Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN COMPUTACIÓN

Aplicación del Aprendizaje Profundo a la clasificación de estados de sueño

Estudante: Antonio Sanjurjo Cortés
Dirección: Elena María Hernández Pereira

A Coruña, febreiro de 2020.

Agradecimientos

Me gustaría agradecer a la tutora de este trabajo, Elena María Hernández Pereira por su ayuda, supervisión y correcciones durante la realización de este trabajo.

Así mismo, también me gustaría agradecer al grupo de investigación LIDIA de la Facultad de Informática por el equipo prestado para realizar el desarrollo.

Resumen

La clasificación automática de los estados del sueño es una tarea compleja basada en el análisis de señales electrofisiológicas. Este trabajo explora la posibilidad de realizar dicha clasificación mediante modelos de Aprendizaje Profundo que extraigan información a partir de imágenes que contienen las formas de onda de las señales, imitando la forma en la que los humanos realizan la clasificación. Los resultados obtenidos son alentadores ya que se consiguen valores de rendimiento próximos a los de trabajos similares.

Abstract

The automatic classification of sleep stages is a complex task based on the analysis of electrophysiological signals. This work explores the possibility to use Deep Learning models that extract information from images containing the waveforms of the signals, imitating the way humans perform the classification. The results obtained are encouraging as the performance values are close to those of similar works.

Palabras clave:

Inteligencia Artificial

Aprendizaje Profundo

Clasificación

Etapas del sueño

Keywords:

Artificial Intelligence

Deep Learning

Classification

Sleep stages

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

El sueño es una de las mayores necesidades de los humanos, siendo una función vital y fisiológicamente necesaria para la vida.

Los problemas relacionados con el sueño tienen consecuencias directas en la calidad de vida de las personas, afectando a áreas como la memoria, la capacidad de concentración y su estabilidad emocional. Es por eso que existe todo un campo de la medicina centrado en su estudio, con el objetivo de comprender el funcionamiento de estos problemas y sus efectos.

Una de las tareas que se incluyen dentro de la medicina del sueño es la monitorización y clasificación de las etapas de sueño que atraviesa el sueño de las personas, un proceso conocido formalmente como polisomnografía. El análisis de este proceso permite identificar desórdenes de sueño que puedan perjudicar la vida de las personas.

El análisis del sueño requiere de personal especializado y de una gran cantidad de tiempo. La informática, y más en concreto, el campo de la Inteligencia Artificial, pueden aportar una mayor agilidad a estos procesos.

Las dificultades asociadas a este dominio han hecho que no exista una solución de análisis automatizado ampliamente aceptada. Por ese motivo, este trabajo explora el uso de una técnica basada en Aprendizaje Profundo para intentar resolver el problema de la clasificación de etapas de sueño.

1.1 Objetivos

El objetivo principal de este trabajo es diseñar modelos de Aprendizaje Profundo que realicen la clasificación de etapas de sueño, usando como entrada para los modelos imágenes de señales extraídas de registros polisomnográficos. Estas señales son, principalmente, el electroencefalograma (EEG) y el electrooculograma (EOG). Este objetivo se descompone en los siguientes subobjetivos:

1. Analizar y procesar las señales de entrada.

2. Generar las imágenes de entrada que usarán los modelos desarrollados.

3. Definir modelos de Aprendizaje Profundo con diferentes señales de entrada y seleccionar los mejores modelos a través de la optimización de sus parámetros.

4. Analizar los resultados para valorar la idoneidad y limitaciones de la aproximación desarrollada.

5. Comparar los resultados obtenidos con otros trabajos existentes en la literatura.

1.2 Organización de la memoria

La memoria de este trabajo Fin de Grado se organiza tal y como sigue.

El capítulo 1, en el que se encuentra este apartado, introduce la temática a tratar en este trabajo junto con los objetivos perseguidos.

El capítulo 2 introduce los conceptos de la medicina del sueño necesarios para tener una perspectiva del problema a resolver.

El capítulo 3 realiza una introducción a los conceptos teóricos que se van a usar en el trabajo.

El capítulo 4 realiza un análisis de algunas de las aproximaciones existentes hasta la fecha para resolver el problema de la clasificación de las etapas del sueño.

El capítulo 5 introduce el material usado para llevar a cabo el desarrollo y experimentación, así como los motivos de su elección.

El capítulo 6 muestra la metodología seguida para realizar el trabajo.

El capítulo 7 muestra el proceso de desarrollo con los resultados específicos de cada parte.

El capítulo 8 realiza un análisis de los resultados en comparación con otros trabajos existentes en el campo.

El capítulo 9 muestra las conclusiones finales del trabajo, así como el posible trabajo futuro.

Capítulo 2

Descripción del dominio

El término *sueño* está asociado tanto al acto de dormir como a las ganas de hacerlo. Cabe destacar que el acto de dormir no se produce de igual manera en todas las especies. Existen especies en las cuales lo habitual resulta ser una reducción de su actividad cerebral y física, pero sin llegar a alcanzar un estado de inconsciencia. En el caso de los humanos, el sueño implica una baja o nula actividad física, acompañada de un estado de inconsciencia que permite evadirnos del entorno.

Entre las acciones más importantes del sueño se encuentra la función restauradora, que permite tanto al cuerpo como al cerebro recuperar la energía necesaria para poder seguir despierto. La calidad del sueño es determinante para poder mantener una vida sana.

Existen múltiples estudios que analizan el efecto que produce la privación del sueño en las personas; un ejemplo puede ser el trabajo de Stickgold [1], en el cual se analiza la importancia que tiene en la consolidación de la memoria. La mala calidad del sueño también afecta a las funciones endocrinas del cuerpo, dificultando por ejemplo la correcta regulación de la glucosa [2].

2.1 Estudios del sueño

Los estudios del sueño son el conjunto de pruebas formales realizadas con el objetivo de comprender el funcionamiento del mismo o de determinar las posibles afecciones de un paciente, como pueden ser los distintos tipos de disomnias.

Debido a la necesidad de realizar una monitorización precisa del paciente, estas pruebas suelen realizarse en entornos médicos controlados, los conocidos como laboratorios del sueño. Estas pruebas intentan realizarse dentro del horario habitual de descanso del paciente, de manera que se pueda observar el comportamiento en un entorno realista, que permita extraer la información de la manera más fiel posible. El hecho de tener que realizar las pruebas en un entorno distinto al habitual supone de por si un problema para el paciente, pudiendo afectar

a su conciliación del sueño. Es por eso que estas pruebas pueden tener que repetirse o trasladarse al lugar de residencia del afectado, con el objetivo de obtener mediciones válidas en decremento de la calidad de la monitorización. Otro factor añadido que complica la conciliación del sueño es la presencia de sensores corporales junto a su correspondiente cableado, necesarios para realizar la adquisición de los datos que posteriormente serán analizados.

Una de las pruebas de estudio más comunes es la polisomnografía. Esta prueba consiste en la realización de un registro de múltiples señales, centrándose principalmente en tres de tipo neurológico: electroencefalograma (EEG), electrooculograma (EOG) y electromiograma (EMG). El polisomnograma suele incluir un mayor número de señales, entre las que se encuentran: mediciones del flujo respiratorio, electrocardiograma, saturación de oxígeno en sangre y temperatura corporal, entre otras (Figura 2.1).

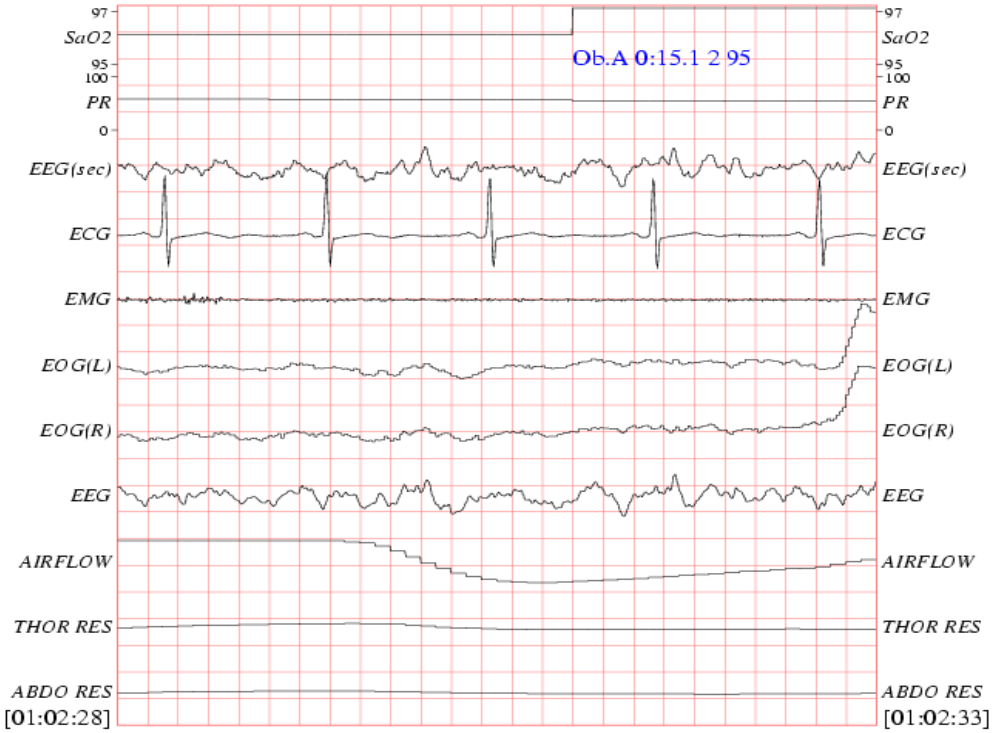


Figura 2.1: Fragmento de un registro polisomnográfico

Esta prueba permite identificar un amplio rango de trastornos del sueño, pero lleva consigo el análisis de un volumen elevado de datos, debido al número de señales y a su duración. El análisis manual de un registro polisomnográfico es por tanto una tarea laboriosa, que requiere de personal médico especializado en el ámbito de la medicina del sueño. Es por eso que las herramientas que permitan una automatización del análisis ayudarán a reducir el coste

económico y temporal del proceso, permitiendo al personal médico centrarse en las regiones de interés.

2.1.1 EEG

El EEG es una señal que mide la actividad eléctrica que se produce en las neuronas. Para realizar estas mediciones es necesaria la colocación de una serie de electrodos extracraneales, capaces de medir la diferencia de potencial eléctrico entre dos puntos, permitiendo así determinar el flujo eléctrico entre ellos. La colocación de los electrodos está estandarizada, existiendo múltiples posiciones que dan lugar a los distintos canales de EEG.

En el EEG existen cuatro tipos principales de ondas cerebrales: [3]:

- **Ondas Delta (δ)**: Presentan frecuencias entre 0.5 y 3.9Hz y amplitudes entre 100 y 200μV.
- **Ondas Theta (θ)**: Presentan frecuencias entre 4 y 7.9Hz y amplitudes entre 50 y 100μV.
- **Ondas Alfa (α)**: Presentan frecuencias entre 8 y 13Hz y amplitudes entre 5 y 50μV. Es una onda que se atenúa con la apertura ocular.
- **Ondas Beta (β)**: Presentan frecuencias entre 14 y 30Hz y amplitudes entre 5 y 50μV. Es la onda que presenta un rango más variado y difícil de concretar.

2.1.2 EOG

El EOG es una señal que tiene como objetivo la medición del movimiento de los ojos, utilizando para ello unos electrodos externos colocados en la zona cercana a estos. Su funcionamiento se basa en la medición de la diferencia de potencial existente entre la córnea y la retina. La duración, signo y magnitud del voltaje medido permiten identificar los distintos tipos de movimientos oculares, diferenciando entre movimientos rápidos y lentos, así como el desplazamiento que abarca el movimiento y su dirección.

2.1.3 EMG

El EMG es una señal que refleja la diferencia de potencial producida por las contracciones de los músculos. Para su registro se colocan en la zona muscular a inspeccionar una serie de electrodos sobre la superficie de la piel. La forma, frecuencia y orden de activación de las unidades motoras permiten caracterizar la información extraída [4]. En el caso de la polisomnografía suelen colocarse electrodos en la barbilla para la obtención del conocido como EMG submental, siendo habitual la colocación de electrodos en ambas piernas del paciente, lo cual permite detectar su movimiento involuntario.

2.2 Estructura del sueño

A medida que avanza el tiempo durante el acto del sueño el cerebro cambia su comportamiento, siguiendo una serie de patrones definidos que permiten distinguir entre una serie de etapas.

En el año 1968, y de mano de Rechtschaffen y Kales (R&K) [5], se establecen los primeros criterios estandarizados para la clasificación de las etapas del sueño. Estos criterios se basan principalmente en el análisis de las tres señales anteriormente descritas: EEG, EMG y EOG. El proceso del sueño es dividido en seis etapas, una de ellas se corresponde a los momentos previos al inicio del sueño, denominada como etapa de vigilia o W. Las demás se corresponden a distintos momentos del sueño propiamente dicho y se denominan como etapa 1, 2, 3, 4 y REM.

La categorización de R&K ha sido considerada de referencia hasta el año 2007, momento en el cual se publica un nuevo estándar por parte de la Academia Americana de Medicina del Sueño (AASM) [6]. Entre sus principales diferencias, destacan las modificaciones realizadas sobre los criterios para determinar las etapas y la unión de las etapas 3 y 4 en un única etapa denominada N3. Su nomenclatura también se modifica, por lo que, a partir de este momento, las etapas se denominan W, N1, N2, N3 y REM.

Tal y como consideran el estándar R&K y la AASM, la clasificación de etapas de sueño se realiza en intervalos de 30 segundos, denominados epochs.

2.2.1 Las etapas del sueño

A continuación se describirán las etapas mencionadas en el manual AASM:

- **Etapa W**: Representa el estado de consciencia o vigilia. La señal de EEG se caracteriza por contener durante al menos un 50% del epoch ondas alfa y en el caso de que los ojos se encuentren abiertos, esta señal será más débil. Durante esta etapa también es visible la onda beta. La señal de EOG mostrará movimientos oculares de velocidad moderada, junto con las variaciones producidas por el parpadeo en caso de tener los ojos abiertos. La frecuencia del EOG suele estar comprendida entre 0.5 y 2 Hz. La señal del EMG submental mostrará valores relativamente altos que irán descendiendo a medida que el cuerpo se relaje y acerque al sueño propiamente dicho.

- **Etapa N1**: Es también conocida como la etapa de sueño ligero y representa la entrada natural en el sueño. Durante esta, el cuerpo sigue siendo capaz de percibir ciertos estímulos exteriores. El EEG se caracteriza por los movimientos rápidos, con ondas theta que deben estar presentes durante más de un 50% del epoch. La presencia de las ondas alfa disminuye y se mezcla con las de las ondas beta. La señal del EOG mostrará

movimientos oculares lentos y la del EMG mostrará valores similares a los producidos durante la última parte de la etapa W. La duración de esta etapa suele ser inferior a los 10 minutos.

- **Etapa N2**: También es conocida como etapa intermedia. La onda más predominante del EEG sigue siendo la onda theta, la onda alfa puede aparecer de forma mínima y la onda delta aparece en mayor cantidad que en la etapa N1, pero sin superar el 20% del epoch. Durante esta etapa aparecen por primera vez dos tipos de forma de onda característicos, los complejos K y los husos del sueño, ambos con una duración mínima de medio segundo y pueden llegar hasta los dos segundos, siendo habitual la presencia de husos de sueño a continuación de los complejos K. En esta etapa el cuerpo entra en un mayor nivel de relajación y evasión de los estímulos, motivo por el cual el EOG será nulo o con movimientos muy ligeros y la señal EMG mostrará un tono muscular bajo.
- **Etapa N3**: También es conocida como sueño profundo. Las ondas que se observan en el EEG son lentas y de gran amplitud. Se caracteriza por contar con la presencia de ondas delta en al menos un 20% del tiempo del epoch. Los complejos K y los husos de sueño también pueden aparecer en esta etapa, aunque con una frecuencia menor que en la etapa N2. La actividad corporal es la menor entre las etapas NREM, lo que conduce a un movimiento ocular nulo en la mayoría de los casos, al igual que ocurrirá con el EEG, que alcanzará los valores más bajos de este grupo. Es durante esta etapa cuando el cuerpo alcanza su mayor grado de recuperación y cuando pueden producirse el sonambulismo o los terrores nocturnos.
- **Etapa REM**: También es conocida como etapa de sueño activo. La señal del EEG se caracteriza por la presencia de ondas theta y alfa con amplitudes bajas. Esta etapa se caracteriza, como su nombre indica, por la presencia de movimientos oculares rápidos. Estos movimientos suelen exhibirse en forma de ráfagas de actividad. Es en estos instantes cuando se alcanzan los niveles de amplitud más bajos en el EMG, permitiendo alcanzar el conocido como estado de atonía muscular. El sueño REM es conocido por ser la etapa en la cual se producen los sueños y las pesadillas, haciendo que exista una mayor actividad cerebral que en las etapas NREM.

2.2.2 Ciclo del sueño

A lo largo de la noche, el cerebro avanza a través de las etapas NREM y REM de manera repetitiva. Esto se conoce como el ciclo del sueño y cada ciclo tiene una duración estimada de entre 90 y 110 minutos en adultos sanos [7], siendo de menor duración en edades más tempranas. La duración del sueño en una persona sana oscila entre las 7 y 9 horas diarias,. Durante este tiempo llegan a producirse 4 o 5 ciclos completos. A medida que avanza el sueño

la duración de la etapa REM aumenta y consecuentemente la de las etapas NREM disminuye. La etapa REM comprende típicamente un 20-25% del tiempo total de sueño.

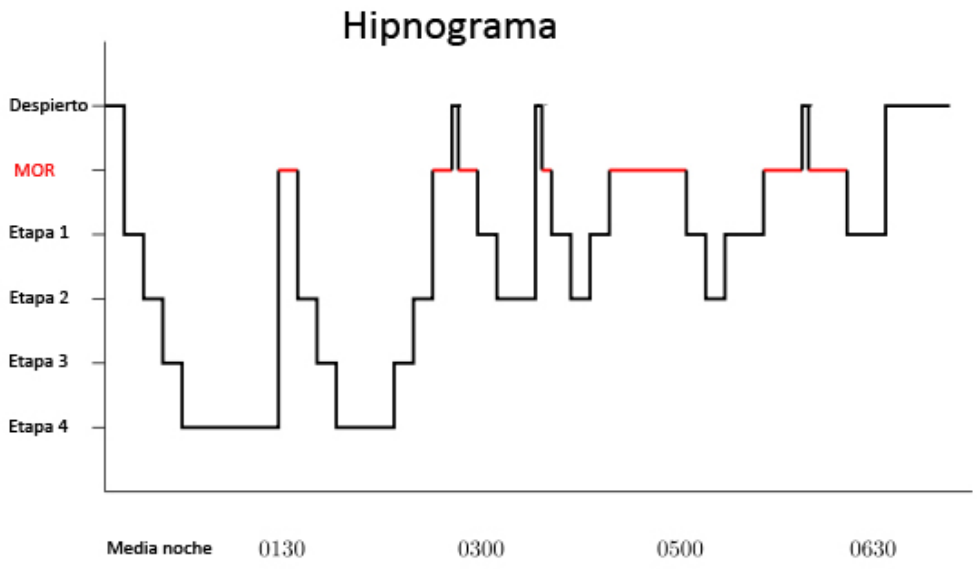


Figura 2.2: Hipnograma de un adulto con anotaciones de etapa

Una de las representaciones más usadas para reflejar las transiciones entre etapas a medida que avanza el tiempo es el hipnograma. Este, tal y como puede observarse en la Figura 2.2, muestra el avance temporal en el eje X y la etapa previamente clasificada en el eje Y. El hipnograma permite identificar de manera sencilla las alteraciones con respecto al ciclo normal del sueño.

Capítulo 3

Introducción teórica

La Inteligencia Artificial (IA) nace en el siglo XX como una rama de investigación dentro del campo de la computación. Su objetivo principal es el estudio y desarrollo de técnicas que permitan imitar la capacidad de los humanos para resolver problemas complejos.

Uno de los campos con mayor recorrido dentro de la IA es el Aprendizaje Automático. Esta rama se centra en en el desarrollo de técnicas que permitan a los ordenadores aprender a partir de los ejemplos a los que se exponen, con el objetivo de generalizar el conocimiento de los patrones. Este aprendizaje a partir de patrones es, por tanto, un proceso de inducción de conocimiento.

El Aprendizaje Automático se divide en: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje por refuerzo. El *aprendizaje supervisado* se caracteriza por disponer de un conjunto de patrones sobre los que se puede definir un criterio de evaluación. Esto es, la existencia de unos valores de entrada asociados a unas salidas deseadas. Su objetivo es por tanto modelizar una función que permita, a partir de las entradas, generar los resultados esperados. Los modelos generados pueden ser usados para realizar clasificación o regresión. Uno de los modelos más populares dentro del aprendizaje supervisado son las redes de neuronas artificiales [8].

3.1 Redes Neuronales Artificiales

Una Red Neuronal Artificial (RNA) es un algoritmo usado en aprendizaje supervisado que define un modelo matemático inspirado en el flujo de información que se produce en las neuronas del cerebro.

Las redes neuronales se componen de múltiples neuronas artificiales, desarrolladas en el año 1943 por McCulloch-Pitts. Uno de los primeros diseños de red neuronal fue el Perceptrón [9]. Esta red fue posteriormente ampliada al Perceptrón multicapa, la cual dispone de una o más capas ocultas situadas entre la capa de entrada y la capa de salida, lo que permite

resolver problemas no linealmente separables.

3.2 Aprendizaje Profundo

El Aprendizaje Profundo (Deep Learning, en inglés) es el conjunto de algoritmos de RNA caracterizados por la presencia de un elevado número de capas ocultas. Estos algoritmos son populares dentro de campos como la Visión Artificial y el Procesamiento del Lenguaje Natural.

Las redes de Aprendizaje Profundo empezaron a desarrollarse a finales de la década de 1970, pero debido a su alto coste computacional, no comenzaron a popularizarse hasta la década de 1990, con el incremento en el uso de las GPU [10].

El coste computacional de este tipo de algoritmos se debe al gran número de neuronas que componen las redes. La profundidad de las redes es superior a la encontrada en los algoritmos tradicionales y el número de neuronas en cada capa también suele ser mayor. Esto último implica tener que realizar más cálculos para obtener la salida de la red a partir de los datos de entrada. De la misma manera, durante la fase de aprendizaje el cálculo del error debe ser obtenido en un mayor número de neuronas, haciendo el proceso más lento. El alto coste computacional no es solo temporal, sino también en necesidades de espacio, ya que se deben almacenar un gran número de pesos en la memoria del equipo.

Otro de los problemas de este tipo de redes se debe a que al disponer de un número elevado de parámetros que deben ser ajustados, el número de patrones de entrenamiento necesarios también debe ser mayor que en las RNA tradicionales. Esto es de especial importancia para evitar el sobreentrenamiento de la red, lo que produce que la optimización se detenga en un mínimo local, condicionado por la variabilidad de los patrones disponibles en el entrenamiento y evitando que el sistema generalice el comportamiento, lo que produce resultados de baja calidad en situaciones nuevas.

A pesar de los problemas de complejidad computacional asociados, el Aprendizaje Profundo ha sido uno de los grandes pilares del incremento en la popularidad de la IA en los últimos años, destacando por sus buenos resultados en múltiples campos con dominios muy variados. Las redes de Aprendizaje Profundo presentan una gran capacidad para modelar conocimiento complejo, estando presentes en tareas tan diversas como la traducción de textos o la conducción autónoma. Otra de sus capacidades más relevantes es la extracción de características automatizada a partir de los patrones disponibles, eliminando la necesidad de diseñar y calcular características complejas de manera manual. Esto reduce los sesgos introducidos por el personal que debe diseñar los algoritmos clásicos de Aprendizaje Automático, donde se tiene que analizar y decidir sobre las características más relevantes para el problema a resolver.

Los tipos de redes más populares son las Redes Neuronales Convolucionales.

3.2.1 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (Convolutional Neural Networks o CNNs, en inglés) son un tipo de RNA inspirado en la conectividad de las neuronas que forman el cortex visual. Este tipo de redes es especialmente común dentro del campo de la Visión Artificial.

A pesar de que el concepto teórico de las CNN se desarrolló en la década de 1980 y 1990, no fue hasta el año 2012 cuando comenzaron a popularizarse. El incremento en popularidad comienza con la publicación de AlexNet [11], una CNN diseñada para realizar clasificación de objetos en imágenes y que en su momento superó los métodos que definían el estado del arte de ese campo (Figura 3.1).

La arquitectura más típica de red CNN es la formada por capas convolucionales seguidas de capas Totalmente Conectadas (Fully Connected o FC, en inglés) o de capas de memoria a largo plazo (Long Short-term Memory o LSTM, en inglés).

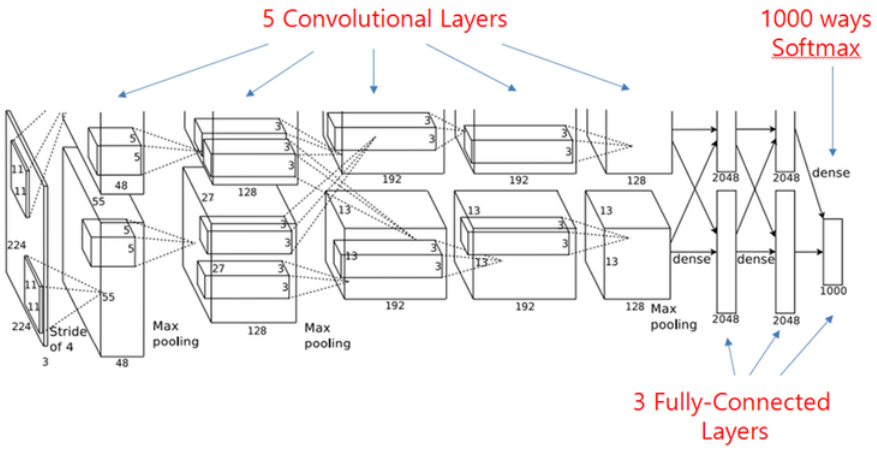


Figura 3.1: Arquitectura de red de AlexNet [11]

Para entender el funcionamiento básico de una red CNN es necesario describir los elementos que la conforman.

Convoluciones

La base de las capas convolucionales es la realización de la operación matemática de convolución, la cual presenta la característica de ser invariante a translaciones. La convolución implementa filtros lineales que pueden definirse como:

$$(f * g)(n) = \sum_{m=-\infty}^{\infty} f(n-m)g(m)$$

siendo f una imagen de tamaño *MxN* y g un filtro o kernel de tamaño *mxn*.

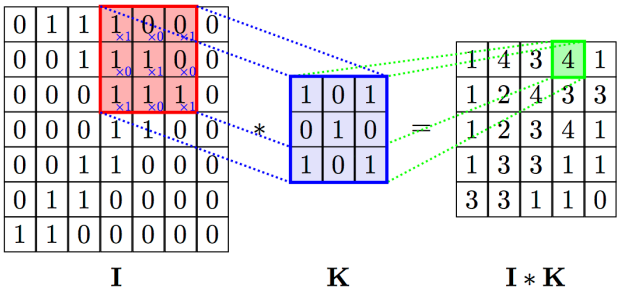


Figura 3.2: Ejemplo de aplicación de convolución 2D

La aplicación de convoluciones mediante filtros genera abstracciones sobre los datos, conocidas como mapas de características. Cada filtro permite detectar una característica del patrón de entrada y la correlación de las activaciones de los diferentes filtros permite identificar el patrón en su totalidad. Puede verse un ejemplo de aplicación de esta operación en la Figura 3.2.

Una vez calculadas las salidas de la aplicación de los filtros, estas son pasadas a la función de activación de la capa convolucional, que determina los valores que serán propagados hacia la siguiente capa de la red.

Los parámetros que definen la convolución mediante filtros son:

- Tamaño de la imagen de entrada.
- Tamaño del filtro convolucional.
- Número de canales de entrada y salida (deben ser los mismos).
- Paso o stride, que constituyen los píxeles de desplazamiento entre cada aplicación del filtro sobre la entrada.
- Relleno o padding, que permite controlar los valores a insertar en las zonas de la imagen de entrada para las cuales la superposición del filtro no sería posible. Las tipos de padding más comunes son: con relleno de ceros y con replicación del valor de la imagen.

Para una imagen NxN y un filtro mxm cuadrado, siendo P el tamaño del relleno (padding) y S el tamaño del paso (stride), la dimensión de la imagen de salida se calculará como:

$$\frac{N - m + 2P}{S} + 1$$

Las capas convolucionales disponen de un parámetro adicional que es el número de filtros a calcular en esa capa de la red. El objetivo de las capas convolucionales es por tanto encontrar de manera automatizada los filtros que extraigan las mejores características para los datos de entrada disponibles.

Pooling

Las capas de pooling tienen como objetivo la simplificación de los datos mediante una reducción en la dimensión de estos. Esta reducción consigue mejorar el coste computacional de la red, al reducir el número de parámetros que deben ser aprendidos, lo que también ayuda a controlar el sobreentrenamiento. Las capas de pooling son muy comunes en las redes CNN y son colocadas a continuación de las capas convolucionales.

Los parámetros de este tipo de capa son:

- Dimensión del elemento de pooling.
- Paso o stride.
- Relleno o padding.

Para una entrada de tamaño MxM, tamaño de relleno P y tamaño de paso S, la dimensión de la imágen de salida será:

$$\frac{M - P}{S} + 1$$

El tipo de pooling más común es el max-pooling, el cual selecciona el elemento de mayor magnitud dentro de la ventana a analizar, tal y como se observa en la Figura 3.3. Otros tipos de pooling comunes son la media o incluso la norma-L2.

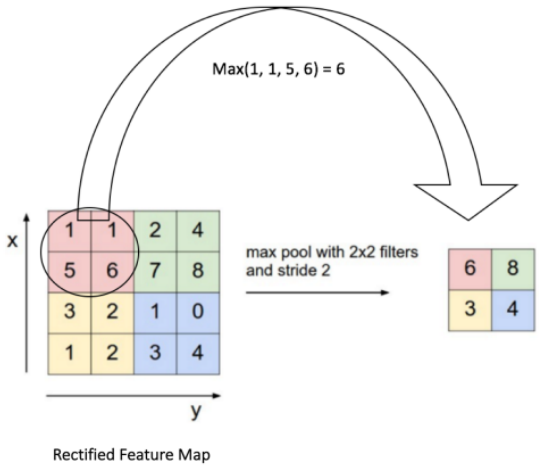


Figura 3.3: Ejemplo de aplicación de max-pool

Batch Normalization

La técnica de Batch Normalization fue introducida en el año 2015 por Sergey Ioffe y Christian Szegedy. Este método se basa en la normalización de las entradas que llegan a la función de activación mediante un ajuste y escalado, que se realiza a partir del cálculo de la media y la desviación estándar del lote (batch, en inglés) de entrada [12].

Su uso implica la pérdida de relevancia en el efecto de la inicialización aleatoria de los pesos de la red y permite trabajar con tasas de aprendizaje mayores. El número de iteraciones necesarias para conseguir la convergencia en el entrenamiento se ve por tanto reducido. La necesidad de reescalar las entradas de manera manual a un rango inferior pierde relevancia, al producirse un escalado interno automatizado. El uso de la técnica de Batch Normalization suele sustituir la necesidad de usar otras técnicas como Dropout dentro de las capas convolucionales.

Dropout

El Dropout es una técnica de regularización que consiste en desactivar neuronas de manera aleatoria con probabilidad $1 - p$ o dejarlas activadas con probabilidad p. El uso de esta técnica consigue, mediante la limitación del número de neuronas, el aprendizaje de características más robustas, lo que permite reducir el sobreentrenamiento. Esto se consigue al reducir la dependencia de la red para predecir valores correctos debidos a la relevancia de unas pocas neuronas.

La reducción del número de neuronas reduce la complejidad computacional de la red, pero la necesidad del ajuste interrelacionado entre un mayor número de neuronas implica una ralentización en el proceso de convergencia.

Sus efectos son mayores en redes con muchos parámetros y con conjuntos de entrenamiento pequeños, ya que son algunas de las situaciones que más favorecen el sobreentrenamiento de las redes.

Flattening

El proceso de Flattening consiste en convertir las salidas de las capas convolucionales, con forma de tensores, a una representación en una dimensión. Este proceso solo modifica la estructura de los datos, no introduce modificaciones en sus valores. El Flattening es necesario para poder trabajar con capas que reciben vectores como entrada, como pueden ser las capas FC y LSTM.

Capas Totalmente Conectadas

Las Capas Totalmente Conectadas (FC) son capas prealimentadas que suelen constituir la parte final de las redes CNN. Estas capas no tienen en cuenta la información espacial extraída por las capas convolucionales, pero son las encargadas de asociar la información representacional de las convoluciones para posteriormente realizar el proceso de clasificación o regresión.

Funciones de activación

Las funciones de activación son funciones matemáticas, utilizadas por las capas convolucionales y FC para introducir la no linealidad en los datos, permitiendo modelizar funciones no triviales.

Las función más populares en las redes tradicionales son la función sigmoide y la tangente hiperbólica (tanh) [13], aunque su uso en las redes de Aprendizaje Profundo es reducido debido a sus problemas de lentitud de convergencia y de desvanecimiento del gradiente.

Dentro de las capas convolucionales, la función de activación más popular es la denominada como Unidad Rectificada Lineal (ReLU, en inglés) [14]. Esta función (Figura 3.4) se ve menos afectada por el problema del desvanecimiento del gradiente al definirse como una rampa de pendiente 1, lo cual permite pasar el error entre las capas. Otra de las ventajas que presenta, es la simplicidad del cálculo, lo cual hace que su evaluación sea más rápida. Uno de los inconvenientes más comunes del uso de ReLUs es la posibilidad de la explosión de gradientes, ya que al ser una función no saturante se permite el paso de valores elevados.

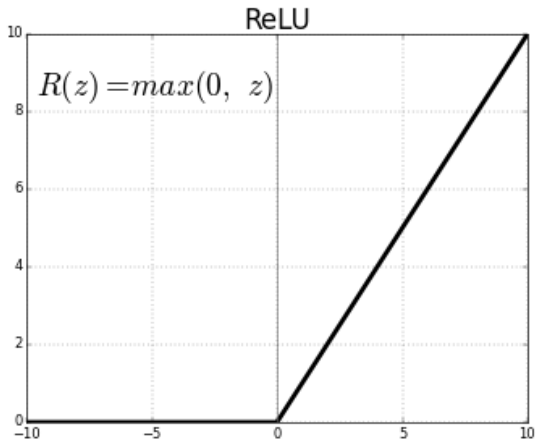


Figura 3.4: Función de activación ReLU.

Optimizador Adam

El optimizador se encarga de realizar el cálculo del error final y de los errores intermedios en el proceso de entrenamiento. A partir de estos errores, el optimizador determina como modificar los pesos de la red para mejorar el modelo.

El optimizador Adam [15], es uno de los más avanzados actualmente, incorporando ventajas de otros optimizadores como AdaGrad y RMSProp [16]. Entre sus características están la existencia de diferentes tasas de aprendizaje para diferentes parámetros de la red, la adaptación de la tasa de aprendizaje a partir del cálculo de dos estimaciones de momentos, un número reducido de parámetros a ajustar y unos valores por defecto que suelen dar buen resultado en la mayoría de problemas.

Capítulo 4

Antecedentes

La clasificación automatizada de los estados de sueño es un problema que se ha intentado resolver mediante diversas técnicas y que ha experimentado un mayor número de propuestas en los últimos años, debido en gran medida a la popularidad del aprendizaje máquina frente a los complejos algoritmos de análisis de señales usados anteriormente. A pesar de las múltiples propuestas, ciertas características de este problema, como los diversos tipos de ruido contenidos en las señales, las variaciones que se producen entre individuos de distintas edades y sexo o la propia complejidad del problema, dificultan la creación de una solución que pueda equiparar sus resultados a los de un experto.

Existen factores que dificultan la comparación entre las soluciones propuestas, así como su verificación y reproducibilidad.

La clasificación realizada por expertos del sueño presenta variaciones significativas, especialmente entre las etapas N2 y N3, así como entre N1 y N2. Danker-Hopfe et al. [17] comparan clasificaciones tanto para el estándar de R&K como para el de la AASM. Los resultados muestran unos índices kappa de 0.73 (80.6%) para R&K y del 0.76 (82.0%) para AASM. Las soluciones que se detallarán a lo largo de este capítulo se encontrarán con estos mismos problemas.

Otros factores que complican la evaluación entre las diferentes soluciones son: la comparación de resultados entre estudios que siguen estándares diferentes (R&K y AASM); el uso no estricto de los estándares, con variaciones respecto a la colocación de electrodos o de las señales usadas; la variación entre los datos de los estudios y la cantidad de estos.

4.1 Análisis de estudios antecedentes

Los primeros trabajos realizados aplicando técnicas de Inteligencia Artificial se centraban principalmente en la extracción manual de características, existiendo una gran cantidad de aproximaciones exploradas, incluyendo: análisis en el dominio del tiempo y/o de la frecuencia, análisis de las formas de onda, cálculos de la energía de las ondas y cálculos estadísticos.

Los algoritmos de clasificación utilizados en estos trabajos eran diversos: árboles de decisión, máquinas de soporte vectorial, vecinos cercados, redes neuronales de diverso tipo, etc.

En el año 2006, Gudmundsson et al. [18] realizan un estudio en el que proponen el uso de máquinas SVM para realizar la clasificación de las etapas del sueño. Su objetivo es determinar si este tipo de algoritmo permite mejorar los resultados obtenidos en estudios previos que usaban como clasificador un algoritmo k-NN. Para simplificar el problema y debido al tamaño reducido de la base de datos utilizada, se realiza la clasificación en etapas despierto, sueño ligero (etapas 1 y 2) y sueño profundo (etapas 3 y 4), siguiendo el estándar R&K. Se utiliza un único canal de EEG al que se le aplica un filtro notch en 50Hz para reducir ruidos provenientes de la corriente eléctrica. En cuanto a la extracción de características, se exploran diferentes alternativas, encontrando el histograma de distribución de amplitud y frecuencia de cada epoch como la característica con mejores resultados, seguido por características de energía y por último el análisis de parámetros de Hjorth. Sobre la clasificación obtenida con la SVM se aplica un postprocesado basado en probabilidades posteriores para evitar cambios bruscos entre etapas que no son habituales. Los resultados obtenidos sin postprocesado, con una precisión del 76%, son similares a los conseguidos con el algoritmo k-NN. Sin embargo, la aplicación del postprocesado consigue aumentar dicho valor de precisión hasta el 81% superando así al algoritmo k-NN.

El trabajo de Längkvist et al. [19] se centra en el uso de técnicas de aprendizaje no supervisado con el objetivo de eliminar el proceso de extracción manual de características sobre las señales. Utilizan las señales de EEG, EMG y dos canales de EOG, siguiendo el estándar de la AASM. El tamaño de epoch usado es de 1 segundo sin solapamiento, frente al uso tradicional de 30 segundos descrito en el estándar R&K. Además, los epochs directamente anteriores o posteriores a un cambio de etapa se eliminan para reducir posibles errores de etiquetado. En cuanto al preprocesado de las señales se aplica un filtro notch en 50Hz. Los experimentos se centran en el uso de *Redes de Creencia Profunda* (Deep Belief Nets o DBN, en inglés) y de Modelos Ocultos de Márkov (Hidden Markov Models o HMM, en inglés). Los autores implementan tres tipos de sistemas, denominados *feat-GOHMM, feat-DBN y raw-DBN*. En los dos primeros se realiza una extracción manual de 28 características. En feat-GOHMM se realiza una selección de características mediante un análisis de componentes principales seguido de un modelo Gaussiano mixto. En raw-DBN la extracción de características es realizada por la propia red a partir de las distintas señales. El sistema feat-DBN obtiene los mejores resultados en términos de precisión (72.2%), seguido por los sistemas raw-DBN (67,4%) y feat- GOHMM (63,9%). Frente a los buenos resultados a la hora de clasificar las etapas despierto y sueño profundo, destacan los pobres resultados en la clasificación de la etapa N1. Las características extraídas de forma automática (raw-DBN) coinciden en gran medida con las extraídas manualmente, sin embargo, a diferencia de estas últimas, no permiten establecer una relación

entre las distintas señales.

En el estudio de Hsu et al. [20] se realizan tres aproximaciones para la clasificación de etapas de sueño con técnicas diferentes: redes neuronales recurrentes Elman (Recurrent Neural Network o RNN, en inglés), redes neuronales alimentadas hacia adelante (Feed Forward Neural Networks o FNN, en inglés) y redes neuronales probabilísticas (Probabilistic Neural Net o PNN, en inglés). Su objetivo es aprovechar la idoneidad de las redes recurrentes para trabajar con información temporal, algo inherente en las señales. Las entradas de las redes se obtienen a partir de las características de energía de un único canal de EEG. El trabajo sigue el estándar de la AASM y la base de datos utilizada ha sido Sleep-EDF [21]. La extracción de características se realiza aplicando filtros de respuesta finita al impulso (FIR) y la Transformada Rápida de Fourier. Los resultados muestran unas precisiones de clasificación del 87.2% para las RNN, seguidas de 81.8% para las PNN y por último, 81.1% para la FFN. En estos resultados se demuestra la dificultad para clasificar la etapa N1, con valores precicsión del 36.7% para las RNN e inferiores para las otras dos aproximaciones.

Hassan y Bhuiyan [22] proponen el uso de un modelo CEEMDAN (Complete Ensemble Empirical Mode Decomposition with Adaptive Noise), junto con la técnica de Boostrap Aggregation o Bagging aplicada a árboles de decisión. Este trabajo extrae sus datos de la base de datos Sleep-EDF y utiliza únicamente un canal de EEG, con el objetivo de diseñar un sistema eficiente pero eficaz que permita el uso de dispositivos de poca potencia. En cuanto al número de etapas clasificadas, este estudio realiza implementaciones que comprenden desde únicamente dos etapas (despierto y dormido) hasta las seis etapas definidas en el estándar R&K. Al modelo de CEEMDAN se incorporan características basadas en estadísticos como la media, la varianza, el sesgo y el coeficiente de Kurtosis. El rendimiento de este sistema se ha evaluado en la identificación de seis, cinco, cuatro, tres y dos etapas, consiguiendo valores de precisión de 86,89%, 90.69%, 92.14%, 94.10% y 99.48% respectivamente. Sus resultados son comparables con otros estudios que utilizan la misma base de datos.

En 2017, Supratak et al. presentan el modelo DeepSleepNet [23]. En este trabajo se utiliza el Aprendizaje Profundo mediante redes convolucionales y redes recurrentes LSTM en el diseño de un algoritmo robusto que permita trabajar con un único canal de EEG, sin la necesidad de extraer características manuales o de realizar un filtrado avanzado sobre la señal. Las bases de datos utilizadas son MASS y Sleep-EDF. El modelo diseñado se adapta a las diferencias presentadas por ambos conjuntos de datos en cuanto a frecuencia de muestreo de las señales y al estándar de clasificación. La arquitectura de red propuesta se divide en dos secciones. La primera parte de la red se encarga del aprendizaje representacional, formado por dos redes convolucionales de una dimensión. El motivo del uso de dos redes reside en la diferencia del tamaño de los filtros usados, centrándose la red con filtros de menor tamaño

en aprender características relativas a patrones temporales pequeños, mientras que la red con filtros de mayor tamaño permite aprender mejor la información relativa a la frecuencia de las ondas. La segunda parte de la red es la encargada del aprendizaje residual y está formada por capas LSTM-bidireccionales y una capa Totalmente Conectada, cuyo objetivo es aprender las características propias de las etapas de sueño a partir de las salidas convolucionales, así como aprender la información temporal relativa a las transiciones entre las mismas. Esta información temporal permite reducir el número de transiciones erróneas entre etapas, especialmente entre aquellas que debido a su caracterización son poco frecuentes. Los resultados de este trabajo demuestran las dificultades de la clasificación correcta del estado N1, con valores de precisión del 43.5% en la base de datos Sleep-EDF y del 60.4% en la base de datos MASS. La mayor parte de confusiones se encuentra entre N2 y N3, siendo N2 la etapa clasificada con mayor precisión. Las precisiones generales obtenidas son del 82.0% para la base de datos SleepEDF y de 86.2% para la base de datos MASS.

Capítulo 5

Tecnologías y material usado

PARA el desarrollo de este proyecto ha sido necesario el uso de un entorno de trabajo compuesto de distintos tipos de software, además de un hardware adecuado para trabajar con redes de Aprendizaje Profundo, así como un conjunto de datos de trabajo.

5.1 Lenguaje de programación Python

Python es un lenguaje de programación creado en el año 1991 por Guido van Rossum [24]. Este lenguaje se caracteriza por ser de alto nivel, interpretado y multiparadigma. Fue concebido inicialmente como un lenguaje orientado a comandos o scripting, disponiendo desde sus inicios de una gran cantidad de funciones a través de su biblioteca estándar.

En la actualidad Python es uno de los lenguajes de programación más populares [25], siendo el lenguaje más usado dentro del campo de la Inteligencia Artificial y la Ciencia de Datos. Esto es debido en gran medida por su sintaxis sencilla, sus capacidades multiplataforma y la gran cantidad de librerías auxiliares de alto nivel que permiten un manejo sencillo de grandes volúmenes de datos, creación de redes neuronales, etc. Muchas de las librerías más importantes para desarrollo de aplicaciones basadas en inteligencia artificial están disponibles para este lenguaje, como pueden ser Scikit-learn, Tensorflow, PyTorch, Keras o Theano entre otras.

En la actualidad existe dos versiones del lenguaje claramente separadas, *Python 2.X* y *Python 3.X*. Ambas versiones tienen una gran popularidad, pero la versión 2.X dejó de ser soportada de manera oficial a comienzos del año 2020. Por este motivo el desarrollo de este Trabajo Fin de Grado se ha realizado con la versión 3.6, la cual cuenta con soporte actualizado por parte de las librerías necesarias para el desarrollo.

5.1.1 Librerías auxiliares de Python

A continuación se describirán las librerías más importantes usadas para complementar las funcionalidades de Python y los motivos de su elección.

PyEDFlib

PyEDFlib es una libería de código abierto basada en EDFlib que permite el manejo de ficheros en formato EDF+ y BDF+[26]. El principal motivo de su elección es la simplicidad de su API, tanto para el manejo de señales como para sus anotaciones. Además permite trabajar con ficheros tanto del estándar EDF/BDF como con EDF+/BDF+, cumpliendo con los requisitos necesarios para trabajar con bases de datos como Sleep-EDFX.

Numpy

Numpy es uno de los paquetes fundamentales para el uso de Python en entornos científicos [27]. Los motivos de su uso son su capacidad para trabajar con *vectores o arrays y matrices*, permitiendo realizar operaciones sobre ellos de manera eficiente y sencilla.

Matplotlib

Matplotlib es una librería orientada a la generación de gráficos a partir de datos contenidos en listas o arrays de Numpy, siendo la librería más popular para este tipo de uso [28].

La librería permite la exportación de los gráficos generados en múltiples resoluciones y formatos. En este trabajo es usada para para generar las imágenes a partir de las señales de EEG y EOG.

OpenCV

OpenCV es una librería diseñada para usar en visión artificial que soporta el lenguaje Python, proporcionando facilidad de uso y elevada eficiencia en el manejo de imágenes [29].

Esta librería es usada para realizar la carga de datos de entrada de la red neuronal, permitiendo realizar ajustes como reescalados eficientes. Además, OpenCV devuelve las imágenes directamente como matrices definidas con Numpy, eliminando la necesidad de realizar conversiones posteriores.

Scikit-learn

Scikit-learn, también conocida como SKlearn, es la librería de referencia para Python dentro del campo del Aprendizaje Automático [30]. Entre sus módulos se encuentran implementaciones de diversos tipos de algoritmos de Aprendizaje Automático, como pueden ser Máquinas

de Soporte Vectorial, Vecinos Cercanos y Árboles de Decisión. También dispone funcionalidades para realizar extracción de características, preprocesado y filtrado de datos entre otros.

El uso de Scikit-learn en este trabajo está centrado en la generación de los conjuntos de entrenamiento y test, la generación de los grupos o folds para realizar validación cruzada y el cálculo de diversas métricas para evaluar los modelos propuestos.

Keras, Tensorflow y PlaidML

Keras es una librería de código abierto que proporciona una API de alto nivel para definir redes neuronales [31]. Los modelos definidos con esta API son traducidos a uno de los diferentes backends soportados, entre los que se encuentran librerías como Tensorflow, Theano, CNTK o PlaidML. Esto permite abstraerse de los cálculos de bajo nivel y centrarse en el proceso de definición de los modelos, la forma de entrenarlos y evaluarlos.

Los motivos de la elección de Keras son: la popularidad de la que dispone dentro del Aprendizaje Profundo, su API de manejo sencillo, su integración con múltiples backends y la facilidad para cambiar entre ellos y los conocimientos previos de esta librería.

En cuanto al backend, se ha optado por usar dos librerías, dependiendo del sistema en el que se vaya a ejecutar.

El backend principal para el desarrollo es Tensorflow, una librería de manejo de tensores desarrollada por Google y usada principalmente dentro del campo del Aprendizaje Automático [32]. Tensorflow es una de las librerías más populares dentro de su área, siendo el backend más usado para Keras.

Tensorflow dispone de implementaciones que permiten su ejecución en CPU, GPU y TPU, siendo la implementación sobre GPU la más interesante para este caso. El uso de una GPU con una potencia de cálculo y memoria dedicada suficiente permite alcanzar velocidades más de 10 veces mayores con respecto a una CPU.

La implementación para GPU disponible de manera oficial está diseñada para ser ejecutada sobre GPUs de Nvidia, con la librería CUDA y con la posibilidad de usar optimizaciones específicas para redes neuronales mediante cuDNN.

Los motivos de la elección de Tensorflow frente a los demás backends soportados por Keras son: su mayor popularidad, el amplio desarrollo del que dispone y los conocimientos previos sobre la librería.

Como backend secundario se ha usado PlaidML [33], una librería desarrollada por Intel que dispone de implementaciones para CPU y GPU, siendo compatible con GPUs de Nvidia y de AMD.

El principal motivo de su uso es el de añadir la posibilidad de ejecución sobre GPUs de AMD sobre el sistema operativo Windows, ya que es el sistema operativo de uno de los equipos de desarrollo del Trabajo Fin de Grado.

Versiones de las principales librerías usadas

- Python == 3.6
- PyEDFlib == 0.1.14
- Numpy == 1.16.5
- Matplotlib == 3.1.0
- OpenCV (Python) == 3.4.2
- Scikit-learn == 0.21.2
- Keras == 2.2.4
- Tensorflow (GPU) == 1.14.0
- CUDA == 10.1
- cuDNN == 7.6.1
- PlaidML == 0.6.4

5.2 Entorno de desarrollo

Para gestionar los distintos paquetes de manera eficiente se ha usado la herramienta Anaconda [34], la cual permite crear distintos entornos de desarrollo de Python, así como sincronizar programas auxiliares como IDEs con esos entornos.

El IDE utilizado ha sido Spyder [35], el cual permite tener una visión sencilla y directa de las variables de los programas y una integración ágil con el depurador. Como herramienta auxiliar para documentar ciertos procesos y pruebas del desarrollo se ha usado la programación mediante las libretas de código de JupyterLab [35].

Para mantener un control sobre las versiones del código y los distintos caminos explorados se ha usado un repositorio de Git [36].

5.3 Fuente de datos de polisomnografías

Los datos usados en este trabajo son extraídos de uno de los conjuntos de datos públicos de PhysioNet [37], una organización de investigación biomédica. La base de datos que se ha seleccionado es Sleep-EDFX [38] en su versión v1.0.0, que es la base de datos de medicina del sueño más extensa de la que disponen. Esta base de datos cuenta con un total de 197 estudios de polisomnografías, en las que se registran dos canales de EEG, un canal de EMG submental y un canal de EOG horizontal además de un marcador de eventos.

Todos los estudios están formados por dos ficheros de datos: los ficheros PSG que contienen los datos de las señales y siguen el formato EDF; y los ficheros de hipnogramas que siguen el estándar EDF+ y contienen anotaciones en las que se incluye la etapa de sueño de cada instante. Estas anotaciones están realizadas manualmente por expertos en medicina del sueño siguiendo las indicaciones de R&K. Las etapas de sueño están definidas como: Wake, 1, 2, 3, 4 , Movement y '?', esta última usada para indicar que ese epoch no ha sido evaluado.

Capítulo 6

Metodología y planificación

Para llegar a desarrollar los modelos que resuelvan el problema de la clasificación de las etapas del sueño es necesario realizar una organización del trabajo que permita avanzar de manera ordenada.

6.1 Fases de desarrollo

Existe una serie de fases necesarias para realizar el desarrollo y que deben definir el trabajo a realizar de manera iterativa, determinando los siguientes pasos a realizar en función de las conclusiones intermedias.

La metodología de desarrollo toma como base una versión simple de Scrum, entendiendo cada fase como un *sprint*. En cada fase se genera o un producto de conocimiento o un producto de implementación software.

Las fases deben implementarse en el siguiente orden:

1. Adquisición de conocimientos teóricos de la medicina del sueño e identificar las características relevantes del dominio.
2. Adquisición de conocimientos teóricos del campo del Aprendizaje Profundo.
3. Diseño de la estructura general del proyecto.
4. Procesado, limpieza y exportación de datos (imágenes) de entrada.
5. Gestionar la división de datos en los conjuntos de entrenamiento y test y manipularlos de manera eficiente.
6. Desarrollar modelos de Aprendizaje Profundo para realizar la clasificación.
7. Evaluar los resultados.

El desarrollo técnico comienza con el diseño de la estructura del proyecto (fase 3). El diseño planteado consiste en la división de las funcionalidades en diferentes scripts de Python, de manera que la estructura general se divida en las siguientes partes:

- Configuración común (config.py). Este contendrá información general del proyecto que necesita ser accedida desde diferentes partes. Esto incluye, información sobre las rutas que contienen las señales y los conjuntos de datos, información sobre la manera de procesar las señales ...

- Funciones para el manejo de ficheros de polisomnografías (psg_utils.py). Contiene los métodos para realizar la apertura y lectura de las señales y anotaciones.

- Funciones para generar el conjunto de datos válido (dataset.py). Contiene los métodos necesarios para realizar la limpieza necesaria y la exportación de las imágenes que formarán la entrada de las redes. Esto es, la implementación de la fase 4.

- Funciones para gestionar el entrenamiento y la realización de los test de los modelos implementados (train.py y test.py). Esto es, la implementación de la fase 5.

- Definición de los modelos de Aprendizaje Profundo. (models.py). Esto es, la implementación de la fase 6.

El desarrollo de los modelos de Aprendizaje Profundo (fase 6) es un bloque de trabajo demasiado grande como para desarrollarlo como una única fase, por ese motivo se descompone en un total de cuatro iteraciones. Cada iteración estará centrada en determinar la idoneidad de un tipo de aproximación para resolver el problema de la clasificación de las etapas del sueño.

Las iteraciones de trabajo son:

1. Determinación del conjunto de datos adecuado.

2. Diseño de modelos con EEG y resolución de entrada 640x480.

3. Diseño de modelos con EEG y resolución de entrada 320x240.

4. Diseño de modelos con EOG y resolución de entrada 320x240.

Dentro de cada iteración se definen los objetivos concretos, el proceso de desarrollo completar los objetivos y las conclusiones específicas de esa iteración. Esas conclusiones determinarán la manera de actuar en las siguientes iteraciones.

6.2 Estimación de costes

Para realizar la estimación de costes se va a tener en cuenta la participación de dos tipos de perfiles. Un ingeniero de aprendizaje automático junior (ingeniero de AA) y un programador junior.

El papel del ingeniero de AA es diseñar la arquitectura, el diseño de la cadena de procesado de datos, la definición de los modelos de Aprendizaje Profundo y su evaluación.

El papel del programador se centra en ayudar en las tareas de implementación de los módulos específicos para apoyar al ingeniero de AA.

El coste temporal se calcula suponiendo un precio de 15€/hora para el ingeniero de AA y un coste de 12€/hora para el programador.

La Tabla 6.1 muestra la estimación de esfuerzo y los costes asociados.

Fase	**Perfil**	**Esfuerzo aprox. (horas)**	**Coste (€)**
1	Ing. AA	15	225
2	Ing. AA	10	150
3	Ing AA	4	60
4	Ing AA	20	300
	Programador	6	72
5	Ing AA	5	75
	Programador	5	60
6	Ing AA		
6.1	-	60	900
6.2	-	90	1350
6.3	-	80	1200
6.4	-	45	675
7	Ing AA	15	225
		Total: 355	5292

Tabla 6.1: Estimación de coste de personal.

Para la realización del proyecto también es necesario disponer de material de trabajo, lo que conlleva con unos gastos que se muestan en la Tabla 6.2. El alto coste de los equipos de desarrollo se debe a las GPU.

Elemento	Coste
Internet	220
Electricidad	300
Equipos desarrollo	2800
Total:	3320

Tabla 6.2: Estimación de coste de materiales.

El coste total de realizar el proyecto se estima en 8612€.

6.3 Medidas de rendimiento

Para evaluar los modelos desarrollados se utilizará un conjunto de medidas de rendimiento que se describen a continuación.

Función de pérdida

La función de pérdida o coste define la función a minimizar durante el entrenamiento de la red neuronal. Su valor indica el nivel de error entre la salida de la red y el valor real.

En el caso de la clasificación, las salidas son de tipo categórico. La función de pérdida más usada para problemas de clasificación multiclase es la entropía cruzada categórica (categorical cross-entropy, en inglés). El valor de esta función disminuye cuando la clase predicha con mayor probabilidad es la de la clase esperada.

$$\text{loss}(p^{\text{true}}, p^{\text{predict}}) = -\sum_i p_i^{\text{true}} \log p_i^{\text{predict}}$$

Para juzgar los resultados de un modelo se usan las métricas de evaluación. Debido a que existen múltiples términos en español para los mismos conceptos y que su uso más frecuente es en inglés, se usará este último idioma para referirse a estas métricas y evitar confusiones.

Accuracy

La métrica de *accuracy* (Acc) se define como la proporción de casos acertados sobre el total de predicciones.

$$accuracy = \frac{VP + VN}{VP + FP + VN + FN}$$

Siendo VP la cantidad de positivos clasificados como tal, VN la cantidad de negativos clasificados como tal, FP la cantidad de negativos clasificados como positivos y FN la cantidad de positivos clasificados como negativos.

Precision

La métrica de *precision* (Prec) puede definirse como la proporción de elementos positivos correctamente clasificados para una determinada clase.

$$precision = \frac{VP}{VP + FP}$$

Recall

La métrica de *recall* (Rec) puede definirse como la proporción de elementos positivos detectados para una determinada clase.

$$recall = \frac{VP}{VP + FN}$$

F1-Score

La métrica *F1-Score* (F1) es la media armónica de *precision* y *recall*.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Macro-F1

La métrica Macro-F1 (MF1) es la media de la métrica F1 para el total de clases a clasificar.

$$MF1 = \frac{\sum_{c=1}^{C} F1_c}{C}$$

Con C=nº de clases y siendo $F1_c$ el valor F1 de cada clase.

Capítulo 7

Desarrollo

El siguiente capítulo tiene como objetivo describir el proceso seguido durante el desarrollo del sistema clasificador de etapas de sueño, así como los resultados intermedios obtenidos.

7.1 Preprocesado de datos de polisomnografías

Del conjunto de datos de Sleep-EDFX [38] se han utilizado 190 de los 197 registros, lo que supone un total de 455.596 epochs.

De las señales de estos registros, se trabaja con un canal de la señal de EEG, y con un canal de EOG. No se ha utilizado la señal de EMG ya que la mayor parte de la información que aporta, se encuentra incluida en la señal de EEG.

El proceso de tratamiento de las señales comienza con la obtención de las etapas asignadas a cada epoch. Esta información se extrae de los ficheros de anotaciones que acompañan a cada registro polisomnográfico.

Dado que se pretende usar la definición de etapas propuesta por la AASM, y que los ficheros de anotaciones siguen el estándar R&K, es necesario considerar las etapas 3 y 4 como etapa 3. Además, los epochs anotados como movimiento se convertirán a etapa despierto (W) tal y como se recomienda en [39].

7.1.1 Filtrado de epochs no válidos

Para cada uno de los registros polisomnográficos, es necesario identificar aquellos epochs que no son válidos para su clasificación y así poder eliminarlos.

Los epochs correspondientes al final del registro se eliminan, ya que no disponen de una asignación de etapa válida. Estos epochs están anotados como desconocidos.

Dado que las señales se obtienen mediante sensores pegados al cuerpo del paciente, existe la posibilidad de que estos realicen mediciones incorrectas debido a los movimientos de dicho

paciente. Al utilizar un único canal de EEG, los epochs correspondientes a estas mediciones deben eliminarse ya que presentarán valores de amplitud anómalos. Los valores de amplitud máximos y mínimos habituales para el EEG son de $\pm 200\mu$V [40]. En el caso de las formas de onda características como los complejos-K, la señal puede presentar picos de $\pm 300\mu$V. En el caso de la señal de EOG, su rango normal de valores también es de $\pm 300\mu$V.

La Figura 7.1 muestra un fragmento del registro de la señal de EEG en el que se puede apreciar la presencia de los valores anómalos mencionados.

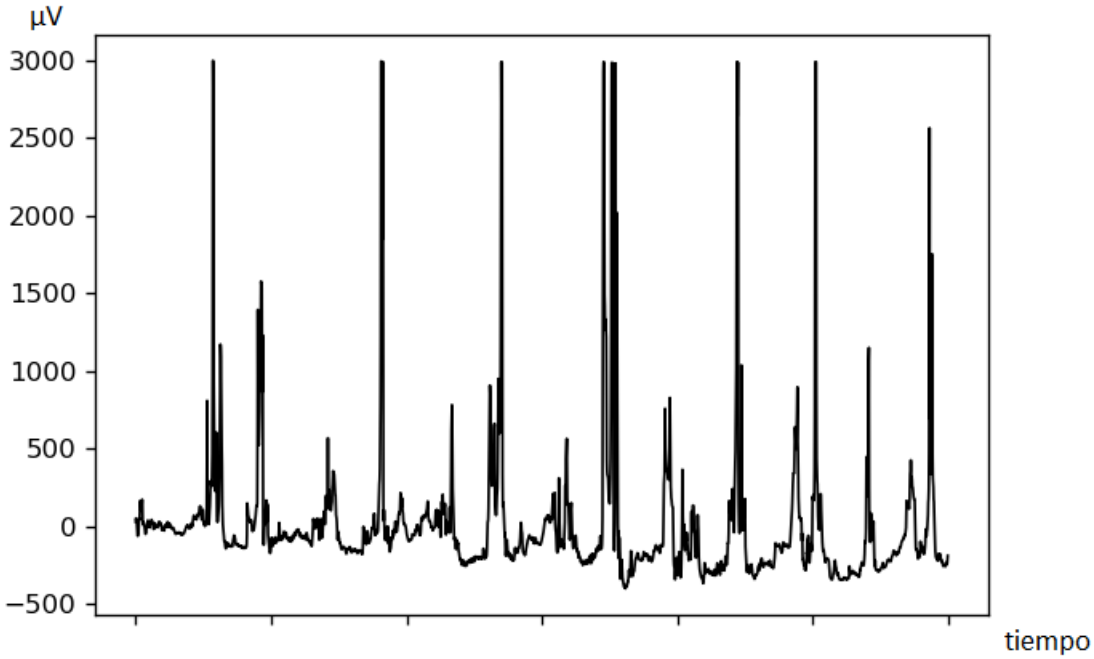


Figura 7.1: Fragmento de un registro de la señal de EEG con valores anómalos.

Aquellos epochs que presenten valores anómalos en un 50% o más de su duración, se eliminarán. En el caso de no superar este porcentaje, los valores que exceden los $\pm 300\mu$V se suavizan mediante la aplicación de un filtrado Hampel [41].

Los epochs adyacentes a los que presentan valores anormalmente altos o bajos suelen caracterizarse por estar formados exclusivamente por valores positivos o negativos, algo que no es posible en términos fisiológicos. Este efecto es producido por el equipamiento que realiza el registro y carece de interés, por lo que también deberán eliminarse para no introducir ruido en el conjunto de datos.

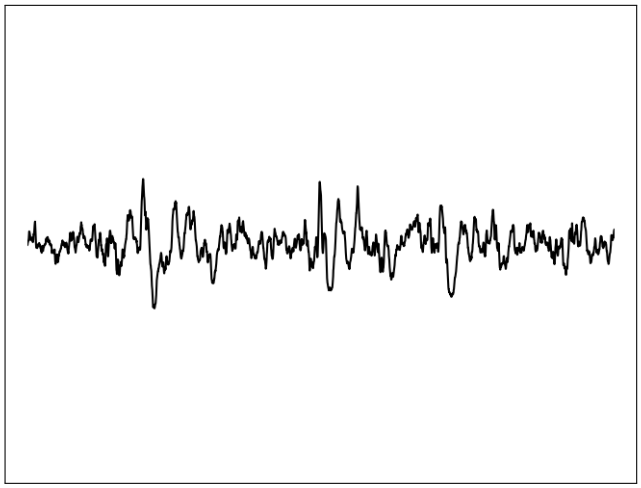
7.1.2 Generación de las imágenes

El objetivo del trabajo es la clasificación de imágenes de la señal de EEG en etapas de sueño. Para generar imágenes similares a las que visualizan los expertos encargados de analizar las polisomnografías, es decir, señales representadas en el dominio del tiempo, es necesario definir parámetros como el rango de valores mostrado y la resolución de las imágenes. El rango de valores está determinado por las características de las señales. El eje X debe representar el avance temporal de los 30 segundos de un epoch. Como la frecuencia de muestreo es de 100Hz tanto para la señal de EEG como de EOG, este eje debe incluir 3.000 valores. En el eje Y se representan los valores de magnitud de la señal y estarán en el rango $\pm 300\mu$V, tal y como se explica en la Sección 7.1.1.

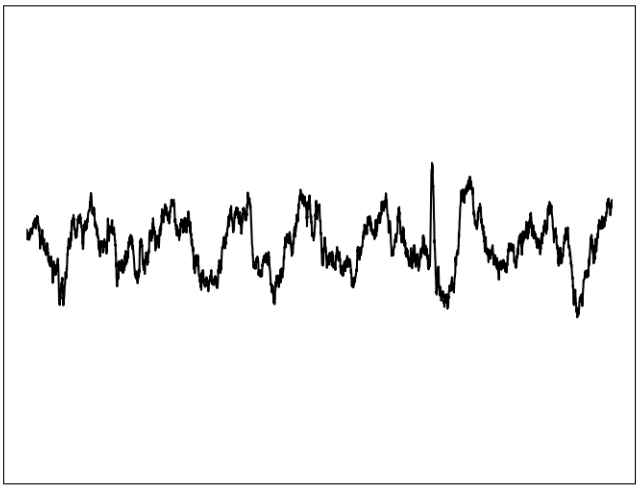
La resolución usada para la exportación de las imágenes es de 640x480 píxeles[1], lo que equivale a una exportación con un valor de dpi en matplotlib de 100. El formato de imagen elegido es *png*. Este formato usa un algoritmo de compresión sin pérdida, algo importante para no perder definición en las formas de onda, ya que un ligero suavizado ayudará a producir valores más continuos de señal y reducir picos y ruidos, pero un exceso de suavizado puede llevar a la pérdida de información relevante. Esto es especialmente importante si los filtros son pequeños, ya que en caso de existir distorsión, esta representará un mayor porcentaje del segmento de la imagen a analizar.

La Figura 7.2 muestra las imágenes generadas a partir de un epoch clasificado como etapa de sueño N3.

[1]Consultar Apéndice A para ver el proceso de generación y almacenamiento de las imágenes



(a) Canal de EEG



(b) Canal de EOG

Figura 7.2: Ejemplo de una imagen generada a partir de un epoch de señal.

7.2 Conjuntos de entrenamiento y test

En los registros polisomnográficos, suele ocurrir un desbalanceo con respecto a las distintas etapas de sueño. Tal y como puede apreciarse en la Tabla 7.1, la presencia de etapas clasificadas como despierto es muy elevada con respecto al resto, siendo unas 3,3 veces superior a la siguiente etapa con más casos, la N2. Por otra parte, la etapa N3 es la que tiene una menor representación, a pesar de que resulta de la unión de dos etapas.

Etapa	W	N1	N2	N3	REM	Descartados
Nº epochs disponibles	289984	24497	86250	18473	32762	3630
%	64.2	5.4	19.1	4.1	7.2	-

Tabla 7.1: Distribución del número de epochs por etapa de sueño.

Los datos disponibles se dividen en dos conjuntos, uno de entrenamiento y uno de test. El porcentaje de división es de un 80% para entrenamiento y un 20% para test.

Para la generación de los conjuntos de entrenamiento y test se usa la librería *scikit-learn*, a través de las funciones *train_test_split* para realizar entrenamientos únicos que permitan probar las capacidades de aprendizaje de una red, y *KFold* para realizar validación cruzada [42].

7.2.1 Carga y procesado de datos de entrada de la red

Para poder realizar el entrenamiento y test de las redes neuronales es necesario cargar todos los datos de las imágenes. Se han implementado dos maneras de gestionar los datos [2]: utilizar la memoria RAM y usar generadores para obtener los datos a medida que se necesitan. Se da preferencia a la opción de utilizar la memoria RAM siempre que sea posible ya que permite reducir el tiempo de entrenamiento.

7.3 Modelos de Aprendizaje Profundo

En este apartado se describe el proceso que ha llevado a la obtención de los distintos modelos basados en Aprendizaje Profundo, que se utilizarán para clasificar las imágenes en etapas de sueño. Basándonos en la metodología de desarrollo especificada en el Capítulo 6, se indican las distintas iteraciones realizadas.

7.3.1 Iteración 1.

Objetivo

El objetivo de la primera iteración es determinar la influencia de disponer de un conjunto de datos tan desbalanceado y valorar las posibles alternativas, así como obtener una primera arquitectura del modelo.

Desarrollo

Tal y como se mencionó en la Sección 7.2, el número de ejemplos disponibles para cada clase es muy diferente, siendo la clase que tiene una menor relevancia dentro de la tarea de

[2]Consultar Apéndice B para ver el proceso de carga de datos de entrada

clasificación de etapas del sueño la de mayor número.

Esta diferencia hace pensar que los modelos van a tener un gran sesgo hacia las clases W y N2, las dos con mayor representación. Para confirmarlo se diseñan dos redes de Aprendizaje Profundo, ambas con tamaños de entrada de 640x480, la resolución a la que se generan las imágenes. La primera consta de tres capas convolucionales seguidas de dos capas FC y la segunda consta de cuatro capas convolucionales seguidas de dos capas FC.

Cada **bloque convolucional** está formado por una capa convolucional 2D, una capa de normalización por lotes, una capa de activación y una capa MaxPooling 2D. En el caso del modelo con cuatro bloques convolucionales, los tamaños de los filtros usados son: 17x17, 13x13, 9x9 y 7x7. La versión con tres bloques usa los tres primeros tamaños de filtro. Estos tamaños permiten capturar la información temporal de las señales y serán refinados durante el desarrollo.

Las **capas FC** están formadas por las neuronas totalmente conectadas, una capa de normalización por lotes, una capa de activación y una capa de Dropout con valor 0.5.

El modelo de cuatro bloques convolucionales cuenta con 1024 y 512 neuronas en la primera y segunda capa FC respectivamente. El modelo de tres bloques convolucionales cuenta con dos capas FC con 256 neuronas.

La última capa de la red es una capa FC (Dense) con 5 neuronas y función de activación *softmax*. En las demás capas la función de activación es *ReLU*.

La división del conjunto de datos en entrenamiento y test se hace mediante una selección aleatoria o hold-out, con el 80% de los casos para entrenamiento, lo que supone un total de 361.572 ejemplos, y el 20% de los casos para test, lo que supone un total de 90.393 ejemplos.

El algoritmo para la optimización de la red que se ha elegido es Adam [15]. Los parámetros usados son: lr=1e-03, beta_1=0.9, beta_2=0.999 y epsilon=1e-08, que son los valores por defecto en Keras para este optimizador. Las métricas de evaluación usadas son las definidas en la Sección 6.3 (página 28).

Ambos modelos son entrenados durante 10 ciclos completos. Este valor es suficiente para comprobar la evolución de las predicciones y el efecto del desbalanceo entre las clases.

Los resultados de los test muestran unos valores de *accuracy* del 85% para el modelo con tres capas convolucionales y del 83% para el modelo con cuatro, donde al continuar la evaluación se observa que los resultados se deben a los aciertos en las clases más representadas (W y N2). Este es el comportamiento habitual de la métrica de *accuracy* cuando las clases están desbalanceadas, confirmado con el cálculo de la métrica *F1* (Tabla 7.2), que muestra valores casi nulos para las clases N1 y REM.

Tipo modelo	W	N1	N2	N3	REM
3 conv + 2FC	0.95	0.02	0.75	0.70	0.01
4 conv + 2FC	0.93	0.01	0.73	0.69	0

Tabla 7.2: Valores de F1 para los modelos estudiados.

Existen múltiples maneras de balancear un conjunto de datos. Como la clase con mayor representación es la más fácilmente clasificable y la de menor interés, la reducción de los ejemplos de esta etapa parece una opción adecuada. Este submuestreo se hará eliminando epochs clasificados como W al inicio y fin del registro, tal y como se muestra en la Figura 7.3.

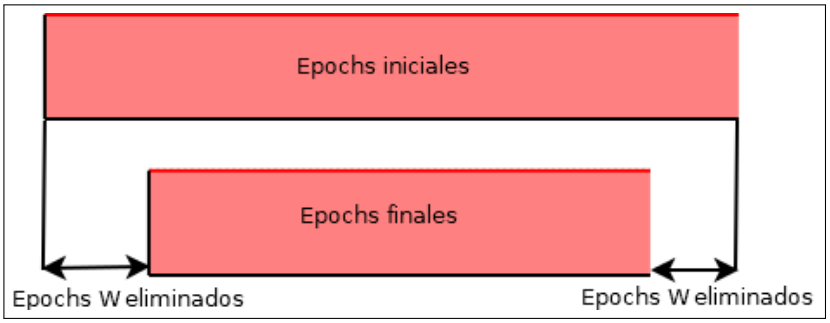


Figura 7.3: Esquema de reducción de epochs W.

Para determinar el número de epochs W que se eliminarán, se han realizado dos pruebas considerando un máximo de 10 y 25 minutos, antes y después del inicio y fin del sueño del paciente. Esto es, contabilizando desde que aparece el primer epoch diferente de W y desde el último diferente de W, al inicio y al final del registro respectivamente.

Teniendo en cuenta estas dos aproximaciones, la Tabla 7.3 muestra la distribución resultante para las distintas etapas de sueño.

Etapa	W	N1	N2	N3	REM
Nº epochs max 10min W	61291	24497	86250	18473	32762
% max 10min W	27.5	10.9	38.6	8.3	14.7
Nº epochs max 25min W	70712	24497	86250	18473	32762
% max 25min W	30.4	10.5	37	8	14.1

Tabla 7.3: Distribución del número de epochs por etapa de sueño.

Con estos datos, se repiten las pruebas iniciales y se obtienen los resultados de la Tabla 7.4.

Modelo	W	N1	N2	N3	REM
10minW 3 conv + 2FC	0.65	0.20	0.65	0.63	0.47
10minW 4 conv + 2FC	0.72	0.19	0.66	0.68	0.39
25minW 3 conv + 2FC	0.81	0.27	0.70	0.73	0.49
25minW 4 conv + 2FC	0.83	0.25	0.75	0.70	0.56

Tabla 7.4: Valores de F1 tras el proceso de submuestreo.

En la tabla se puede comprobar un descenso en el valor *F1* de la etapa W debido a la gran reducción de ejemplos de esta etapa, pero a cambio, se observa una mejora considerable de esta medida para las etapas N1 y REM, para las que la aproximación inicial no era capaz de definir modelos eficientes.

Las diferencias existentes en los resultados de la Tabla 7.4 para los modelos 10min y 25min, indican que el submuestreo debe ser controlado para evitar la limitación de la capacidad de aprendizaje y generalización de los modelos.

La clase con peores resultados sigue siendo N1, algo habitual en la clasificación automática de etapas de sueño. Dado que N1 presenta valores de *F1* inferiores a N3, a pesar de tener un 2.5% más de casos disponibles sobre el total del conjunto de datos (unos 6000 casos más), se considera que los resultados se deben a la dificultad de caracterizar la clase y no tanto al reducido número de casos disponibles, en comparación a las clases con mayor representación.

Conclusiones

Después de realizar las primeras pruebas, puede concluirse la relevancia de disponer de un conjunto de datos balanceado. La técnica de submuestreo resulta adecuada para ser utilizada en el resto de iteraciones del desarrollo, siendo la versión con límite de 25 minutos la que proporciona los mejores resultados.

7.3.2 Iteración 2.

Objetivo

El objetivo de esta iteración es el diseño de varios modelos que tomen como entrada las imágenes obtenidas a partir de la señal de EEG con una resolución de 640x480, comprobando su capacidad de aprendizaje e identificando los posibles problemas de sobreentrenamiento.

Desarrollo

Dado que el número de ejemplos disponibles es bajo en comparación a las complejidad de las redes con las que se va a trabajar, es necesario comprobar como afecta la selección de los

datos al proceso de entrenamiento.

Para conseguir verificar la independencia de los resultados y los datos seleccionados deben usarse técnicas como la validación cruzada, que consiste en repetir el proceso de entrenamiento n veces, calculando finalmente la media entre las diferentes pruebas, siendo habitual calcular también la varianza. Debido a la relevancia del volumen de datos disponibles para el entrenamiento al trabajar con modelos complejos, se decide usar validación cruzada aleatoria o con hold-out, también conocida como validación cruzada Monte-Carlo [43], usando un 90% de los casos para entrenamiento y un 10% para test sobre un total de 5 conjuntos. Esta versión de la validación cruzada tiene como ventaja la independencia del tamaño del conjunto de entrenamiento y test en relación al número de pruebas que se realizan. Su inconveniente principal es que los datos pueden solaparse en las múltiples iteraciones.

Debido a que el tiempo de entrenamiento de un epoch en el sistema disponible supera los 20 minutos, se considera que el uso de este tipo de metodología de validación es más adecuada que usar un esquema de selección aleatoria único o validación cruzada 10-fold. La validación cruzada aleatoria consigue un buen compromiso en cuanto a rango de exploración y cantidad de tiempo necesario para ejecutar las pruebas.

En cuanto a los modelos, se diseñan versiones con tres y cuatro capas convolucionales y una y dos capas FC, tomando como base las conclusiones de la iteracion anterior. Se experimenta con diferentes tamaños y número de filtros en las capas convolucionales, que van desde 7x7 hasta 25x25. La selección del tamaño de pool y de paso, se ha realizado tanto con solapamiento como sin el. También se prueba con diferentes números de neuronas en las capas FC, con valores de 256, 512 y 1024. El optimizador de entrenamineto usado es Adam con los siguientes parámetros: 1e-04, beta_1=0.9, beta_2=0.999 y epsilon=1e-08. La función de activación es ReLU, con softmax de 5 neuronas en la última capa de la red. El tamaño de batch usado es 32, estando limitado por la cantidad de memoria de GPU disponible. El entrenamiento se realiza sobre un total de 20 ciclos de entrenamiento. La elección del número de ciclos se basa en intentar reducir el sobreentrenamiento y obtener tiempos de ejecución aceptables que permitan realizar la validación con los múltiples paquetes.

La selección y optimización de los hiperparámetros se realiza manualmente a partir de la creación de un conjunto de búsqueda inicial generado automáticamente. Las técnicas como la búsqueda en cuadrícula, búsqueda aleatorizada o técnicas más complejas, junto con la validación cruzada requieren de una gran cantidad de tiempo de ejecución, no ajustándose correctamente a los modelos complejos planteados.

Cada prueba se realiza en función de los valores obtenidos anteriormente, orientando los hiperparámetros hacia los valores que muestran unos mejores resultados. El número total de modelos probados para esta iteración es de 12, de los cuales una parte es descartada por presentar poca capacidad de aprendizaje o por tener una gran variabilidad de resultados en

función de los datos de entrada.

La mayor parte de los modelos explorados se basan en el diseño mencionado en la Sección 7.3.1 (página 35), al que denominaremos como 1er diseño convolucional. Dentro de la sección de los bloques convolucionales también se explora el siguiente diseño, denominado como 2º diseño convolucional (Figura 7.4).

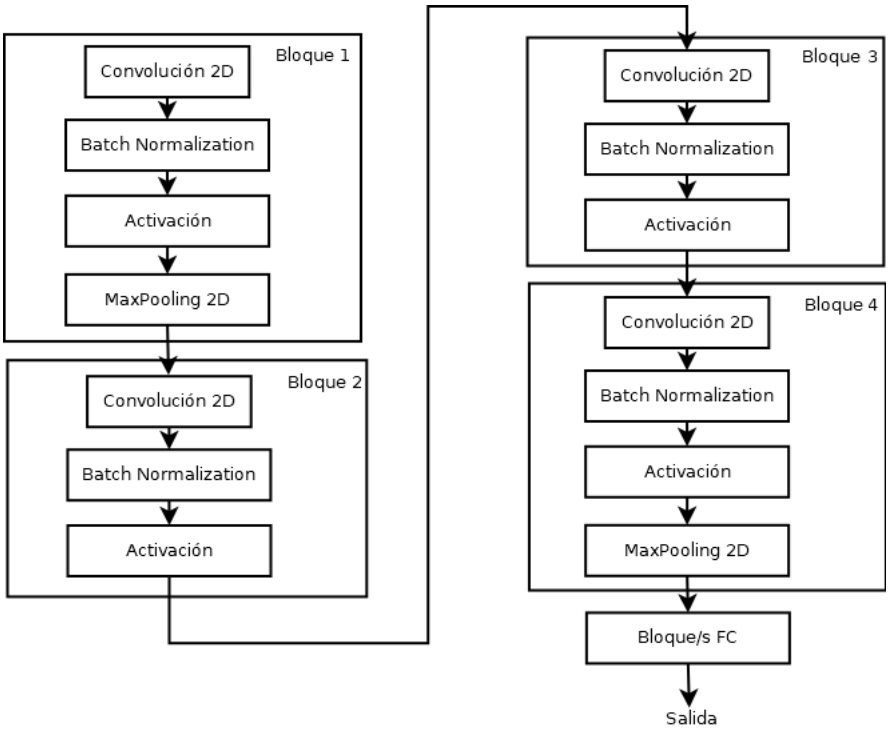


Figura 7.4: Esquema del 2º diseño convolucional.

En caso de usar 3 bloques convolucionales se suprime el tercer bloque de entre los cuatro planteados en la figura.

En la Tabla 7.5 se muestran los resultados de las arquitecturas más significativas probadas[3].

[3]La arquitectura de cada uno de los modelos puede consultarse en el Anexo C

				F1				
Nº modelo	**Tipo**	**Acc**	**MF1**	**W**	**N1**	**N2**	**N3**	**REM**
2	4 conv tipo 1 + 2 FC	0.68	0.57	0.80	0.21	0.75	0.65	0.42
3	4 conv tipo 1 + 2 FC	0.67	0.58	0.81	**0.26**	0.72	0.70	0.39
5	4 conv tipo 2 + 2 FC	0.53	0.38	0.65	0.11	0.69	0.35	0.12
6	3 conv tipo 2 + 2 FC	0.52	0.39	0.51	0.10	0.62	0.45	0.29
7	3 conv tipo 1 + 2 FC	0.67	**0.62**	**0.97**	0.23	0.60	**0.83**	0.48
9	4 conv tipo 1 + 2 FC	**0.70**	0.58	0.84	0.10	**0.76**	0.65	0.52
10	4 conv tipo 1 + 2 FC	**0.70**	0.60	0.85	0.15	**0.76**	0.72	**0.53**

Tabla 7.5: Medidas de *accuracy*, *MF1* y *F1* para los distintos modelos.

La Tabla 7.6 muestra la matriz de confusión para el mejor modelo, así como las medidas de rendimiento obtenidas con dicho modelo.

	Predicción							
	W	**N1**	**N2**	**N3**	**REM**	**Prec**	**Rec**	**F1**
W	6237	171	90	17	420	0.88	0.90	0.89
N1	543	298	635	8	876	0.41	0.13	0.19
N2	131	172	6618	359	1477	0.79	0.76	0.77
N3	9	1	522	1357	21	0.77	0.71	0.74
REM	146	92	516	10	2589	0.48	0.77	0.59

Tabla 7.6: Matriz de confusión y medidas de rendimiento del modelo 10.

Uno de los mayores problemas que presentan estos modelos es la gran facilidad de que se produzca sobreentrenamiento, esto es debido al gran número de parámetros a entrenar. El modelo 10 es el modelo con menos parámetros entrenables (1.142.793), pero en modelos como el 5 y 6, al no disponer todas las capas convolucionales de max-pooling, el número de parámetros es de 39.897.341 y 40.033.637 respectivamente. Los modelos intermedios, como el 7 y el 9 disponen de 10.492.197 y 5.151.689 parámetros respectivamente. Existe una clara relación entre el efecto del sobreentrenamiento y el número de parámetros entrenables, ya que en los modelos con mayor número de parámetros el rendimiento del mismo se mantiene durante casi todo el entrenamiento.

Conclusiones

Los resultados muestran una clara dificultad en la clasificación de la etapas N1 y REM, especialmente en la N1, lo que es habitual en la mayoría de las aproximaciones existentes.

Este es el motivo del bajo valor de *accuracy*. Se demuestra la capacidad de aprendizaje de la red, pero es necesario reducir la dimensión del tamaño de la entrada para simplificar los modelos y conseguir una mejor generalización y eficiencia.

7.3.3 Iteración 3.

Objetivo

El objetivo de esta iteración es la creación de modelos que tomen como entrada imágenes con menor resolución que en la segunda iteración. La resolución propuesta, de 320x240, sigue permitiendo apreciar las diferencias en las formas de onda de manera visual, con lo que se prevé que los resultados no empeoren y se mejore el proceso de entrenamiento en términos de tiempo de ejecución.

Desarrollo

La reducción del tamaño de la entrada a la mitad (76800) permite reducir notablemente los tiempos de ejecución en el entrenamiento. Esto implica una reducción en los parámetros del modelo que se necesita establecer, y conlleva a disminuir la dependencia del número de ejemplos para el entrenamiento. La consecuencia más inmediata es la posibilidad de realizar una evaluación más exhaustiva al poder usar más casos de test. El esquema de validación usado para esta iteración pasa a ser la validación cruzada de 5 iteraciones. En cada iteración de la validación 5-fold se dispone de un 80% de los datos para el entrenamiento y el 20% restante para el test. Este esquema de validación es más estricto que la validación cruzada aleatoria usada en la segunda iteración, ya que los modelos se prueban con conjuntos de test independientes, asegurándose de que no hay datos repetidos o no seleccionados para los conjuntos generados. El número de datos disponibles en cada paquete es de 186.155 para entrenamiento y 46.539 para test.

Las arquitecturas probadas en esta iteración coinciden con las de la iteración anterior. Dado que la resolución pasa a ser la mitad, el tamaño de los filtros se reduce por el mismo factor, ajustando el resultado al número impar más cercano y mayor que 1. Lo mismo ocurre con el tamaño del pool usado y de su paso. Por ejemplo, un modelo en el que el tamaño de los filtros de la primera capa convolucional fuese de 13 pasa a ser de 7 para esta resolución. De esta manera, la capacidad de representación espacial de los datos se mantiene, obteniendo modelos equivalentes. Además también se prueban variaciones sobre el modelo, como pueden ser la incorporación del dropout a las capas convolucionales o la modificación del número de filtros de ciertas capas.

Los entrenamientos se realizan durante 50 ciclos completos, con el optimizador Adam y los parámetros lr=1e-04, beta_1=0.9, beta_2=0.999 y epsilon=1e-08. Se realizan pruebas con

tamaños de batch usado es de 32 y 64, sin observar diferencias entre ellos, lo que lleva a decantarse por el de menor tamaño, al necesitar menos memoria de GPU.

Los resultados de la validación cruzada de los modelos más significativos[4] de cada tipo pueden verse en la Tabla 7.7.

				F1				
Nº modelo	**Tipo**	**Acc**	**MF1**	**W**	**N1**	**N2**	**N3**	**REM**
7	3 conv tipo 1 + 2 FC	0.71	0.62	0.88	0.17	0.76	0.70	0.57
9	4 conv tipo 1 + 2 FC	**0.75**	**0.64**	**0.89**	0.22	0.77	**0.72**	0.60
10	4 conv tipo 1 + 2 FC	0.71	0.63	0.87	**0.29**	0.76	0.70	0.52
12	4 conv tipo 1 + 2 FC	0.74	0.61	0.88	0.18	0.78	0.67	0.56
13	3 conv tipo 1 + 1 FC	**0.76**	**0.64**	**0.89**	0.19	**0.79**	0.71	**0.61**

Tabla 7.7: Medidas de *accuracy*, *MF1* y *F1* de cada clase para los distintos modelos.

Los modelos 9 y 13, con dos arquitecturas de red que difieren en número de capas convolucionales y capas FC, consiguen resultados muy similares. El modelo 9 ofrece una mayor estabilidad en sus resultados, con una variabilidad inferior al 5% para cada una de las métricas analizadas en los distintos paquetes. Por otra parte, el modelo 13 tiene valores de *accuracy* que oscilan entre un 79% y un 68%. En cuanto al número de parámetros de cada modelo, el modelo 9 dispone de 2.000.409 parámetros entrenables, mientras que el 13 dispone de 19.707.637. Esta diferencia de parámetros se debe a que la última capa convolucional del modelo 13 tiene el doble de filtros que la última capa convolucional del modelo 9 y a que tiene una capa menos de pooling.

Debido a su mayor estabilidad se considera que el modelo 9 es más efectivo, al tener una dependencia menor de los datos disponibles.

La Tabla 7.8 muestra la matriz de confusión para el el mejor modelo. La mayor parte de confusiones se producen asignando a etapas N1 la clase N2. La baja calidad en la detección de la etapa N1 reduce notablemente el valor de MF1.

[4]La arquitectura de cada uno de los modelos puede consultarse en el Anexo D

	Predicción							
	W	**N1**	**N2**	**N3**	**REM**	**Prec**	**Rec**	**F1**
W	13184	302	272	13	447	0.86	0.93	0.89
N1	1312	686	1710	11	1254	0.47	0.14	0.21
N2	325	277	14504	652	1504	0.76	0.84	0.80
N3	26	4	1089	2510	13	0.79	0.69	0.73
REM	428	192	1452	10	4455	0.58	0.68	0.63

Tabla 7.8: Matriz de confusión y medidas de rendimiento del modelo 9.

Conclusiones

El uso de modelos con una resolución de entrada menor (320x240) permite obtener mejoras en cuanto a coste computacional con respecto a la versión desarrollada en la segunda iteración. La capacidad de aprendizaje consigue mejorar gracias a la mayor simplicidad de los modelos. La clasificación con mayor dificultad sigue siendo la de la etapa N1, para todos los modelos evaluados.

7.3.4 Iteración 4.

Objetivo

El objetivo de esta iteración es el de comprobar las capacidades de clasificación de las etapas del sueño usando únicamente la señal de EOG con los modelos diseñados anteriormente. El uso de esta señal debería proporcionar una mejor precisión de clasificación para las etapas del sueño en las que los movimientos oculares tienen una presencia más marcada, como son las etapas W y REM.

Desarrollo

Las pruebas con la señal de EOG se realizan utilizando los modelos de la iteracón anterior, ya que son los que proporcionan los mejores resultados. Las señales de EOG contienen formas de onda más sencillas que las de EEG, por lo que la capacidad de los modelos desarrollados debería ser suficiente para aprender las características de esta señal de EOG. Sin embargo, lo que se pretende es determinar si esta señal con los modelos mencionados permiten identificar las etapas de sueño.

El esquema de validación se mantiene idéntico al de la iteración previa, ya que es un buen compromiso entre tiempo de ejecución y exhaustividad de validación.

Los entrenamientos se realizan durante 50 ciclos completos, usando como optimizador Adam con los parámetros lr=1e-03, beta_1=0.9, beta_2=0.999 y epsilon=1e-08. El tamaño de

batch usado es de 32.

				F1				
Nº modelo	**Tipo**	**Acc**	**MF1**	**W**	**N1**	**N2**	**N3**	**REM**
7	3 conv tipo 1 + 2 FC	0.71	0.63	0.84	0.27	0.76	0.64	0.62
9	4 conv tipo 1 + 2 FC	**0.74**	**0.66**	**0.86**	**0.29**	**0.77**	**0.69**	**0.70**

Tabla 7.9: Medidas de *accuracy*, *MF1* y *F1* para los distintos modelos.

En esta iteración se probaron un total de 4 modelos, de los cuales solamente dos de ellos mostraron cierta estabilidad entre paquetes[5]. La Tabla 7.9 muestra los resultados en términos de *accuracy* y *MF1* para los dos modelos más estables. La diferencia entre los distintos paquetes para las métricas *F1* y *accuracy* no supera el 8% y el 3% respectivamente.

	Predicción							
	W	**N1**	**N2**	**N3**	**REM**	**Prec**	**Rec**	**F1**
W	12237	588	879	48	466	0.88	0.86	0.87
N1	944	1107	2454	37	431	0.47	0.22	0.30
N2	427	471	14955	846	563	0.71	0.87	0.78
N3	53	8	1045	2526	10	0.72	0.69	0.71
REM	256	201	1699	34	4347	0.75	0.66	0.70

Tabla 7.10: Matriz de confusión y medidas de rendimiento del modelo 9.

La matriz de confusión del mejor modelo (Tabla 7.10) muestra que el valor de *precision* más elevado se encuentra en las clases W y REM. La clase N1 sigue siendo la que presenta los peores valores, con un valor de *precision* que no llega al 50% y un valor de *recall* del 22%. La mayor parte de los errores se producen entre las etapas N1 y N2, siendo superior el número de ejemplos de N1 clasificados como N2 que como N1.

Conclusiones

Los modelos basados en señal EOG muestran tener una capacidad de aprendizaje similar a la de los modelos basados en la señal EEG, permitiendo una buena identificación de las etapas W y REM, ofreciendo también unos resultados de *recall* para N1 superiores a los obtenidos con señales de EEG. El uso de la señal EOG presenta un sobreentrenamiento temprano más acusado. Esto es debido a que las formas de onda presentes en las señales son más sencillas de modelizar.

[5]La arquitectura de cada uno de los modelos puede consultarse en el Anexo D

Capítulo 8

Resultados

Este capítulo pretende resumir y evaluar los resultados generales extraídos de las diferentes iteraciones, interpretar dichos resultados y compararlos con otros estudios existentes.

8.1 Evaluación de resultados de los modelos desarrollados

El análisis que se realizará esta basado en las pruebas en cuanto a resolución de las imágenes generadas, número de filtros de los modelos, variaciones sobre el modelo inicial y señal utilizada.

En cuanto a las resoluciones probadas, la versión de resolución de entrada de 320x240 demuestra tener una mejor capacidad de generalización junto con una mejor eficiencia computacional.

El número de filtros que se utilice no debe ser necesariamente elevado, entre los modelos probados se encontraban implementaciones con hasta 128 filtros en la última capa, consiguiendo sin embargo los modelos con un número mucho menor (32) mejores resultados.

En cuanto a las formas de los filtros, los mejores resultados se han encontrado con filtros de dimensiones reducidas, como puede ser 9x9, 7x7 o 5x5 para la primera de las capas convolucionales, llegando a un tamaño mínimo en la última capa de 3x3. En cuanto a su forma, los filtros cuadrados han superado en resultados a los rectangulares.

La reducción con max-pool resulta más efectiva cuando se aplica después de cada una de las capas convolucionales. En cuanto al tamaño del pool, los mejores valores son con selección de 2x2 y mismo valor para el paso, o el uso de una selección de dimensión 4x4 y un paso de 2x2, es decir, con un solapamiento de la mitad del tamaño del pool.

Con respecto a la señal usada, los modelos de EEG y EOG han conseguido resultados generales similares, pero con diferencias en cuanto a los valores específicos de cada etapa. La señal de EEG consigue mejores resultados para las etapas W, N2 y N3, mientras que el EOG consigue mejores resultados para N1 y REM.

Una forma de comparar los resultados generados por los expertos con los obtenidos mediante el uso de los modelos de Aprendizaje Profundo es a través de la visualización de los hipnogramas de un registro polisomnográfico completo. La Figura 8.1 muestra el hipnograma del experto frente a los hipnogramas resultantes utilizando el modelo 9 con resolución de entrada 320x240 con las señales de EEG (Figura 8.2) y de EOG (Figura 8.3) respectivamente.

Al comparar los tres hipnogramas puede verse que la forma general del hipnograma original está presente en las versiones generadas a partir de las predicciones de los modelos. La mayor diferencia se encuentra en el mayor número de transiciones entre etapas en los hipnogramas generados con los modelos. Esto es debido en gran parte a la ausencia de información temporal y contextual del epoch a clasificar, lo que impide mantener una inercia de la etapa de sueño actual que se sobreponga a los ligeros cambios en las señales que no se corresponden con cambios reales de etapa.

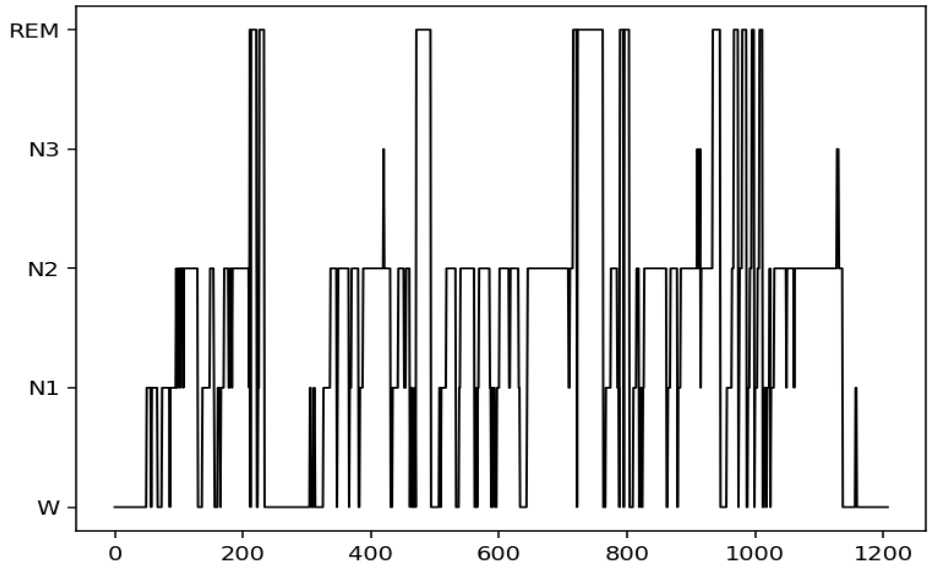


Figura 8.1: Hipnograma original.

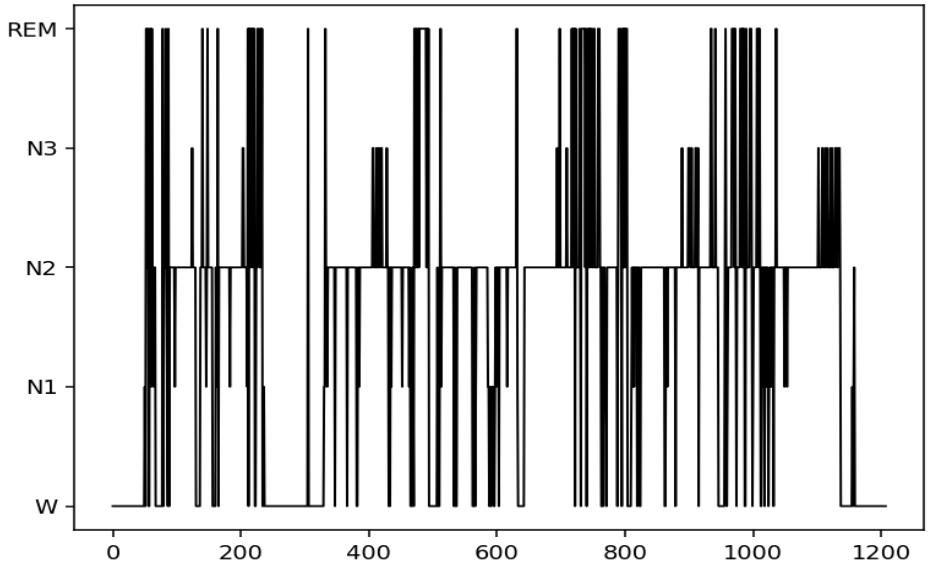


Figura 8.2: Hipnograma generado a partir de señal EEG.

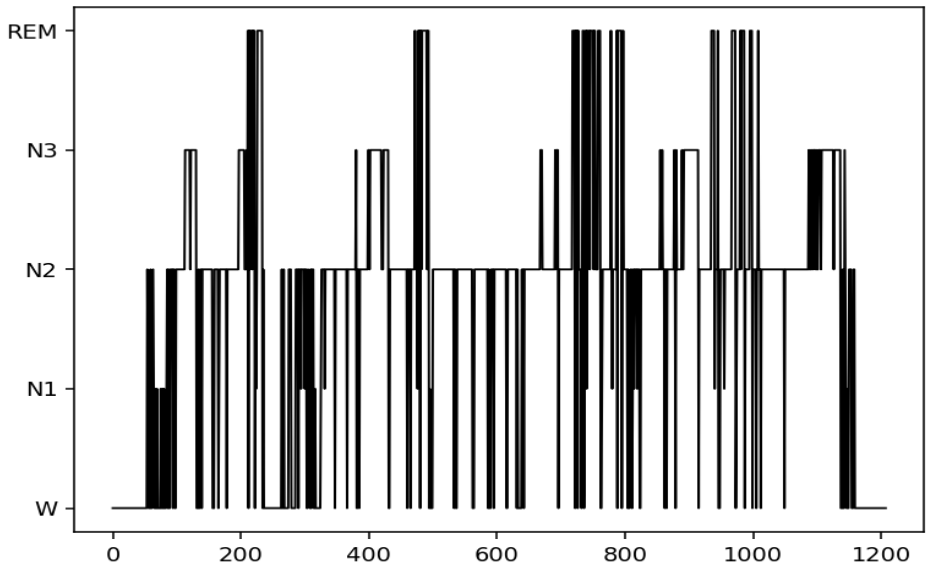


Figura 8.3: Hipnograma generado a partir de señal EOG.

8.2 Visualización interna de los modelos

Uno de los problemas de usar sistemas basados en redes neuronales es el efecto de *"caja negra"*, es decir, desconocer el funcionamiento interno de los modelos. En esta sección intenta mostrar gráficamente el conocimiento que almacenan los modelos desarrollados mediante los Saliency Maps.

Los Saliency Maps o mapas destacados [44] permiten representar el gradiente de la salida del modelo con respecto a la entrada. Esto permite identificar la relevancia relativa de los píxeles y determinar su influencia en la asignación de la clase por parte del modelo. La librería usada para realizar este cálculo es keras-vis [45].

La Figura 8.4 muestra un epoch de señal EEG para una etapa N3 junto con su Saliency Map. Los colores con tonos amarillos o verdes claros indican un mayor gradiente, lo que significa que tienen una mayor relevancia. El eje Y de las figuras representa la amplitud de la señal, con valores entre -300 y 300μV. El eje X representa el avance temporal del epoch desde los 0 hasta los 30 segundos.

La visualización de esta figura permite ver que los píxeles de mayor interés son aquellos que se encuentran en los picos de las formas de onda (voltajes de intensidad media) o en las secciones en las que la frecuencia de la señal es baja, lo que entra dentro de la definición de las etapas N3.

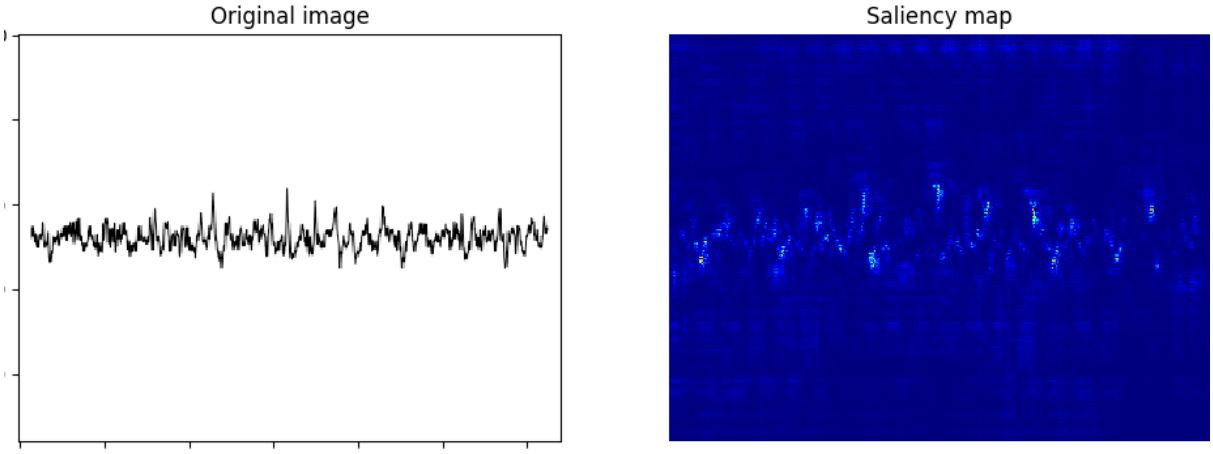


Figura 8.4: Epoch de EEG y su Saliency map correspondiente.

La Figura 8.5 muestra un epoch de señal EOG clasificado como W. En esta etapa, la señal de EOG presenta movimientos oculares perfectamente identificables. El Saliency Map asociado la imagen original muestra la relevancia de las zonas de gran amplitud, algo característico de la etapa W.

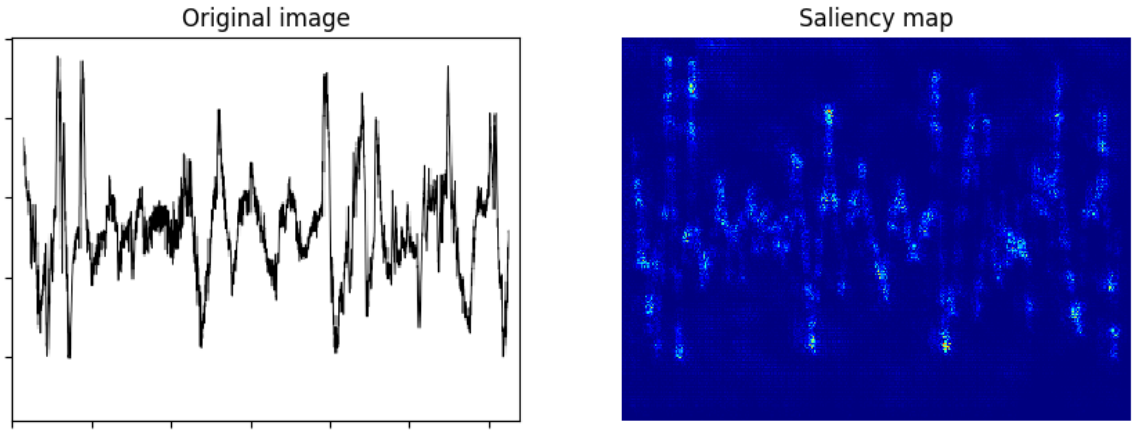


Figura 8.5: Epoch de EOG y su Saliency map correspondiente.

8.3 Comparación con otros estudios

La Tabla 8.1 muestra una comparación entre distintos estudios. Existen una serie de factores que dificultan la comparación de los valores, como puede ser el uso de diferentes conjuntos de datos, distintos canales de las señales, reportar solamente los mejores resultados obtenidos y no los de la validación, etc.

A pesar de que la comparación no es del todo justa, nuestros modelos muestran unos resultados similares a los de otros autores, especialmente respecto a aquellos más próximos en cuanto a la información que extraen (Längkvist, Tsinalis y Virkkala), superando tanto la versión con EEG como la de EOG los valores de *accuracy* general de Längkvist y de Virkkala, y con unos valores de *F1* similares. Las etapas que sufren peores resultados en comparación a los demás estudios son la N1 y la REM, especialmente para el modelo con señal de EEG.

Los resultados son por tanto alentadores ya que existe margen de mejora a partir de la base planteada.

			F1				
Trabajo	**Fuente de datos (# Num reg.)**	**Acc**	**W**	**N1**	**N2**	**N3**	**REM**
Längkvist et al. [19]	St Vicent's Univer. Hosp. #25, EEG, EOG, EMG	0.72	0.78	0.37	0.76	0.84	0.78
Hassan et al. [46]	Sleep-EDF. #8, EEG	0.90	-	-	-	-	-
Supratak et al. [23]	Sleep-EDF. #20, EEG	0.80	0.88	0.37	0.83	0.77	0.80
Supratak et al. [23]	MASS. #62, EEG	0.86	0.87	0.60	0.90	0.82	0.89
Tsinalis et al. [47]	Sleep-EDF. #20, EEG	0.78	0.67	0.44	0.81	0.85	0.76
Virkkala et al. [48]	Privada. #256, EOG	0.72	0.76	0.34	0.78	0.74	0.73
Este trabajo	Sleep-EDFX. #190, EEG	0.75	0.89	0.22	0.77	0.72	0.60
Este trabajo	Sleep-EDFX. #190, EOG	0.74	0.86	0.29	0.77	0.69	0.70

Tabla 8.1: Comparación entre distintos estudios.

Capítulo 9

Conclusiones

Este capítulo muestra las conclusiones generales de este trabajo, así como los conocimientos adquiridos y el posible trabajo futuro.

9.1 Conclusiones

El uso de modelos de Aprendizaje Profundo basados en 2D permite realizar la tarea de clasificación de estados de sueño de una manera próxima a la extracción de características que realiza el personal especializado, con unos valores de *accuracy* similares a los de otras aproximaciones existentes que no incorporan información temporal.

La elección del conjunto de datos debe realizarse de manera que se disponga del mayor número de patrones posible, pero siempre manteniendo un equilibrio entre el número de casos de cada clase.

La complejidad de los modelos debe ser controlada mediante la selección de un tamaño de entrada como la propuesta, de 320x240 o inferior, para poder disponer de tiempos de ejecución aceptables y evitar problemas de sobreentrenamiento de los modelos, que de otro modo modelizan no solo las formas de onda relevantes, sino también aquellas que contienen ruidos.

9.2 Conocimientos adquiridos

El desarrollo de este trabajo está centrado en los contenidos específicos de la mención de Computación, expandiendo los conceptos sobre Inteligencia Artificial estudiados en asignaturas como Aprendizaje Automático.

Los conocimientos específicos adquiridos son:

- Conocimiento teórico de las pruebas de polisomnografía.

- Estudio teórico de las bases del Aprendizaje Profundo y tipos de redes comunes dentro de esta rama.
- Manejo de librerías de alto nivel para Aprendizaje Automático en Python.
- Configuración y uso de GPUs en múltiples entornos para el desarrollo de algoritmos de Aprendizaje Profundo.

9.3 Trabajo futuro

Tras la realización del trabajo pueden determinarse ciertas lineas que continúen con la base desarrollada.

Teniendo en cuenta los resultados obtenidos y la comparación con los últimos trabajos que definen el estado del arte, puede verse la relevancia de incorporar conocimiento sobre las transiciones entre etapas, tanto mediante técnicas estadísticas como mediante modelos de Aprendizaje Profundo, que permitan reflejar esta información, como puede ser las redes recurrentes de Elman o las redes CNN con capas LSTM.

Por otra parte, dado que las señales de EEG y EOG presentan características que permiten caracterizar unas etapas mejor que otras, resulta obvio el diseño de un sistema que utilice ambas señales para realizar las clasificaciones en lugar de un único canal de información.

La realización de las pruebas con otra fuente de datos también puede ayudar a comprobar las capacidades de adaptación de los modelos desarrollados.

Apéndices

Apéndice A

Generación de imágenes

Dado que el proceso de generación de las imágenes requiere de una gran cantidad de tiempo de CPU, se realiza una única vez para el conjunto de datos, permitiendo la posibilidad de realizar reescalados futuros para trabajar con resoluciones de entrada inferiores. Además, la generación y exportación de la imagen de cada epoch puede ser realizada de manera paralela. Para optimizar el uso de los recursos de CPU y RAM disponibles, se usa la librería integrada en Python *multiprocessing*, creando un *Pool* con un número de trabajadores igual al número de threads de CPU disponibles en el sistema. Sobre el *Pool* se hace la llamada a *starmap_async*, la cual permite pasar la ejecución en paralelo de una función que reciba múltiples parámetros. En este caso las llamadas se realizan a la función encargada de generar y exportar las imágenes. Esta función recibe como parámetros la señal, la ruta en la que guardar la imagen, el nombre del estudio, el número del epoch y la etapa en la que ha sido clasificado.

La ruta y nombre de las imágenes de salida contienen por tanto toda la información necesaria para referenciar la señal que representan, eliminando la necesidad de almacenar un array o fichero auxiliar en el que se realice una asociación entre la imagen y la etapa de sueño que representa.

La estructura de almacenamiento de los datos generados tiene un diseño de fácil extensión en caso de querer usar más señales y sigue el esquema de la Figura A.1.

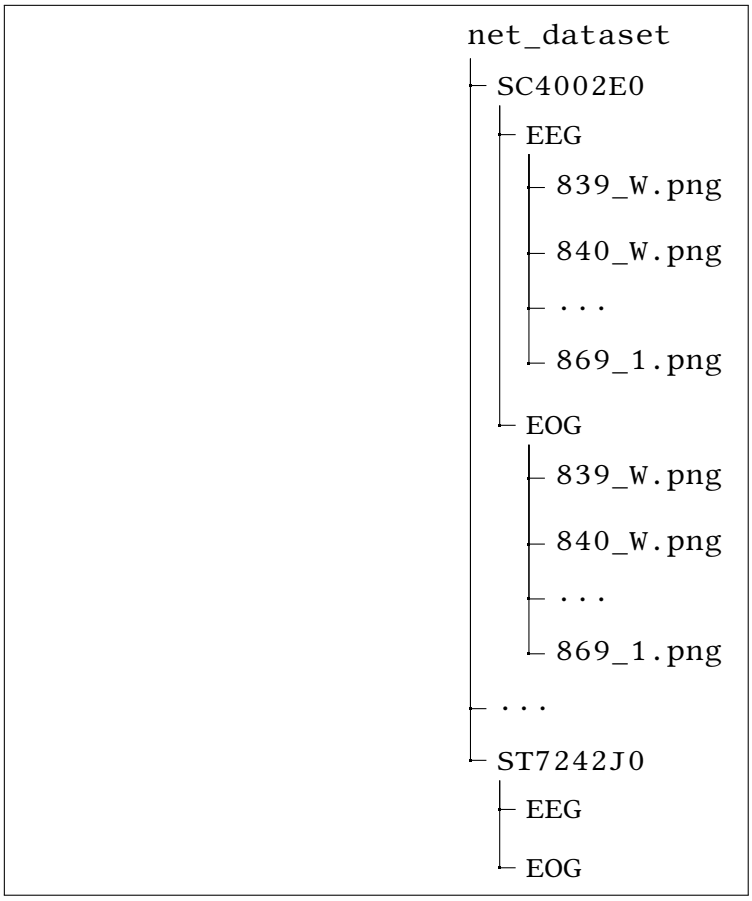


Figura A.1: Estructura de carpetas del conjunto de datos de imágenes

Apéndice B

Proceso de carga de datos de entrenamiento y test

Dentro de Keras existen dos modos diferenciados para gestionar los datos de entrenamiento y test. La primera opción es cargar todo el conjunto de datos en la RAM del equipo. La segunda opción es usar generadores de Python para cargar los datos a medida que se necesitan [42].

Trabajar con los datos en memoria es la manera más eficiente en cuanto a alcanzar mayor velocidad de entrenamiento o test. El inconveniente principal es la necesidad de disponer de un equipo con una gran cantidad de memoria. Por eso motivo también se han implementado generadores, de manera que se pueda realizar el proceso en cualquier sistema a cambio de una penalización en cuanto a tiempo de ejecución. Estos generadores solo devuelven un batch de datos en cada llamada. Pueden crearse generadores que se ejecuten concurrentemente para reducir los tiempos de espera por motivos de carga de datos.

Las funciones para cargar datos de entrenamiento, tanto si implementan la carga en memoria como los generadores, realizan el siguiente proceso:

1. Para cada nuevo epoch, reordenar los datos.
2. Cargar las imágenes con OpenCV (escala de grises).
3. Redimensionar las imágenes al tamaño de entrada de la red con OpenCV.
4. Obtener las etapas de los datos cargados a partir del nombre de los ficheros.
5. Convertir nombres de etapa a valor numérico de clase.
6. Devolver los datos.

En el caso de las funciones de test se suprime el primer paso, ya que la reordenación no afecta a los resultados de los test.

Apéndice C

Modelos para resolución de 640x480

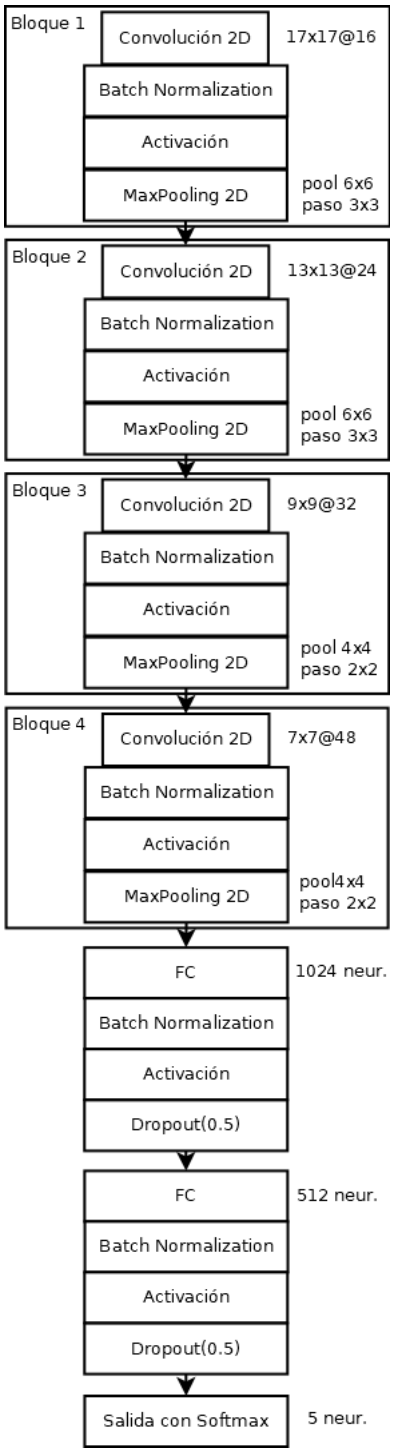


Figura C.1: Modelo 2.

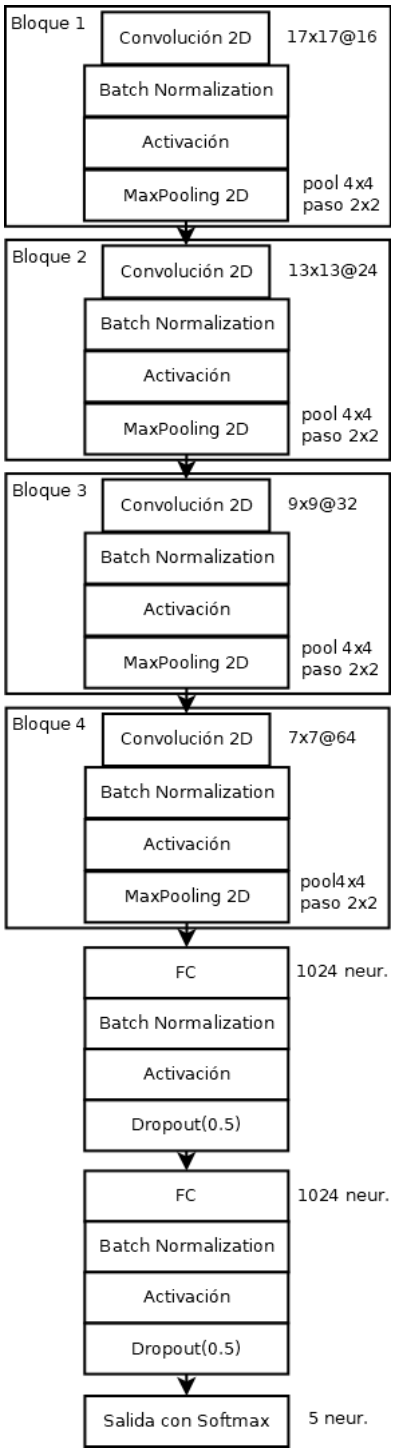


Figura C.2: Modelo 3.

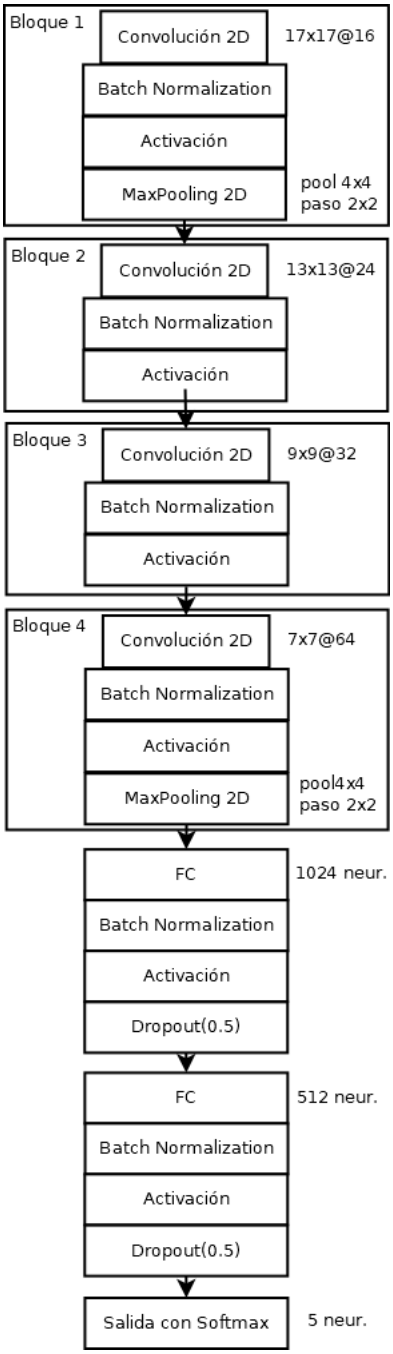


Figura C.3: Modelo 5.

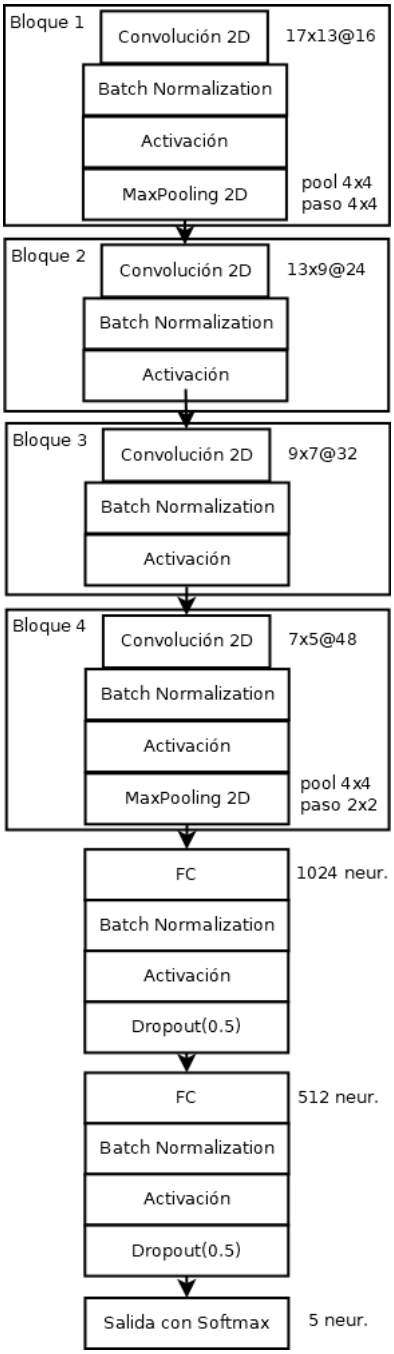


Figura C.4: Modelo 6.

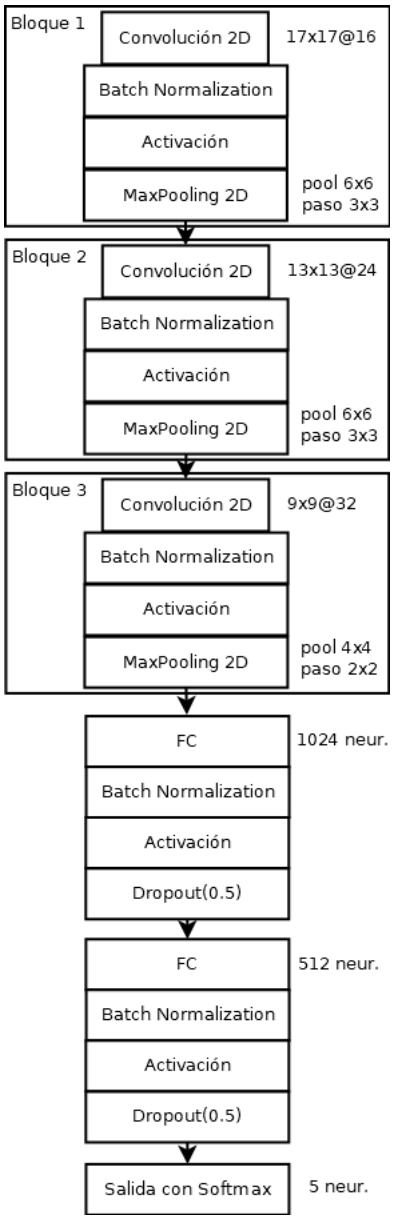


Figura C.5: Modelo 7.

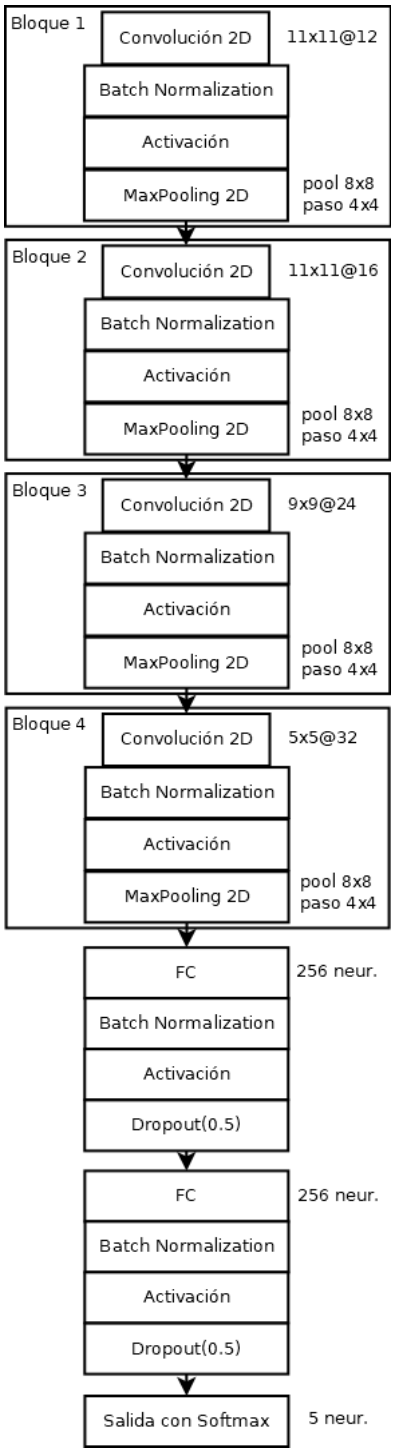


Figura C.6: Modelo 9.

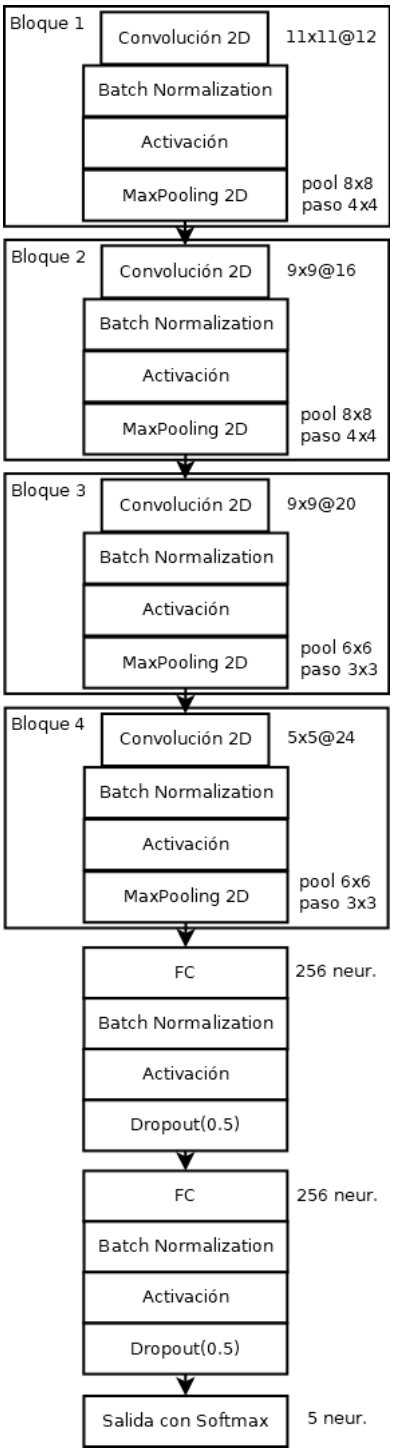


Figura C.7: Modelo 10.

Apéndice D

Modelos para resolución de 320x240

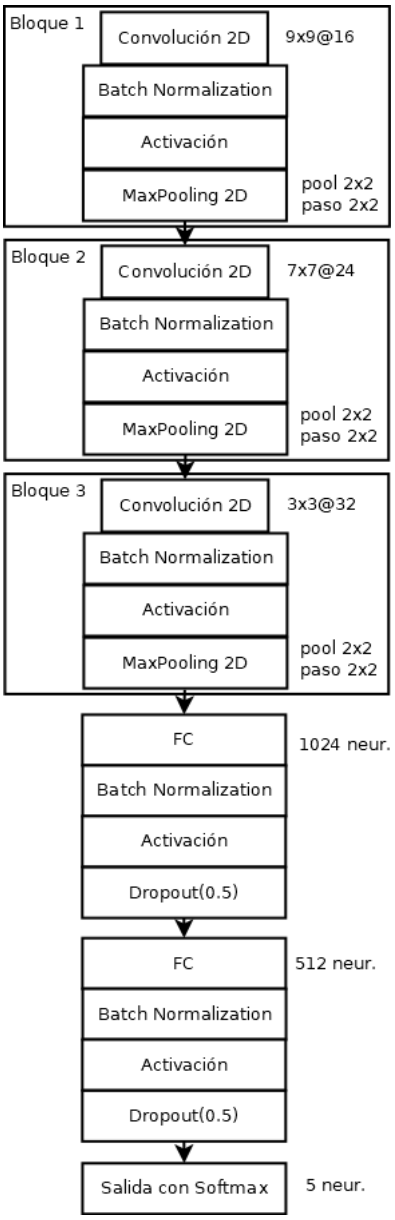


Figura D.1: Modelo 7.

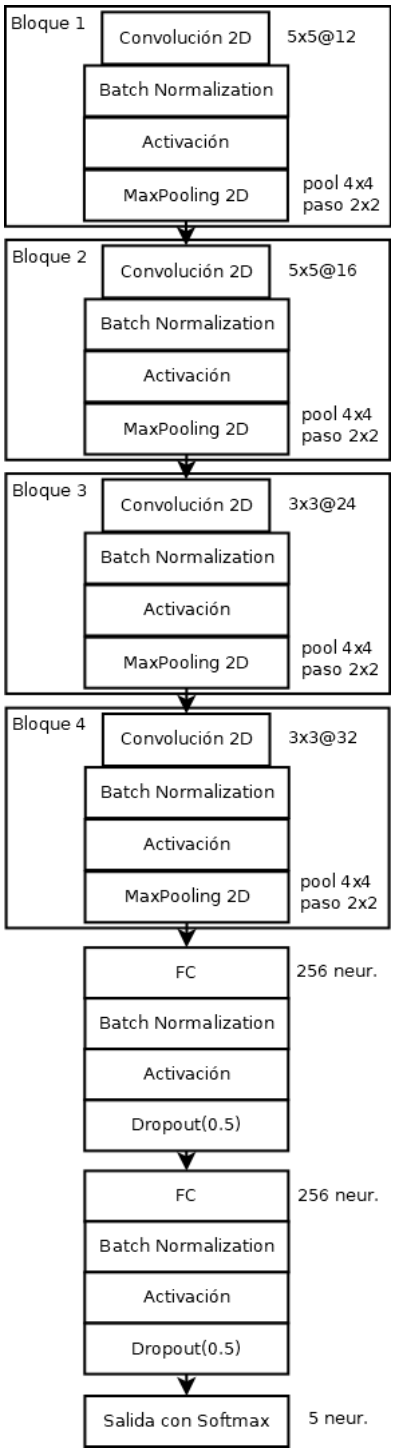


Figura D.2: Modelo 9.

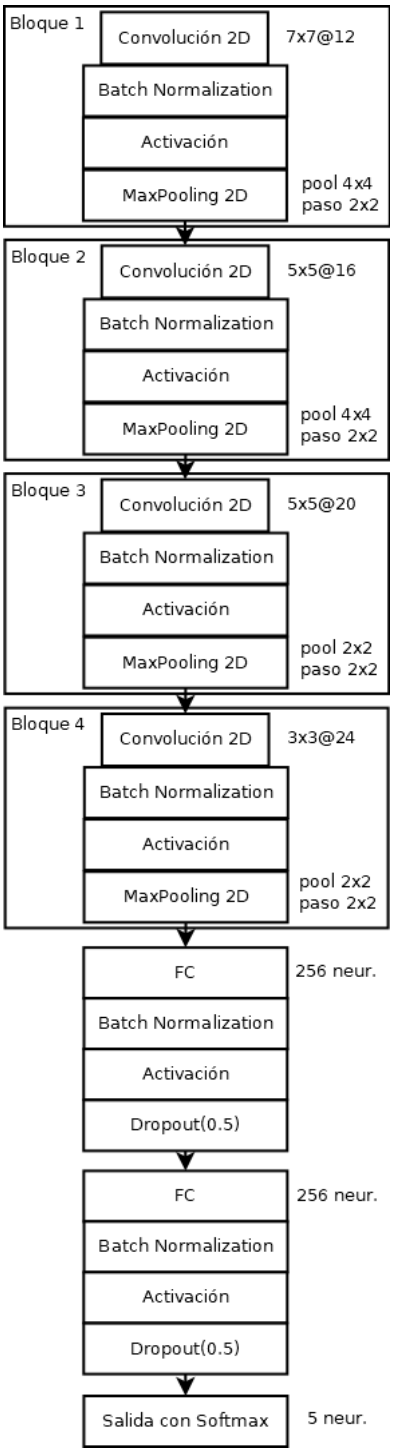


Figura D.3: Modelo 10.

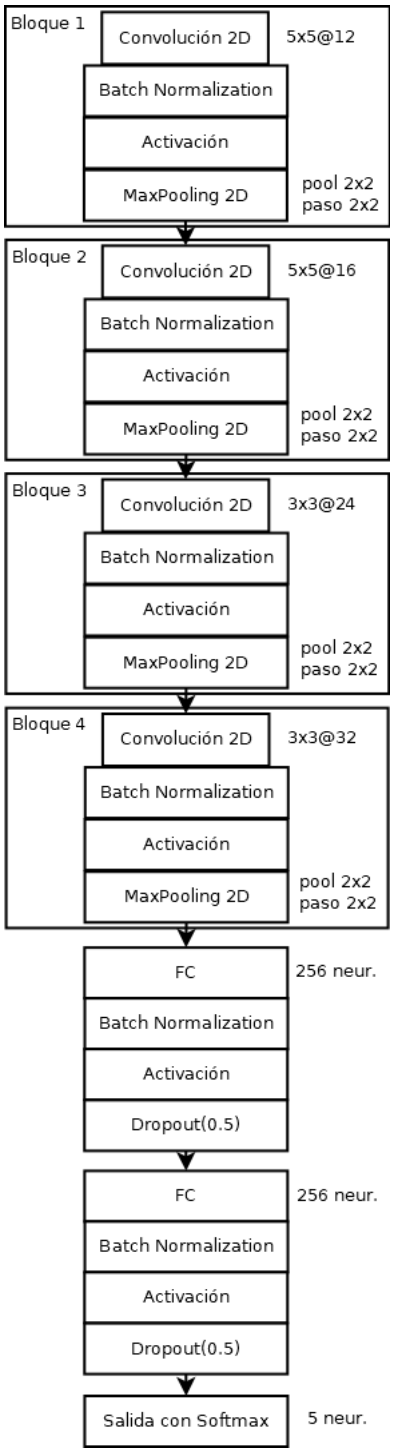


Figura D.4: Modelo 12.

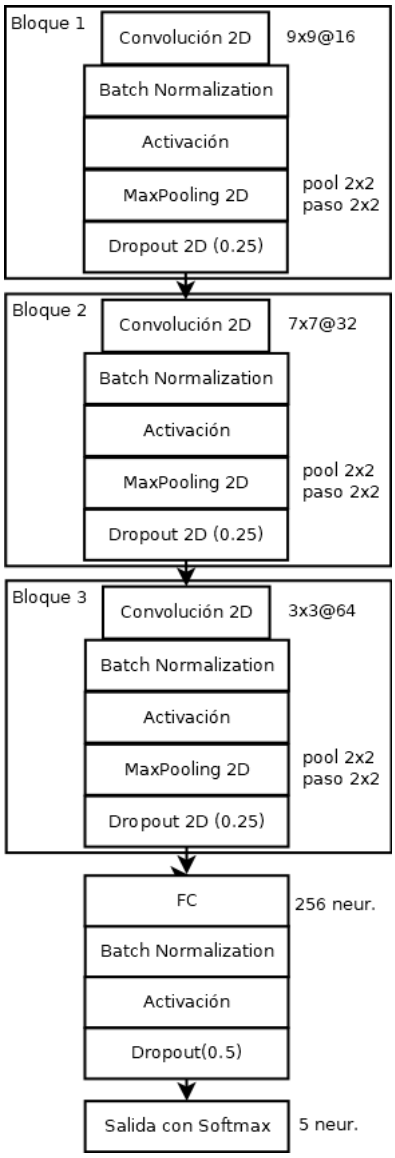


Figura D.5: Modelo 13.

Lista de acrónimos

PSG *Polisomnografía.*

EEG *Electroencefalograma.*

EOG *Electrooculograma.*

EMG *Electromiograma.*

REM *Rapid Eye Movement.*

NREM *No Rapid Eye Movement.*

MOR (traducción al español de REM) *Movimiento ocular rápido.*

NMOR (traducción al español de NREM) *Sin movimiento ocular rápido.*

AASM *American Academy of Sleep Medicine. En español, Academia Americana de Medicina del Sueño*

IA *Inteligencia Artificial*

DL *Deep Learning. En español, Aprendizaje Profundo*

FC *Capa Fully-connected. En español, Capa totalmente conectada*

RNA *Red de Neuronas Artificial*

CNN *Convolutional Neural Network. En español, Red Neuronal Convolucional*

k-NN *K-Nearest Neighbors. En español, K-Vecinos más Cercanos*

DBN *Deep Belief Nets. En español, Redes de Creencia Profunda*

HMM *Hidden Markov Models. En español, Modelos Ocultos de Markov*

PNN *Probabilistic Neural Net. En español, Red Neuronal Probabilística*

FFN *Feed-forward Net. En español, Red Neuronal Prealimentada*

RNN *Recurrent Neural Network. En español, Red Neuronal Recurrente*

LSTM *Long Short-term Memory*

SVM *Support Vector Machine*

MF1 *Macro F1-Score*

ReLU *Rectifier Linear Unit. En español, Unidad Rectificadora Lineal*

GPU *Graphics Processing Unit. En español, Unidad de Procesamiento Gráfico*

Glosario

Orientación a comandos (scripting, en inglés). Pequeño programa de software que suele ejecutarse mediante un intérprete.

Depurador. Programa que se usa para facilitar la eliminación de los errores de programación.

Epoch de una señal. Segmento de una señal con una duración determinada.

Ciclo de entrenamiento (epoch, en inglés). Iteración completa sobre el conjunto de datos de entrenamiento.

Bibliografía

[1] R. Stickgold, "Sleep-dependent memory consolidation," *Nature*, vol. 437, no. 7063, p. 1272—1278, October 2005. [En línea]. Disponible en: https://doi.org/10.1038/nature04286

[2] K. L. Knutson, K. Spiegel, P. Penev, and E. V. Cauter, "The metabolic consequences of sleep deprivation," *Sleep Medicine Reviews*, vol. 11, no. 3, pp. 163 – 178, 2007. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S1087079207000202

[3] C. Novo-Olivas, L. Chacón Guitiérrez, and J. Alberto Barradas Bribiesca, *Mapeo Electroencefalográfico y Neurofeedback*, 2010, pp. 371–412.

[4] M. B. Reaz, M. S. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: Detection, processing, classification and applications," *Biological Procedures Online*, vol. 8, no. 1, pp. 11–35, 2006.

[5] A. Rechtschaffen and A. Kales, *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects.* Bethesda, Md., U. S. National Institute of Neurological Diseases and Blindness, Neurological Information Network, 1968.

[6] C. Iber, S. Ancoli-Israel, A. Chesson, and S. Quan, "The AASM manual for the scoring of sleep and associated events: Rules, terminology and technical specifications," *Westchester, IL: American Academy of Sleep Medicine*, 01 2007.

[7] E. Hartmann, "The 90-minute sleep-dream cycle," *Archives of general psychiatry*, vol. 18, pp. 280–6, 04 1968.

[8] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[9] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, pp. 65–386, 1958.

[10] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, p. 85–117, Jan 2015. [En línea]. Disponible en: http://dx.doi.org/10.1016/j.neunet.2014.09.003

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [En línea]. Disponible en: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[13] T. M. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[14] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[16] S. Ruder, "An overview of gradient descent optimization algorithms," 2016.

[17] H. Danker-Hopfe, P. Anderer, J. Zeitlhofer, M. Boeck, H. Dorn, G. Gruber, E. Heller, E. Loretz, D. Moser, S. Parapatics, B. Saletu, A. Schmidt, and G. Dorffner, "Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard," *Journal of Sleep Research*, vol. 18, no. 1, pp. 74–84, mar 2009. [En línea]. Disponible en: http://doi.wiley.com/10.1111/j.1365-2869.2008.00700.x

[18] S. Gudmundsson, T. Runarsson, and S. Sigurdsson, "Automatic Sleep Staging using Support Vector Machines with Posterior Probability Estimates," pp. 366–372, 2006.

[19] M. Längkvist, L. Karlsson, and A. Loutfi, "Sleep Stage Classification Using Unsupervised Feature Learning," *Advances in Artificial Neural Systems*, vol. 2012, pp. 1–9, 2012.

[20] Y. L. Hsu, Y. T. Yang, J. S. Wang, and C. Y. Hsu, "Automatic sleep stage recurrent neural classifier using energy features of EEG signals," *Neurocomputing*, vol. 104, pp. 105–114, 2013. [En línea]. Disponible en: http://dx.doi.org/10.1016/j.neucom.2012.11.003

[21] B. Kemp, "The Sleep-EDF database," accessed: 2019-07-30. [En línea]. Disponible en: https://www.physionet.org/physiobank/database/sleep-edf/

[22] A. R. Hassan and M. I. H. Bhuiyan, "Computer-aided sleep staging using Complete Ensemble Empirical Mode Decomposition with Adaptive Noise and bootstrap

aggregating," *Biomedical Signal Processing and Control*, vol. 24, pp. 1–10, 2016. [En línea]. Disponible en: http://dx.doi.org/10.1016/j.bspc.2015.09.002

[23] A. Supratak, H. Dong, C. Wu, and Y. Guo, "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.

[24] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.

[25] "PYPL popularity of Programming Language," accessed: 2019-12-12. [En línea]. Disponible en: https://pypl.github.io/PYPL.html

[26] "PyEDFlib edf/bdf toolbox in python," accessed: 2019-11-22. [En línea]. Disponible en: https://pyedflib.readthedocs.io/en/latest/

[27] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[28] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in science & engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[29] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[31] F. Chollet *et al.*, "Keras," 2015. [En línea]. Disponible en: https://keras.io

[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[33] Intel, "Plaidml," 2017. [En línea]. Disponible en: https://github.com/plaidml/plaidml

[34] Anaconda, "Anaconda," 2012. [En línea]. Disponible en: https://www.anaconda.com/

[35] P. Raybaut, "Spyder ide," 2009. [En línea]. Disponible en: https://www.spyder-ide.org/

[36] L. Torvalds, "Git," 2005. [En línea]. Disponible en: https://git-scm.com/

[37] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit,

and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000 (June 13), circulation Electronic Pages: http://circ.ahajournals.org/content/101/23/e215.full PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[38] B. Kemp, "The Sleep-EDF extended database," accessed: 2019-08-25. [En línea]. Disponible en: https://www.physionet.org/content/sleep-edfx/1.0.0/

[39] A. Imtiaz and E. Rodriguez-Villegas, "Recommendations for performance assessment of automatic sleep staging algorithms," *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC 2014*, vol. 2014, pp. 5044–7, 08 2014.

[40] J. T. Martínez, *Electroencefalografía clínica básica*, 2005, ch. IV EEG Normal.

[41] A. Quintero-Rincon, S. Liberczuk, and M. Risk, "Preprocesamiento de eeg con filtros hampel," vol. 89, 2012.

[42] J. Brownlee, *Deep Learning With Python.* Machine Learning Mastery, eBook. [En línea]. Disponible en: https://machinelearningmastery.com/deep-learning-with-python/

[43] Q. Xu, Y.-Z. Liang, and Y.-P. Du, "Monte carlo cross-validation for selecting a model and estimating the prediction error in multivariate calibration," *Journal of Chemometrics*, vol. 18, pp. 112 – 120, 02 2004.

[44] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," 2013.

[45] R. Kotikalapudi and contributors, "keras-vis," 2017. [En línea]. Disponible en: https://github.com/raghakot/keras-vis

[46] A. R. Hassan, S. K. Bashar, and M. I. H. Bhuiyan, "On the classification of sleep states by means of statistical and spectral features from single channel electroencephalogram," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug 2015, pp. 2238–2243.

[47] O. Tsinalis, P. M. Matthews, and Y. Guo, "Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders," *Annals of Biomedical Engineering*, vol. 44, no. 5, pp. 1587–1597, 2016. [En línea]. Disponible en: https://doi.org/10.1007/s10439-015-1444-y

[48] J. Virkkala, J. Hasan, A. Värri, S.-L. Himanen, and K. Müller, "Automatic sleep stage classification using two-channel electro-oculography," *Journal of Neuroscience*

Methods, vol. 166, no. 1, pp. 109 – 115, 2007. [En línea]. Disponible en: http://www.sciencedirect.com/science/article/pii/S0165027007003044