogramas y polisomnografías. Es un formato sencillo y flexible para el almacenamiento de señales biológicas y físicas.

Las dimensiones iniciales de los datos es de alrededor de 2 Tb, consiguiendo reducir su dimen-

sión a alrededor de 165 GB, en formato `.mat`. Estos ficheros contienen toda la información que se recoge en una noche de sueño en una polisomnografía.

Las señales que van a emplearse en el desarrollo de este Trabajo de Fin de Grado son las tomadas gracias a unas bandas torácicas y abdominales que se colocan sobre el cuerpo del paciente. Estas señales se miden durante todas las horas de sueño del paciente, con una frecuencia de muestreo de 10 Hz, es decir, se toman 10 muestras por segundo. Además, se utilizarán unas variables indicador de apnea obstructiva, apnea central e hipopnea, también medidas con una frecuencia de muestreo de 10 Hz. Estas variables nos muestran si en ese preciso instante en el que se toma la medición hay un evento apneico (1) o no (0).

5.1.2. Filtrado de las señales

Una vez que tenemos todas las señales en formato `.mat`, debemos filtrarlas, debido a que pueden tener diferentes artefactos que empeoren la calidad de las predicciones en etapas posteriores del estudio. Estos artefactos pueden producirse por ruido generado por el instrumental utilizado para realizar las mediciones o por movimientos del paciente durante el sueño.

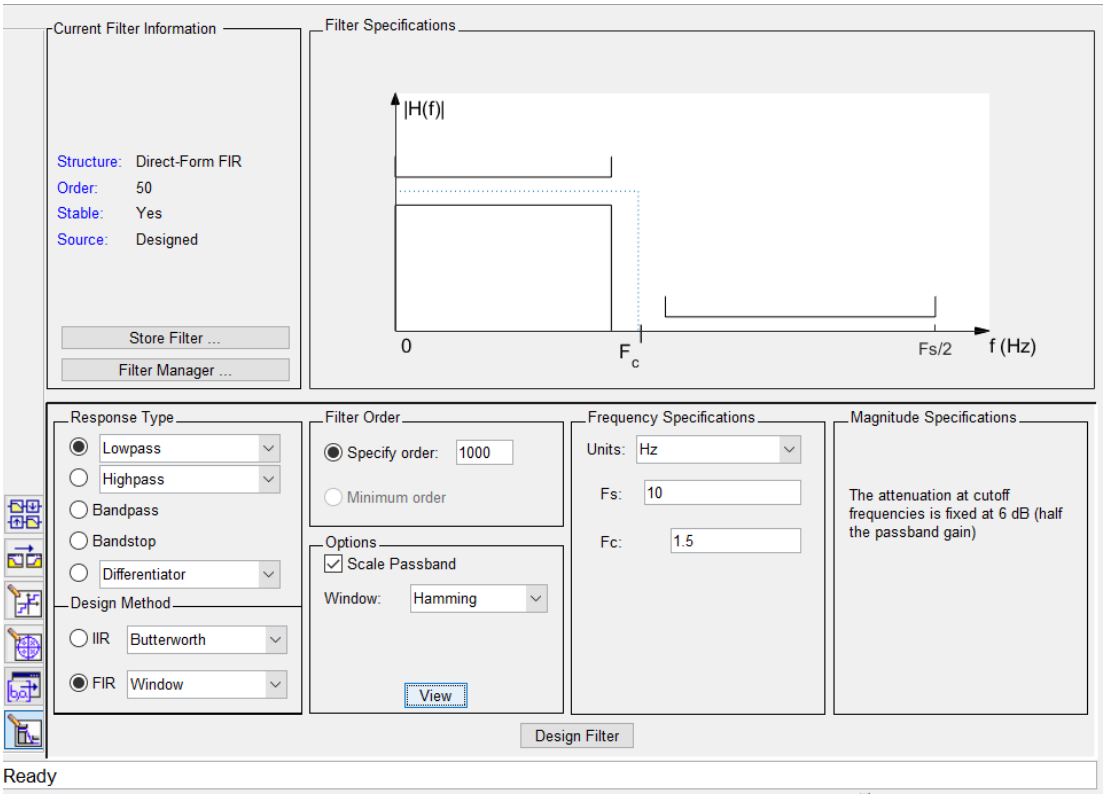

Figura 5.2: Esquema del procesamiento de los datos

Para el filtrado de las señales utilizaremos una herramienta de MATLAB llamada `filterDesigner`. En la figura 5.2 podemos ver el aspecto de su interfaz y la configuración del filtro que se va a utilizar.

El filtro que se va a utilizar es un **filtro de paso bajo** (*lowpass filter*). Es un tipo de filtro que permite el paso de frecuencias más bajas y atenúa las frecuencias más altas, es decir, atenúa el efecto del ruido. Como método de diseño utilizaremos **FIR** (*Finite Impulse Response*), para reducir el ruido en un análisis prospectivo lineal a lo largo del tiempo. Utilizamos FIR en vez de IIR (*Infinite Impulse Response*) porque afecta de forma lineal a la fase de las señales (en vez de logarítmica) y porque IIR en ocasiones puede provocar el desfase de la señal. El tipo de ventana que se utilizará será **Hamming**, porque no tiene rizado en la banda de paso, es decir, no añade pequeños valores en la banda de la señal que queremos analizar.

El orden distorsiona el número de muestras equivalente al orden que se utilice, en este caso, **1000**. Esto significa que vamos a eliminar las primeras y últimas 1000 observaciones, lo que equivale a eliminar 200 segundos en total, porque la frecuencia de muestreo de nuestros datos es de 10 Hz.

Respecto a las configuraciones de la frecuencia, utilizaremos los siguientes parámetros:

- **Unidades**: **Hz**, es la unidad de medida de la frecuencia en el Sistema Internacional de Unidades.
- **Fs**: indica la frecuencia de muestreo, es decir, la frecuencia con la que se han tomado los datos, en este caso, **10 Hz**. Por lo tanto, a la ahora de tomar los datos, se recogen 10 muestras por segundo.
- **Fc**: indica la frecuencia de corte: a partir de qué frecuencia queremos que se recoja la menor información posible, en este caso nos interesa que sea **1.5 Hz**.

Una vez hayamos completado todos los campos, generamos el filtro y obtendremos un vector de coeficientes. A continuación, aplicamos el filtro a la función:

```
filtered_signal = filtfilt(num, den, signal);
```

Siendo `num` el vector de coeficientes que acabamos de generar, `den` otro vector de coeficientes que, para esta caso concreto de ventana de Hamming, su valor es simplemente 1; y `signal` la señal que queremos filtrar, en nuestro caso, las señales torácicas y abdominales.

A continuación, en la figura 5.3 podemos ver un ejemplo de una representación de una señal y esta misma señal tras aplicarle un filtro.

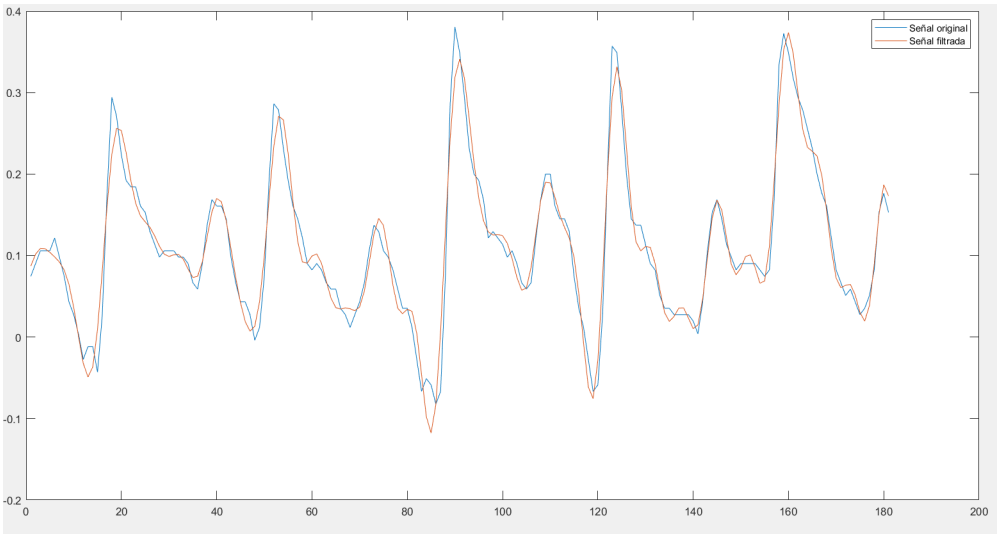

Figura 5.3: Comparación de una señal observada y esta misma tras aplicar un filtro

Como podemos ver en la figura 5.3, el filtro nos ha permitido eliminar cierto ruido que presenta la señal, haciendo que la nueva señal ya filtrada presente una mayor suavidad, pero que conserve a la perfección la forma de la original. La aplicación de este filtro permite que la red neuronal no se pierda con el ruido, influyendo de forma negativa en las posteriores predicciones.

Una vez que se han filtrado las señales, se divide cada una de ellas en fragmentos de 20 minutos. Cada señal se corresponde con una noche de sueño de un paciente, pero utilizar esta señal completa para el entrenamiento de modelos de *deep learning* provocaría dos limitaciones fundamentales: el número de observaciones sería muy inferior y se incrementaría significativamente la complejidad de la red, al tener observaciones con un tamaño de entrada mucho mayor, pudiendo resultar inabordable con los recursos disponibles. Por lo tanto, la señal de cada individuo va a dividirse en el máximo número posible de fragmentos de 20 minutos no solapados, para poder ser estudiado mediante una red neuronal. En [42] se estudian diferentes tamaños de los segmentos (5 minutos, 10 minutos, 20 minutos, 30 minutos y 60 minutos), siendo 20 minutos el tamaño de fragmento que proporciona mejores resultados.

Tras dividir cada señal, los fragmentos de cada una se almacenan en una matriz, lo que significa que cada matriz de datos representa a un individuo. Después, se estandariza cada uno de los fragmentos, de la siguiente forma:

```
stand_signal = zscore(signal);
```

A continuación, se deben etiquetar cada uno de los fragmentos. Como posteriormente se va a desarrollar un problema de regresión, siguiendo el enfoque de regresión de un modelo que resultó

exitoso para otro tipo de señal fisiológica [42], la etiqueta será el número de eventos apneicos, de cualquier tipo, que se produzcan en los 20 minutos de duración del fragmento. Las etiquetas, tal y como se extraen del fichero `.mat`, son un indicador binario de presencia o ausencia de apnea en cada instante de la medición. Como la frecuencia de muestreo en la recogida de los datos es de 10 Hz, para considerar que se produce un evento de apnea, deberíamos contar, al menos, 100 instantes seguidos de presencia de apnea. Al dividir la señal completa de un individuo en fragmentos de 20 minutos, un evento de apnea puede quedar dividido en dos fragmentos diferentes, por lo que a la hora de hacer el conteo de eventos se tiene especial cuidado en las primeras y últimas instancias de los indicadores de apnea. Para ello, en caso de detectarse apnea en un extremo, se cuenta tan solo la parte proporcional a lo que sería un evento completo. De esta forma, la etiqueta de los fragmentos puede incluir números decimales.

Una vez filtradas las señales, fragmentadas y etiquetadas, se almacenan los fragmentos y sus etiquetas en un fichero `.mat` que posteriormente serán utilizados. Cabe destacar que, por simplicidad, se ha descartado un pequeño grupo de individuos cuya frecuencia de muestreo era inferior a la de las demás.

5.1.3. Distribución de los datos en entrenamiento, validación y test

Como se comentó en el capítulo 3, el estudio de SHHS del que se disponen los datos, se dividió en dos etapas: SHHS1 y SHHS2. Los individuos de esta última etapa, participaron también en la primera. Por este motivo se deben separar con cautela los datos de entrenamiento y test, ya que los datos de un mismo individuo, de cualquiera de las dos etapas en las que participó, no pueden encontrarse en distintos grupos de entrenamiento, validación o test.

En el siguiente esquema, podemos ver cómo ha sido el reparto de individuos entre los distintos conjuntos de datos y el número de individuos en cada grupo.

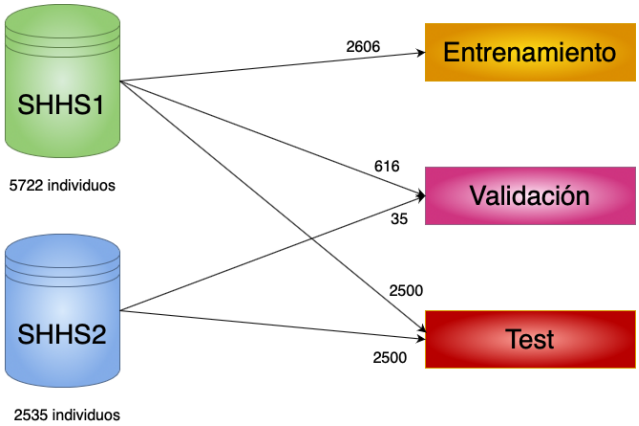

Figura 5.4: Distribución de los datos en entrenamiento, validación y test

En la figura 5.4 podemos observar que, para el conjunto de test se han empleado 5000 individuos. De estos, 2500 individuos han participado en SHHS2 y por lo tanto, como se ha comentado antes, sus datos en SHHS1 deben utilizarse también en test, nunca en entrenamiento o validación, para evitar posibles sesgos.

Los individuos pertenecientes a SHHS1 y que no están en SHHS2, y los individuos en SHHS2 que no están en SHHS1 (debido a los individuos que hemos descartado por tener una frecuencia de muestreo más baja) se utilizarán por tanto para entrenamiento y validación.

La división entre entrenamiento y validación ha sido un 80 % de los individuos para entrenamiento y un 20 % para validación, teniendo la precaución de nuevo de que todos los fragmentos de un mismo individuo se encuentren en un mismo conjunto de datos: o entrenamiento o validación, pero nunca en los dos a la vez.

5.1.4. *Data augmentation*

Como se ha podido comprobar, para evitar que se produzca sobreajuste al utilizar datos de un mismo individuo para entrenamiento y test, se han tenido que dejar muchas observaciones para test, quedando un porcentaje de datos más bajo de lo que nos gustaría para entrenamiento. Además, al estudiarse individuos con enfermedades cardiovasculares, no es difícil intuir que muchos padecerán apnea con un grado de severidad moderado o grave, y muy pocos no padecerán apnea o la padecerán de forma leve, por lo tanto, no contamos con unos datos balanceados. En la tabla 5.1 podemos ver el número de individuos tanto en entrenamiento como en validación, para cada grado de severidad de la apnea, siendo el caso de pacientes que no padecen apnea muy minoritario, como ya se intuía:

	No apnea	Apnea leve	Apnea moderada	Apnea severa	Total
Entrenamiento	77 (3 %)	529 (20 %)	958 (37 %)	1042 (40 %)	2606
Validación	11 (2 %)	143 (22 %)	214 (33 %)	283 (43 %)	651

Tabla 5.1: Número de individuos según tipo de apnea

Debido a los dos motivos expuestos previamente, se ha decidido hacer *data augmentation* sobre los individuos que no padecen apnea del sueño, es decir, aquellos que sufren menos de cinco eventos por hora. Para ello, se van a dividir los fragmentos de 20 minutos de estos individuos de diferente forma: en vez de ser fragmentos no solapados, vamos a hacer un solapamiento de 10 minutos entre individuos, de tal forma que un fragmento comenzará 10 minutos después de que comience el anterior fragmento. No estamos incrementando el número de individuos, pero sí el número de

fragmentos diferentes para cada individuo. Cabe destacar que esta técnica de *data augmentation* solo se utiliza para el conjunto de datos de entrenamiento, ya que es en el que necesitamos más datos para entrenar un modelo de más precisión.

5.1.5. *Data augmentation* sobre todas las clases

Como se verá en capítulos posteriores, el uso de *data augmentation* sobre la clase minoritaria no es suficiente para eliminar el sobreajuste, por lo tanto, se ha decidido aplicar *data augmentation* sobre todos los individuos, con el objetivo de obtener un conjunto de datos mucho más grande.

Los datos se obtienen de la misma forma que en el apartado anterior: se dividen las señales de una noche sueño en fragmentos de 20 minutos solapados en 10 minutos. Así, se consigue prácticamente el número de datos de entrenamiento, por lo que se espera que los resultados mejoren.

5.1.6. Transformación de señales en imágenes

Una forma completamente diferente de analizar las señales abdominal y torácica para el diagnóstico de apnea del sueño es el estudio de la representación tiempo-frecuencia de las señales, también llamado **espectrograma**.

Un espectrograma nos permite no solo representar el espectro de una señal, sino también cómo evoluciona a lo largo del tiempo. Para construir esta representación tiempo-frecuencia, utilizaremos una transformación corta de Fourier sobre cada fragmento de señal de 20 minutos, mediante la siguiente instrucción en MATLAB:

```
spectrogram(signal,300*fs,299*fs,1024,10, 'yaxis', 'MinThreshold',-50);
ax=gca;
ylim(ax, [0,1.6]);
```

En esta función introducimos, en primer lugar, los datos que queremos transformar (`signal`). A continuación, se introduce el tamaño de la ventana de muestras, es decir, la ventana sobre la que se hace la transformada corta de Fourier medido en número de muestras. En este caso se ha elegido una ventana de 5 minutos, es decir, 300 segundos multiplicado por la frecuencia de muestreo, `fs` (recordemos que en nuestro problema se toman 10 mediciones por segundo, por lo tanto, $fs = 10$). El motivo es que nos interesa la información presente en frecuencias especialmente bajas, entre 0.025 y 0.05 Hz. El siguiente parámetro nos dice cuántas muestras se solapan. En este caso se va a solapar todo menos un segundo, es decir, 290 segundos multiplicado por la frecuencia

de muestreo. Además, se representarán 1024 puntos para el cálculo de la transformada de Fourier. Para un correcto funcionamiento, se recomienda que esta cantidad sea potencia de 2. El parámetro 10 nos indica cuál es la frecuencia de muestreo con la que se han tomado los datos. Por último, "MinThreshold" y -50 se utiliza para que todo lo que esté atenuado 50 decibelios o más, sea represente de color azul. Por encima de 1.6 Hz no tenemos información, por lo tanto, en el código anterior también podemos ver cómo se omiten todas las frecuencias mayores a 1.6 Hz.

En la figura 5.5 podemos ver un ejemplo de un fragmento de 20 minutos de sueño de una señal torácica, en el que se producen tres eventos de apnea. En el espectrograma efectivamente, existen tres zonas bastante diferenciadas de color amarillo, en bajas frecuencias que corresponden con estos tres eventos. También se puede apreciar la respiración normal, cuya información espectral se encuentra alrededor de los 0.25 Hz (1 respiración cada 4 segundos). Cabe destacar que cuando se utilicen estas imágenes, se eliminarán tanto los ejes como la leyenda de colores en la derecha del espectrograma, para conseguir un correcto funcionamiento de la red neuronal.

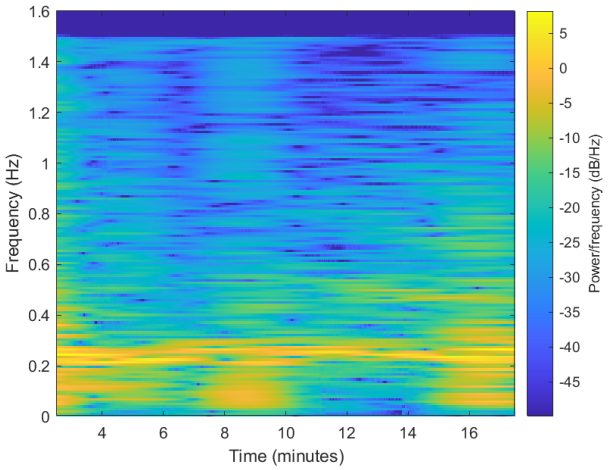

Figura 5.5: Espectrograma para fragmento de 20 minutos de una señal torácica

5.2. CNN 1 dimensión

Se ha decidido tratar un problema de regresión, de tal forma que podamos predecir el número de eventos de apnea que se producen en un intervalo de tiempo de 20 minutos para las dos señales: abdominal y torácica. Para ello, vamos a tratar las dos señales como dos canales. De esta forma tendríamos unas entradas a la red de una única dimensión, pero con dos canales, debido a que ambas señales pueden contener información complementaria que se vea recogida en los diferentes canales.

5.2.1. Arquitectura del modelo base

Se ha elegido una arquitectura de red basada en el diseño mostrado en [42] para la predicción de apnea obstructiva del sueño en niños mediante una señal de oximetría. Se ha elegido para el modelo base, una arquitectura de 6 capas convolucionales más una *fully-connected* en la que se aplica una regresión lineal, con un tamaño de *kernel* en las capas de convolución de 32. También 32 canales en la capa final y un *dropout* de 0.1 en todas las capas.

Podemos observar en la figura 5.6 el diseño de capas de la arquitectura, y en la figura 5.7, los componentes de cada capa: una convolución en 1 dimensión, una normalización del *batch*, una capa ReLu, una capa de *pooling* máximo y una capa de *dropout*.

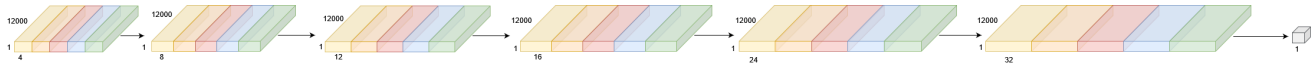

Figura 5.6: Arquitectura de la red neuronal

Figura 5.7: Estructura de cada capa de la arquitectura

El diseño de esta red convolucional en PyTorch se escribe de la siguiente forma:

```
class MyModel(nn.Module):
    def __init__(self):
        super().__init__()
        self.layer1 = nn.Sequential(
            nn.Conv1d(in_channels = 2, out_channels = 4, kernel_size = 32, ...
   stride= 2) ,
            nn.BatchNorm1d(num_features = 4),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1) )
        self.layer2 = nn.Sequential(
            nn.Conv1d(in_channels = 4, out_channels = 8, kernel_size = 32, ...
   stride= 2) ,
            nn.BatchNorm1d(num_features = 8),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1)  )
```

```
        self.layer3 = nn.Sequential(
            nn.Conv1d(in_channels = 8, out_channels = 12, kernel_size = 32, ...
    stride= 2) ,
            nn.BatchNorm1d(num_features = 12),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1) )
        self.layer4 = nn.Sequential(
            nn.Conv1d(in_channels = 12, out_channels = 16, kernel_size = 32,...
     stride= 2) ,
            nn.BatchNorm1d(num_features = 16),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1)  )
        self.layer5 = nn.Sequential(
            nn.Conv1d(in_channels = 16, out_channels = 24, kernel_size = 32,...
     stride= 2) ,
            nn.BatchNorm1d(num_features = 24),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1)  )
        self.layer6 = nn.Sequential(
            nn.Conv1d(in_channels = 24, out_channels = 32, kernel_size = 32,...
     stride= 2) ,
            nn.BatchNorm1d(num_features = 32),
            nn.ReLU(),
            nn.MaxPool1d(kernel_size = 2, stride = 1),
            nn.Dropout(p = 0.1) )
        self.lin = nn.Linear(in_features = 32*156, out_features = 1)

    def forward(self,x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = self.layer6(out)

        out = out.reshape(out.shape[0], out.shape[1]*out.shape[2]) # Flatten
        out = self.lin(out)

        return out
```

En la tabla 5.2 podemos ver los elementos que componen cada capa, el número de parámetros en cada una y sus dimensiones de salida, es decir, la dimensión que tienen al pasar a la siguiente capa.

Capa	Dimensión de salida	Nº parámetros
Convolución 1D	4, 5985	260
Norm. *Batch*	4, 5985	8
ReLu	4, 5985	0
Pooling	4, 5984	0
Dropout	4, 5984	0
Convolución 1D	8, 2977	1032
Norm. *Batch*	8, 2977	16
ReLu	8, 2977	0
Pooling	8, 2976	0
Dropout	8, 2976	0
Convolución 1D	12, 1473	3084
Norm. *Batch*	12, 1473	24
ReLu	12, 1473	0
Pooling	12, 1472	0
Dropout	12, 1472	0
Convolución 1D	16, 721	6160
Norm. *Batch*	16, 721	32
ReLu	16, 721	0
Pooling	16, 720	0
Dropout	16, 720	0
Convolución 1D	24, 345	12312
Norm. *Batch*	24, 345	48
ReLu	24, 345	0
Pooling	24, 344	0
Dropout	24, 344	0
Convolución 1D	32, 157	24608
Norm. *Batch*	32, 157	64
ReLu	32, 157	0
Pooling	32, 156	0
Dropout	32, 156	0
Capa lineal	1	4993

Tabla 5.2: Dimensiones y número de parámetros en cada capa

5.3. CNN 2 dimensiones

Al igual que en el apartado anterior, se va a tratar de resolver un problema de regresión para predecir el número de eventos que se producen en 20 minutos para las dos señales abdominal y torácica. A diferencia de la anterior, en este caso vamos a tratar estas dos señales como dos dimensiones de una misma matriz, es decir, vamos a apilar las dos señales. Es una alternativa a la arquitectura anterior, para estudiar si existen diferencias entre estas dos formas de plantear este problema.

5.3.1. Arquitectura

La arquitectura de red escogida es similar a la anterior, pero adaptada a dos dimensiones. Igualmente, para el modelo base se han escogido 6 capas convolucionales (pero de dos dimensiones), más una capa *fully-connected* en la que se aplica una regresión lineal, con un tamaño de kernel de (1, 32) en todas las capas, es decir, un kernel rectangular, de forma que mantenemos siempre la segunda dimensión añadida. La capa final tiene 32 canales y un *dropout* de 0.1 en todas las capas.

En la figura 5.7 ya vimos cómo era la estructura de cada una de las 6 capas. En el caso de dos dimensiones, la estructura es exactamente la misma pero transformando las capas a dos dimensiones. En la figura 5.8 se muestra la estructura de capas de la arquitectura.

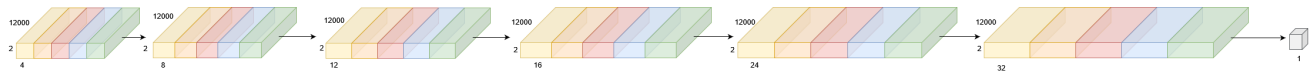

Figura 5.8: Estructura de cada capa de la arquitectura

El diseño de esta red convolucional en Pytorch se escribe de la siguiente forma:

```
class MyModel(nn.Module):
    def __init__(self):
        super().__init__()

        self.layer1 = nn.Sequential(
            nn.Conv2d(in_channels = 1, out_channels = 4, kernel_size = (1, ...
   32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 4),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )
        self.layer2 = nn.Sequential(
```

```
            nn.Conv2d(in_channels = 4, out_channels = 6, kernel_size = (1, ...
    32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 6),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )
        self.layer3 = nn.Sequential(
            nn.Conv2d(in_channels = 6, out_channels = 8, kernel_size = ...
    (1,32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 8),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )
        self.layer4 = nn.Sequential(
            nn.Conv2d(in_channels = 8, out_channels = 10, kernel_size = (1, ...
    32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 10),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )

        self.layer5 = nn.Sequential(
            nn.Conv2d(in_channels = 10, out_channels = 16, kernel_size = (1,...
     32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 16),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )

        self.layer6 = nn.Sequential(
            nn.Conv2d(in_channels = 16, out_channels = 32, kernel_size = (1,...
     32), stride= (1,2)) ,
            nn.BatchNorm2d(num_features = 32),
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = (1,2), stride = (1,1)),
            nn.Dropout(p = 0.1) )

        self.lin = nn.Linear(in_features = 32* 2* 156, out_features = 1)

    def forward(self,x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = self.layer3(out)
        out = self.layer4(out)
        out = self.layer5(out)
        out = self.layer6(out)

        out = out.reshape(out.shape[0], out.shape[1]*out.shape[2]*out.shape...
    [3]) # Flatten
```

```
        out = self.lin(out)

        return out
```

En la tabla 5.3 se muestran los elementos que componen cada capa, el número de parámetros de cada una de ellas y sus dimensiones de salida, esto es, la dimensión que tienen al pasar a la siguiente capa.

Capa	Dimensión de salida	Nº parámetros
Convolución 1D	4, 2, 5985	132
Norm. *Batch*	4, 2, 5985	8
ReLu	4, 2, 5985	0
Pooling	4, 2, 5984	0
Dropout	4, 2, 5984	0
Convolución 1D	6, 2, 2977	774
Norm. *Batch*	6, 2, 2977	12
ReLu	6, 2, 2977	0
Pooling	6, 2, 2976	0
Dropout	6, 2, 2976	0
Convolución 1D	8, 2, 1473	1544
Norm. *Batch*	8, 2, 1473	16
ReLu	8, 2, 1472	0
Pooling	8, 2, 1472	0
Dropout	8, 2, 1472	0
Convolución 1D	10, 2, 721	2570
Norm. *Batch*	10, 2, 721	20
ReLu	10, 2, 721	0
Pooling	10, 2, 720	0
Dropout	10, 2, 720	0
Convolución 1D	16, 2, 345	5136
Norm. *Batch*	16, 2, 345	32
ReLu	16, 2, 345	0
Pooling	16, 2, 344	0
Dropout	16, 2, 345	0
Convolución 1D	32, 157	16416
Norm. *Batch*	32, 2, 157	64
ReLu	32, 2, 157	0
Pooling	32, 2, 156	0

Dropout	32, 2, 156	0
Capa lineal	1	4993

Tabla 5.3: Dimensiones y número de parámetros en cada capa

5.4. *Transfer learning*

Con el objetivo de desarrollar otro enfoque completamente diferente a los anteriores, se ha decidido utilizar una red convolucional pre-entrenada, para la predicción de la apnea del sueño con las mismas señales abdominal y torácica, pero a partir de sus espectrogramas.

5.4.1. ResNet

Los modelos muy profundos tienen problemas con el desvanecimiento y la explosión del gradiente. Para arreglarlo, se introducen bloques de capas residuales. Se produce una conexión directa que se salta algunas capas del modelo, a esto se le llama *skip connection*. Este tipo de redes se denominan ResNet, y fueron propuestas por primera vez en *Deep Residual Learning for Image Recognition* [44].

Las redes ResNet han sido entrenadas previamente con una base de datos de acceso libre llamada ImageNet. Existen diferentes variantes de ResNet: 18, 34, 50, 101 y 152, siendo estos, el número de capas de cada variante. En este proyecto, debido a la gran cantidad de datos de entrenamiento, se ha elegido la red más pequeña: ResNet18, con un tamaño de *batch* de 16.

Este tipo de redes están preparadas para recibir como entrada imágenes en los tres canales RGB. Nuestras imágenes tienen estos tres canales, por lo que no debemos hacer ningún tipo de cambio en el modelo. En cambio, al seguir tratando con un problema de regresión para predecir el número de eventos de apnea que se producen en los fragmentos de 20 minutos de sueño, añadimos una capa con una regresión lineal con una única neurona de salida. El código necesario para implementar ResNet18 y adaptarlo al problema que se plantea, en Python, es el siguiente:

```
model = torch.hub.load('pytorch/vision:v0.10.0', 'resnet18',pretrained=True)
num_features = model.fc.in_features
model.fc = nn.Linear(num_features, 1)
model = model.to(device)
```

A continuación, en la figura 5.9 se muestra el diseño de la arquitectura de ResNet18, adaptada al problema de apnea:

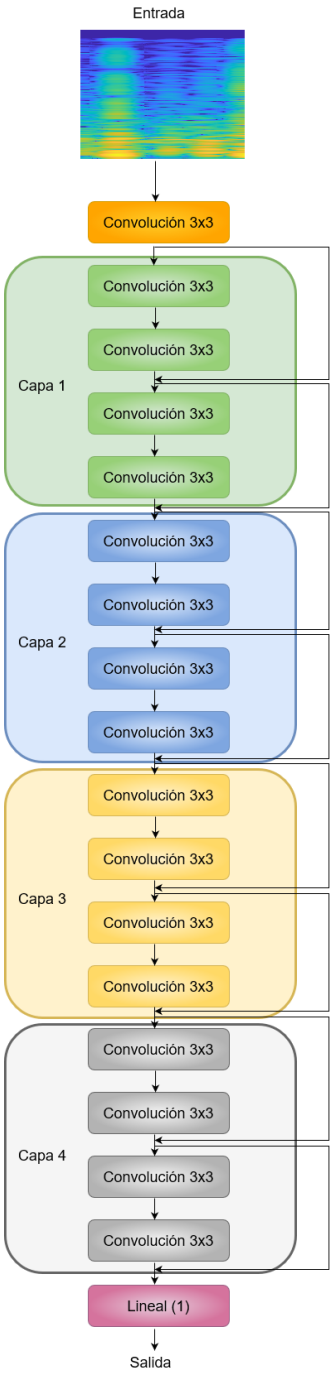

Figura 5.9: Arquitectura ResNet18 adaptada al problema de apnea

5.4.2. EfficientNet

Los modelos de redes convolucionales son muy grandes, muy profundos o con una resolución muy alta. Los modelos rápidamente se saturan y no mejoran, solo tienen más parámetros. Las redes EfficientNet escalan uniformemente todas las dimensiones de profundidad, anchura y resolución de forma eficaz. Con un menor número de parámetros, estos modelos son más eficientes y proporcionan mejores resultados. Este tipo de redes fue propuesto por Google AI en *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* [45].

Existen diferentes de modelos de *EfficientNet*: b0, b1, b2, b3, b4, b5, b6 y b7. En la figura 5.10 podemos ver cómo son cada uno de estos modelos en función del número de parámetros que tienen y el *accuracy* que alcanzan al ser entrenados con el conjunto de datos de ImageNet. Vemos que los primeros modelos, con unos pocos más parámetros se aumenta bastante la precisión del modelo, sin embargo, en los últimos modelos, con un mayor número parámetros, la precisión mejora bastante poco. El modelo elegido, por motivos de falta de memoria en la GPU debido a las grandes dimensiones de las imágenes, es el modelo b0, con un tamaño de *batch* de 4, por este mismo motivo.

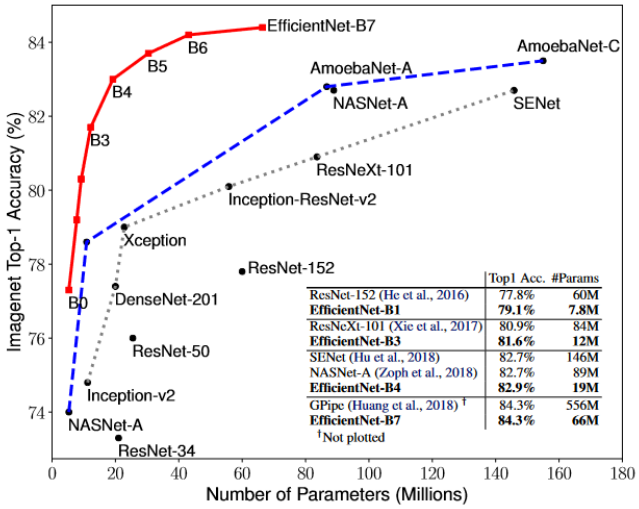

Figura 5.10: Comparación de los distintos modelos de EfficientNet (extraído de [45])

Cabe destacar que además, en la figura 5.10 también podemos ver la comparación con otras muchas redes ya entrenadas, entre las que se encuentran las explicadas previamente: ResNet. Apreciamos que el modelo de ResNet18 ni siquiera aparece debido a tener una menor precisión que los otros modelos. Por lo tanto, el uso de los modelos EfficientNet debería mejorar los resultados que obtengamos con ResNet18.

El código necesario para implementar el modelo de EfficientNet b0 es:

```
model = torchvision.models.efficientnet_b0(pretrained=True)
model.classifier[1] = nn.Linear(1280, 1, bias = True)
model = model.to(device)
```

En la figura 5.11 se muestra el diseño de la arquitectura EfficientNet b0, adaptada al problema de apnea: cambiando la última capa por una capa lineal con una salida de dimensión 1.

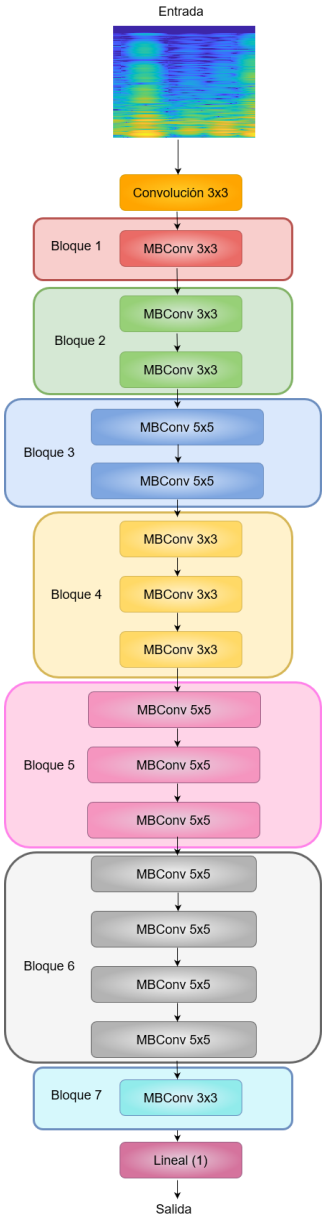

Figura 5.11: Arquitectura EfficientNet b0 adaptada al problema de apnea

Capítulo 6

Resultados

En este capítulo se presentan los resultados de la experimentación de las redes convolucionales de una y de dos dimensiones. Además, se muestran los resultados para test de los modelos que tengan un mejor funcionamiento.

6.1. CNN 1 dimensión

A continuación se muestra la experimentación para la optimización de hiperparámetros a partir del modelo base explicado en el capítulo anterior. A la vista de los resultados, se elegirán los modelos con mejores resultados y se evaluarán con un conjunto de datos de test.

6.1.1. Experimentación

En esta sección vamos a llevar a cabo una optimización de los hiperparámetros del modelo base. Para los entrenamientos, utilizaremos 200 épocas, una tasa de aprendizaje de 0.001 y un tamaño de *batch* de 32. Como criterio de optimización utilizaremos la función de pérdida de Hubber y Adam como optimizador. Podemos definir la función de pérdida de Hubber de la siguiente forma:

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{si } |a| \leq \delta \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right) & \text{resto} \end{cases} \tag{6.1}$$

El optimizador Adam ya fue definido en el capítulo 3.

A continuación, a partir de la arquitectura del modelo base ya definido, vamos a buscar la mejor configuración de hiperparámetros. Buscaremos el modelo que nos ofrezca una mejor concordancia en validación entre las predicciones y los valores reales. Utilizaremos para ello el índice kappa, que se calculará a partir del índice de apnea-hipopnea (IAH), medido en eventos por hora, para cada individuo. El IAH se calcula para cada paciente de forma individual, a partir de las predicciones de número de eventos apneicos en fragmentos de 20 minutos. El motivo de utilizar esta métrica es el interés por el rendimiento del diagnóstico final del paciente. Una vez que se calcule el IAH de cada paciente, se procederá a discretizar en cuatro clases clínicamente relevantes:

- **No apnea**: IAH < 5 e/h
- **Apnea leve**: 5 e/h $\leq$ IAH < 15 e/h
- **Apnea moderada**: 15 e/h $\leq$ IAH < 30 e/h
- **Apnea severa**: IAH ≥ 30 e/h

Tras discretizar las predicciones, calcularemos el índice kappa de la siguiente forma:

```
kappa = sklearn.metrics.cohen_kappa_score(AHI_prediction, AHI_real)
```

El índice kappa es el que nos permitirá medir la calidad de los modelos ajustados. Además, como técnica para evitar el sobreajuste, nos quedaremos con el modelo en validación que mejor resultados de, en este caso, que mejor índice kappa tenga.

En la tabla 6.1 podemos ver el conjunto de hiperparámetros con los que buscaremos los mejores resultados en validación. Cabe destacar que se han tenido que utilizar diferentes tamaños de *kernel* para 6 y 8 capas debido a que, en los modelos de 8 capas, al aplicar *kernels* tan grandes, estábamos reduciendo la dimensión de las entradas tanto que nos quedábamos sin parámetros.

Hiperparámetros	**Espacio de búsqueda**
Nº capas	6, 8
Tamaño *kernel*	32, 64, 128 (6 capas), 16, 32 (8 capas)
Nº canales	16, 32, 64, 128
Dropout	0.1, 0.2, 0.3, 0.4

Tabla 6.1: Hiperparámetros

En los siguientes apartados se mostrarán los resultados de los múltiples experimentos elaborados con las distintas combinaciones de hiperparámetros que se han mencionado en la tabla 6.1.

6.1.1.1. *Dropout* = 0.1, 6 capas

En primer lugar probaremos modelos con 6 capas y un *dropout* bajo, de 0.1.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.418	0.454	0.454
32	0.498	0.562	0.553
64	0.570	0.666	0.676
128	0.751	0.702	0.848

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.408	0.403	0.402
32	0.404	0.406	0.418
64	0.409	0.408	0.405
128	0.412	0.399	0.401

Tabla 6.2: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.1, 6 capas

En la tabla 6.2 podemos ver los resultados de los primeros experimentos, con los índices kappa calculados tanto en entrenamiento como en validación. Se aprecia que en el caso de validación, el índice kappa varía muy poco, siendo más alto en el caso de 32 canales y un tamaño de *kernel* de 128. En entrenamiento podemos ver que, con un mayor tamaño de *kernel* y un mayor número de canales, se obtienen mejores resultados, aunque, como hemos dicho antes, nos debemos fijar en los datos en validación, ya que son los que realmente nos dicen si el modelo está aprendiendo. Un índice kappa muy alto en entrenamiento, pero bastante más bajo en validación, nos indica que se está produciendo sobreajuste, por lo que debemos intentar reducirlo.

6.1.1.2. *Dropout* = 0.1, 8 capas

En la tabla 6.3 se observan los resultados de los experimentos con un *dropout* de 0.1 y 8 capas. Los resultados en validación son algo mejores que los que se han obtenido para los experimentos con el mismo *dropout* y con 6 capas. El mayor índice kappa lo encontramos para el experimento con 64 canales y un tamaño de *kernel* de 16, alcanzando un valor de 0.429.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.386	0.406
32	0.440	0.488
64	0.610	0.703
128	0.806	0.886

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.409	0.421
32	0.416	0.418
64	0.429	0.421
128	0.420	0.406

Tabla 6.3: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.1, 8 capas

6.1.1.3. *Dropout* = 0.2, 6 capas

En la tabla 6.4 vemos los resultados de los experimentos con 6 capas y un *dropout* de 0.1. Si comparamos los resultados con los de la tabla 6.2, podemos ver que por norma general, los resultados mejoran ligeramente, aunque no en todos los casos ni tampoco es una mejora importante. El mayor valor del índice kappa en validación lo alcanzamos tanto con 64 canales y un tamaño de *kernel* de 32, como con 16 canales y un tamaño de *kernel* de 128.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.390	0.406	0.413
32	0.424	0.453	0.48
64	0.481	0.518	0.563
128	0.618	0.558	0.726

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.406	0.402	0.428
32	0.405	0.418	0.411
64	0.428	0.407	0.416
128	0.399	0.404	0.420

Tabla 6.4: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.2, 6 capas

6.1.1.4. *Dropout* = 0.2, 8 capas

En la tabla 6.5 vemos los resultados de la experimentación para los modelos con 8 capas y un *dropout* de 0.2. Observamos que el índice kappa en los casos en los que tenemos 16 y 32 canales, por lo general, es menor que en el caso de un *dropout* de 0.1. Sin embargo, el caso de tener 128 canales, para los dos tamaños de *kernel* obtenemos un índice kappa mejor, siendo el de un *kernel* de tamaño 16, el mejor encontrado hasta el momento, con un valor de 0.435.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.374	0.385
32	0.413	0.427
64	0.497	0.553
128	0.657	0.765

Validación		
Kernel / Canales	**16**	**32**
16	0.380	0.402
32	0.398	0.420
64	0.417	0.412
128	0.435	0.426

Tabla 6.5: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.2, 8 capas

6.1.1.5. *Dropout* = 0.3, 6 capas

En la tabla 6.6 estudiamos los resultados de los experimentos para los modelos con 6 capas y un *dropout* de 0.3. Se aprecia que al ajustar un modelo con un *dropout* mayor que en los casos anteriores, no estamos encontrando una mejora, sino al contrario, el índice kappa es algo menor o se mantiene en valores parecidos.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.382	0.395	0.396
32	0.401	0.416	0.433
64	0.430	0.429	0.467
128	0.526	0.505	0.649

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.402	0.403	0.397
32	0.407	0.410	0.400
64	0.405	0.413	0.406
128	0.404	0.405	0.404

Tabla 6.6: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.3, 6 capas

6.1.1.6. *Dropout* = 0.3, 8 capas

En la tabla 6.7 podemos ver los valores del índice kappa en validación para los modelos con 8 capas y un *dropout* de 0.3. Se aprecia que para los casos en los que el modelo tiene pocos canales, el índice kappa es menor que en casos de un *dropout* menor. En el caso de los modelos con 128 canales, el valor del índice kappa se mantiene en valores similares a los que ya se obtuvieron con un *dropout* de 0.2.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.364	0.363
32	0.383	0.392
64	0.448	0.460
128	0.545	0.642

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.322	0.364
32	0.397	0.402
64	0.386	0.411
128	0.431	0.432

Tabla 6.7: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.3, 8 capas

6.1.1.7. *Dropout* = 0.4, 6 capas

En la tabla 6.8 podemos ver el índice kappa de los modelos con 6 capas y un *dropout* de 0.4. Cuando comentamos la tabla 6.6 ya intuíamos que un mayor *dropout* no mejoraría la calidad del

modelo. Al ver los resultados de estos nuevos experimentos, lo comprobamos. Podemos ver que para los modelos en validación, apenas unos pocos superan un índice kappa de 0.4, bastante por debajo de los valores más altos que se han obtenido con otros experimentos.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.365	0.387	0.375
32	0.376	0.398	0.402
64	0.391	0.405	0.435
128	0.415	0.429	0.494

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.403	0.390	0.374
32	0.374	0.387	0.375
64	0.401	0.418	0.362
128	0.401	0.400	0.399

Tabla 6.8: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.4, 6 capas

6.1.1.8. *Dropout* = 0.4, 8 capas

En la tabla 6.9 vemos los resultados de los experimentos con 8 capas y un *dropout* de 0.4. Al igual que ha ocurrido con los experimentos de 6 capas y un *dropout* de 0.4, observamos que el índice kappa es bastante bajo en la mayoría de los datos, y en el caso de 128 canales, el índice kappa es algo más grande que el resto de estos experimentos, pero aún así, es menor que estos mismos experimentos con un menor *dropout*.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.349	0.361
32	0.373	0.385
64	0.408	0.435
128	0.475	0.547

Validación		
Kernel / Canales	**16**	**32**
16	0.357	0.352
32	0.357	0.387
64	0.388	0.410
128	0.413	0.419

Tabla 6.9: Resultados índice kappa modelo CNN 1 dimensión, *dropout* = 0.4, 8 capas

6.1.2. Experimentación con *data augmentation*

En esta sección vamos a aplicar *data augmentation* a los individuos que no padecen apnea, en el conjunto de datos de entrenamiento. Este tipo de individuos, representan un grupo minoritario, por lo que resultan de gran importancia unos entrenamientos con más datos, también así para eliminar el sobreajuste en los modelos. A continuación se analizan los resultados de la experimentación.

6.1.2.1. *Dropout* = 0.1, 6 capas

En la tabla 6.10 estudiamos los resultados de la experimentación con los modelos con 6 capas y 0.1 *dropout*.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.403	0.447	0.468
32	0.504	0.571	0.559
64	0.586	0.659	0.692
128	0.745	0.795	0.834

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.392	0.409	0.415
32	0.396	0.415	0.420
64	0.403	0.408	0.399
128	0.413	0.418	0.407

Tabla 6.10: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.1, 6 capas

Podemos ver que sí que ha aumentado en unos cuantos casos el valor del índice kappa respecto a los experimentos sin *data augmentation* mostrados en la tabla 6.2. Es más notoria esta diferencia en el caso del modelo con 128 canales y un tamaño de *kernel* de 64, pasando de 0.399 a 0.418.

6.1.2.2. *Dropout* = 0.1, 8 capas

En la tabla 6.11 analizamos ver los resultados de la experimentación de los modelos con 8 capas y 0.1 *dropout*. Si comparamos estos resultados con los obtenidos en la experimentación sin *data augmentation* de la tabla 6.3, se ve que tan solo en el caso de los modelos con 128 canales se logran superar los resultados que se obtuvieron previamente.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.399	0.411
32	0.459	0.517
64	0.616	0.716
128	0.823	0.875

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.409	0.413
32	0.406	0.410
64	0.422	0.413
128	0.431	0.412

Tabla 6.11: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.1, 8 capas

6.1.2.3. *Dropout* = 0.2, 6 capas

En la tabla 6.12 analizamos los índices kappa de los modelos con *data augmentation*, 6 capas y un *dropout* de 0.2. Si lo comparamos con los experimentos de este mismo tipo pero con un *dropout* de 0.1, vemos que en unos cuantos casos los resultados mejoran ligeramente. Si comparamos estos resultados con los obtenidos con esta misma configuración de hiperparámetros pero sin *data augmentation*, vemos que los resultados son muy similares.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.389	0.412	0.430
32	0.43	0.459	0.514
64	0.481	0.540	0.583
128	0.600	0.661	0.706

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.401	0.402	0.414
32	0.419	0.423	0.399
64	0.405	0.414	0.412
128	0.406	0.398	0.415

Tabla 6.12: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.2

6.1.2.4. *Dropout* = 0.2, 8 capas

En la tabla 6.13 revisamos los resultados de la experimentación con modelos con 8 capas y un *dropout* de 0.2, además de *data augmentation*. Si comparamos estos resultados con los obtenidos para un valor de *dropout* de 0.1, recogidos en la tabla 6.11, podemos ver que se obtiene por norma general, unos mejores resultados.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.378	0.385
32	0.418	0.432
64	0.510	0.561
128	0.679	0.784

Validación		
Kernel / Canales	**16**	**32**
16	0.385	0.403
32	0.412	0.416
64	0.431	0.415
128	0.427	0.418

Tabla 6.13: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.2, 8 capas

6.1.2.5. *Dropout* = 0.3, 6 capas

En la tabla 6.14 se presentan los resultados de los experimentos con modelos con 6 capas, 0.3 *dropout* y *data augmentation*. Apreciamos que los resultados no mejoran mucho respecto a usar

un *dropout* de 0.2. Si nos fijamos en la tabla 6.6 podemos ver que sí que mejora ligeramente el índice kappa al utilizar *data augmentation*, aunque las diferencias no son elevadas.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.385	0.389	0.389
32	0.404	0.422	0.448
64	0.427	0.455	0.461
128	0.511	0.543	0.591

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.405	0.411	0.386
32	0.396	0.383	0.409
64	0.422	0.416	0.411
128	0.397	0.412	0.409

Tabla 6.14: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.3, 6 capas

6.1.2.6. *Dropout* = 0.3, 8 capas

En la tabla 6.15 vemos el índice kappa de los experimentos de los modelos con 8 capas, un *dropout* de 0.3 y *data augmentation*. Si lo comparamos con los modelos con estas mismas características pero un *dropout* de 0.2, los resultados apenas cambian. Si nos fijamos en el modelo con 128 canales y un tamaño de *kernel* de 32, vemos que es un valor del índice kappa más alto que el resto y más alto que en el modelo con un *dropout* de 0.2 y también mayor que en el modelo con los mismos hiperparámetros pero sin *data augmentation*.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.359	0.364
32	0.388	0.398
64	0.446	0.482
128	0.552	0.618

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.367	0.350
32	0.377	0.407
64	0.405	0.413
128	0.423	0.439

Tabla 6.15: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.3, 8 capas

6.1.2.7. *Dropout* = 0.4, 6 capas

En la tabla 6.16 analizamos los resultados de los experimentos de los modelos con 6 capas, un *dropout* de 0.4 y *data augmentation*. Podemos ver que un *dropout* de 0.4 no presenta ventajas respecto a un *dropout* de 0.3, ya que los índices kappa no mejoran, por lo que tampoco mejora la calidad de los ajustes. Si comparamos estos resultados con los experimentos sin *data augmentation*

(tabla 6.8), se aprecia que tampoco hay una mejora sustancial, los valores del índice kappa parece que se mantienen.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.375	0.381	0.386
32	0.387	0.389	0.407
64	0.390	0.394	0.415
128	0.428	0.485	0.478

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.398	0.412	0.384
32	0.366	0.396	0.377
64	0.415	0.384	0.377
128	0.381	0.380	0.405

Tabla 6.16: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.4

6.1.2.8. *Dropout* = 0.4, 8 capas

En la tabla 6.17 podemos ver el resultado de los experimentos de los modelos con 8 capas, un *dropout* de 0.4 y *data augmentation*. Si lo comparamos con los modelos que tienen un *dropout* de 0.3, no mejora nada un mayor *dropout*. También, si lo comparamos con los modelos sin *data augmentation*, los resultados son muy similares, tampoco hay mejoría alguna.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.358	0.360
32	0.379	0.385
64	0.425	0.442
128	0.484	0.523

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.204	0.355
32	0.338	0.364
64	0.395	0.395
128	0.406	0.415

Tabla 6.17: Resultados índice kappa CNN 1 dim. con *data augmentation*, *dropout* = 0.4, 8 capas

6.1.3. *Data augmentation* en todas las clases

En este apartado vamos a seleccionar los modelos que han obtenido un mejor índice kappa en validación y vamos a aplicar *data augmentation* a todos los tipos de apnea, no solo a los pacientes que no padecen apnea. El motivo para aplicar *data augmentation* es que en los modelos entrenados previamente, se ha visto que existe sobreajuste, al tener un valor del índice kappa en entrenamiento bastante mayor que en validación. De esta forma se esperan obtener unas mejores predicciones, al tener más datos para entrenar. Los modelos seleccionados han sido los cuatro mejores sin *data*

augmentation y los cuatro mejores con *data augmentation* en los individuos que según su índice de apnea-hipopnea, no padecen apnea. En caso de que un mismo modelo coincida en estos dos grupos, se elegirá el siguiente modelo con un mayor índice kappa.

En la tabla 6.18 vemos una comparación del índice kappa de los modelos seleccionados en su forma base (es decir, sin *data augmentation* o con *data augmentation* en el grupo de pacientes sin apnea) y con *data augmentation* en todas las clases.

	Entrenamiento		Validación	
	Base	*D.augm*	Base	*D.augm*
8 capas, 0.1 *dropout*, 64 canales, 16 *kernel*	0.610	0.585	0.429	0.447
6 capas, 0.2 *dropout*, 64 canales, 32 *kernel*	0.481	0.455	0.428	0.426
8 capas, 0.2 *dropout*, 128 canales, 16 *kernel*	0.657	0.636	0.435	0.437
8 capas, 0.2 *dropout*, 128 canales, 32 *kernel*	0.765	0.619	0.426	0.441
	Data augmentation			
8 capas, 0.1 *dropout*, 128 canales, 16 *kernel*	0.823	0.801	0.431	0.449
8 capas, 0.2 *dropout*, 64 canales, 16 *kernel*	0.510	0.497	0.431	0.431
8 capas, 0.3 *dropout*, 128 canales, 32 *kernel*	0.618	0.635	0.439	0.441
8 capas, 0.3 *dropout*, 128 canales, 16 *kernel*	0.552	0.554	0.423	0.434

Tabla 6.18: Resultados índice kappa CNN 1 dimensión: *data augmentation* en todas las clases

Podemos comprobar que, el valor del índice kappa en entrenamiento es muy similar en los modelos base y con *data augmentation*, incluso llegando a ser algo inferior en el segundo caso. Sin embargo, si nos fijamos en los resultados en validación, que son los que realmente más nos interesan, podemos ver que en casi todos los casos el índice kappa crece con respecto a los modelos base. El modelo que mejor resultado ofrece es el modelo con 8 capas, 128 canales, un tamaño de *kernel* de 16 y un *dropout* de 0.1, alcanzando un índice kappa de 0.449.

6.1.4. Regresión Ridge

Al igual que que se ha comentado en el apartado anterior, se ha visto que se produce sobreajuste en el entrenamiento de muchos de los modelos anteriores. Se ha decidido aplicar una técnica de regularización, una regresión Ridge (L2), ya explicada en el capítulo 3 para evitar que se produzca sobreajuste y se obtenga un mejor resultado en validación.

En la tabla 6.19 se estudian los modelos que se seleccionaron en el apartado anterior. Se comparan los valores del índice kappa del modelo en su forma base con este mismo modelo pero aplicando regresión Ridge.

	Entrenamiento		Validación	
	Base	Ridge	Base	Ridge
8 capas, 0.1 *dropout*, 64 canales, 16 *kernel*	0.610	0.498	0.429	0.442
6 capas, 0.2 *dropout*, 64 canales, 32 *kernel*	0.481	0.427	0.428	0.406
8 capas, 0.2 *dropout*, 128 canales, 16 *kernel*	0.657	0.464	0.435	0.425
8 capas, 0.2 *dropout*, 128 canales, 32 *kernel*	0.765	0.466	0.426	0.434
	Data augmentation			
8 capas, 0.1 *dropout*, 128 canales, 16 *kernel*	0.823	0.540	0.431	0.441
8 capas, 0.2 *dropout*, 64 canales, 16 *kernel*	0.510	0.425	0.431	0.410
8 capas, 0.3 *dropout*, 128 canales, 32 *kernel*	0.618	0.423	0.439	0.429
8 capas, 0.3 *dropout*, 128 canales, 16 *kernel*	0.552	0.430	0.423	0.414

Tabla 6.19: Resultados índice kappa CNN 1 dimensión: regresión Ridge

Vemos que el uso de una regresión Ridge apenas mejora el valor del índice kappa en validación, tan solo mejora en la mitad de los modelos con los que se ha comparado. Se observa que sí que se reduce la diferencia entre el índice kappa de los modelos en entrenamiento y en validación, aunque no es lo que estamos buscando ya que se está produciendo una disminución del índice kappa en entrenamiento, no en validación, que es realmente lo que se busca.

6.1.5. CNN 1 dimensión por tipos de apnea

En este apartado se plantea la posibilidad de que las redes neuronales no estudien e intenten predecir el número de apneas totales, sino por tipos de eventos, es decir, intentar predecir por separado las apneas obstructiva, central e hipopnea. Según el tipo de apnea al que nos enfrentamos, puede que se obtengan resultados diferentes. Por ejemplo, en el caso de las apneas centrales, el flujo de aire a los pulmones para completamente, por lo que las bandas torácicas y abdominal no registrarán movimiento alguno. Este tipo de señal, al tener unos eventos más marcados, debería resultar más fácil para la red neuronal predecir estos eventos. Para el caso de la hipopnea, se puede intuir que los resultados serán peores, debido que esta es tan solo una reducción del flujo aéreo, por lo que es más difícil que sea captada por las bandas torácicas y abdominales.

Se ha escogido el mejor modelo, aquel con un mayor índice kappa en validación, tanto en los modelos base como en los modelos con *data augmentation*, y se han ajustado estos modelos con los tres tipos diferentes de apnea que se desean predecir, junto con el entrenamiento de estos mismos modelos pero aplicando técnicas de regularización que ya se han aplicado en otros modelos previamente: regresión Ridge y *data augmentation* en todos los individuos.

6.1.5.1. 8 capas, 0.2 *dropout*, 128 canales, 16 *kernel*

En la tabla 6.20 se estudian los resultados de los experimentos para el modelo con 8 capas, un *dropout* de 0.2, 128 canales y un tamaño de *kernel* de 16. Como se ha comentado antes, se ve que para el caso de la apnea central, el índice kappa es bastante mejor que para los otros dos tipos de apnea, llegando a alcanzar un valor de 0.797. Para el caso de la apnea obstructiva, se siguen obteniendo resultados más elevados que los que se obtuvieron haciendo predicciones para los tres tipos de apnea de forma conjunta. Para el caso de la hipopnea, como se pudo intuir previamente, se obtienen resultados inferiores, apenas alcanzando un valor del índice kappa de 0.35.

	Entrenamiento			Validación		
	OSA	Central	Hipopnea	OSA	Central	Hipopnea
Base	0.713	0.892	0.664	0.563	0.797	0.339
Ridge	0.514	0.460	0.427	0.558	0.783	0.347
D. augmentation todas las clases	0.736	0.893	0.615	0.594	0.730	0.354

Tabla 6.20: Resultados índice kappa CNN 1 dimensión según tipo de apnea

Si nos fijamos en los valores de los índices kappa, para la apnea obstructiva y la hipopnea, el mejor valor se alcanza en el modelo en el que se aplica *data augmentation* sobre todas las clases. Para la apnea central, se alcanza en el modelo base.

6.1.5.2. 8 capas, 0.3 *dropout*, 128 canales, 32 *kernel*, *data augmentation*

En la tabla 6.21 se muestran los resultados de los entrenamientos del modelo con 8 capas, un *dropout* de 0.3, 128 canales y un tamaño de *kernel* de 16. Si nos fijamos en los resultados obtenidos, se ve que tanto en el caso de la apnea obstructiva como en el caso de la apnea central, se obtiene el mejor resultado al aplicar *data augmentation* a todas las clases. En el caso de la hipopnea, el mejor resultado se obtiene al aplicar una regresión Ridge al modelo.

Si comparamos los resultados con los obtenidos con el modelo anterior, vemos que apenas existen diferencias en el índice kappa de los mejores resultados obtenidos.

	Entrenamiento			Validación		
	OSA	Central	Hipopnea	OSA	Central	Hipopnea
Base	0.639	0.766	0.606	0.574	0.683	0.339
Ridge	0.477	0.519	0.380	0.541	0.580	0.352
D. augmentation todas las clases	0.703	0.892	0.611	0.592	0.783	0.348

Tabla 6.21: Resultados índice kappa CNN 1 dimensión según tipo de apnea

6.1.6. Resultados sobre test

En esta sección vamos a analizar más en profundidad los modelos con los que se han obtenido los índices kappa en validación más altos. Vamos a analizar la evolución del índice kappa en cada época para los conjuntos de entrenamiento y validación. También, para cada modelo, se mostrará la sensibilidad, especifidad, valor predictivo positivo y valor predictivo negativo, medidos en función de los distintos umbrales de severidad de la apnea (5 e/h, 15 e/h y 30 e/h), es decir, contando los que caen por encima como positivos y los que caen por debajo como negativos, para el conjunto de datos de test. Estas métricas sirven para saber si predice un determinado grado de severidad o superior. Además, para cada modelo, se mostrará el índice kappa, medido sobre los conjuntos de entrenamiento, validación y test, junto con la tasa de acieto (*accuracy*) en el conjunto de test. Por último, se mostrará la matriz de confusión de estos modelos para el conjunto de datos de test.

Se ha decidido estudiar modelos variados, es decir, un modelo en su forma base o con *data augmentation* solo en los individuos sin apnea, otro con regresión Ridge y otro aplicando *data augmentation* a todas las clases. A pesar de que los modelos elegidos no son los modelos con los mejores resultados absolutos, sí son los que mejor índice kappa tienen dentro de los de su tipo. Los modelos elegidos han sido los siguientes:

1. **Modelo 1**: modelo con 8 capas, un *dropout* de 0.3, 128 canales y un tamaño de *kernel* de

32, aplicando *data augmentation* a los individuos que no tienen apnea.

2. **Modelo 2**: modelo con 8 capas, un *dropout* de 0.1, 64 canales y un tamaño de *kernel* de 16, aplicando regresión Ridge.

3. **Modelo 3**: modelo con 8 capas, un *dropout* de 0.1, 128 canales y un tamaño de *kernel* de 16, aplicando *data augmentation* a todos los individuos.

En la tabla 6.22 podemos ver los resultados obtenidos para estos tres modelos.

	Modelo 1			Modelo 2			Modelo 3		
	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h
Sensibilidad	0.164	0.664	0.894	0.140	0.618	0.869	0.068	0.526	0.842
Especifidad	0.991	0.871	0.639	0.990	0.890	0.673	0.995	0.911	0.705
VPP	0.482	0.747	0.871	0.427	0.763	0.878	0.436	0.772	0.886
VPN	0.957	0.819	0.690	0.956	0.802	0.655	0.953	0.770	0.622
Kappa entrenam.	0.618			0.498			0.801		
Kappa val.	0.439			0.442			0.449		
Kappa test	0.406			0.383			0.343		
Accuracy test	0.597			0.580			0.555		

Tabla 6.22: Resultados sobre test, índice kappa sobre entrenamiento y validación

Se aprecia que la sensibilidad, es decir, la probabilidad de que, si un individuo tiene un IAH superior al umbral considerado en cada caso, ser predicho como tal, de los tres modelos es muy baja en el caso de los individuos que no tienen apnea, es decir, hasta 5 e/h. Esto significa que existe una gran cantidad de individuos que se clasifican como si tuvieran más de 5 eventos por hora, cuando realmente no los tienen. El motivo fundamental de este resultado es la desequilibrada distribución de las clases, hay muy pocos individuos que tengan menos de 5 e/h. La especifidad para este grupo de menos de 5 e/h en cambio es muy alta, en todos los casos cercana a 0.9. Esto se debe a que muy pocos individuos que realmente tengan apnea con más de 5 eventos por hora son predichos como si tuvieran menos de 5 eventos. Estudiando el valor predictivo positivo (VPP) para este mismo grupo de individuos, vemos que tiene un valor de entre 0.42 y 0.48. Esto nos dice, para los individuos que se han predicho como si no tuvieran apnea, cual es la probabilidad de que realmente no la tengan. Fijándonos en el valor predictivo negativo (VPN) para los individuos con menos de 5 e/h vemos que se obtienen valores muy altos, alrededor de 0.95, lo que nos dice que

los tres modelos, cuando un individuos se clasifica como si tiene más 5 e/h, es muy probable que esto sea así. El valor predictivo positivo (VPP) y el valor predictivo negativo (VPN) son métricas más robustas y menos sesgadas por la distribución de clases.

Si nos fijamos en las sensibilidad de los individuos para 15 e/h y 30 e/h, vemos que va aumentando a medida que aumenta el número de eventos, como se ha comentado antes, esta métrica está sesgada por la distribución de clases. El grupo de individuos que tienen 30 eventos de apnea o menos, es mayor que el grupo de los individuos que tienen 15 eventos o menos y por supuesto mucho mayor que el grupo de individuos que tienen 5 eventos o menos. Este mismo sesgo se aprecia en la especifidad, va disminuyendo a medida que cambia la distribución de clases. Si comparamos estas métricas entre los distintos modelos, el modelo 1 es el que nos ofrece una mejor sensibilidad, mientras que el modelo 3 es el que nos ofrece mejores resultados para la especifidad. Tanto el valor predictivo positivo como el valor predictivo negativo para los individuos con umbrales de 15 eventos por hora y 30 e/h tiene valores muy similares a la sensibilidad y especifidad para estos mismos grupos.

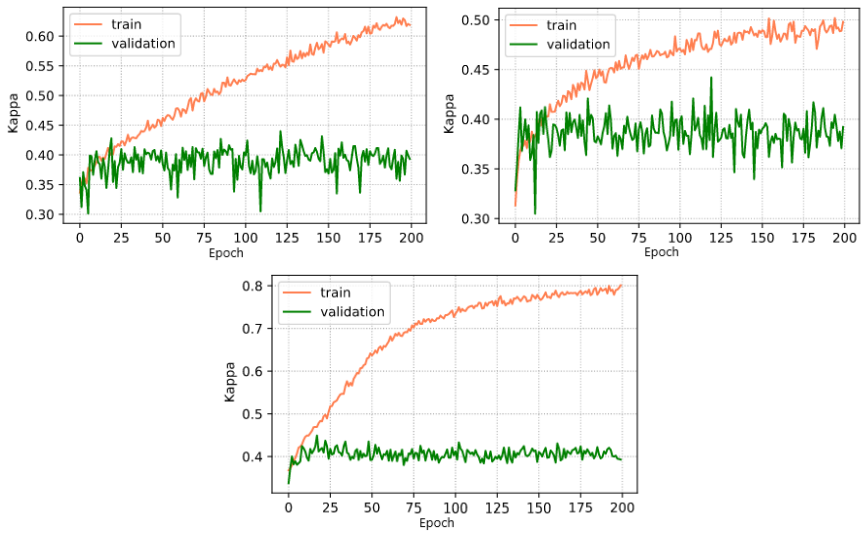

Figura 6.1: Evolución del índice kappa para los modelos 1 (izquierda), 2 (derecha) y 3 (abajo)

En la tabla 6.22 también podemos ver los resultados del índice kappa para los tres conjuntos de datos: entrenamiento, validación y test. El mejor resultado en validación se obtiene en el tercer modelo sin embargo, el mejor índice kappa en test se obtiene para el modelo 1. La tasa de acierto (*accuracy*) sobre el conjunto de test también es mayor en el primer modelo. Si comparamos el índice kappa de los modelos en entrenamiento y validación, sobre todo para el modelo 3, vemos que es mucho mayor en el primer caso, lo que nos indica que se está produciendo sobreajuste. Si nos fijamos en la figura 6.1, podemos ver la evolución del índice kappa a lo largo de las épocas para los modelos en entrenamiento (naranja) y en validación (verde). Observando las figuras comprobamos

que existe sobreajuste, la curva del índice kappa en entrenamiento crece a lo largo de las épocas, sin embargo, en validación, al llegar a un cierto aprendizaje, la curva deja de crecer, es decir, deja de aprender.

En la figura 6.2 podemos ver las matrices de confusión de los tres modelos, a la izquierda el modelo 1, a la derecha el modelo 2 y abajo el modelo 3. Las matrices de confusión están construidas de forma que los porcentajes y los colores se muestran por filas, es decir, para los valores reales de apnea de los individuos. La clase 0 se corresponde con no apnea (IAH $<$ 5 e/h), la clase 1 se corresponde con apnea leve (5 e/h $\leq$ IAH $<$ 15 e/h), la clase 2 se corresponde con apnea moderada (15 e/h $\leq$ IAH $<$ 30 e/h) y la clase 3 se corresponde con apnea severa (IAH $\geq$ 30 e/h).

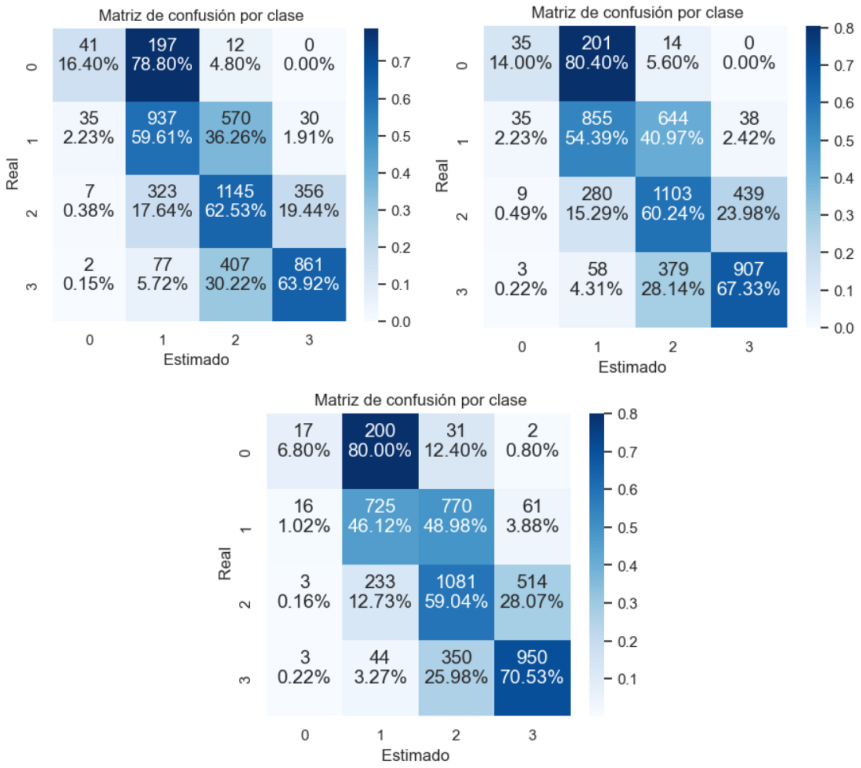

Figura 6.2: Matrices de confusión de los modelos 1 (izquierda), 2 (derecha) y 3 (abajo)

En las matrices de confusión vemos lo que ya se ha comentado, a medida que aumenta el grado de severidad de la apnea, las predicciones son mejores. También, para el caso de los individuos que no tienen apnea, se ve por qué la sensibilidad era tan baja, la mayoría de los individuos que no tienen apnea se predicen como si tuvieran apnea leve.

A continuación vamos a analizar el modelo con el mejor índice kappa para cada tipo de evento de apnea. Se utilizarán las mismas métricas que para los tres modelos anteriores. Los modelos que

vamos a estudiar son:

1. **OSA**: modelo con 8 capas, un *dropout* de 0.2, 128 canales y un tamaño de *kernel* de 16, aplicando *data augmentation* a todos los individuos.
2. **Apnea central**: modelo con 8 capas, un *dropout* de 0.2, 128 canales y un tamaño de *kernel* de 16, aplicando regresión Ridge.
3. **Hipopnea**: modelo con 8 capas, un *dropout* de 0.2, 128 canales y un tamaño de *kernel* de 16, aplicando *data augmentation* a todos los individuos.

En la tabla 6.23 se muestran los resultados obtenidos para estos tres modelos.

	OSA			Central			Hipopnea		
	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h
Sensibilidad	0.905	0.979	0.995	0.999	0.999	-	0.153	0.681	0.944
Especifidad	0.715	0.707	0.615	0.406	0.227	-	0.986	0.816	0.409
VPP	0.940	0.989	0.997	0.988	0.996	-	0.407	0.737	0.872
VPN	0.604	0.556	0.500	0.848	0.833	-	0.950	0.772	0.633
Kappa entrenam.	0.736			0.460			0.615		
Kappa val.	0.563			0.783			0.354		
Kappa test	0.503			0.445			0.334		
Accuracy test	0.844			0.984			0.565		

Tabla 6.23: Resultados sobre test, índice kappa sobre entrenamiento y validación según el tipo de apnea

Antes de comentar la tabla, se debe mencionar que no se presentan los resultados para un umbral de 30 eventos por hora para la apnea central ya que ningún individuo presenta tantos tipos de apnea de este tipo.

Se aprecia que la sensibilidad para los modelos de apnea obstructiva y central es muy alta para los tres umbrales de apnea, lo que significa que la probabilidad de clasificar a los individuos en un determinado grupo de apnea, cuando realmente pertenecen a ese grupo, es muy alta. En el caso de la hipopnea, la sensibilidad es bastante baja para los individuos que tienen menos de 5 eventos por hora, pero a medida que aumenta el número de eventos, también lo hace la sensibilidad.

La especifidad para la apnea obstructiva es bastante similar para todos los diferentes umbrales de número de eventos por hora. Para el caso de la apnea central, la especifidad es bastante peor. Y para el caso de la hipopnea, es muy alta para el grupo de individuos con menos de 5 eventos por hora, pero va disminuyendo a medida que aumenta el número de eventos por hora.

Respecto al valor predictivo positivo podemos ver que para los casos de la apnea obstructiva y central se obtienen valores muy altos, indicando que al predecir que un individuo pertenece a un grado de severidad concreto, es muy probable que realmente pertenezca a este grupo. El valor predictivo positivo para el caso de la hipopnea es bastante más bajo, aunque a medida que aumenta el número de eventos, aumenta, alcanzando un valor de 0.87.

El valor predictivo negativo nos da resultados peores para el caso de la apnea obstructiva y la hipopnea cuando hay muchos eventos, lo que significa que si asignamos a un individuo la no pertenencia a un grado de severidad concreto, es algo probable que sí que pertenezca a este grupo.

A continuación nos fijamos en los valores del índice kappa para los distintos modelos. Se aprecia que el mejor resultado sobre test pertenece al modelo que ajusta la apnea obstructiva del sueño, con un valor de 0.503, siendo este valor próximo al índice kappa del modelo en validación. Esto mismo ocurre con el modelo que ajusta la hipopnea, aunque con un índice kappa mucho peor. Para el caso de la apnea central, se aprecia que hay una gran diferencia entre el índice kappa del modelo en validación y en test, esta diferencia no debería ser tan grande. Como se ha comentado antes, para los datos de test no existe ningún individuo que tenga más de 30 eventos por hora, por lo que se intuye que estos problemas pueden estar originados por este motivo. A pesar de esto, el índice kappa en test es bastante bueno, es mejor que en los modelos en los que se estudiaban los eventos de apneicos de forma conjunta.

Al estudiar la tasa de acierto (*accuracy*) de los tres modelos, vemos que la de la apnea central es muy alta, por lo que se podría esperar que el índice kappa también lo fuera, o al menos, que fuera más alto que en los otros modelos, sin embargo, esto no ocurre así. El índice kappa es una métrica que mide la mejora relativa respecto a una predicción aleatoria. Debido a la escasez de individuos con apnea central leve, media y grave, y el pobre rendimiento de este clasificador para dichas categorías, el índice kappa obtiene un valor bajo con respecto a la tasa de acierto, ya que esta última no es robusta al desbalanceo. Cabe esperar que si aumenta la muestra de individuos para dichas categorías, el comportamiento mejore sustancialmente tal y como se extrae de los resultados de validación. La tasa de acierto para la apnea obstructiva es alta, acercándose al 85 %, mientras que la tasa de acierto de la hipopnea es bastante baja, tan solo del 56 %.

En la figura 6.3 podemos ver la evolución del índice kappa a lo largo de las épocas para los modelos en entrenamiento (naranja) y en validación (verde). Se aprecia que existe sobreajuste, sobre todo en el caso de la hipopnea (figura de abajo), debido a que el índice kappa crece a lo largo de todo el entrenamiento, pero no lo hace en el caso de validación.

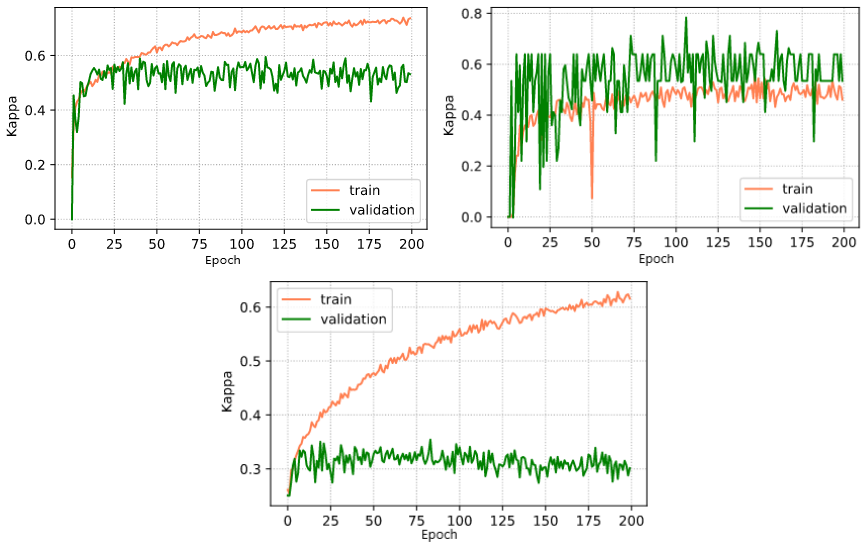

Figura 6.3: Evolución del índice kappa para los modelos de OSA (izquierda), apnea central (derecha) e hipopnea (abajo)

Cabe destacar que en la evolución del índice kappa en validación para la apnea central, es muy inestable al principio, pero en las últimas épocas, se muestra mucho más estable. Vemos que para los modelos para la apnea obstructiva y la hipopnea, los modelos son bastante más estables.

En la figura 6.4 se representan las matrices de confusión para los tres modelos de apnea obstructiva (izquierda), apnea central (derecha) e hipopnea (abajo). Están construidas de la misma forma que las matrices de confusión de la figura 6.2, es decir, los cálculos de los porcentajes y los colores de la matriz se calculan por filas, es decir, para cada grupo de tipos de eventos reales.

Como podemos ver, la matriz de confusión de la apnea central tan solo tiene tres clases, esto se debe a lo que ya se ha comentado, no existe ningún individuo que padezca apnea central con más de 30 eventos por hora.

En la matriz de confusión de la apnea obstructiva se puede ver que los colores más fuertes son los que se encuentran en la diagonal, es decir cuando se acierta con las predicciones. En la matriz de confusión de la apnea central se puede ver también por qué se obtiene una tasa de acierto tan alta: realmente se acierta en la mayoría de individuos, pero esto es porque la mayoría no tienen apnea. Las predicciones para las otras clases son sensiblemente inferiores, por lo que el índice kappa, al ser menos sensible a la distribución de clases, nos da un peor resultado. La matriz de confusión de la hipopnea se equivoca bastante, vemos colores bastante intensos en las celdas cercanas a la diagonal. Este es el principal motivo por el que tanto el índice kappa como la tasa de acierto, son las más bajas.

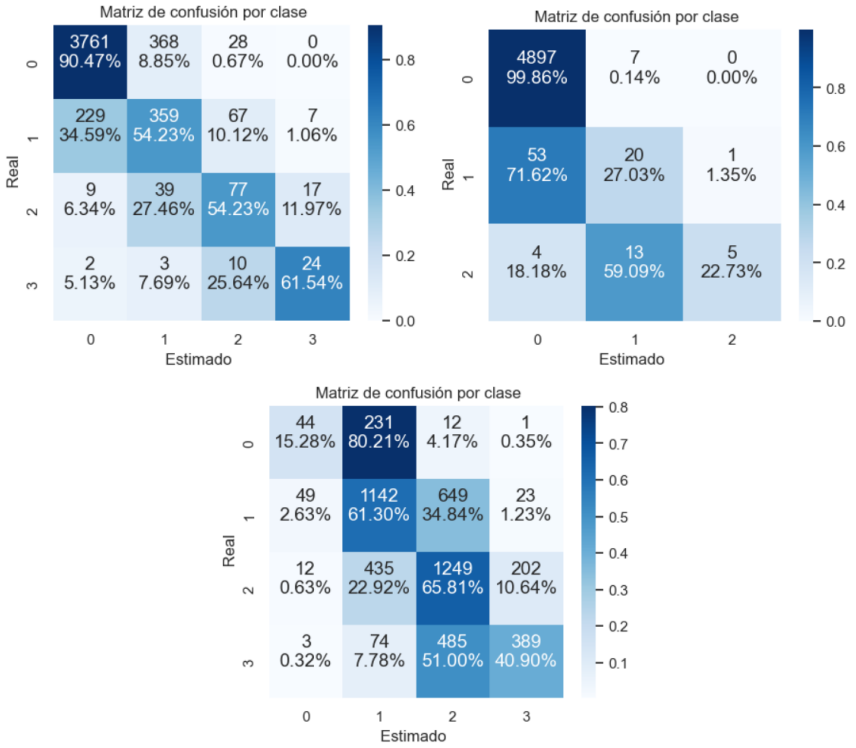

Figura 6.4: Matrices de confusión de los modelos de OSA (izquierda), apnea central (derecha) e hipopnea (abajo)

6.2. CNN 2 dimensiones

A continuación se muestra la experimentación para la optimización de hiperparámetros a partir del modelo base explicado en el anterior capítulo. Posteriormente, se elegirán los modelos que presenten los mejores resultados y se evaluarán sobre un conjunto de datos de test.

6.2.1. Experimentación

En esta sección vamos optimizar los hiperparámetros del modelo base. Procederemos de la misma forma que con los modelos de redes convolucionales de una dimensión. Para los entrenamientos, utilizaremos 200 épocas, una tasa de aprendizaje de 0.001 y un tamaño de *batch* de 32. Como criterio de optimización, se utilizará la función de pérdida de Hubber y Adam como optimizador. El espacio de búsqueda de hiperparámetros también será el indicado en la tabla 6.1 y también estudiaremos el índice kappa, tal y como se ha explicado para las redes convolucionales

de una dimensión.

En los siguientes apartados se mostrarán los resultados de los múltiples experimentos elaborados con las distintas combinaciones de hiperparámetros de la tabla 6.1.

6.2.1.1. *Dropout* = 0.1, 6 capas

En primer lugar, probamos con modelos con 6 capas y un *dropout* de 0.1. En la tabla 6.24 podemos ver que los índices kappa en validación obtenidos son más bajos que los que se obtuvieron para estos mismos modelos pero con capas convolucionales de una dimensión. El valor del índice kappa más alto en validación lo encontramos para el modelo con 64 capas y un tamaño de *kernel* de 64.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.419	0.447	0.459
32	0.469	0.472	0.507
64	0.617	0.633	0.691
128	0.732	0.777	0.826

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.403	0.394	0.387
32	0.388	0.395	0.392
64	0.383	0.413	0.388
128	0.379	0.395	0.380

Tabla 6.24: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.1, 6 capas

6.2.1.2. *Dropout* = 0.1, 8 capas

En la tabla 6.25 se muestran los resultados de los entrenamientos de los modelos con 8 capas y un *dropout* de 0.1. Si comparamos estos resultados con los modelos de 6 capas, vemos que los índices kappa son más altos, superando valores de 0.4 en casi todas las ocasiones. El índice kappa más alto en validación lo encontramos en el modelo con 64 capas y un tamaño de *kernel* de 16.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.403	0.417
32	0.466	0.494
64	0.626	0.714
128	0.821	0.888

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.400	0.408
32	0.403	0.410
64	0.426	0.393
128	0.405	0.400

Tabla 6.25: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.1, 8 capas

6.2.1.3. *Dropout* = 0.2, 6 capas

En la tabla 6.26 vemos los resultados de los entrenamientos de los modelos con 6 capas y un *dropout* de 0.2. Comparando los resultados con los de los modelos con un *dropout* de 0.1 en la tabla 6.24, vemos que los resultados mejoran levemente en muchos casos. Si comparamos estos resultados con los obtenidos con los modelos con capas convolucionales de una dimensión, vemos que los resultados son muy similares, con valores algo más pequeños del índice kappa en algunos casos.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.408	0.414	0.428
32	0.418	0.448	0.452
64	0.511	0.498	0.540
128	0.612	0.666	0.701

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.407	0.402	0.410
32	0.399	0.391	0.398
64	0.397	0.403	0.388
128	0.400	0.408	0.395

Tabla 6.26: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.2, 6 capas

6.2.1.4. *Dropout* = 0.2, 8 capas

En la tabla 6.27 se muestran los resultados de los modelos con 8 capas y un *dropout* de 0.2. Se aprecia que, por lo general, se obtienen resultados algo peores que en los modelos con un *dropout* de 0.1, salvo en el caso de los modelos con 128 capas, en los que se obtienen mejores resultados.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.391	0.389
32	0.425	0.439
64	0.498	0.555
128	0.694	0.807

Validación		
Kernel / Canales	**16**	**32**
16	0.389	0.382
32	0.390	0.389
64	0.396	0.416
128	0.418	0.409

Tabla 6.27: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.2, 8 capas

6.2.1.5. *Dropout* = 0.3, 6 capas

En la tabla 6.28 vemos los resultados de los modelos con 6 capas y un *dropout* de 0.3.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.394	0.415	0.414
32	0.405	0.421	0.436
64	0.440	0.452	0.471
128	0.504	0.624	0.679

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.397	0.398	0.408
32	0.403	0.396	0.388
64	0.382	0.403	0.394
128	0.398	0.398	0.383

Tabla 6.28: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.3, 6 capas

Comparando estos resultados con los de los modelos con *dropout* de 0.2, vemos que se obtienen unos valores del índice kappa muy similares, lo que nos indica que un mayor *dropout* no está mejorando nuestros modelos.

6.2.1.6. *Dropout* = 0.3, 8 capas

En la tabla 6.29 aparecen los resultados del entrenamiento de los modelos con 8 capas y un *dropout* de 0.3. Si nos fijamos en los modelos con un menor número de canales y los comparamos con los correspondientes modelos con un *dropout* de 0.2, vemos que no se aprecian diferencia en los índices kappa, incluso en los modelos con solo 16 canales, empeora. Sin embargo, en los modelos con un mayor número de canales, los resultados son iguales o mejores que los obtenidos con un menor *dropout*.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.369	0.387
32	0.412	0.416
64	0.443	0.473
128	0.560	0.679

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.316	0.349
32	0.381	0.390
64	0.407	0.417
128	0.418	0.418

Tabla 6.29: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.3, 8 capas

6.2.1.7. *Dropout* = 0.4, 6 capas

En la tabla 6.30 se recogen los resultados del entrenamiento de los modelos con 6 capas y un *dropout* de 0.4.

Entrenamiento			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.385	0.396	0.388
32	0.388	0.393	0.410
64	0.409	0.412	0.422
128	0.455	0.476	0.547

Validación			
Canales \ *Kernel*	**32**	**64**	**128**
16	0.388	0.388	0.396
32	0.391	0.387	0.402
64	0.381	0.395	0.391
128	0.340	0.400	0.381

Tabla 6.30: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.4, 6 capas

Al igual que ocurría al comparar los modelos con un *dropout* de 0.3, estos modelos, al tener un mayor *dropout*, tampoco están mejorando, o incluso están obteniéndose valores del índice kappa peores.

6.2.1.8. *Dropout* = 0.4, 8 capas

En la tabla 6.31 se muestran los resultados de los modelos con 8 capas y un *dropout* de 0.4. Se puede ver que estos resultados no son muy altos y que tampoco mejoran los que se obtuvieron con un *dropout* de 0.3, aunque tampoco empeoran enormemente los resultados.

Entrenamiento		
Canales \ *Kernel*	**16**	**32**
16	0.354	0.366
32	0.378	0.392
64	0.410	0.426
128	0.469	0.552

Validación		
Canales \ *Kernel*	**16**	**32**
16	0.362	0.321
32	0.362	0.337
64	0.362	0.395
128	0.414	0.410

Tabla 6.31: Resultados índice kappa CNN 2 dimensiones, *dropout* = 0.4, 8 capas

6.2.2. Experimentación con *data augmentation*

En esta sección se va a aplicar *data augmentation* a los individuos que no padecen apnea, en el conjunto de datos de entrenamiento. El motivo es un intento de mejorar los resultados obtenidos previamente y de evitar el sobreajuste, debido a que los individuos que no padecen apnea son bastante minoritarios. A continuación, se analizan los resultados de la experimentación.

6.2.2.1. *Dropout* = 0.1, 6 capas

En la tabla 6.32 vemos ver los resultados de la experimentación para los modelos con 6 capas y un *dropout* de 0.1. Se aprecia que los resultados mejoran ligeramente en muchos de los casos, respecto a los modelos en los que no se aplica *data augmentation*. No se reduce el sobreajuste en modelos con muchos canales, al comparar los índices kappa de estos modelos en entrenamiento con los de validación. Los primeros siguen teniendo valores altos, mientras que los segundos siguen teniendo valores más bajos.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.426	0.466	0.482
32	0.481	0.507	0.505
64	0.632	0.668	0.687
128	0.781	0.813	0.783

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.403	0.399	0.406
32	0.402	0.409	0.391
64	0.387	0.387	0.393
128	0.381	0.399	0.410

Tabla 6.32: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.1, 6 capas

6.2.2.2. *Dropout* = 0.1, 8 capas

En la tabla 6.33 se muestran los resultados de la experimentación con los modelos con 8 capas y un *dropout* de 0.1. Comparando estos resultados con los que se han obtenido para el modelo sin *data augmentation* vemos que los índices kappa son muy similares. El mayor índice kappa se obtiene para el modelo con 32 canales y un tamaño de *kernel* de 32.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.408	0.418
32	0.465	0.504
64	0.655	0.733
128	0.835	0.902

Validación		
Kernel / Canales	**16**	**32**
16	0.405	0.401
32	0.425	0.409
64	0.406	0.413
128	0.407	0.403

Tabla 6.33: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.1, 8 capas

6.2.2.3. *Dropout* = 0.2, 6 capas

En la tabla 6.34 podemos ver los resultados de los experimentos de los modelos con 6 capas y un *dropout* de 0.2. Al igual que en la tabla anterior, si comparamos con los modelos entrenados sin *data augmentation* no se aprecian diferencias en el índice kappa de ambos. El modelo que en este caso da un mejor índice kappa en validación es el que tiene 16 canales y un tamaño de *kernel* de 128.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.408	0.423	0.428
32	0.426	0.448	0.458
64	0.489	0.543	0.526
128	0.624	0.673	0.702

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.399	0.385	0.411
32	0.407	0.399	0.406
64	0.386	0.406	0.400
128	0.415	0.394	0.389

Tabla 6.34: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.2, 6 capas

6.2.2.4. *Dropout* = 0.2, 8 capas

En la tabla 6.35 se presentan los resultados para los modelos con 8 capas y un *dropout* de 0.2. Comparando estos resultados con los obtenidos para este mismo modelo pero con un *dropout* de 0.1 vemos que por lo general, para los modelos con un menor número de capas, el índice kappa disminuye, pero para los modelos con 128 capas, el índice kappa en un caso se mantiene y en otro mejora.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.396	0.385
32	0.412	0.441
64	0.517	0.571
128	0.685	0.793

Validación		
Kernel / Canales	**16**	**32**
16	0.387	0.364
32	0.398	0.412
64	0.398	0.411
128	0.422	0.403

Tabla 6.35: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.2, 8 capas

6.2.2.5. *Dropout* = 0.3, 6 capas

En la tabla 6.36 se muestran los resultados para los modelos con 6 capas y un *dropout* de 0.3. Comparando estos resultados con los de este mismo modelo pero con un *dropout* de 0.2, vemos que se obtienen resultados muy parecidos, por lo que para este tipo de modelos, un mayor *dropout* ya no mejora los resultados.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.400	0.400	0.410
32	0.404	0.425	0.429
64	0.451	0.452	0.478
128	0.505	0.547	0.630

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.406	0.400	0.398
32	0.383	0.389	0.395
64	0.396	0.395	0.399
128	0.401	0.394	0.401

Tabla 6.36: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.3, 6 capas

6.2.2.6. *Dropout* = 0.3, 8 capas

En la tabla 6.37 vemos los resultados para los modelos con 8 capas y un *dropout* de 0.3. Si nos fijamos en los resultados de estos mismos modelos pero con un *dropout* de 0.2, vemos lo mismo que nos ocurría en el caso anterior, un mayor *dropout* no mejora los resultados. Se aprecia una gran diferencia entre los modelos de esta tabla según el número de capas. Los modelos con 16 capas tienen un índice kappa bastante inferior al obtenido con modelos con 128 capas.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.380	0.388
32	0.408	0.422
64	0.444	0.486
128	0.573	0.678

Validación		
Kernel / Canales	**16**	**32**
16	0.361	0.362
32	0.373	0.408
64	0.393	0.395
128	0.410	0.403

Tabla 6.37: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.3, 8 capas

6.2.2.7. *Dropout* = 0.4, 6 capas

En la tabla 6.38 se muestran los resultados para los modelos con 6 capas y un *dropout* de 0.4. Comparando estos resultados con los obtenidos con un *dropout* de 0.3, vemos que empeoran,

por lo que el uso de un mayor *dropout* resultaría inútil. Los modelos de esta tabla con un mayor tamaño de *kernel* parece que presentan unos mejores resultados, pero siguen siendo peores que otros resultados que hemos obtenido para otros modelos.

Entrenamiento			
Kernel / Canales	**32**	**64**	**128**
16	0.386	0.394	0.397
32	0.386	0.388	0.400
64	0.398	0.414	0.425
128	0.450	0.455	0.535

Validación			
Kernel / Canales	**32**	**64**	**128**
16	0.333	0.390	0.397
32	0.378	0.383	0.401
64	0.361	0.396	0.391
128	0.383	0.402	0.397

Tabla 6.38: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.4, 6 capas

6.2.2.8. *Dropout* = 0.4, 8 capas

En la tabla 6.39 se presentan los resultados para los modelos con 8 capas y un *dropout* de 0.4. Como ya se comentó antes, un mayor *dropout* para estos modelos, no mejora su aprendizaje. Se aprecia diferencia entre los índices kappa de los modelos con un menor número de capas y los que tienen 128 capas. El modelo con un mayor índice kappa es el que cuenta con 64 capas y un tamaño de *kernel* de 32.

Entrenamiento		
Kernel / Canales	**16**	**32**
16	0.538	0.370
32	0.385	0.391
64	0.411	0.428
128	0.479	0.540

Validación		
Kernel / Canales	**16**	**32**
16	0.384	0.320
32	0.374	0.375
64	0.377	0.418
128	0.412	0.413

Tabla 6.39: Resultados índice kappa CNN 2 dim. con *data augmentation*, *dropout* = 0.4, 8 capas

6.2.3. *Data augmentation* en todas las clases

Al igual que en los modelos con capas convolucionales de una dimensión, vamos a seleccionar los modelos que han obtenido un mejor índice kappa en validación y vamos a aplicar *data augmentation* a todos los individuos, para evitar el sobreajuste que se ve en los modelos con un índice kappa en entrenamiento mucho mayor que en validación.

	Entrenamiento		Validación	
	Base	D.augm	Base	D.augm
8 capas, 0.1 *dropout*, 64 canales, 16 *kernel*	0.626	0.588	0.426	0.433
8 capas, 0.2 *dropout*, 128 canales, 16 *kernel*	0.694	0.634	0.418	0.427
8 capas, 0.3 *dropout*, 128 canales, 16 *kernel*	0.560	0.562	0.418	0.451
8 capas, 0.3 *dropout*, 128 canales, 32 *kernel*	0.679	0.626	0.418	0.432
	Data augmentation			
6capas, 0.1 *dropout*, 128 canales, 128 *kernel*	0.783	0.689	0.410	0.410
8 capas, 0.1 *dropout*, 64 canales, 32 *kernel*	0.655	0.660	0.406	0.432
6 capas, 0.2 *dropout*, 128 canales, 32 *kernel*	0.624	0.537	0.415	0.421
8 capas, 0.4 *dropout*, 128 canales, 32 *kernel*	0.540	0.546	0.413	0.450

Tabla 6.40: Resultados índice kappa CNN 2 dimensiones: *data augmentation* en todas las clases

En la tabla 6.40 se presenta una comparación del índice kappa de los modelos seleccionados en su forma base (es decir, sin *data augmentation* o con *data augmentation* en el grupo de pacientes sin apnea) y con *data augmentation* en todas las clases. Podemos comprobar que el índice kappa en entrenamiento es muy similar para los modelos base y con *data augmentation* en todas las clases, incluso en algunos casos disminuye para los segundos. Sin embargo, en todos los modelos salvo en uno, el índice kappa en validación mejora. En algunos casos esta mejora es bastante grande, pasando de un índice kappa de 0.418 o 0.413 a 0.451 y 0.45 respectivamente. Existe un modelo que no mejora, pero tampoco empeora, manteniendo su índice kappa en 0.41. El modelo con un mejor índice kappa es el formado por 8 capas, con un *dropout* de 0.3, 128 canales y u tamaño de *kernel* de 16.

6.2.4. Regresión Ridge

Al igual que en el apartado anterior, al ver que se produce sobreajuste en el entrenamiento de muchos modelos, se ha decidido aplicar una técnica de regularización: una regresión Ridge (L2), buscando obtener unos mejores resultados en validación.

	Entrenamiento		Validación	
	Base	Ridge	Base	Ridge
8 capas, 0.1 *dropout*, 64 canales, 16 *kernel*	0.626	0.484	0.426	0.417
8 capas, 0.2 *dropout*, 128 canales, 16 *kernel*	0.694	0.471	0.418	0.422
8 capas, 0.3 *dropout*, 128 canales, 16 *kernel*	0.560	0.427	0.418	0.433
8 capas, 0.3 *dropout*, 128 canales, 32 *kernel*	0.679	0.424	0.418	0.427
	Data augmentation			
6 capas, 0.1 *dropout*, 128 canales, 128 *kernel*	0.783	0.512	0.410	0.403
8 capas, 0.1 *dropout*, 64 canales, 32 *kernel*	0.655	0.488	0.406	0.410
6 capas, 0.2 *dropout*, 128 canales, 32 *kernel*	0.624	0.455	0.415	0.392
8 capas, 0.4 *dropout*, 128 canales, 32 *kernel*	0.540	0.404	0.413	0.416

Tabla 6.41: Resultados índice kappa CNN 2 dimensiones: regresión Ridge

En la tabla 6.41 se presentan los resultados de los modelos que ya se seleccionaron para aplicar *data augmentation* en todas las clases. Se comparan los índices kappa del modelo en su forma base y aplicando regresión Ridge.

A la vista de los resultados se aprecia que el uso de la regresión Ridge no es tan bueno como la aplicación de *data augmentation* del apartado anterior. En más de la mitad de los casos sí que mejora el índice kappa, pero en otros tres casos, se reduce. El modelo con el que se ha obtenido un mayor índice kappa en validación es el formado por 8 capas, un *dropout* de 0.3, 128 canales y un tamaño de *kernel* de 16, con un valor de 0.433.

6.2.5. CNN 2 dimensiones por tipos de apnea

Al igual que se hizo para los modelos con capas convolucionales de una dimensión, se va a intentar predecir por separado el número de apneas obstructivas, centrales e hipopneas. Se ha escogido el mejor modelo de entre los modelos base y el mejor modelo de entre los modelos en los que se ha aplicado *data augmentation* a los individuos que no tienen apnea. A estos modelos se les ha aplicado regresión Ridge y *data augmentation* a todas sus clases.

6.2.5.1. 8 capas, 0.1 *dropout*, 64 canales, 16 *kernel*

En la tabla 6.42 se presentan los resultados de los experimentos para el modelo con 8 capas, un *dropout* de 0.1, 64 canales y un tamaño de *kernel* de 16. Podemos ver que los mejores resultados en validación se obtienen para la predicción de la apnea central, alcanzando un índice kappa de 0.873 tanto para el modelo base como para el modelo con *data augmentation* en todas las clases. Para este caso de apnea central, se ve que el uso de técnicas de regularización no están funcionando, porque no está mejorando el índice kappa, incluso, al aplicar regresión Ridge, está bajando mucho. Al igual que ocurría para los modelos con capas convolucionales de una dimensión, la predicción de la hipopnea es la que peores resultados nos proporciona, pero en este caso sí que se aprecia que las técnicas de regularización están funcionando.

	Entrenamiento			Validación		
	OSA	Central	Hipopnea	OSA	Central	Hipopnea
Base	0.672	0.776	0.597	0.573	0.873	0.327
Ridge	0.530	0.515	0.44	0.563	0.689	0.339
D. augmentation todas las clases	0.689	0.901	0.547	0.560	0.873	0.350

Tabla 6.42: Resultados índice kappa CNN 2 dimensiones según tipo de apnea

Para el caso de la apnea obstructiva vemos que sí que parece que se ha conseguido reducir el sobreajuste en el caso de la regresión Ridge, debido a que se obtiene un índice kappa muy parecido tanto en entrenamiento como en validación.

6.2.5.2. 6 capas, 0.2 *dropout*, 128 canales, 32 *kernel*, *data augmentation*

En la tabla 6.43 se muestran los resultados de los experimentos para el modelo con 6 capas, un *dropout* de 0.2, 128 canales, un tamaño de *kernel* de 32 y *data augmentation*. Podemos ver los resultados de entrenar los modelos sin *data augmentation* de ningún tipo, aplicando regresión Ridge y aplicando *data augmentation* a todas las clases. Volvemos a obtener los mejores resultados para los modelos que predicen la apnea central, llegando a alcanzar un índice kappa de 0.873 en validación. El mejor índice kappa para la apnea obstructiva también lo encontramos al aplicar *data augmentation* a todas las clases, alcanzando un valor de 0.569. Por último, las predicciones de hipopnea son las que nos proporcionan unos peores resultados, alcanzando un índice de kappa máximo de 0.338.

	Entrenamiento			Validación		
	OSA	Central	Hipopnea	OSA	Central	Hipopnea
Base	0.577	0.835	0.614	0.539	0.798	0.324
Ridge	0.478	0.514	0.376	0.550	0.783	0.328
D. augmentation todas las clases	0.626	0.818	0.488	0.569	0.873	0.338

Tabla 6.43: Resultados índice kappa CNN 2 dimensiones según tipo de apnea

6.2.6. Resultados sobre test

En esta sección vamos a analizar en profundidad los modelos con los que se han obtenido los índices kappa en validación más altos. Se estudiarán las mismas métricas que ya se han mencionado para las redes convolucionales de una dimensión. El método para elegir los modelos que se estudiarán es el mismo que se ha seguido al elegir los mejores modelos con capas convolucionales de una dimensión. Estos modelos son:

1. **Modelo 1**: modelo con 8 capas, un *dropout* de 0.1, 64 canales y un tamaño de *kernel* de 16.
2. **Modelo 2**: modelo con 8 capas, un *dropout* de 0.3, 128 canales, un tamaño de *kernel* de 16 y regresión Ridge.
3. **Modelo 3**: modelo con 8 capas, un *dropout* de 0.3, 128 canales, un tamaño de *kernel* de 16 y *data augmentation* en todas las clases.

En la tabla 6.44 se muestran los resultados obtenidos para estos tres modelos.

Al igual que en los modelos con capas convolucionales de una dimensión, la sensibilidad de los tres modelos con los individuos que tienen menos de 5 eventos por hora, es muy baja, es decir, muchos individuos se clasifican como si tuvieran más de 5 eventos por hora cuando realmente tienen menos. El motivo es la desequilibrada distribución de clases, si nos fijamos, de estas tres sensibilidades tan bajas, la mejor es la del modelo en la que se aplica *data augmentation* a todas las clases. Con un mayor número de individuos que tienen menos de 5 eventos por hora, los resultados serían mejores. La sensibilidad de los individuos para 15 eventos por hora y 30 eventos por hora va aumentando a medida que aumenta el número de eventos, alcanzando un valor aproximado en los tres modelos de 0.86.

La especifidad en el grupo de individuos con menos de 5 eventos por hora es de 0.99, es decir, la probabilidad de que individuos con más de 5 eventos de apnea sean clasificados como tal, es

muy alta. Para los individuos con un umbral de 15 eventos por hora y 30 eventos por hora, la especifidad va disminuyendo, alcanzando valores de 0.65.

	Modelo 1			Modelo 2			Modelo 3		
	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h
Sensibilidad	0.084	0.553	0.858	0.152	0.615	0.866	0.180	0.616	0.863
Especifidad	0.995	0.893	0.651	0.990	0.879	0.657	0.990	0.887	0.658
VPP	0.467	0.747	0.869	0.447	0.744	0.873	0.489	0.757	0.872
VPN	0.954	0.777	0.628	0.957	0.799	0.644	0.958	0.801	0.638
Kappa entrenam.	0.626			0.484			0.562		
Kappa val.	0.426			0.433			0.451		
Kappa test	0.336			0.366			0.371		
Accuracy test	0.550			0.568			0.572		

Tabla 6.44: Resultados sobre test, índice kappa sobre entrenamiento y validación

El valor predictivo positivo nos indica para los individuos que son clasificados en un determinado grupo, cual es la probabilidad de que realmente pertenezcan a este grupo. Para el caso de 5 eventos por hora, esta mayor probabilidad se encuentra en el tercer modelo, con un valor de 0.489. Para el caso de 15 eventos por hora, el mejor resultado se encuentra también en el tercer modelo, con un valor de 0.757. Para el caso de 30 eventos por hora, el modelo que da un mejor resultado es el segundo, con un VPP de 0.873.

El valor predictivo negativo nos indica para los individuos que se predicen por encima de un umbral de eventos por hora, cual es la probabilidad de que realmente estén por encima de este umbral. En el caso de 5 eventos por hora y 15 eventos por hora, los mejores resultados se encuentra en el modelo 3 en ambos casos, con un valor de 0.958 y 0.801 respectivamente. En el caso de 30 eventos por hora, el mejor resultados es el del modelo 2, con un valor de 0.644.

En la tabla 6.44 también se muestran los resultados del índice kappa para los conjuntos de entrenamiento, validación y test. El mejor resultado obtenido sobre test es el del modelo 3, con un valor de 0.371, que se encuentra por debajo del mejor resultado sobre test que se obtuvo con un modelo de capas convolucionales de una dimensión, que tenía un valor de 0.406. El mejor índice kappa en validación de estos modelos con capas convolucionales de dos dimensiones y la mejor tasa de acierto también se obtiene para este tercer modelo. En la figura 6.5 podemos ver la evolución del índice kappa para los tres modelos tanto en entrenamiento como en validación. Para los modelos 1 y 3 vemos como la evolución del índice kappa en entrenamiento es creciente, pero

no en validación, el índice kappa a partir de una determinada época, deja de crecer, por lo que estamos ante un claro caso de sobreajuste. En el modelo 2 podemos observar que la evolución del índice kappa es bastante inestable a lo largo de todas las épocas.

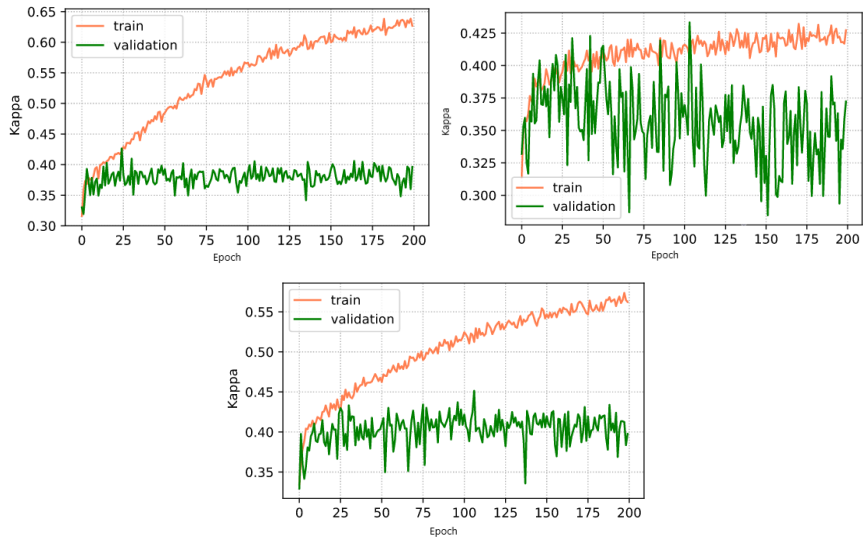

Figura 6.5: Evolución del índice kappa para los modelos 1 (izquierda), 2 (derecha) y 3 (abajo)

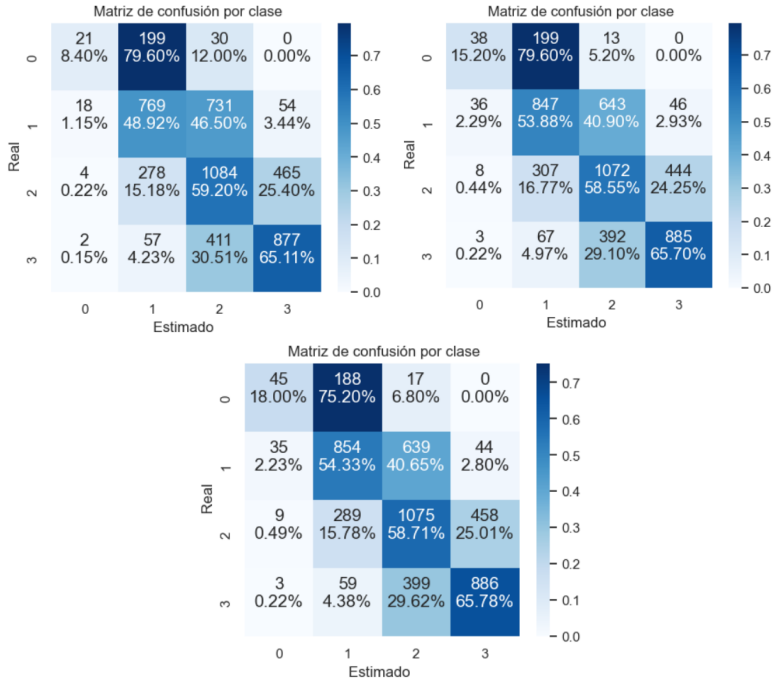

Figura 6.6: Matrices de confusión de modelos 1 (izquierda), 2 (derecha) y 3 (abajo)

En la figura 6.6 vemos las matrices de confusión de los tres modelos: 1 (izquierda), 2 (derecha) y 3 (abajo). Estas matrices de confusión nos muestran lo que ya hemos visto antes, a medida que aumenta el grado de severidad de la apnea, las predicciones son mejores. Para los individuos sin apnea, en los tres modelos, resultan más difíciles de clasificar las nuevas instancias. Vemos que muchos de los individuos que realmente no tienen apnea, se predicen como si tuvieran apnea leve. En el resto de clases, no ocurre lo mismo. Las clases con un grado de severidad más alto resultan más sencillas de clasificar. Cuanto mayor sea el grado de severidad, es más sencillo obtener resultados correctos.

A continuación vamos a analiza el modelo con el mejor índice kappa para cada tipo de evento de apnea. Se van a utilizar las mismas métricas que para los tres modelos anteriores. Los modelos que vamos a estudiar son:

1. **OSA**: modelo con 6 capas, un *dropout* de 0.2, 128 canales, un tamaño de *kernel* de 32 y *data augmentation* en todas las clases.
2. **Apnea central**: modelo con 6 capas, un *dropout* de 0.2, 128 canales, un tamaño de *kernel* de 32 y *data augmentation* en todas las clases.
3. **Hipopnea**: modelo con 8 capas, un *dropout* de 0.1, 64 canales, un tamaño de *kernel* de 16 y regresión Ridge.

En la tabla 6.45 se muestran los resultados obtenidos para estos tres modelos.

	OSA			Central			Hipopnea		
	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h
Sensibilidad	0.903	0.979	0.995	0.996	1	-	0.090	0.554	0.898
Especifidad	0.703	0.608	0.615	0.531	0.227	-	0.994	0.866	0.483
VPP	0.938	0.985	0.997	0.991	0.997	-	0.464	0.757	0.881
VPN	0.595	0.516	0.511	0.729	1	-	0.947	0.720	0.528
Kappa entrenam.	0.626			0.818			0.440		
Kappa val.	0.569			0.873			0.339		
Kappa test	0.477			0.536			0.275		
Accuracy test	0.837			0.985			0.522		

Tabla 6.45: Resultados sobre test, índice kappa sobre entrenamiento y validación según el tipo de apnea

Al igual que en el caso del estudio de los modelos con capas convolucionales de una dimensión sobre test, no se presentan los resultados para un umbral de 30 eventos por hora para la apnea central debido a que ningún individuos presenta tantos eventos apneicos.

La sensibilidad para la apnea central y obstructiva en cualquiera de los tres umbrales, igual que en el caso de los resultados sobre test, es muy alta. Lo mismo ocurre para la sensibilidad de los modelos con capas convolucionales de una dimensión. Para el caso de la sensibilidad en el estudio de la hipopnea, también se obtienen unos resultados similares. La especifidad más alta se encuentra en el modelo que intenta predecir la hipopnea, con un valor de 0.994, sin embargo, para el modelo de estudio de la apnea obstructiva, se ve que el valor de la especifidad para los tres umbrales de apnea, es el más parecido.

El valor predictivo positivo es muy alto para todos los umbrales en la apnea obstructiva y central. En el caso de la hipopnea, el VPP crece a medida que crece el umbral del número de eventos. Vemos que esto mismo ocurría en los modelos con una capa convolucional.

El valor predictivo negativo da peores resultados peores para el caso de la apnea obstructiva. En el caso de un umbral de 15 eventos por hora para predecir la apnea central y un umbral menor de 5 eventos para la hipopnea, el VPN es muy alto.

A continuación, en esta misma tabla 6.45 se muestran los índices kappa medidos en entrenamiento, validación y test. En el modelo para la apnea obstructiva, sobre test se obtiene un índice kappa de 0.477. Para el modelo de la apnea central, se obtiene un índice kappa en test de 0.536, que contrasta mucho con el índice kappa en validación, pero sobre todo con la tasa de acierto (*accuracy*) . En los modelos de capas convolucionales con una dimensión, ocurre exactamente lo mismo. Los motivos por los que esto ocurre, quedaron expuestos para los modelos con capas convolucionales de una dimensión. El índice kappa sobre test del modelo para medir la hipopnea es muy bajo, tan solo 0.275. Comprobamos de nuevo que los peores resultados se obtienen para predecir la hipopnea.

En la figura 6.7 podemos ver la evolución del índice kappa a lo largo de las distintas épocas de entrenamiento (color naranja) y en validación (color verde) para los tres modelos de apnea obstructiva (izquierda), central (derecha) e hipopnea (abajo). Podemos ver que para el modelo para predecir la hipopnea, se produce un fuerte sobreajuste. Para el modelo de apnea obstructiva, también se aprecia sobreajuste, pero mucho menos marcado. En el caso de la apnea central, parece que no hay tanto sobreajuste debido a que los valores del índice kappa en entrenamiento y validación son bastante similares, aunque se ve que para el caso de validación, el modelo es un poco inestable, pero bastante menos de lo que hemos podido observar en el modelo para apnea central de capas convolucionales de una dimensión.

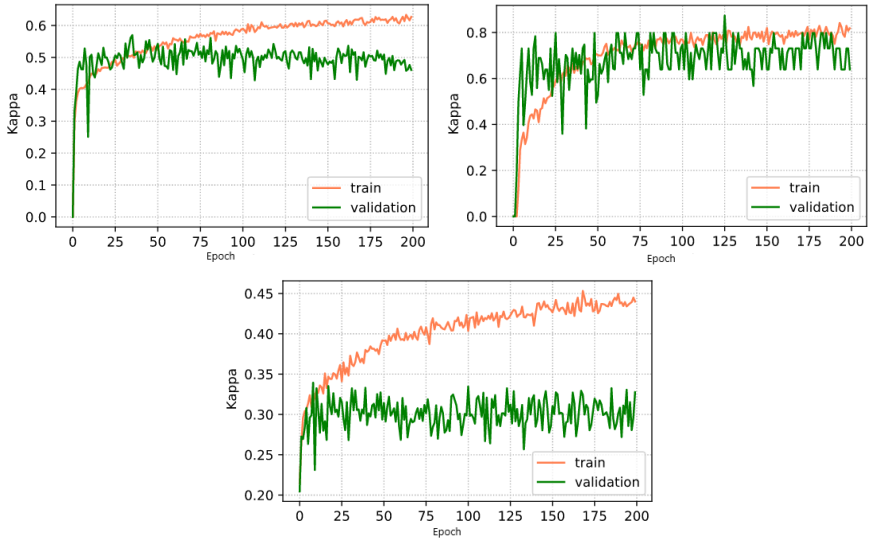

Figura 6.7: Evolución del índice kappa para los modelos de OSA (izquierda), apnea central (derecha) e hipopnea (abajo)

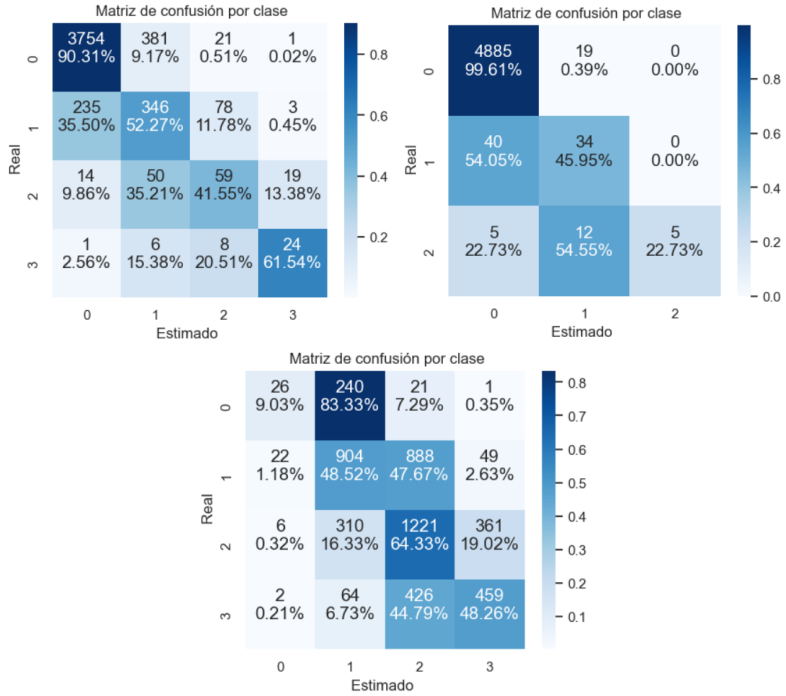

Figura 6.8: Matrices de confusión de modelos de OSA (izquierda), apnea central (derecha) e hipopnea (abajo)

En la figura 6.8 se muestran las matices de confusión para los tres modelos de apnea obstructiva (izquierda), central (derecha) e hipopnea (abajo). Están construidas de la misma forma que todas las matrices de confusión que se han presentado anteriormente. Como se comentó antes, no contamos con individuos con más de 30 eventos por hora de apnea central, por lo que la matriz de confusión para este modelo es de tan solo tres clases.

En la matriz de confusión de la apnea obstructiva vemos que la diagonal está marcada con colores más fuertes, lo que nos indica que para cada grado de severidad de la apnea, la clase más veces predicha es la correcta. En el caso de la apnea central, vemos que la mayoría de los individuos no padecen apnea y que se clasifican correctamente. Este es el motivo por el que la tasa de acierto es tan alta para este modelo. Sin embargo, vemos que para los otros dos grados de severidad de la apnea, hay menos proporción de aciertos por clase, por lo que el índice kappa se verá afectado, empeorando. En el caso de la hipopnea, ya vimos que su índice kappa en test era bastante bajo. Al ver la matriz de confusión vemos que se equivoca mucho más que los otros clasificadores. Ese es el motivo por el cual obtuvo peor índice kappa en test.

6.3. *Transfer learning*

En esta sección vamos a estudiar los resultados del ajuste de los modelos ResNet 18 y EfficientNet b0, con las imágenes generadas a partir de los fragmentos de 20 minutos de sueño de los pacientes. Ambos modelos están entrenados con tan solo 40 épocas debido al largo tiempo que requiere, en el primer caso el entrenamiento del modelo duró 26 horas y en el segundo, 51 horas. En la tabla 6.46 podemos ver los resultados del ajuste de estos modelos, tanto en entrenamiento como en validación.

	ResNet18			EfficientNet b0		
	5 e/h	15 e/h	30 e/h	5 e/h	15 e/h	30 e/h
Sensibilidad	0	0.240	0.935	0	0.221	0.891
Especifidad	0.994	0.968	0.470	0.997	0.966	0.569
VPP	0	0.698	0.696	0	0.667	0.728
VPN	0.983	0.804	0.847	0.983	0.780	0.801
Kappa entrenamiento	0.785			0.513		
Kappa validación	0.298			0.328		
Accuracy validación	0.540			0.564		

Tabla 6.46: Resultados para los modelos de *transfer learning*

La sensibilidad y el valor predictivo positivo de ambos modelos para el caso de 5 eventos por hora o menos es 0, lo que significa que estos modelos, para los pacientes que no tienen apnea, en ningún caso los clasifican como tal. A medida que aumentamos el número de eventos de apnea por hora, la sensibilidad y el valor predictivo positivo mejoran. Para 30 eventos por hora, se obtiene la mejor sensibilidad en el modelo ResNet18 y el mejor valor predictivo positivo en el modelo EfficientNet b0.

Con la especifidad y el valor predictivo negativo ocurre lo contrario que en el caso de la sensibilidad y el valor predictivo positivo, ahora se obtienen resultados muy altos para 5 eventos por hora, pero peores para el caso de 30 eventos por hora.

Si nos fijamos en el índice kappa de los dos modelos, vemos que, en entrenamiento, el índice kappa es mucho más grande para el modelo ResNet18, sin embargo, si nos fijamos en el índice kappa en validación, es mejor en el modelo EfficientNet b0, que en el modelo ResNet18. De esta forma vemos que con el modelo EfficientNet b0 se consigue un menos sobreajuste que con el modelo Resnet18.

En ambos casos la tasa de acierto en validación es muy baja, tan solo del 54 % y del 56 % (*accuracy*), por lo que, estas métricas unidas a las del índice kappa en validación, nos dicen que estos dos modelos no están obteniendo un buen resultado.

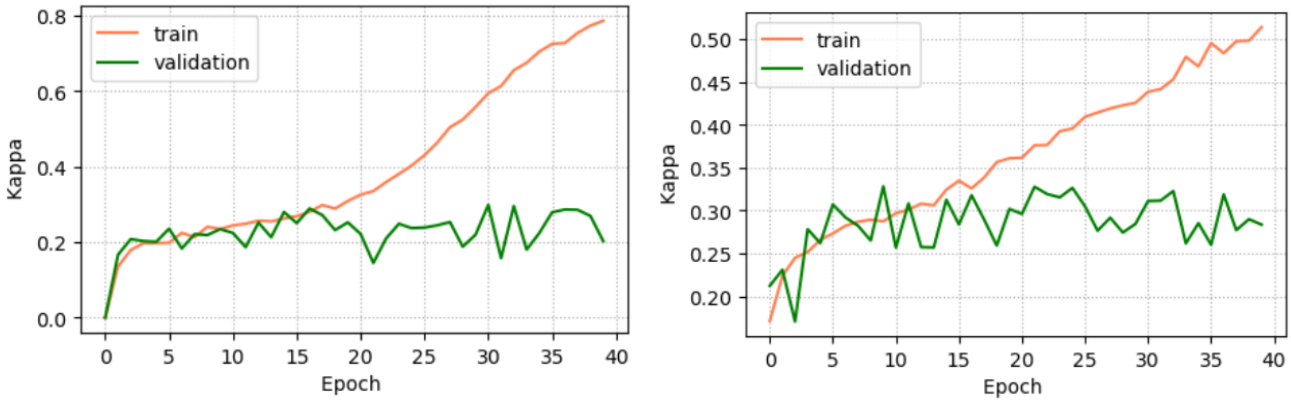

Figura 6.9: Evolución del índice kappa en entrenamiento y validación para los modelos ResNet18 (izquierda) y EfficientNet b0 (derecha)

En la figura 6.9 podemos ver la gráfica de evolución del índice kappa para los dos modelos: Resnet18 a la izquierda y EfficientNet b0 a la derecha. En ambos casos vemos que la evolución del índice kappa en entrenamiento es ascendente a lo largo de las épocas, pero no en validación, en este caso se mantienen valores muy similares en todas las épocas, lo que nos está mostrando la existencia de sobreajuste. Este sobreajuste es más marcado en el caso del modelo ResNet18

debido a un mayor crecimiento del índice kappa en entrenamiento y un menor índice kappa en validación.

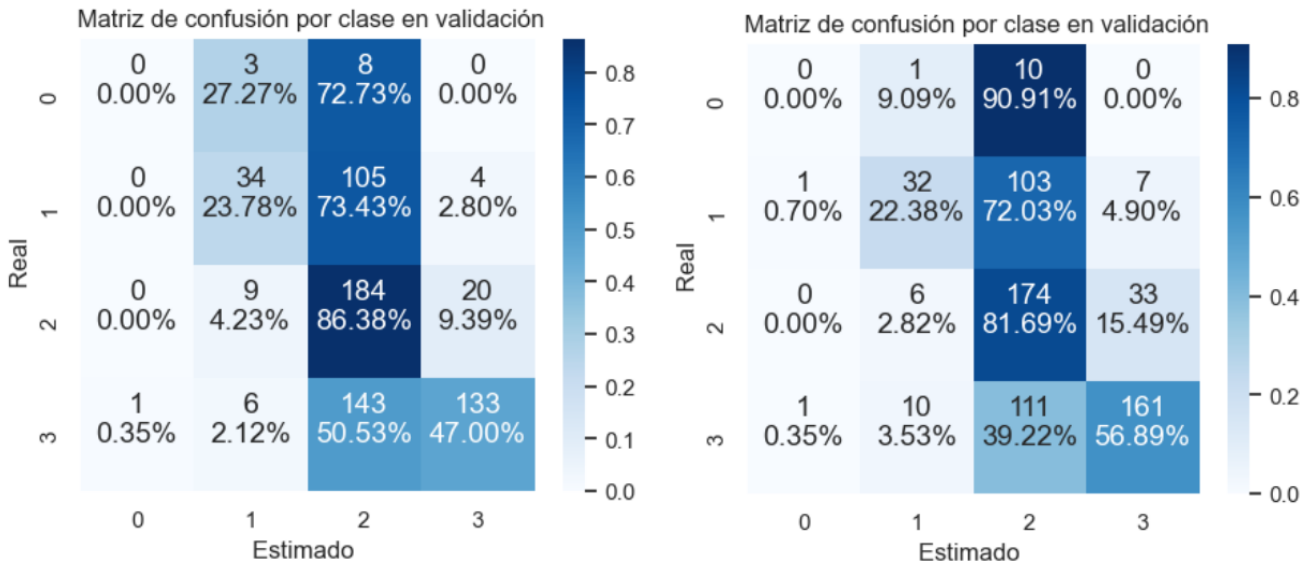

Figura 6.10: Matrices de confusión en validación para los modelos ResNet18 (izquierda) y EfficientNet b0 (derecha)

En la figura 6.10 podemos ver las matrices de confusión para los dos modelos: a la izquierda ResNet18 y a la derecha EfficientNet b0. En ambos casos vemos que las redes tienden a clasificar la mayoría de instancias en la tercera clase, es decir, como apnea moderada. Se aprecia en ambos casos que las predicciones para los individuos sin apnea o apnea leve, son bastante equivocadas, sin embargo, en el caso de apnea moderada y grave, sobre todo en el caso del modelo EfficientNet b0, mejoran mucho, por lo tanto, con un mayor grado de severidad, es más sencillo obtener mejores predicciones. Esto es debido a la desigualdad en la distribución de las distintas clases en las que se agrupan los individuos.

Capítulo 7

Conclusiones y líneas futuras

7.1. Conclusiones

En este trabajo se han estudiado diferentes modelos de redes neuronales aplicados a un problema de regresión con el objetivo de predecir el número de eventos apneicos en fragmentos de 20 minutos de sueño. Los datos que se han utilizado para este proyecto pertenecen a la base de datos de SHHS (Sleep Heart Health Study), en concreto, las señales obtenidas mediante las bandas torácicas y abdominales colocadas en el paciente durante una noche de sueño que monitorizan sus movimientos respiratorios.

Los dos enfoques de redes convolucionales con una y dos dimensiones nos han proporcionado resultados similares, aunque finalmente, el modelo que ha obtenido un resultado diagnóstico más alto sobre test, es un modelo con capas convolucionales de una dimensión, con un índice kappa en test de 0.406 y 0.439 en validación y una sensibilidad y especificidad de 0.164-0.991, 0.664-0.871, y 0.894-0.639, respectivamente, en cada uno de los 3 umbrales de IAH que determinan la severidad de la enfermedad (5 eventos/hora, 15 e/h, y 30 e/h).

El estudio de los eventos de apnea por separado nos ha permitido obtener un modelo con un buen rendimiento para la predicción de eventos de apnea obstructiva, con un índice kappa de cuatro clases sobre test de 0.503. También se ha visto que la predicción de eventos de apnea central genera resultados algo inestables pero también bastante elevados, con un índice kappa en test de 0.536 y 0.873 en validación. La predicción de eventos de hipopnea nos proporciona unos rendimientos inferiores que el resto de modelos estudiados.

Por último, el uso de modelos de *transfer learning* nos ha permitido realizar un estudio de las señales de apnea completamente diferente a los modelos anteriores. A partir de las señales transformadas a imágenes, se han entrenado dos modelos bien conocidos: ResNet18 y EffifientNet

b0, aunque se han obtenido resultados inferiores que en el resto de modelos entrenados con señales.

En resumen, los resultados muestran que la aplicación de modelos *deep learning* sobre las señales respiratorias torácicas y abdominales podrían ayudar a la simplificación del diagnóstico de la apnea del sueño, especialmente en lo referente a la apnea central.

7.1.1. Limitaciones

Todos los individuos que participaron en el estudio de SHHS eran mayores de 40 años, por lo que nuestros resultados deberían evaluarse en otros rangos de edad. Particularmente interesante sería el análisis en los niños, en los que la apnea del sueño, también es una enfermedad con una alta prevalencia, en el caso de la apnea obstructiva, entre un 1 % y un 5 % [46].

Además, como se ha ido comentando a lo largo del trabajo, los grupos de severidad de la apnea del sueño están desbalanceados, ya que muchos individuos padecen apnea severa pero muy pocos no padecen la enfermedad. Esto hace que los entrenamientos de los modelos, y por tanto, también las predicciones, puedan estar influenciadas por esta circunstancia, quedando sesgadas hacia las clases con más eventos de apnea.

7.1.2. Líneas futuras

Como se ha comentado en el apartado de limitaciones, los resultados obtenidos en este proyecto no son extrapolables a niños, quedando por estudiar los modelos desarrollados en un conjunto de datos elaborado a partir de pruebas diagnosticas con sujetos pediátricos. En relación con la base de datos, queda pendiente utilizar otras técnicas de *data augmentation*, para intentar balancear el número de instancias de cada clase.

Respecto a los modelos de redes neuronales, se pueden explorar nuevas arquitectura en busca de mayores rendimientos diagnósticos, así como modelos más explicables, tanto para la predicción de todos los eventos apneicos de forma conjunta, como por separado, en especial, para los eventos de apnea central, en los que se han obtenido resultados prometedores pero aún algo inestables. Es de especial importancia este tipo de eventos debido a que revelan información de la relación entre el corazón y el cerebro.

Por último, queda por explorar más en profundidad la vía del uso de los espectrogramas como datos para el entrenamiento de modelos en vez de la propia señal, especialmente, pasando por una reducción de la dimensión de las imágenes, permitiendo entrenamientos más rápidos y modelos más complejos.

Siglas

ACS *Apnea Central del Sueño*. VII, 6, 7, 8

AOS *Apnea Obstructiva del Sueño*. VII, 3, 4, 6, 7, 8, 12, 13

ECG *Electrocardiograma*. 8, 11, 14

EEG *Electroencefalograma*. 8, 9, 10, 11, 14

EMG *Electromiograma*. VIII, 9, 10, 11, 14

EOG *Electrooculograma*. 8, 11, 14

FC fully-connected. 43

HSAT Home Sleep Apnea Testing. VIII, 12

IAH *Índice de apnea-hipopnea*. III, IV, 4, 76, 89, 117

PAP *Estudio de la presión positiva en las vías respiratorias (Positive Airway Pressure)*. VIII, 11

PSG *Polisomnografía*. III, V, VIII, 1, 3, 4, 8, 9, 10, 12

RIP *Cinturón de inductancia respiratoria (respiratory inductance plethysmography)*. 10

Glosario

hipoxemia Disminución anormal de la presión del oxígeno en la sangre arterial. 3

hipercapnia Aumento de la presión parcial de dióxido de carbono en sangre. 3

úvula Porción triangular de tejido que cuelga del paladar blando. 5

escala de somnolencia de Epworth Es una medida subjetiva de la somnolencia. Se calcula evaluando las respuestas del paciente ante ciertas preguntas acerca de la probabilidad de quedarse dormido en ciertas situaciones. 6

electroencefalograma Prueba diagnóstica que detecta la actividad eléctrica del cerebro mediante electrodos fijados en el cuero cabelludo. Permite diagnosticar trastornos cerebrales, principalmente epilepsia u otros trastornos convulsivos. 8, 9

electrooculograma Prueba diagnóstica que consiste en colocar unos pequeños electrodos en las zonas cercanas a los músculos de los ojos con el objetivo de estudiar el movimiento de los músculos de los ojos. 8, 9

electrocardiograma Prueba diagnóstica que registra las señales eléctricas del corazón. Se utiliza para detectar rápidamente problemas cardíacos. 8

hipoxia Estado de deficiencia de oxígeno en sangre, células y tejidos del organismo, que provoca alteraciones en su funcionamiento. 8

narcolepsia Trastorno crónico del sueño caracterizado por una somnolencia extrema durante el día y ataques repentinos de sueño. Provoca dificultades para mantenerse despierto durante períodos largos de tiempo, sin importar las circunstancias. 8

electromiograma Prueba diagnóstica que se realiza para ver si existe algún daño en la médula espinal o en los nervios periféricos de los músculos. 9

Bibliografía

[1] Berry, R., Budhiraja, R., Gottlieb, D., Gozal, D., Iber, C., & Kapur, V. et al. (2012). Rules for Scoring Respiratory Events in Sleep: Update of the 2007 AASM Manual for the Scoring of Sleep and Associated Events. Journal Of Clinical Sleep Medicine, 08(05), 597-619. https://doi.org/10.5664/jcsm.2172

[2] Apnea del sueño. Último acceso: 20 de Febrero de 2022 de https://www.mayoclinic.org/es-es/diseases-conditions/sleep-apnea/symptoms-causes/syc-20377631

[3] Muñoz-Lombo, J., Garrido-Valencia, G., & Pacheco, R. (2022). Prequency and Factors Associated with Obstructive Sleep Apnea in Adults. Último acceso: 21 de febrero de 2022. http://www.ramr.org/articulos/volumen_20_numero_4/articulos_originales/articulos_originales_frequency_and_factors_associated_with_obstructive_sleep_apnea_in_adults._cali_colombia.pdf

[4] Senaratna, C., Perret, J., Lodge, C., Lowe, A., Campbell, B., & Matheson, M. et al. (2017). Prevalence of obstructive sleep apnea in the general population: A systematic review. Sleep Medicine Reviews, 34, 70-81. https://doi.org/10.1016/j.smrv.2016.07.002

[5] Young, T., Evans, L., Finn, L., & Palta, M. (1997). Estimation of the Clinically Diagnosed Proportion of Sleep Apnea Syndrome in Middle-aged Men and Women. Sleep, 20(9), 705-706. https://doi.org/10.1093/sleep/20.9.705

[6] Apnea central del sueño. Último acceso: 25 de Febrero de 2022. https://www.mayoclinic.org/es-es/diseases-conditions/central-sleep-apnea/symptoms-causes/syc-20352109

[7] Eckert, D., Jordan, A., Merchia, P., & Malhotra, A. (2007). Central Sleep Apnea: Pathophysiology and treatment. Chest, 131(2), 595-607. https://doi.org/10.1378/chest.06.2287

[8] Donovan, L. M., & Kapur, V. K. (2016). Prevalence and Characteristics of Central Compared to Obstructive Sleep Apnea: Analyses from the Sleep Heart Health Study Cohort. Sleep, 39(7), 1353–1359. https://doi.org/10.5665/sleep.5962

[9] Khan, M., & Franco, R. (2014). Complex Sleep Apnea Syndrome. Sleep Disorders, 2014, 1-6. https://doi.org/10.1155/2014/798487

[10] Rundo J.V. & Downey R. (2019). Polysomnography. Handbook of Clinical Neurology, Elsevier, 160, 381-392. https://doi.org/10.1016/B978-0-444-64032-1.00025-4

[11] Summer, J. (2022). At-Home Sleep Studies and Tests — Sleep Foundation. Último acceso: 23 de febrero. https://www.sleepfoundation.org/sleep-studies/at-home-sleep-study

[12] Sleep Data - National Sleep Research Resource - NSRR. Último acceso: 15 de febrero de 2022, de https://sleepdata.org/datasets/shhs

[13] S. F. Quan, B. V. Howard, C. Iber, J. P. Kiley, F. J. Nieto, G. T. O'Connor, D. M. Rapoport, S. Redline, J. Robbins, J. M. Samet & P. W. Wahl, The Sleep Heart Health Study: Design, Rationale, and Methods. (1997). Sleep. https://doi.org/10.1093/sleep/20.12.1077

[14] PyTorch. (2022). Último acceso: 20 de febrero de 2022 de https://pytorch.org/

[15] Fisicalandia. (n.d.). Último acceso: 20 de febrero de 2022 de https://fisicalandia.com/matematicas/tensor/.

[16] PyTorch documentation - PyTorch 1.11.0 documentation. (n.d.). Último acceso: 20 de febrero de 2022 de https://pytorch.org/docs/stable/index.html

[17] Matlab. (n.d.). Último acceso: 2 de marzo de 2022 de https://es.mathworks.com/products/matlab.html

[18] Comet - Build better ML models faster. (n.d.). Último acceso: 5 de abril de 2022 de https://www.comet.ml/

[19] Blank, D. (n.d.). Experiment - Comet.ml. Último acceso: 5 de abril de 2022 de https://www.comet.ml/docs/python-sdk/Experiment/

[20] Bengio, Y., Courville, A., & Vincent, P. (2014). Representation Learning: A Review and New Perspectives. IEEE Transactions On Pattern Analysis And Machine Intelligence, 35(8), 1798-1828. https://doi.org/10.1109/tpami.2013.50

[21] Nielsen, M. (2018) Neural Networks and Deep Learning http://neuralnetworksanddeeplearning.com/

[22] McCulloch, W., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin Of Mathematical Biophysics, 5(4), 115-133. https://doi.org/10.1007/bf02478259

[23] Akshay L Chandra. McCulloch-Pitts Neuron — Mankind's First Mathematical Model Of A Biological Neuron. Último acceso: 6 de Marzo de 2022 de: https://towardsdatascience.com/mcculloch-pitts-model-5fdf65ac5dd1

[24] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408. https://doi.org/10.1037/h0042519

[25] Las partes que conforman una neurona. Último acceso: 28 de enero de 2022. https://puzzlefactory.pl/es/rompecabezas/jugar/gente/283421-las-partes-que-conforman-una-neurona

[26] Brownlee, J. (2022). How to Choose an Activation Function for Deep Learning. Último acceso: 30 de enero de 2022. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

[27] Great Learning Team (2020). Activation Functions in Neural Networks Explained. Último acceso: 30 de enero de 2022. https://www.mygreatlearning.com/blog/activation-functions/

[28] Sheehan, D. (2022). Visualising Activation Functions in Neural Networks. Último acceso: 30 de enero de 2022. https://dashee87.github.io/deep%20learning/visualising-activation-functions-in-neural-networks/

[29] Higham, N. (2021). What Is the Softmax Function?. Último acceso: 31 de enero de 2022. https://nhigham.com/2021/01/12/what-is-the-softmax-function/

[30] Singh, S. (2022). Leaky ReLU as an Activation Function in Neural Networks. Último acceso: 31 de enero 2022. https://deeplearninguniversity.com/leaky-relu-as-an-activation-function-in-neural-networks/

[31] Brownlee, J. (2020). How to Choose Loss Functions When Training Deep Learning Neural Networks.Último acceso: 3 de febrero de 2022. https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

[32] Vimukthi Jayalath, K. (2020). Feedforward and Backpropagation Mathematics Behind a Simple Artificial Neural Network. Último acceso: 5 de febrero de 2022. https://medium.com/analytics-vidhya/feedforward-and-backpropagation-mathematics-behind-a-simple-artificial-neural-networ

[33] De Silva, S. (2020). The Maths behind Back Propagation. Último acceso: 7 febrero de 2022. https://towardsdatascience.com/the-maths-behind-back-propagation-cf6714736abf

[34] Doshi, S. (2019). Various Optimization Algorithms For Training Neural Network. Último acceso: 8 de febrero de 2022. https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6

[35] Goodfellow, I., Bengio, Y., & Courville, A. (2017). Deep learning. Cambridge, Mass: The MIT Press.

[36] Saha, S. (2018). A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way. Último acceso: 10 de febrero de 2022. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[37] Powell, V. Image Kernels explained visually. Último acceso: 11 de febrero de 2022. https://setosa.io/ev/image-kernels/

[38] Mallick, S., & Nayak, S. (2018). Number of Parameters and Tensor Sizes in a Convolutional Neural Network (CNN). Último acceso: 11 de febrero de 2022. https://learnopencv.com/number-of-parameters-and-tensor-sizes-in-convolutional-neural-network/

[39] Pykes, K. (2020). The Vanishing/Exploding Gradient Problem in Deep Neural Networks. Último acceso: 15 de febrero de 2022. https://towardsdatascience.com/the-vanishing-exploding-gradient-problem-in-deep-neural-networks-191358470c11

[40] Nikolaiev, D. (2021). Overfitting and Underfitting Principles. Último acceso: 16 de febrero de 2022. https://towardsdatascience.com/overfitting-and-underfitting-principles-ea8964d9c45c

[41] Martinez Heras, J. (2020). Regularización Lasso L1, Ridge L2 y ElasticNet - IArtificial.net. Último acceso: 16 de febrero de 2022. https://www.iartificial.net/regularizacion-lasso-l1-ridge-l2-y-elasticnet/

[42] Vaquerizo-Villar, F., Alvarez, D., Kheirandish-Gozal, L., Gutierrez-Tobal, G., Barroso-Garcia, V., & Santamaria-Vazquez, E. et al. (2021). A Convolutional Neural Network Architecture to Enhance Oximetry Ability to Diagnose Pediatric Obstructive Sleep Apnea. IEEE Journal Of Biomedical And Health Informatics, 25(8), 2906-2916. doi: https://doi.org/10.1109/jbhi.2020.3048901

[43] Hicks, Michael & Foster, Jeffrey. (2010). Adapting Scrum to Managing a Research Group.

[44] He, K., Zhang, X., Ren, S. & Sun, J. Deep Residual Learning for Image Recognition (2016). IEEE Conference on Computer Vision and Pattern Recognition https://arxiv.org/pdf/1512.03385.pdf

[45] Tan, Mingxing and Le & Quoc V. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. (2019). CoRR. http://arxiv.org/abs/1905.11946

[46] Schechter, M. S., Ward, S. D., Sheldon, S. H., Shiffman, R. N., Lehmann, C., Spruyt, K., & American Academy of Pediatrics (2012). Diagnosis and management of childhood obstructive sleep apnea syndrome. Pediatrics, 130(3), e714–e755. https://doi.org/10.1542/peds.2012-1672