UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
ESCUELA DE POSTGRADO Y EDUCACIÓN CONTINUA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

MODELOS DE ATENCIÓN NEURONAL PARA LA REPRESENTACIÓN DE ELECTROENCEFALOGRAMAS Y LA DETECCIÓN DE EVENTOS TRANSITORIOS DEL SUEÑO

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIA DE DATOS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

ALONSO IGNACIO VARGAS ESTAY

PROFESOR GUÍA:
PABLO ESTÉVEZ VALENCIA

MIEMBROS DE LA COMISIÓN:
CLAUDIO PEREZ FLORES
RODRIGO SALAS FUENTES

Este trabajo ha sido parcialmente financiado por:
ANID – Programa Iniciativa Científica Milenio – ICN2021_004 y Fondecyt 1220829

SANTIAGO DE CHILE
2023

RESUMEN DE LA TESIS PARA OPTAR AL
GRADO DE MAGÍSTER EN CIENCIA DE DATOS
Y DE LA MEMORIA PARA OPTAR AL TÍTULO
DE INGENIERO CIVIL ELÉCTRICO
POR: ALONSO IGNACIO VARGAS ESTAY
FECHA: 2023
PROF. GUÍA: PABLO ESTÉVEZ VALENCIA

MODELOS DE ATENCIÓN NEURONAL PARA LA REPRESENTACIÓN DE ELECTROENCEFALOGRAMAS Y LA DETECCIÓN DE EVENTOS TRANSITORIOS DEL SUEÑO

Los eventos transitorios del sueño, como los husos de sueño y los complejos-k, están asociados a diferentes funciones cognitivas y al diagnóstico de diferentes enfermedades neurológicas. Sin embargo, la detección de estos eventos implica una inversión importante en capital humano experto y de tiempo para generar el etiquetado de estos eventos. Por lo tanto, existe una motivación inherente a desarrollar detectores automáticos que tengan la capacidad aprender de datos de Electro Encefalogramas (EEG) con eventos transitorios del sueño.

El modelo propuesto para desarrollar el detector automático es un modelo basado en aprendizaje profundo, particularmente en atención neuronal. Los modelos de atención neuronal (*Transformers*) han tenido gran éxito en el área de procesamiento de lenguaje natural y, más recientemente, en el procesamiento de imágenes. Los *Transformers* pueden aprender sobre el contexto de los datos no etiquetados previo a las etiquetas mismas, mejorando así su desempeño. En este trabajo se proponen variaciones importantes en la arquitectura del *Transformer*, agregando operaciones convolucionales que ayudan a llevar a cabo el procesamiento de las señales EEG, las que a diferencia de las secuencias de texto, presentan un largo considerablemente mayor. Además, como aprendizaje previo a la tarea de detección, se utiliza una tarea de contraste para que así el *Transfomer* aprenda a contextualizar el EEG con mejores representaciones iniciales. El aprendizaje por contraste resultó tener una mejora en el F1-score de detecciones de eventos transitorios, permitiendo además tener desempeños que alcanzan y en algunos casos superan el estado del arte.

Dedicado a toda mi familia y especialmente a mis hermanos.

Pero sobre todo ... a mi madre

Agradecimientos

Agradezco a mi profesor guía, el Dr. Pablo Estévez por creer en mí y en mis decisiones a la hora de abordar el problema que se trabaja en esta tesis. Le agradezco por darme acceso a un computador con GPU en el laboratorio de Inteligencia Computacional del Departamento de Ingeniería Eléctrica (DIE), el que ha sido realmente de mucha ayuda. También le agradezco por darme acceso al servidor del laboratorio el cual posee componentes de alto rendimiento para el entrenamiento de modelos de gran tamaño y que han sido imprescindibles a la hora de entrenar los modelos desarrollados en este trabajo de tesis. Quiero agradecer también a Nicolás Tapia, quien fue alumno del Dr. Estévez y quien trabajó en la misma tarea que se aborda en esta tesis. Él ha sido un gran apoyo a lo largo del desarrollo de mi trabajo, ayudándome a entender el trasfondo del problema en cuestión y prácticas que se aplican en esta tarea en particular, entre otras cosas.

Agradezco a la Agencia Nacional de Investigación y Desarrollo (ANID) por financiar parcialmente este trabajo a través del proyecto FONDECYT - 1220829. También agradezco al Laboratorio Nacional de Computación de Alto Rendimiento o NLHPC, por poner a mi disposición una cuenta con 150.000 horas de cómputo por 12 meses, 88 cores y 500Gb de almacenamiento en sus máquinas de alto rendimiento para poder entrenar los modelos desarrollados en este trabajo de tesis con mayor rapidez.

De manera más personal, quiero agradecer a mi profesor de redes neuronales en el curso de *Deep Learning* en la FCFM, el Dr. Jorge Pérez, quien despertó en mí el gran interés que tengo por los modelos neuronales y quien me aceptó como alumno en un trabajo dirigido utilizando modelos neuronales, lo que me devolvió la confianza que necesitaba para seguir desarrollándome en el área de aprendizaje de máquinas. Quiero agradecer a mi madre y a mi hermano, quienes siempre me apoyaron y me entendieron cuando más lo necesité y que estuvieron ahí conmigo en los momentos más difíciles. Finalmente quiero agradecer a mi pareja y compañera, Patricia, quien le devolvió la alegría a mis días después de una época personalmente muy difícil, quien ha sido mi confidente siempre que la he necesitado y quien me ha brindado un apoyo incondicional.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

El cerebro y su comportamiento siempre han sido de gran interés en el mundo de la ciencia, particularmente, de la neurociencia. Hoy se sabe que el comportamiento del cerebro determina las capacidades cognitivas, de memorización y de aprendizaje de las personas, además de ciertos desórdenes y enfermedades neurológicas como la esquizofrenia o la epilepsia. Una forma en la que la neurociencia puede estudiar el comportamiento del cerebro es mediante los llamados electroencefalogramas (EEG) y el estudio de ciertos patrones identificables en estas señales. Estos patrones identificables deben ser anotados, ya sea por un experto o por un detector automático. Sin embargo, la extensión de estas señales de estudio y cantidad de estos patrones es tan grande que identificarlos requiere de mucho tiempo y recursos expertos que se desperdician en tareas repetitivas y tediosas.

Los patrones o eventos mencionados arriba, son la razón de estudio de esta tesis y la motivación principal es desarrollar un detector automático que pueda detectar y localizar con precisión estos eventos. Los eventos en cuestión son eventos transitorios del sueño, más específicamente, los husos de sueño y los complejos-k. Para investigar estos eventos se estudia un conjunto de señales fisiológicas llamado polisomnograma (PSG), entre las cuales se encuentra el EEG, que es la señal de interés en este trabajo para detectar los eventos transitorios del sueño. En particular, ambos patrones se quieren detectar en la fase N2 del sueño, la cual, es la fase más duradera y libre de ruido y de artefactos del sueño.

1.2. Identificación y formulación del problema

Como ya se mencionó, la detección de estos eventos es de suma importancia para el análisis o diagnóstico de pacientes a nivel neurológico. De la figura 1.1 –que muestra un segmento de un canal de EEG– se puede inferir que el trabajo necesario para anotar los eventos es extenso y arduo, necesitando el tiempo y la opinión de expertos en la materia. Como se puede apreciar en la figura 1.1, en tan solo 20 segundos de la etapa N2 (en este segmento) hay 5 eventos: 4 husos de sueño y 1 complejo K. Además, la etapa N2 comprende alrededor de 4 horas de señal para solamente una noche de un solo sujeto. Dado esto último, es evidente la necesidad de detectores automáticos que puedan agilizar la identificación de estos eventos para poder concentrar los recursos expertos humanos en el análisis de éstos y los diagnósticos pertinentes.

Lamentablemente, los detectores automáticos que tienen mejor desempeño hoy en día sufren las consecuencias del mismo problema que pretenden resolver, la necesidad de muchos datos etiquetados para poder entrenar estos modelos de detección automática. Como ya se mencionó, generar bases de datos con eventos marcados de etapas N2 de EEG supone un trabajo que involucra múltiples expertos y varios meses de trabajo en conjunto, lo que aún así deja una enorme cantidad de datos no etiquetada disponible. Por lo tanto, esta tesis propone atacar el problema con un tipo de red neuronal llamado *Transformer*, el cual ha demostrado poder extraer información valiosa de datos no etiquetados en tareas de *natural language processing* (NLP), procesamiento de imágenes y de sonido.

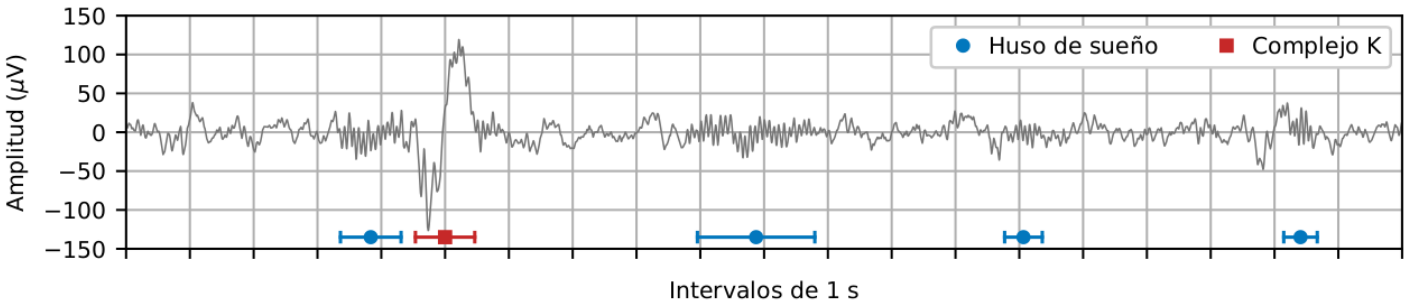

Figura 1.1: 20 segundos de un EEG en la fase N2 del sueño con eventos marcados de husos de sueño y complejos-k

Parte del desafío inherente del presente trabajo es lograr adaptar el diseño del *Transformer* para poder procesar señales univariadas (o eventualmente multivariadas) y aprovechar su arquitectura para extraer información de datos que vienen en forma de series temporales. Esto es un gran desafío ya que la arquitectura del *Transformer*, si bien es simple conceptualmente, en la práctica se traduce en modelos de muy alta complejidad, con muchas posibles variaciones, que no fueron pensados para procesar secuencias de largo de varios ordenes de magnitud. Para lograr esto, se proponen distintas variaciones a la arquitectura y pre-entrenamiento de los *transformers* utilizados en el área de procesamiento de lenguaje.

1.3. Hipótesis

La hipótesis principal del presente trabajo es que el *Transformer* es capaz de aprender a representar las series de tiempo provenientes del EEG en forma de series de vectores de características contextualizadas, mejorando la detección localización de husos del sueño y complejos-k con respecto al estado del arte. Esta hipótesis se basa en el supuesto de que el *Transformer* es capaz de aprender a extraer características generales en cada punto o zona del EEG que dependerían del contexto. Es importante que esto pueda aprenderlo mediante aprendizaje semi-supervisado para poder aprovechar la enorme cantidad de datos no-etiquetados, lo cual significaría un gran avance en la representación de series temporales extensas como lo son las señales de EEG. Si las características aprendidas son lo suficientemente útiles y generales, este modelo podría re-entrenarse (*fine-tuning*) en datos etiquetados para resolver un problema particular como lo es la detección de eventos del sueño. Cabe destacar que esto podría extenderse a otros problemas que puedan utilizar y beneficiarse de representaciones de EEG.

1.4. Objetivo general

El objetivo general del trabajo de tesis es desarrollar un detector de husos de sueño y complejos-k (eventos) durante la fase NREM 2 del sueño. El diseño de este detector está basado en redes neuronales profundas, particularmente en el diseño de un *transformer* especialmente adaptado para el procesamiento de EEG.

1.5. Objetivos específicos

Para cumplir con el objetivo general, se han propuesto los siguientes objetivos específicos que incluyen y complementan el objetivo principal del trabajo de tesis. Cabe destacar, que estos objetivos fueron cambiando durante el transcurso del trabajo de tesis a medida que se descartaban e incluían ciertas hipótesis respecto al entrenamiento de un *transformer* para el procesamiento de EEG.

- Diseñar una arquitectura basada en un *transformer* que esté adaptada para el procesamiento del EEG.
- Evaluar la efectividad de los *transformers* diseñados en distintas bases de datos de EEG etiquetados con eventos transitorios del sueño.
- Comparar distintos diseños y entrenamientos de *transformers*, abordando distintas funciones de pérdida, arquitecturas de atención neuronal y transferencias de aprendizaje.
- Interpretar las detecciones resultantes entregadas por el *transformer* diseñado para diferentes EEG, analizando las vecindades donde se pone atención en las diferentes capas del *transformer*.
- Comparar el desempeño de los *transformers* en la detección de eventos del sueño con modelos del estado del arte basados en redes neuronales que utilizan capas convolucionales y LSTM para la detección.

1.6. Organización del trabajo

Capítulo 2

En el capítulo 2 se presentan los antecedentes necesarios para comprender el modelo propuesto en este trabajo de tesis. Primero se presenta el electroencefalograma (EEG) del sueño. Se describen los eventos transitorios del sueño y las fases del sueño. Luego, se enuncian de forma general las redes neuronales y las distintas arquitecturas que se han desarrollado a lo largo del tiempo, incluyendo los *transformers* en los cuales se inspiró el modelo propuesto. El capítulo termina con una revisión general sobre los distintos detectores de eventos transitorios del sueño en la literatura.

Capítulo 3

En el Capítulo 3 se presentan los procedimientos de procesamiento de datos utilizados, el modelo propuesto y los métodos de evaluación utilizados. Primero se presentan las bases de

datos utilizadas para pre-entrenamiento y para el ajuste fino. Luego se presentan la forma de particionar las bases de datos, lo cual es necesario para comprender como se evalúan los modelos que se comparan en este trabajo de tesis. Para terminar, se presentan las métricas de evaluación y cómo estás se aplican a la forma en la que se procesan las detecciones de los modelos que se comparan en este trabajo de tesis.

Capítulo 4

En el capítulo 4 se comienza mostrando los tiempos de computo de los distintos modelos que se comparan en este trabajo de tesis, incluyendo su tiempo de inferencia y de entrenamiento. Luego, se muestran las comparaciones de distintos modelos en cuanto a su desempeño incluyendo el nivel máximo de confianza obtenido de *t-test* de hipótesis de diferencia de medias. Finalmente en el capítulo se presentan los análisis de explicabilidad de las detecciones que facilita el modelo propuesto.

Capítulo 5

En el Capítulo 5 se presentan las conclusiones del trabajo de tesis, las ventajas y limitaciones del modelo propuesto, y las principales contribuciones del trabajo realizado. También se presentan los aprendizajes obtenidos del trabajo y la justificación del cumplimiento de los objetivos planteados. Por último, se presentan distintas hipótesis y propuestas en relación a posibles trabajos futuros.

Capítulo 2

Antecedentes

2.1. Electroencefalogramas del sueño

Un electroencefalograma (EEG) es un examen que sirve para medir la actividad eléctrica del cerebro en forma de series de tiempo de mediciones de voltaje. Este voltaje es del orden de los micro-volt y es producido por la comunicación electroquímica entre las neuronas del cerebro. A través de las sinapsis de estas células se transmiten pulsos de voltaje entre sí a lo largo de sus membranas celulares y axones, activando o inhibiendo conexiones neuronales. Dentro de un tejido de neuronas, estas recepciones y emisiones de pulsos pueden entrar en sincronía, generando así, patrones de oscilaciones eléctricas que son identificables. En el campo de la neurociencia, se cree que estos patrones de oscilaciones eléctricas cumplen un rol esencial en el funcionamiento y comprensión del cerebro. Por lo tanto, caracterizarlas se ha vuelto una tarea fundamental en el área.

Los EEG capturan las señales eléctricas del cerebro mediante electrodos ubicados en el cuero cabelludo de los sujetos. Esto permite capturar diferencias de potencial entre distintos electrodos. Para una señal de voltaje formada por el mismo par de electrodos se tiene un "canal", y dentro de estos canales es posible identificar patrones en las señales de voltaje, los cuales son normalmente identificados y analizados por expertos. Las intensidades absolutas de estas señales rondan en torno a los 100μV, pudiendo llegar incluso hasta los 200μV. La presencia de distintos patrones de oscilación en estas señales da lugar una composición espectral que se ha diferenciado en distintas bandas de frecuencia. Entre las bandas que se pueden encontrar en un EEG son de particular interés en este trabajo la banda delta (0.5-4 Hz) y la banda sigma (11-16 Hz). La figura 2.1 muestra el estándar de los electrodos presentes en un EEG, cómo se identifican y donde se ubican en el cuero cabelludo.

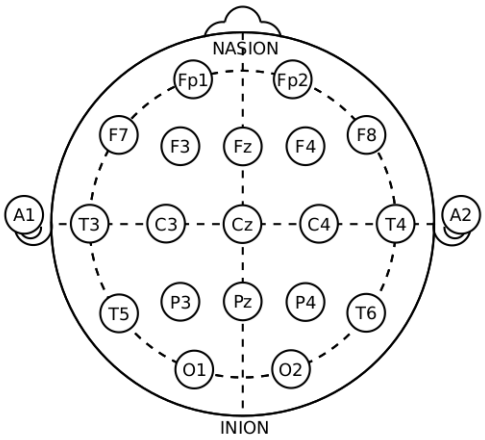

Figura 2.1: Sistema internacional 10-20 para los electrodos del EEG, donde el NASION se ubica a la altura de la nariz y los electrodos A1 y A2 a la altura de las orejas (imagen de dominio público)

2.1.1. Fases del sueño

Las señales de interés en este trabajo de tesis son los EEG durante el sueño, en particular, durante la fase N2 del sueño. Existen 5 fases del sueño: *Wake* (W), *Rapid Eye Movement* (REM), no REM 1 (N1), no REM 2 (N2), no REM 3 (N3) (Carskadon y Dement, 2005). La etapa de interés en este trabajo es la N2, en la cual se desea detectar eventos transitorios ya que es la etapa con menor tono muscular, actividad cerebral y ocular, lo que permite obtener señales de voltaje más "limpias", es decir, sin artefactos externos o ruido. Cabe destacar, además, que la etapa N2 del sueño es la etapa más extensa del sueño, llegando a comprender alrededor del 50 % de la noche total de sueño, teniendo que por noche de sueño se registrarían aproximadamente 4 horas de la señal de interés.

En adultos normales, el sueño suele iniciar con una fase No REM del sueño hasta pasar a la primera fase REM del sueño. Luego, durante la noche, estas fases continúan alternándose mientras la etapa N2 va predominando en el tiempo. En el presente trabajo se trabaja siempre con segmentos de 20 segundos de la etapa N2 del sueño. Este segmento coincide con el tiempo que se recomienda marcar con fases de sueño según Wolpert (1969).

2.1.2. Eventos transitorios del sueño

Dentro de las señales capturadas en los EEG se pueden reconocer ciertos eventos particulares durante la etapa N2 (y parcialmente durante la N3), llamados husos de sueño y complejos-k. A estos eventos se les llama eventos transitorios del sueño y su ocurrencia durante el sueño es fundamental, dado que, la etapa N2 del sueño abarca aproximadamente la mitad de la noche de sueño. Estos eventos forman patrones de actividad eléctrica que se han vuelto objetos de estudio importantes para identificar desórdenes neurológicos del sueño y ciertas patologías. La figura 1.1 muestra un canal de un EEG en la que identifica estos eventos.

Los husos de sueño se destacan del resto de la señal del EEG, dado que dentro de la señal hay pequeños segmentos con frecuencias entre 11 y 16 Hz (Berry, Albertario, Harding, Lloyd, y Plante, 2018). Esta banda de frecuencia es la banda sigma y en esta banda se pueden

encontrar todos los husos de sueño presentes en el EEG, aunque esto no quiere decir que toda la banda sigma de la señal EEG sean husos de sueño. Este evento suele tener en general una duración entre 0.5 y 1.2 segundos, pero puede llegar a salirse de esos rangos levemente según Stern (2005). Los husos de sueño han sido asociados a los procesos de aprendizaje y de memorización (Clawson, Durkin, y Aton, 2016) y su actividad anormal ha sido vinculada con enfermedades como la esquizofrenia (Fernandez y Lüthi, 2020).

En cuanto a los complejos-k, lo que los define es que estos son eventos de baja frecuencia pero de gran amplitud con respecto al resto del EEG. La banda en la que estos eventos se localizan es principalmente la banda delta (0.5-4 Hz), suelen tener una duración entre 0.5 y 1 segundos y su amplitud varía entre los 100 y 400 μV, todo esto según Stern (2005). La actividad anormal de los complejos-k ha sido asociada a patologías neurológicas como la epilepsia (El Helou et al., 2008).

2.2. Redes neuronales

Las redes neuronales son un tipo de modelo basado en el perceptrón de Rosenblatt (1958), que combina operaciones lineales en cascada intercalando estas operaciones con funciones no lineales. Dadas estas condiciones, el Teorema de Aproximación Universal o *Universal Aproximation Theorem* (UAT) dice que es posible aproximar cualquier función continua con tan solo una capa oculta y una neurona de salida. La figura 2.2 muestra la unidad básica de una red neuronal, el perceptrón propuesto por Rosenblatt (1958).

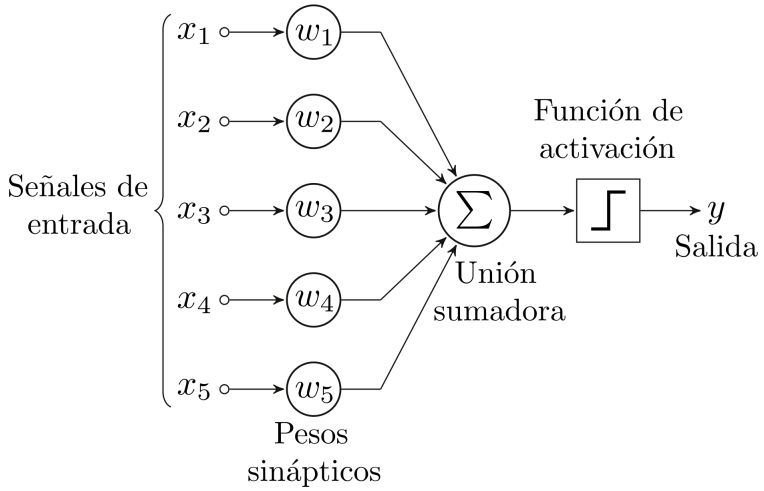

Figura 2.2: Perceptrón de Frank Rosenblatt. https://es.wikipedia.org/wiki/Perceptr%C3%B3n

De la figura 2.2 se puede ver el funcionamiento de un único perceptrón con 5 pesos sinápticos asociados a la entrada de tamaño 5. Luego, en las ecuaciones 2.1 y 2.2 se puede ver que concatenando 3 perceptrones en paralelo, donde cada uno asocia distintos pesos a la misma entrada, se forma una capa lineal junto con una función no lineal (función de activación) a la salida, lo que representa una capa de una red neuronal quedando de la forma $y = f(xW + b)$. Donde cada una de las columnas de la matriz W representa los pesos asociados a una neurona o perceptron, junto con cada uno de los sesgos (bias) asociados a cada una de las 3 neuronas

para este ejemplo.

$$x^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}, \qquad W = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \end{bmatrix}, \qquad b^T = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \tag{2.1}$$

$$y^T = \begin{bmatrix} y_1 = x_1w_{11} + x_2w_{21} + x_3w_{31} + x_4w_{41} + x_5w_{51} + b_1 \\ y_2 = x_1w_{12} + x_2w_{22} + x_3w_{32} + x_4w_{42} + x_5w_{52} + b_2 \\ y_3 = x_1w_{13} + x_2w_{23} + x_3w_{33} + x_4w_{43} + x_5w_{53} + b_3 \end{bmatrix} \tag{2.2}$$

2.2.1. Redes convolucionales

Las redes neuronales convolucionales son un tipo de red neuronal donde las capas de la red son operaciones convolucionales, las cuales se componen por filtros cuyos valores son los pesos de la capa neuronal convolucional. Las operaciones convolucionales pueden ser en distintas dimensiones, por ejemplo, cuando se trata del procesamiento de imágenes –que es el caso para el cual estaban pensados originalmente las redes convolucionales– las convoluciones son en 2 dimensiones dado que el filtro se desplaza a lo largo de la imagen en 2 direcciones. Los primeros acercamientos a este tipo de redes fueron propuestos por Hubel y Wiesel (1959) llamándolas "celdas" y luego Fukushima (1980) quién desarrolló un modelo neuronal adaptado al reconocimiento de patrones.

Posteriormente, Lecun, Bottou, Bengio, y Haffner (1998) propusieron una de las primeras arquitecturas de red convolucional más moderna y cuya investigación fue de las más citadas de la década. Sin embargo, no fue hasta que Krizhevsky, Sutskever, y Hinton (2012) propusieron una arquitectura basada puramente en redes neuronales convolucionales para competir en la clasificación de ImageNet (Deng et al., 2009) que resurgió el interés en la comunidad científica por el aprendizaje profundo de redes neuronales.

La figura 2.3 muestra una capa convolucional de una red neuronal con dos filtros de 3x3 en una imagen de entrada con 3 canales, lo que resulta en una nueva imagen con menor dimensión espacial y con dos canales dado que las salidas de los filtros se concatenan. La operación $*$ en la figura 2.3, es la operación de convolución en dos dimensiones con un paso de 1 píxel y sin *zero padding*, lo que ocasiona una disminución de 2 píxeles en la resolución de la imagen (uno por cada lado) en cada una de las dimensiones espaciales, resultando en una imagen de 4x4. Finalmente, si se volviera a aplicar una capa convolucional, cada uno de los filtros tendría tantos canales como la imagen, en este caso cada filtro de 3x3 tendría dos canales, teniendo que cada filtro en esta nueva capa hipotética tendría $3 * 3 * 2 = 18$ pesos o parámetros distintos.

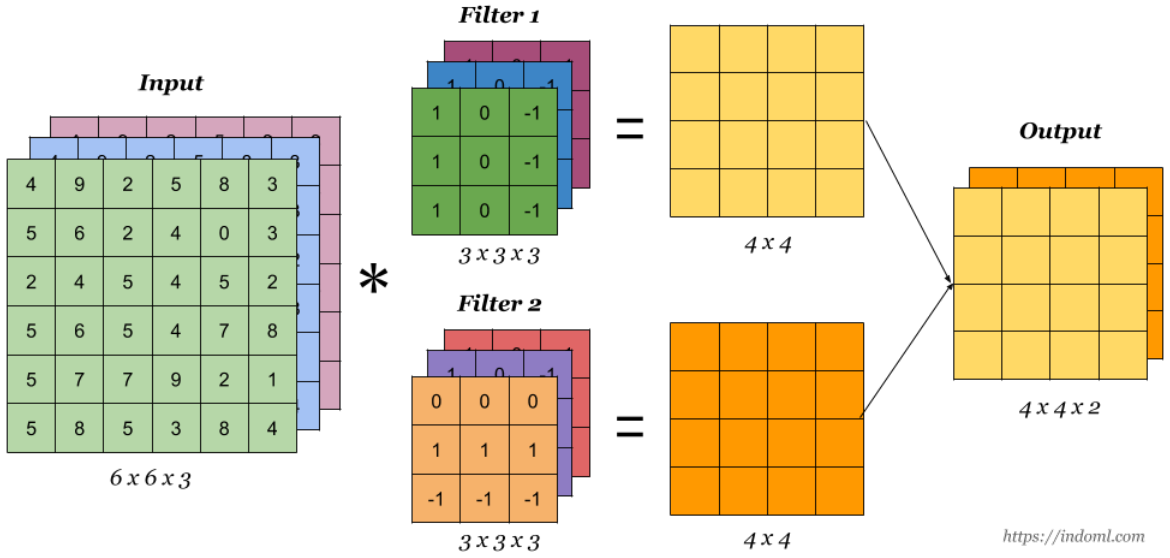

Figura 2.3: Ejemplo de operaciones convolucionales con 2 filtros. https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

2.2.2. Redes recurrentes

Las redes recurrentes son un tipo de red neuronal que puede asimilarse a lo que es una máquina de estados. Esto porque las redes recurrentes están diseñadas para procesar secuencias y fueron introducidas como tal por Rumelhart y McClelland (1985). La red recurrente más básica se compone de dos capas lineales con la tangente hiperbólica como función de activación, siendo estas mismas capas las que procesan cada estado de la secuencia de manera recurrente. Cada valor de la secuencia entra como input a una misma capa lineal (misma para todos los inputs) y el estado anterior entra a la otra capa lineal (la misma para todos los estados), cuyos resultados sumados pasan por la función de activación, lo que definiría el estado actual. Esto permite procesar secuencias teniendo representaciones vectoriales de la secuencia desde el inicio hasta cada uno de los puntos. La figura 2.4 muestra esquemáticamente el funcionamiento de una red recurrente básica, donde el bloque **A** representa las mismas capas lineales con la tangente hiperbólica como función de activación. Este bloque recurrente procesa una secuencia de entradas x_t, generando una secuencia de salidas h_t.

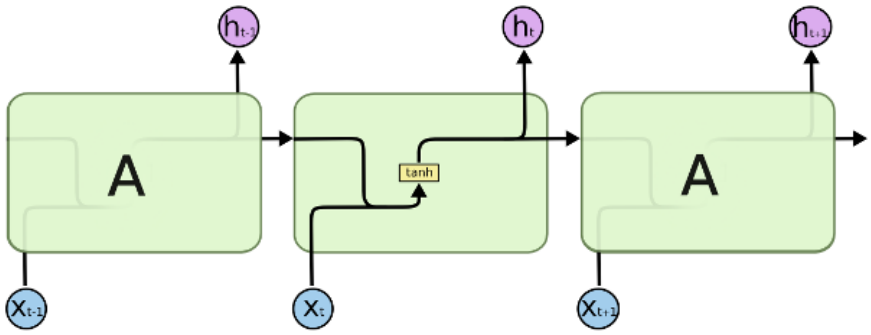

Figura 2.4: Esquema de una RNN básica. https://www.codeproject.com/Articles/1272354/ANNT-Recurrent-neural-networks

La arquitectura de la red recurrente mostrada en la figura 2.4 presenta dos problemas para secuencias que comiencen a ser más extensas. En primer lugar, está el problema del desvanecimiento del gradiente el cual es típico en arquitecturas donde la función de activación es la tangente hiperbólica. Esto es porque para el cálculo de los gradientes en el *backpropagation* existen multiplicaciones de salidas de la tangente hiperbólica con salidas de la derivada de esta misma función, teniendo que para valores lejanos al cero la derivada es cercana a cero y para valores cercanos al cero la multiplicación también se va a cero. Por lo que, luego de varias multiplicaciones, los valores de entrada de las secuencias en general casi no afectan en el entrenamiento de los parámetros de la red recurrente. En segundo lugar, la información que se aprende para valores del comienzo de la secuencia se va perdiendo a medida que se modifica el estado, por lo que si, por ejemplo, se toma el último estado para clasificar la secuencia, la información extraída del comienzo de la secuencia prácticamente no influye en nada.

Para corregir los problemas mencionados, surge una nueva arquitectura de red recurrente llamada *Long Short Term Memory* (LSTM) propuesta por Hochreiter y Schmidhuber (1997). Esta arquitectura de red añade un estado adicional llamado "celda de memoria" y una serie de compuertas con más capas lineales que controlan la lectura y escritura de la celda, además de la opción de olvido. Estos componentes añadidos a la arquitectura base de la red recurrente mitigan en gran medida los problemas mencionados arriba. La figura 2.5 muestra un esquema del funcionamiento de la red LSTM, donde el bloque **A** representa las mismas capas lineales (pesos compartidos) que representa la red LSTM. Finalmente, al igual que en el caso de la red recurrente básica, la red LSTM procesa una secuencia de entradas x_t de manera recurrente, generando una secuencia de salidas h_t.

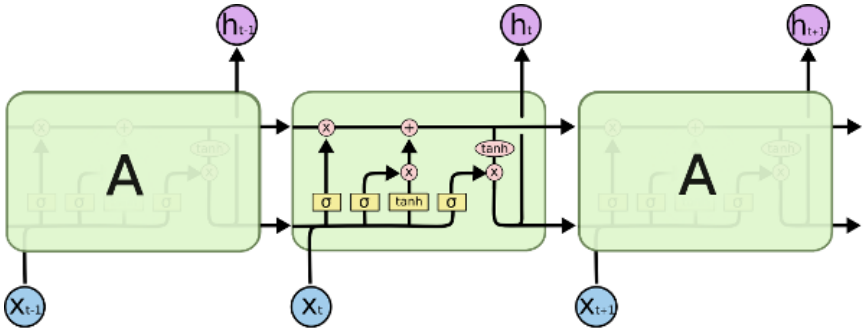

Figura 2.5: Esquema de una LSTM. https://www.codeproject.com/Articles/1272354/ANNT-Recurrent-neural-networks

2.2.3. Atención neuronal

En esta sección se explica la unidad básica del *Transformer*, que es el tipo de modelo que se propone en este trabajo. El *Transformer*, propuesto por primera vez por Vaswani et al. (2017), se basa principalmente en la idea de "Atención Neuronal", la que propone procesar secuencias en donde las conexiones neuronales determinan en qué partes de la secuencia se va a extraer más información o se va a poner "atención". La idea de "Atención Neuronal" fue propuesta de manera más concreta por primera vez por Bahdanau, Cho, y Bengio (2015) y ésta se basa en que cada punto de una secuencia genera ponderadores sobre todo el resto de la secuencia (salida de *softmax* en figura 2.6). Son los ponderadores más altos los que

determinan dónde se quiere poner más atención.

Una consecuencia directa de la Atención Neuronal es que el *Transformer* se basa en una arquitectura donde no importa que tan lejos esté un valor en la secuencia, siempre se puede poner atención sobre ese valor. Esto elimina completamente el segundo problema mencionado de las redes recurrentes y que, además, las conexiones neuronales que determina la atención no son fijas como en el resto de las redes neuronales, es decir, que los ponderadores dependen de cada secuencia.

La forma en que la atención neuronal procesa una secuencia es mediante representaciones vectoriales para cada punto de la secuencia. Una secuencia puede ser vectorizada de diferentes maneras, pero suponiendo que cada valor de la secuencia tiene un vector que la represente, cada uno de estos vectores puede verse como una columna de una matriz que representa la secuencia completa. En un *Transformer*, se pasa esta matriz por 3 capas lineales paralelas, lo que resulta en 3 matrices diferentes llamadas Q (*querys*), K (*keys*) y V (*values*).

En la literatura, suele llamarse auto-atención cuando se tiene que a partir de una misma secuencia se forman las matrices de *querys*, *keys* y *values*. Como estaba pensada originalmente por Bahdanau et al. (2015), cuando las *querys* vienen de una secuencia distinta se llama solamente Atención Neuronal. En el proceso de atención (o auto-atención), las *querys* y las *keys* se multiplican para formar un *score* en relación a las *keys* (cuya matriz de secuencia es siempre la misma que la de los *values*), para así calcular una ponderación distribuida mediante la función *softmax* y, finalmente, ponderar los *values*. Este proceso se ilustra en la figura 2.6.

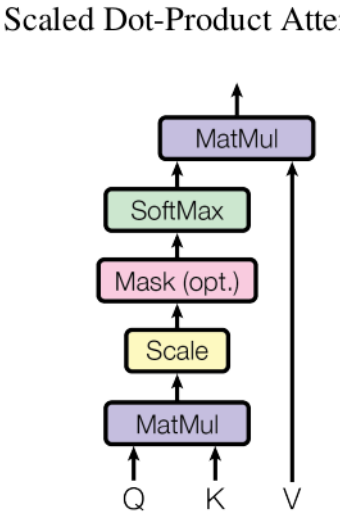

Figura 2.6: Esquema del proceso de Atención Neuronal. Imagen extraída de publicación del *Transformer* original de Vaswani et al. (2017).

En una secuencia dada, puede que quizás haya más de un valor en la secuencia donde se quiera poner atención pero, al usar la *softmax*, se está forzando a que sea solo un valor el que tome importancia en la secuencia, lo cual puede ser problemático si es que en realidad existe más de un valor en la secuencia que sea importante. Es por esto que el *Transformer*

utiliza, además, la Atención Multicabezal (o *Multi-Head Attention*), proceso en el cual simplemente se realizan múltiples atenciones en paralelo y en donde las matrices resultantes de los diferentes cabezales de atención se concatenan en el sentido de las características, para luego reducir la dimensionalidad mediante una capa lineal. Este proceso se esquematiza en la figura 2.7.

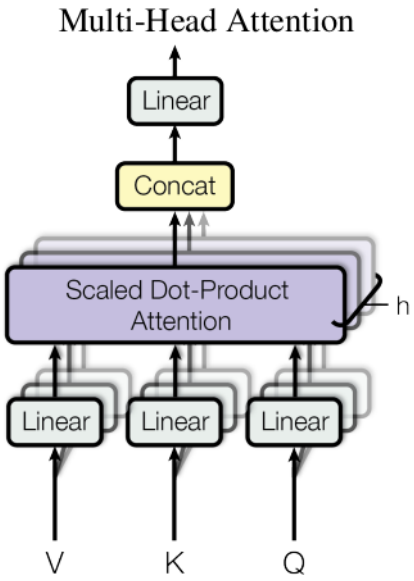

Figura 2.7: Esquema del proceso de Atención Multicabezal con h cabezales. Imagen extraída de publicación del *Transformer* original de Vaswani et al. (2017).

Esta arquitectura ha demostrado ser sumamente poderosa, superando el estado del arte en prácticamente todas las tareas de *Natural Language Processing* (NLP) y, más recientemente, en tareas de procesamiento de imágenes y de señales de audio. Esto se debe a su gran capacidad de aprender representaciones contextualizadas de las secuencias sobre las que se entrena, y también, a su gran capacidad de paralelización en entrenamiento. Esto permite entrenar modelos gigantescos con grandes volúmenes de datos en tiempos razonables. Además, esta arquitectura posee los beneficios inherentes de la Atención Neuronal para conectar información lejana en secuencias más extensas. Finalmente, cabe mencionar que se demostró por Pérez, Marinković, y Barceló (2019) que la Atención Neuronal es Turing completa, lo que significa que es posible simular una máquina de Turing mediante solo mecanismos de atención.

2.2.4. Transformer original

El *Transformer* fue pensado originalmente por Vaswani et al. (2017) para procesar texto (secuencias de palabras), en particular, para traducción automática. Este modelo es de tipo "Seq2Seq", lo que significa que el modelo se divide en dos partes. Una parte de la red está hecha para procesar una secuencia completa la cual se quiere traducir (el *encoder*) y la otra parte de la red va generando la secuencia de salida palabra por palabra, que es la frase traducida (el *decoder*). La figura 2.8 muestra la arquitectura de un solo bloque de atención

multicabezal del *Transformer* original.

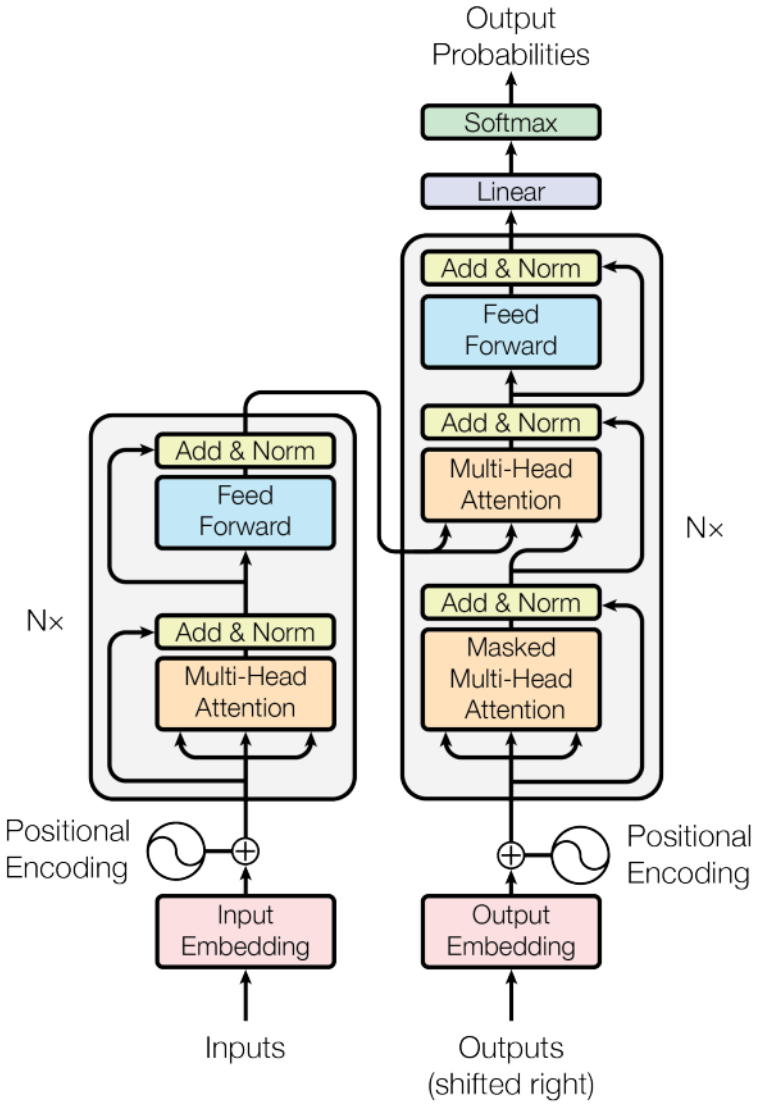

Figura 2.8: Arquitectura del *Transformer* original. Imagen extraída de publicación del *Transformer* original de Vaswani et al. (2017).

La figura 2.8 muestra que en el *decoder* que después de la capa de auto-atención, hay una capa de atención. En ésta, el *decoder* pone atención sobre el *encoder* para incluir la información contextual que aprendió en éste. La forma en la que se realiza esta atención es que la salida de la capa de auto-atención del *decoder* corresponden a las *querys* que ponen atención sobre la salida del *encoder* que pasan a ser los *keys* y los *values*. Además de lo anterior, en la arquitectura se pueden ver unos módulos llamados "*Feed Forward*", los cuales son redes neuronales de 2 capas lineales con una función de activación *Rectified Linear Unit* (ReLU) entre ellas. Finalmente, notar que después de cada capa de atención y de cada capa *Feed Forward* hay conexiones residuales "*Add & Norm*" donde se suman las entradas de cada módulo a su salida y se normalizan. Esto último es equivalente a las conexiones residuales propuestas por He, Zhang, Ren, y Sun (2016).

Por último, se puede ver que en el proceso de Atención Neuronal no existe un orden en los valores de las secuencias, esto porque si una de las columnas que representa la secuencia se intercambiara de lugar con otra, matemáticamente no habría diferencia en el resultado de esa columna. Esto significa que la posición de los valores de la secuencia se pierde, a diferencia de lo que pasa en las redes recurrentes –en las cuales por su estado recurrente– existe un orden intrínseco. Es por esto que en la arquitectura del *Transformer* se agrega la información posicional de los valores de la secuencia, lo cual es llamado "*Positional Encoding*" (ver figura 2.8). Esto consiste en sumar (o concatenar) los valores vectorizados de la secuencia con una capa de funciones seno y coseno de las posiciones numeradas, que puede estar o no parametrizada.

2.2.5. BERT

BERT es un acrónimo para *Bidirectional Encoder Representations from Transformers*, el cual fue desarrollado por Devlin et al. (2019) y, como su nombre sugiere, es solo un *encoder* de lo que sería el *Transformer* original. Este modelo fue diseñado para aprender representaciones a partir de texto no etiquetado en la etapa de pre-entrenamiento. Esta consiste en entregarle secuencias de texto con palabras ocultas para que el modelo pueda predecir qué palabras faltan en los espacios que quedan al ocultar estas palabras. Es por esto, que se dice que BERT es un *Masked Language Model*, ya que en pre-entrenamiento se pone una máscara que cubre un 15 % de las palabras de manera aleatoria para que el modelo trate de completarlas, aprendiendo así representaciones vectoriales contextualizadas de las palabras.

BERT demostró ser sumamente efectivo cuando se entrena en una segunda etapa llamada *fine-tuning*, la cual consiste en entrenar el modelo –después del pre-entrenamiento– en una base de datos etiquetada para resolver alguna tarea particular. Es importante destacar que la máscara de palabras solo se aplica en el pre-entrenamiento, ya que en *fine-tuning* son las etiquetas las que permiten entrenar el modelo realizando predicciones de éstas. La principal ventaja de BERT es su capacidad de aprovecharse de los datos no etiquetados, además de poder entrenarse de manera muy eficiente al ser un *transformer* de tipo *encoder*, por lo que las operaciones en las capas de atención son todas paralelizables. La figura 2.9 muestra un esquema de entrenamiento completo de BERT para evidenciar lo explicado.

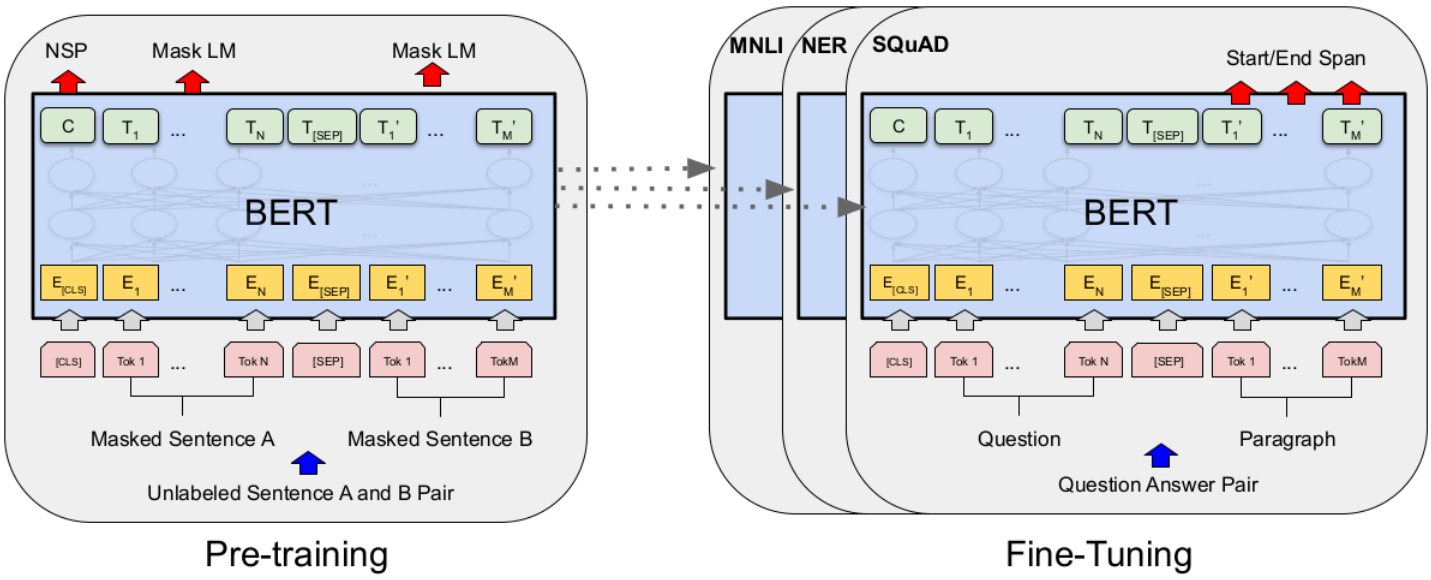

Figura 2.9: Pre-entrenamiento de BERT en el lado izquierdo como un *Masked Language Model* (Mask LM) con un tarea de *Next Sentence Prediction* (NSP) incluida. Ajuste fino de BERT en el lado derecho de la figura con posibles tareas como *genre classification* (MNLI), *Named Entity Recognition* (NER) y *auestion answering completion* (SQuAD). El mismo modelo (arquitectura) es utilizado en ambos lados de la figura, a excepción de la capa de salida. Los *tokens* [CLS] (clasificación) y [SEP] (separación de pregunta/respuesta) están presentes en todas las secuencias. Imagen extraída de publicación original de Devlin et al. (2019).

En la figura 2.9 –tomada de la publicación original de Devlin et al. (2019)– se muestra el mecanismo de máscaras en el pre-entrenamiento. También se muestra que al comienzo de cada secuencia se añade un *token* de clasificación llamado [CLS] que es el mismo para todas las secuencias durante el pre-entrenamiento. Este *token* sirve para tareas de *fine-tuning* donde lo que se busca es clasificar de alguna forma las secuencias en su totalidad. Además, se muestra que en pre-entrenamiento se le entregan dos secuencias consecutivas para que pueda aprender a contestar preguntas. La forma en que el *Transformer* puede distinguir entre una secuencia y otra (a priori, no puede), es que se le entrega esta información en forma de *embeddings* (representaciones únicas) sumado a la información posicional.

Los *transformers* que se modelan en este trabajo para representar EEG están inspirados en BERT. Esto porque los *transformers* propuestos en este trabajo son del tipo *encoder bidireccional*, y además, se pre-entrenan utilizando máscaras en la señal que tienen las mismas proporciones que BERT.

2.2.6. Transformer que mejora la localidad

Esta sección explica una variante del *transformer* propuesto por Li et al. (2019), solo que intenta resaltar vecindades dentro de una secuencia, a diferencia de resaltar puntos como en un proceso de atención normal. En cada cabezal se trata de poner atención sobre una vecindad y no sobre solamente un punto. Esto viene de la diferencia entre procesar secuencias en forma de series de tiempo versus procesar secuencias de palabras, que es para lo que se pensó

el *transformer* de Vaswani et al. (2017) y el de Devlin et al. (2019).

La principal diferencia entre los *transformers* mencionados radica en que en una oración una palabra por sí sola puede tener un significado dado el resto de las palabras en la oración, en cambio en una serie de tiempo, en general una vecindad o un conjunto de puntos tienen un significado dado el resto de la serie de tiempo. Esto es más evidente cuando se ve la cantidad de puntos que debe tener una serie de tiempo para que represente algo versus cuántas palabras debe tener una oración para que signifique algo, siendo la diferencia en el largo de la secuencia de hasta varios ordenes de magnitud.

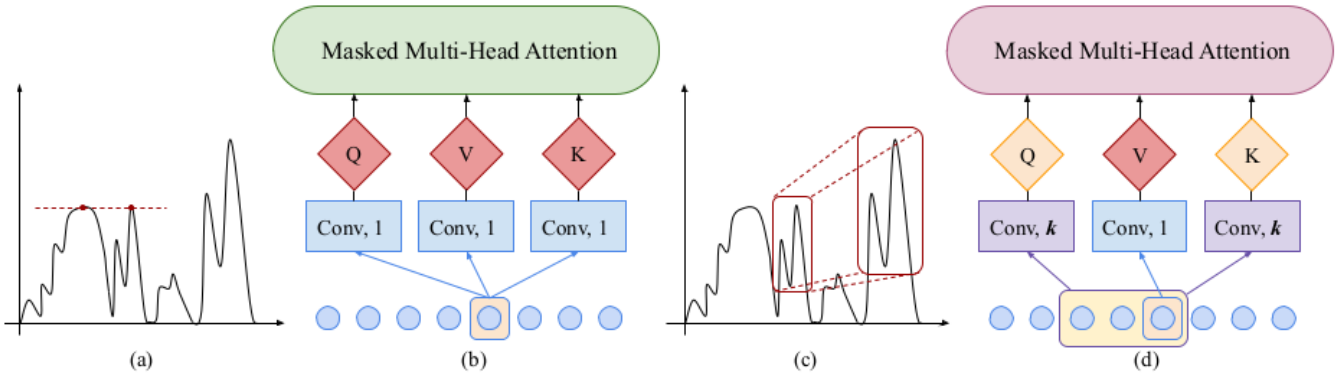

Figura 2.10: a) y b) muestran la Atención Vanilla. c) y d) muestran la Atención Local sobre vecindades. Imagen extraída de la publicación original de Li et al. (2019)

El mecanismo de atención del *transformer* propuesto por Li et al. (2019) se resume en la figura 2.10. Este mecanismo incluye una variación en las capas lineales que definen las *querys*, *keys* y *values* del *transformer* original. Las capas lineales que forman estas matrices (o tensores) se pueden ver como una capa convolucional donde la ventana del filtro es de largo uno (una sola muestra). Lo que propone Li et al. es que las convoluciones que forman las *querys* y las *keys* tengan un largo mayor que 1, por ejemplo, una ventana de 3 o 5 muestras. Esto permitiría calcular *scores* sobre vecindades más que sobre una muestra en secuencias con cientos o miles de muestras.

Cabe destacar que Li et al. (2019) utiliza una capa lineal normal (convolución de largo 1) para formar los *values*. La razón para no computar los *values* con ventanas convolucionales de largo mayor que 1 no queda muy clara, pero al no hacer esto, se elimina la posibilidad de utilizar un *stride* para reducir la cantidad de *scores* en el mecanismo de Atención Neuronal, ya que las *keys* y los *values* deben tener necesariamente el mismo largo. Esto es porque cada *query* computa tantos *scores* como *keys* hayan y estos ponderan a cada uno de los *values*, es decir, por cada *score* proveniente de una operación con una *key*, debe haber un *value*. Lo anterior tomará más relevancia luego, dado que, en este trabajo se propone utilizar ventanas convolucionales en los *values*, para poder reducir la cantidad de *scores* a calcular utilizando un *stride* mayor a 1.

2.2.7. BENDR - Wav2vec 2.0

BENDR es un acrónimo para *BErt-inspired Neural Data Representations*. Este modelo fue propuesto por Kostas, Aroca-Ouellette, y Rudzicz (2021) y fue desarrollado como una variante del *transformer* llamado Wav2vec 2.0 propuesto por Baevski, Zhou, Mohamed, y Auli (2020). BENDR es el único *transformer* encontrado en la literatura que se pre-entrena en señales EEG para resolver tareas particulares de este tipo de señales. BENDR tiene muchas similitudes con Wav2vec 2.0, pero las principales son el aprendizaje de tipo contrastivo para pre-entrenar el modelo y la reducción del largo de la secuencia con capas convolucionales 1D para extraer características mediante pérdida de resolución. La figura 2.11 esquematiza la arquitectura general que tienen los *rransformers* BENDR y Wav2Vec 2.0.

En la figura 2.11 se puede ver que la arquitectura de estos *transformers* consiste, inicialmente, en procesar como input la señal de entrada $\mathcal{X}$. Luego, reducir la resolución a manera de extraer características mediante capas convolucionales con *strides* obteniendo representaciones intermedias (q) de la señal. Estas representaciones entran como input al *transformer* tipo BERT, enmascarando un porcentaje de las representaciones intermedias (q) y tratando de reconstruir las reqpresentaciones enmascaradas en las representaciones contextualizadas $\mathcal{C}$ por el *transformer*. Cabe destacar que ambos *transformers* (BENDR y Wav2vec) relativizan la información posicional utilizando una capa convolucional para aprender el llamado "*positional encoding*", a diferencia del *positional encoding* estático que se utiliza en BERT.

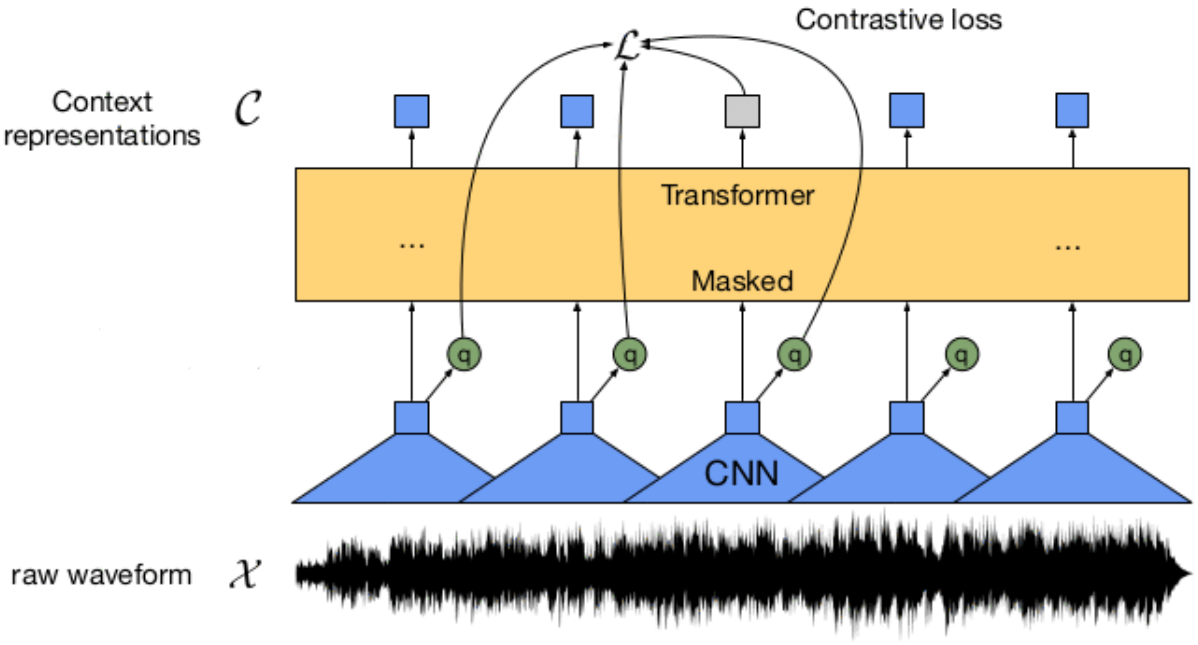

Figura 2.11: Arquitectura y pre-entrenamiento de *transformer* tipo BENDR/Wav2vec 2.0. Imagen extraída de publicación original de Wav2vec 2.0.

Las principales diferencias entre BENDR y Wav2vec 2.0 son, en primer lugar, que las señales EEG de entrada en BENDR están compuestas de multiples canales a diferencia de Wav2Vec 2.0 que solo señales de entrada de 1 canal. Segundo, la salida del extractor de características de BENDR (capas convolucionales) no pasa por una capa de cuantización como en

Wav2vec 2.0, la que pretende discretizar los vectores de características aprendidos. Tercero, el pre-entrenamiento de BENDR es netamente de aprendizaje contrastivo, es decir, solamente utiliza la llamada "*Contrastive Loss*", mientras que, Wav2vec 2.0 suma una segunda función de pérdida, en la que calcula la entropía de la salida del *transformer*. Por último, ambos modelos difieren en algunos hiperparámetros de arquitectura en cuanto al tipo extractor de características y las variaciones del *transformer* tipo BERT, como por ejemplo, la cantidad de capas del *transformer* o el número de características para representar cada muestra.

La función de pérdida contrastiva es la función de pérdida que se utiliza para pre-entrenar los *transformers* que se proponen en este trabajo de tesis. Lo anterior, por los buenos resultados que obtuvieron Kostas et al. (2021) y Baevski et al. (2020) con BENDR y Wav2vec 2.0 en el pre-entrenamiento de *transformers* para procesar y representar señales. Esta función de pérdida calcula las similitudes coseno entre las salidas del *transformer* en las posiciones que fueron enmascaradas y las representaciones de entrada que fueron enmascaradas. La forma en que esto se realiza se detalla en la ecuación 2.3.

$$\mathcal{L} = -log\frac{exp(sim(c_t, q_t) \cdot k)}{\sum_{\tilde{q}\in Q_t} exp(sim(c_t, \tilde{q}) \cdot k)} \tag{2.3}$$

En la ecuación 2.3 se calcula la similitud coseno (sim) entre cada instante de tiempo que fue enmascarado, donde c_t es la salida del *transformer* en el tiempo t y q_t es la salida de las capas convolucionales o extractor de características. En esta misma ecuación se está calculando la función *softmax* pero solamente en el instante de tiempo enmascarado t y las similitudes coseno que hacen contraste con el instante de tiempo t son los llamados negativos Q_t. Los negativos (Q_t) son 20 instantes de tiempo de la entrada al *transformer* que también fueron enmascarados. Las similitudes coseno de entrada a la softmax puntual son multiplicadas por un hiper-parámetro llamado temperatura k. Este hiper-parametro tiene un valor entre 0 y 1, lo que suaviza la *softmax* agregando un sesgo inductivo al modelo que vuelve más exigente que la similitud coseno del denominador sea 1 y el resto de las similitudes coseno sea -1. Para terminar, la consecuencia directa de esta función de pérdida es que el *transformer* trata de aprender que cada instante de tiempo es completamente diferente (en ángulo) al resto de los instantes de tiempo. Es necesario recordar que cada instante de tiempo representa un conjunto de puntos de la señal original debido a las capas convolucionales previas al *transformer* tipo BERT.

2.3. Trabajos previos y estado del arte

La detección automática de eventos transitorios del sueño representa un gran desafío dada la variabilidad entre expertos para generar bases de datos etiquetadas. Esto no solo hace difícil la tarea de evaluar la detección automática, sino que también de comparar distintos algoritmos y métodos de aprendizaje de máquinas. Este problema se ha ido atenuando con el tiempo y con la aparición de bases de datos de alta calidad en su etiquetado, como la base de datos de MASS utilizada en este trabajo. Otra gran dificultad que existe para los detectores automáticos es la variabilidad en las señales EEG entre sujetos, sobretodo en la banda sigma (Fernandez y Lüthi, 2020; Shinomiya, Nagata, Takahashi, y Masumura, 1999)

que es la banda donde se encuentran los husos de sueño.

2.3.1. Detectores automáticos basados en características

Entre los métodos basados en características, existen detectores que se basan en la clasificación de segmentos cortos mediante la utilización de umbrales (Purcell et al., 2017). Dentro de estos detectores existen, por ejemplo, métodos de comparación de amplitudes dentro de bandas de frecuencias o métodos que se basan en la implementación de sistemas expertos (reglas diseñadas a mano). También existen métodos no paramétricos de aprendizaje de máquinas que utilizan *Support Vector Machines* (Ulloa et al., 2016) y *Random Forest* (Jiang, Ma, y Wang, 2021).

Algunos de estos detectores tienen desempeños estables y más competitivos, como el detector A7 (Lacourse, Delfrate, Beaudry, Peppard, y Warby, 2019) que utiliza ventanas con un paso para calcular 4 distintas características, las que luego de pasar por un umbral combinado con reglas expertas, entregan detecciones de husos de sueño. Otro detector con buen desempeño es el detector Spinky (Lajnef et al., 2017), que descompone la señal en dos componentes para detectar husos de sueño y complejos-k. Sin embargo, a pesar de que algunos métodos tienen un buen desempeño, estos detectores tienen varios problemas. Estos problemas se ven particularmente reflejados en los falsos positivos, dada las complicaciones y desafíos intrínsecos a la detección automática de eventos transitorios del EEG.

2.3.2. Detectores automáticos basados en redes neuronales

Los detectores basados en redes neuronales son de particular interés en este trabajo, ya que son este tipo de detectores los que tienen los mejores desempeños en la tarea de detección de eventos transitorios del sueño. Los métodos de detección basados en redes neuronales han demostrado poder sobrellevar de mejor manera los desafíos de la detección automática de eventos transitorios en el EEG. Estos métodos alcanzan un mejor equilibrio entre falsos positivos y falsos negativos, y además, logran generalizar mejor las detecciones en sujetos que no forman parte del conjunto de entrenamiento de los modelos.

Entre estos detectores se encuentra DOSED (Chambon, Thorey, Arnal, Mignot, y Gramfort, 2019), el cual procesa señales EEG en segmentos de 20 segundos para detectar husos de sueño y complejos-k. Este modelo se compone de una red convolucional para la extracción de características y de una red *fully connected* que toma en cuenta todo el contexto temporal. Otro detector basado en redes neuronales es el detector SpindleNet (Kulkarni et al., 2019) integrado 2 redes neuronales, donde una red procesa ventanas de 0.25 segundos de señal EEG, mientras la otra procesa la ventana filtrada en la banda sigma. Ambas redes se componen de capas convolucionales seguidas de capas LSTM, para finalmente concatenar las salidas de ambas redes y predecir la clase de la ventana con capas *fully connected*. Existen otros detectores basados en redes neuronales, pero en este trabajo se da énfasis y se compara el modelo propuesto con el detector del estado del arte, RED.

RED

El estado del arte en la detección automática de eventos transitorios de sueño es RED, propuesto por Tapia y Estevez (2020). La principal limitación de este modelo es que solamente está entrenado sobre bases de datos de EEG etiquetados con eventos transitorios del sueño y no aprovecha por tanto la gran cantidad de datos no etiquetados que existe en registros

de EEG. Este modelo se compone de una primera etapa de extracción de características, la cual se compone de una serie de capas convolucionales que disminuyen el largo de la serie de tiempo para luego pasar a una etapa de contextualización, la cual consiste de dos capas recurrentes tipo LSTM bidireccional. Esto se puede ver más claramente en detalle en la figura 2.12.

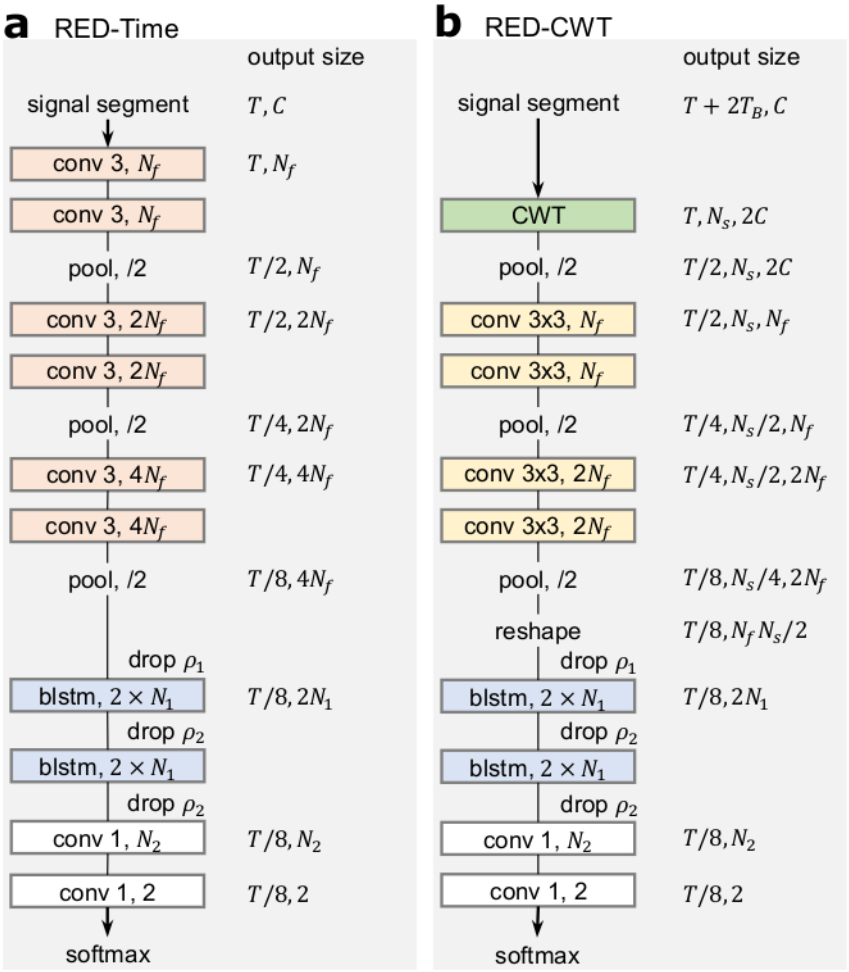

Figura 2.12: Arquitecturas de modelos RED. **a** es la arquitectura RED-Time que utiliza un solo canal (temporal) de la señal EEG. **b** es la arquitectura RED-CWT que utiliza dos canales, el temporal y la *Continuous Wavelet Transform* (CWT) del canal temporal de la señal EEG. Imagen extraída de la publicación original de Tapia y Estevez (2020).

En la figura 2.12 se puede ver claramente que hay dos modelos RED con arquitecturas muy similares. La principal diferencia entre estas arquitecturas es que una de ellas (**b** - RED-CWT) incluye una característica extra en la serie de tiempo como input además del valor mismo del EEG. Esta característica extra fue construida a partir de la misma serie de tiempo al calcular su CWT (*Continuous Wavelet Transform*). Se puede apreciar un ejemplo de esta transformada en la figura 2.13.

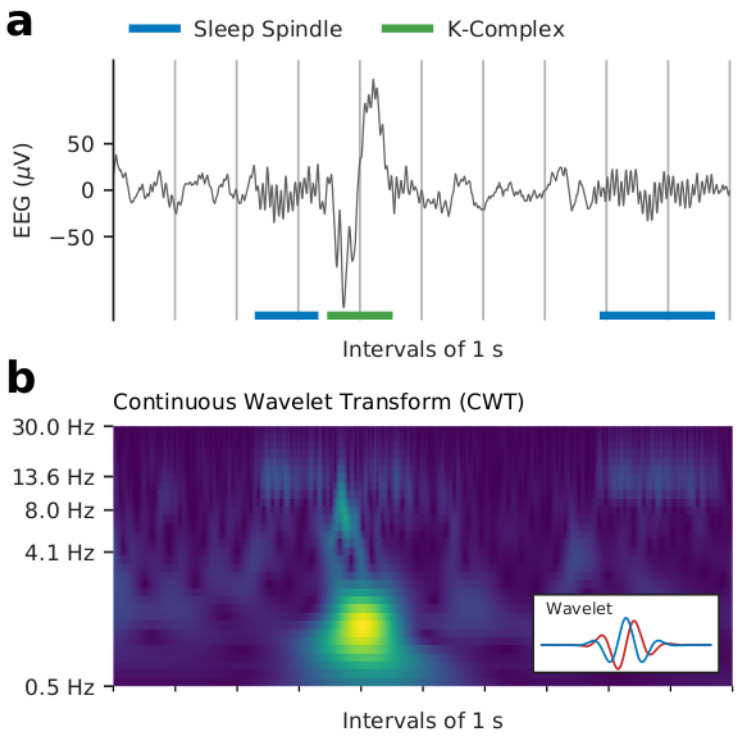

Figura 2.13: Ejemplo de un EEG y su CWT. Imagen extraída de la publicación original de Tapia y Estevez (2020).

Si bien la inclusión de transformadas puede parecer intuitiva a nivel de ingeniería de características, una de las principales razones por las que se utilizan los modelos de aprendizaje profundo de redes neuronales es su gran capacidad de extraer características, eliminando la necesidad de ingresar otras características. Esta observación se apoya fuertemente en el desempeño de estos modelos al ser evaluados y comparados con otros detectores automáticos que realizan la misma tarea. Estos resultados se pueden ver en la figura 2.14, en la cual se ve claramente como el añadir características (RED-CWT) no influye de manera significativa en el desempeño del modelo.

Task	Method	F1-score	Recall	Precision
SS-E1	**RED-CWT**	80.9 ± 0.4	81.5 ± 1.6	81.2 ± 1.2
	RED-Time	81.2 ± 0.5 [b]	81.7 ± 2.2	81.8 ± 1.8
	DOSED [9]	78.4 ± 0.8 [c]	78.3 ± 2.0	80.0 ± 1.9
	A7 [15]	71.4 ± 0.3 [c]	75.3 ± 0.4	69.3 ± 0.8
SS-E2	**RED-CWT**	84.7 ± 0.4	82.6 ± 1.2	88.1 ± 1.1
	RED-Time	84.5 ± 0.3 [b]	82.9 ± 1.0	87.4 ± 1.1
	DOSED [9]	82.0 ± 0.6 [c]	79.2 ± 1.3	87.0 ± 0.7
	A7 [15]	73.9 ± 0.4 [c]	82.5 ± 2.0	68.0 ± 1.0
	SpindleNet [10] [a]	83.0 ± 2.0	85.2	81.0 ± 3.2
KC	**RED-CWT**	82.8 ± 0.4	81.7 ± 1.1	84.9 ± 0.4
	RED-Time	82.6 ± 0.4 [b]	82.1 ± 1.3	83.9 ± 0.8
	DOSED [9]	77.1 ± 0.8 [c]	76.5 ± 1.2	78.6 ± 1.8
	Spinky [17]	64.9 ± 0.1 [c]	62.6 ± 0.5	68.7 ± 0.6

Figura 2.14: Desempeño de modelos RED y otros detectores. Imagen extraída de la publicación original de Tapia y Estevez (2020).

Capítulo 3

Metodología

A manera de recapitulación, el problema en cuestión es la detección automática de eventos transitorios del sueño en registros de electroencefalogramas, más específicamente, la detección de husos de sueño y de complejos-k. Para esto, se propone un modelo basado en atención neuronal llamado *Transformer*. Es importante destacar que el diseño y adaptación del *Transformer* para poder representar series de tiempo fue un problema desafiante y su resolución fue fundamental en el trabajo realizado.

Dicho lo anterior, a continuación se describen los procedimientos, métodos, propuestas y cálculos de los resultados que se utilizan en el siguiente capítulo donde se enuncian los resultados y las discusiones. Además, se describen en detalle los modelos propuestos, la forma de entrenar estos modelos y el procesamiento de datos realizado.

3.1. Datos para entrenamiento y evaluación

Los datos de señales EEG que se utilizaron en este trabajo fueron procesados utilizando el lenguaje python y las librerías que este lenguaje dispone para tratar datos y señales. En cuanto a la disponibilidad de datos, este fue facilitado por el laboratorio de Inteligencia Computacional a cargo del profesor guía del presente trabajo, el Dr. Pablo Estévez. Estos datos fueron almacenados y administrados en un *data warehouse*, utilizando el motor de bases de datos relacionales, Postgresql.

3.1.1. Bases de datos

Las bases de datos con señales de electroencefalogramas de las cuales se dispone son múltiples y variadas. Dentro de estas bases de datos, algunas cuentan con eventos transitorios del sueño etiquetados y otras bases de datos no tienen etiquetas aunque sí las marcas de las fases del sueño. Es importante destacar que en este trabajo, como ya se mencionó anteriormente, se procesan únicamente los EEG del sueño de la fase N2.

Por un lado, se tienen bases de datos sin eventos transitorios de sueño etiquetados, pero que poseen las marcas de las fases del sueño nocturno. Dentro de estas bases están las bases de datos proporcionadas por el repositorio *National Sleep Research Resource* (NSRR). Estas bases de datos se utilizan netamente para pre-entrenar el *transformer* propuesto, considerando que estas bases de datos son las bases con más registros de EEG que se tienen a disposición. Otra base de datos con registros sin etiquetas es la base de datos *Montreal*

Archive of Sleep Studies, MASS por sus siglas en inglés. Por otro lado, se tienen las bases de datos con eventos transitorios del sueño etiquetados por expertos. Entre estas se encuentran las bases de datos MASS-SS2, con etiquetas de husos de sueño y etiquetas de complejos-k (por separado) y la base de datos MASS-MODA, la cual solo contiene etiquetas de husos de sueño. Cabe destacar que ambas bases de datos etiquetadas son subconjuntos de registros de la base de datos MASS.

NSRR

El repositorio NSRR fue desarrollado por Zhang et al. (2018) y se compone diferentes bases de datos, dentro de las cuales se utilizan 6 bases de datos abreviadas por; CFS desarrollada por Redline et al. (1995), CHAT desarrollada por Marcus et al. (2013), CCSHS desarrollada por Rosen et al. (2003), SHHS desarrollada por Quan et al. (1997), MrOS desarrollada por Blackwell et al. (2011) y SOF desarrollada por Spira et al. (2008). Estas bases de datos poseen páginas de sueño de 30 segundos, anotadas con la fase de sueño correspondiente según el estándar R&K. El conjunto de estas bases de datos engloba un total de 11.630 sujetos con diferentes frecuencias de muestreo en los registros. Los canales de interés de estas bases de datos son el C3-A2 y C4-A1, ya que los canales centrales suelen ser los canales más "limpios" de las señales EEG.

MASS

La base de datos de MASS fue desarrollada por Christian, Gosselin, Carrier, y Nielsen (2014). Esta base contiene registros polisomnográficos de noches completas de sueño de 200 sujetos. Dentro de estos sujetos, hay 97 hombres con una edad promedio de 42.9 ± 19.8 años y hay 103 mujeres con una edad promedio de 38.3 ± 18.9 años. Todos los registros presentes en la base de datos tienen una frecuencia de muestreo de 256 Hz. Además, cada sujeto tiene entre 4 y 20 canales de señal EEG, pero para efectos de este trabajo, solamente se toman en cuenta los canales C3-LER (*linked ear resistor*) o C3-CLE (*computed linked ear*) dependiendo del que esté disponible, y el canal A2-CLE cuando este está disponible para poder referenciarlo con el canal C3 y así tener disponible el canal C3-A2. Al igual que las bases de datos de NSRR, MASS tiene marcas de las fases de sueño en páginas de 30 segundos de registro, pero también tiene marcas de fases de sueño en páginas de 20 segundos de sueño.

MASS-SS2

La base de datos MASS-SS2 es uno de los 5 subconjuntos entre los que está dividida la base de datos MASS. Este subconjunto está compuesto de 19 adultos jóvenes entre 18 y 33 años con marcas de fases del sueño en páginas de 20 segundos. Esta base de datos contiene etiquetas de eventos transitorios del sueño en toda la etapa N2 del sueño. Los eventos fueron etiquetados por dos expertos, E1 y E2. El experto E1 etiquetó todas las etapas N2 con husos de sueño y complejos-k, mientras que, el experto E2 solo etiquetó husos de sueño y a sólo 15 de los 19 sujetos de MASS-SS2. Además, el experto E2, a diferencia del E1, tuvo acceso a las señales EEG filtradas en la banda sigma (11-16 Hz) y no planteó una duración mínima en la detección de husos de sueño. Esto tuvo como consecuencia que las detecciones de E1 estuvieran casi completamente contenidas en las detecciones de E2. Por lo tanto, se decidió solamente experimentar con las señales etiquetadas por el experto E1, ya que para efectos de evaluación, se consideró que este experto era un punto de comparación suficiente. A partir de

este punto, se refiere como SS2-E1 a la base de datos MASS-SS2 con las etiquetas de husos de sueño del experto E1, y se refiere como SS2-KC a la base de datos MASS-SS2 con las etiquetas de complejos-k.

MASS-MODA

La base de datos MODA fue desarrollada por Lacourse, Yetton, Mednick, y Warby (2020). MODA es el acrónimo para *Massive Online Data Annotation*, la cual, es una plataforma en donde varios expertos anotaron husos de sueño en extractos de señales de 115 segundos de 180 sujetos de la base MASS. Estas anotaciones combinadas forman una base de datos de anotaciones husos de sueño de alta calidad. Esta base de datos está compuesta por dos fases que diferencia sujetos por edad. La fase 1 consiste en 100 adultos con una edad promedio de 24.1 años y la fase 2 consiste en 80 adultos con una edad promedio de 62.0 años. Las dos fases en conjunto consideran 30 sujetos con 10 bloques de 115 segundos de señal anotada y 150 sujetos entre 2 y 3 bloques de 115 segundos de señal. A partir de este punto, se refiere como MODA a la base de datos MASS-MODA con husos de sueño etiquetados.

3.1.2. Procesamiento de señales

Para procesar las señales en bruto de las bases de datos, primero se conectan todas las páginas consecutivas de señales EEG en etapa N2. Esto se realiza para tener señales más continuas a la hora de realizar filtrados o evaluaciones. También, se agregan 15 segundos en los bordes de cada señal continua en la etapa N2 para eliminar los efectos de borde en el filtrado de señales. Estas señales se guardan en forma de arreglos en tablas de *Postgresql* para poder acceder a ellas y poder estudiarlas más directamente como conjunto. En la figura 3.1 se muestra un ejemplo de una señal N2 bruta de 40 segundos con los 15 segundos extra de borde añadidos por lado que son eliminados al final de procesamiento de señales.

Filtrado de señales

A razón de eliminar ruido y pequeños artefactos, se aplica un filtro pasa banda sin distorsión de fase y con frecuencias de corte en 0.1 Hz y 35 Hz, utilizando la librería *scipy.signal* de python. El filtro utilizado es un filtro *Butterworth*, propuesto por Butterworth (1930). Se utiliza un orden 10 en el filtro *Butterworth*, dado que este filtro necesita de ordenes mayores para alcanzar ciertos requerimientos de filtrado. Además, el filtro es aplicado hacia adelante y hacia atrás para no distorsionar la fase y en formato SOS (*series second-order sections*). La forma en la que se aplica el filtro es, primero mediante un pasa-alto con frecuencia de corte en 0.3 Hz y luego mediante un filtro pasa-bajo con frecuencia de corte en 30 Hz.

Remuestreo de señales

Luego del filtrado de señales, éstas se remuestrean a 200 Hz utilizando un filtro polifásico, nuevamente de la librería *scipy.signal* de python. Para poder remuestrear la señal, internamente se calcula el máximo común divisor entre la frecuencia de muestreo original y la nueva frecuencia de muestreo. Luego se sobremuestrea la señal, intercalando ceros, y se aplica un filtro pasa bajos de respuesta finita (FIR) sin distorsión de fase. Finalmente se submuestrea la señal de 250 Hz (frecuencia original), para llegar así a la frecuencia de muestreo de 200 Hz deseada.

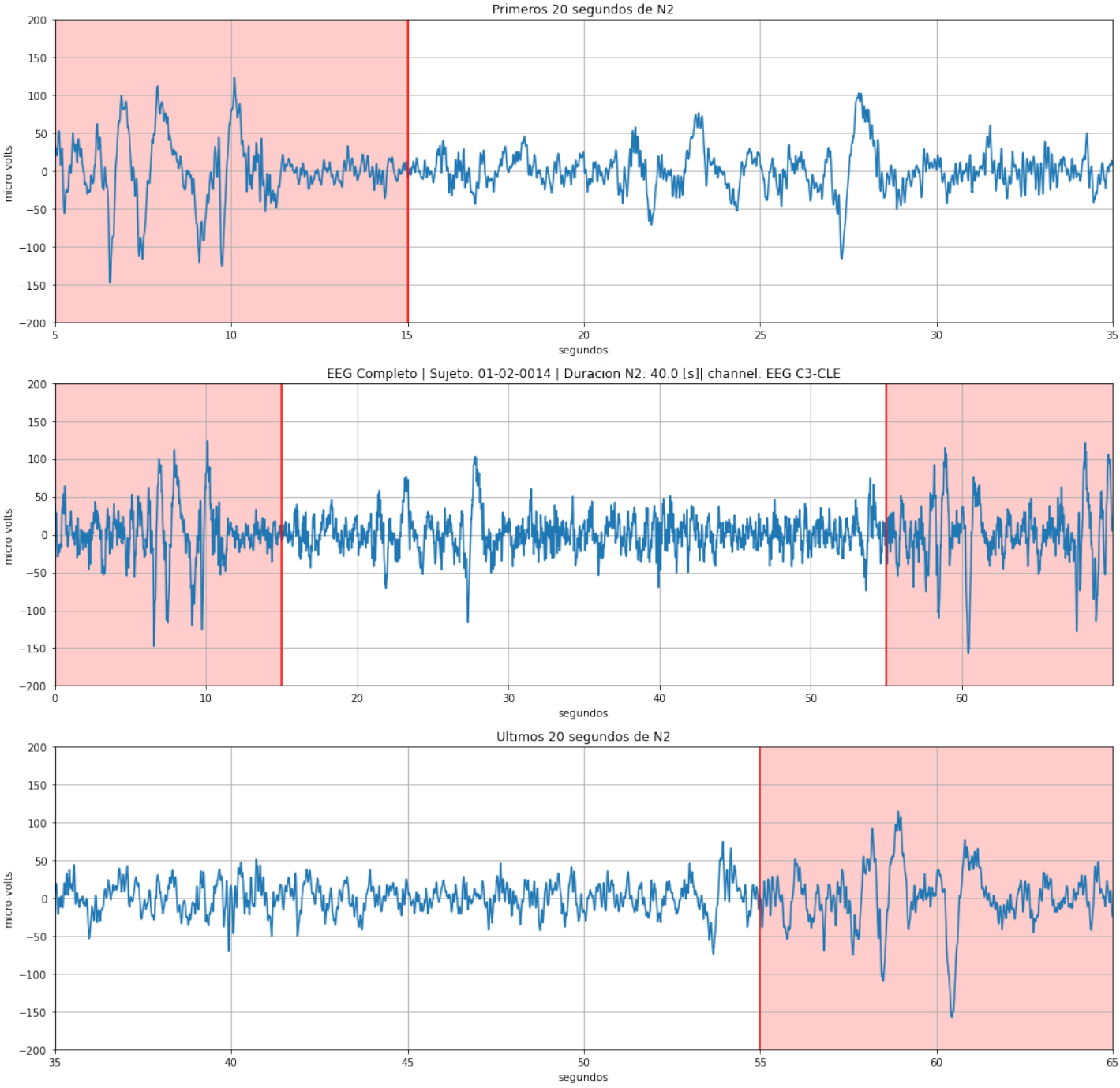

Figura 3.1: Señal EEG bruta de MASS. Segundo gráfico muestra señal completa con 40 segundos de N2 y 15 segundos extra añadidos a cada lados (70segundos en total). Primer y tercer gráfico muestra un acercamiento a los bordes de los 70 segundos de señal.

3.1.3. Particiones de validación cruzada

El *transformer* propuesto, al igual que los detectores automáticos en la literatura, se evalúa en la detección de eventos transitorios realizando validaciones cruzadas con 5 *folds* por validación cruzada a nivel de sujetos, es decir, con una partición de MASS en 5 conjuntos de sujetos. La partición se realiza 3 veces con distintas combinaciones de sujetos por partición.

La razón por la cual se utilizan los sujetos para particionar, y no las señales en sí, es porque existe una gran correlación en las señales de un mismo sujeto en cuanto a las características de sus husos de sueño y complejos-k. En cada una de las 3 validaciones cruzadas, se seleccionan 3 conjuntos de los 5 para entrenar, un conjunto para validar y un conjunto para testear. Esto significa que en cada validación cruzada se realizan 5 entrenamientos distintos, donde cada uno de los 5 conjuntos es utilizado exactamente una vez para validación y una para testeo.

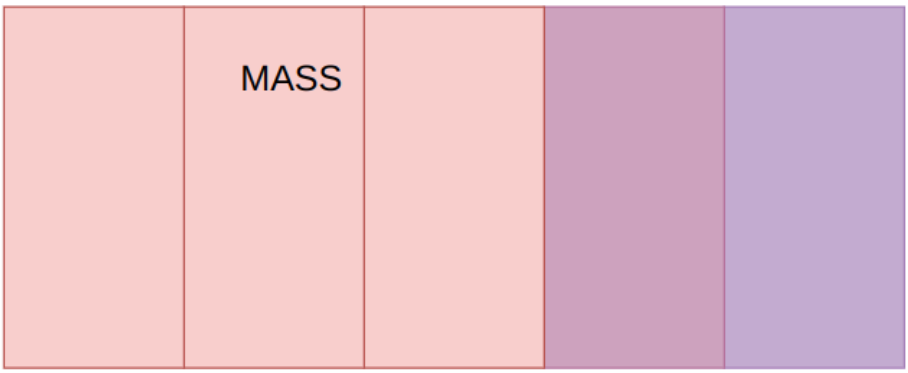

Figura 3.2: Ejemplo de una partición (*fold*) de una de las 3 validaciones cruzadas de 5 particiones de MASS. En cada partición se dividen los sujetos en 5 conjuntos, 3 para entrenamiento, 1 para validación y 1 para testeo.

De todos las bases de datos descritas en la subsección anterior, las bases de datos de NSRR solo se ocupan para pre-entrenar (ver sección 3.2.2) el *transformer* con datos no etiquetados. Es decir, no se utilizan para entrenar o evaluar las detecciones de eventos transitorios del sueño. Por lo tanto las particiones de datos para realizar validaciones cruzadas como método de evaluación, se realizan solamente sobre la base de datos MASS y sus subconjuntos etiquetados MASS-SS2 y MASS-MODA. La figura 3.3 ejemplifica esquemáticamente las intersecciones entre los conjuntos MASS, MASS-SS2 y MASS-MODA.

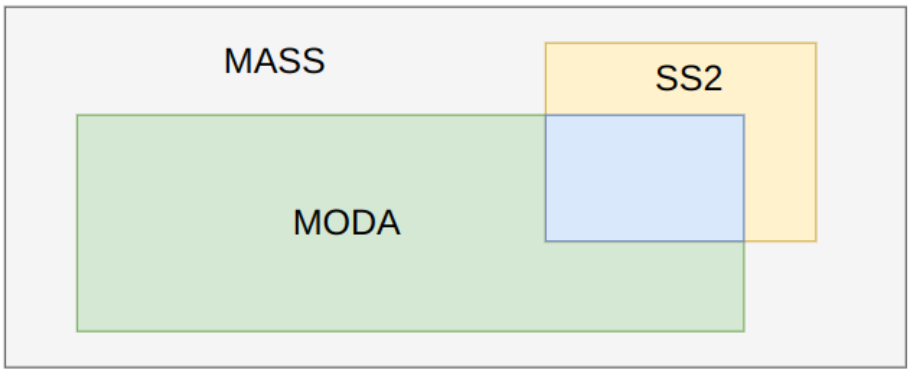

Figura 3.3: Intersección de MASS con conjuntos de datos etiquetados

Como ya se mencionó en la sección anterior, la base de datos MASS contiene noches completas de 200 sujetos. En particular, las señales que se utilizan son las señales en la etapa N2

de los canales C3 y A2, cuando este último está disponible para referenciar el canal C3 con el fin de tener señales más "limpias". La razón por la que se eligen estos canales es porque la base de datos MASS-SS2 se etiquetó utilizando el canal C3-LER y la base de datos MASS-MODA se etiquetó utilizando el canal C3-CLE y el canal C3-A2 (siempre que el canal A2 esté disponible). En resumen, las señales de MASS utilizadas abarcan, aproximadamente, un total de 4.6 millones de segundos, mientras que las señales utilizadas en sus subconjuntos etiquetados, MASS-SS2 y MASS-MODA, abarcan un total aproximado de 260 mil y 86 mil segundos respectivamente.

En la figura 3.3, la zona marcada en azul representa la intersección entre MODA y SS2 con una presencia de 14 sujetos, teniendo que MASS-MODA posee 180 sujetos y SS2 posee 19 sujetos. Esto implica que hay 15 sujetos en la base MASS, que no tienen señales con eventos etiquetados ni en MASS-MODA, ni en MASS-SS2. Lo anterior es sumamente importante de comprender para poder realizar las validaciones cruzadas. Esto porque la base de datos MASS se utiliza para pre-entrenar el *transformer* y el conjunto de entrenamiento generado en cada uno de los 5 *folds* de las validaciones cruzadas debe coincidir con el conjunto de entrenamiento generado para el ajuste fino en los datos etiquetados de MODA y SS2. Esto quiere decir que los 3 conjuntos de sujetos utilizados para entrenamiento en la zona roja de la figura 3.2 deben contener los sujetos que se utilizarán en el ajuste fino (entrenamiento en datos etiquetados) del *transformer* para entrenar en MASS-MODA y SS2. Se debe considerar que para ambas bases de datos, MODA y SS2, también se genera una partición de 5 conjuntos en cada una de las validaciones cruzadas, ya que en los conjuntos de prueba de estas bases de datos se calculan las métricas para evaluar el modelo.

La manera en la que se aborda el problema anterior es que para cada una de las 3 validaciones cruzadas siempre se generan ciertas combinaciones de sujetos. Estas combinaciones son tales que las particiones de la figura 3.2 se forman de ciertas cantidades de sujetos de los conjuntos de la figura 3.3, tal como se muestra en la figura 3.4. En cada una de las 3 validaciones cruzadas siempre se tiene esta composición y lo que cambia son los sujetos que conforman cada uno de los conjuntos mostrados en 3.3, repartidos en cada una de las 3 particiones generadas para las validaciones cruzadas.

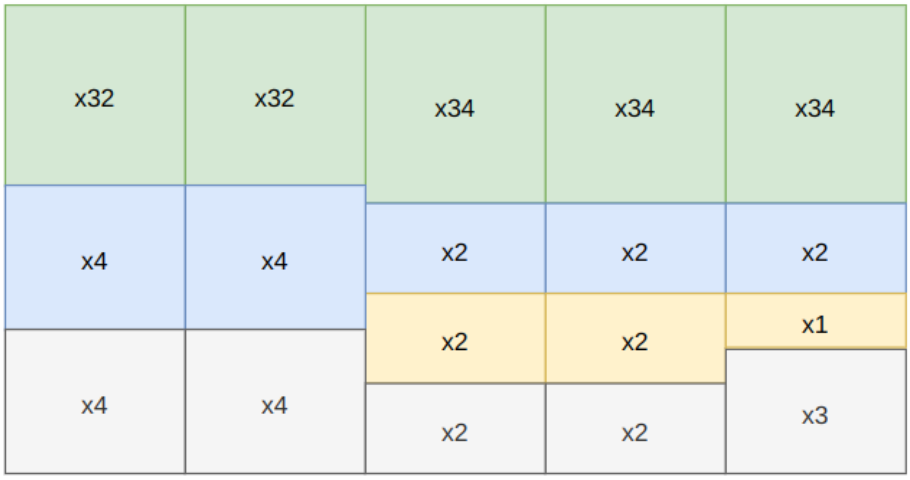

Figura 3.4: Composición en cantidad de sujetos pertenecientes a MASS, MODA y SS2 según 3.3 en cada uno de los 5 conjuntos formados por las particiones de cada validación cruzada.

De la figura 3.4, se puede verificar que para el pre-entrenamiento en datos no etiquetados se tiene 40 sujetos de MASS en cada una de los conjuntos. En cuanto a las particiones de los datos etiquetados, en cada una se tienen 36 sujetos de MODA y 4 sujetos de MASS-SS2 por conjunto generado, excepto en el último conjunto, donde se tienen solo 3 sujetos de MASS-SS2. Recordando que la base de datos MASS-SS2 contiene solo 19 sujetos, no se puede dividir el total de sujetos de forma exacta en 5 conjuntos.

3.2. Modelo propuesto - Locally enhancing BENDR (LeBENDR)

El detector automático propuesto en esta tesis está inspirado principalmente en el modelo de red neuronal, BENDR. La figura 3.5 esquematiza de manera general la arquitectura neuronal del modelo propuesto. El modelo, así como todas las variaciones y los entrenamientos fueron implementadas en pytorch. En la figura, se puede ver que aparte de la capa de codificación posicional o *positional encoding* y las capas del *transformer*, la arquitectura posee capas convolucionales previas como en BENDR. Pero estos bloques convolucionales son para reducir la resolución de la señal tal como lo hace el detector RED propuesto por Tapia y Estevez (2020), al igual que el bloque de clasificación. Este último bloque está compuesto por una capa lineal para reducir la dimensionalidad y una neurona de salida para utilizar la función sigmoidea que genera la probabilidad positiva (presencia de evento) y la entropía cruzada binaria como función de pérdida.

El modelo propuesto trabaja con señales de entrada de largo fijo. Eventualmente, estos largos podrían ser de largo variable, pero por simplicidad y para ser consistentes con los detectores automáticos desarrollados hasta el momento se trabaja con señales de 20 segundos muestreadas a 200 Hz (4000 puntos de resolución). En la figura 3.5 a la salida de cada bloque se especifica la dimensión de salida mediante una tupla. La primera posición de la tupla representa el largo de la secuencia y la segunda posición de la tupla representa la cantidad de

características. En la salida de los bloques convolucionales la cantidad de características es E, pero dentro de cada bloque convolucional la cantidad de características es constante. Esto es, porque la primera capa convolucional dentro de uno de estos bloques duplica la cantidad de características, excepto por el primer bloque, en el que la primera capa convolucional aumenta las características de 1 a $\frac{E}{4}$.

En la figura 3.5 se ilustra que después de los bloques convolucionales se utiliza *dropout* (Srivastava, Hinton, Krizhevsky, Sutskever, y Salakhutdinov, 2014) para reducir el sobreajuste en los datos de entrenamiento. También se usa *dropout* después de la primera capa lineal y función de activación ReLU en el bloque de clasificación. Además, se utiliza *batch normalization* (Ioffe y Szegedy, 2015) para añadir estabilización al entrenamiento del modelo. Una diferencia importante con el modelo propuesto por Tapia y Estevez (2020) y con el modelo propuesto por Kostas et al. (2021), es que las señales no son normalizadas en la etapa de preprocesamiento. Esta función se le deja netamente a la primera capa de *batch normalization* previa a los bloques convolucionales.

Por último, la figura 3.5 muestra la salida de las capas del *transformer* conectadas con una línea punteada con el bloque de clasificación. Esto se debe a que la tarea de pre-entrenamiento explicada en la sección 3.2.2, no utiliza el bloque de clasificación, dado que en este entrenamiento se busca aprender representaciones de las señales EEG, sin datos etiquetados. Solo en la tarea de ajuste fino o de entrenamiento con datos etiquetados, se utiliza la capa de clasificación para poder detectar los eventos transitorios del sueño.

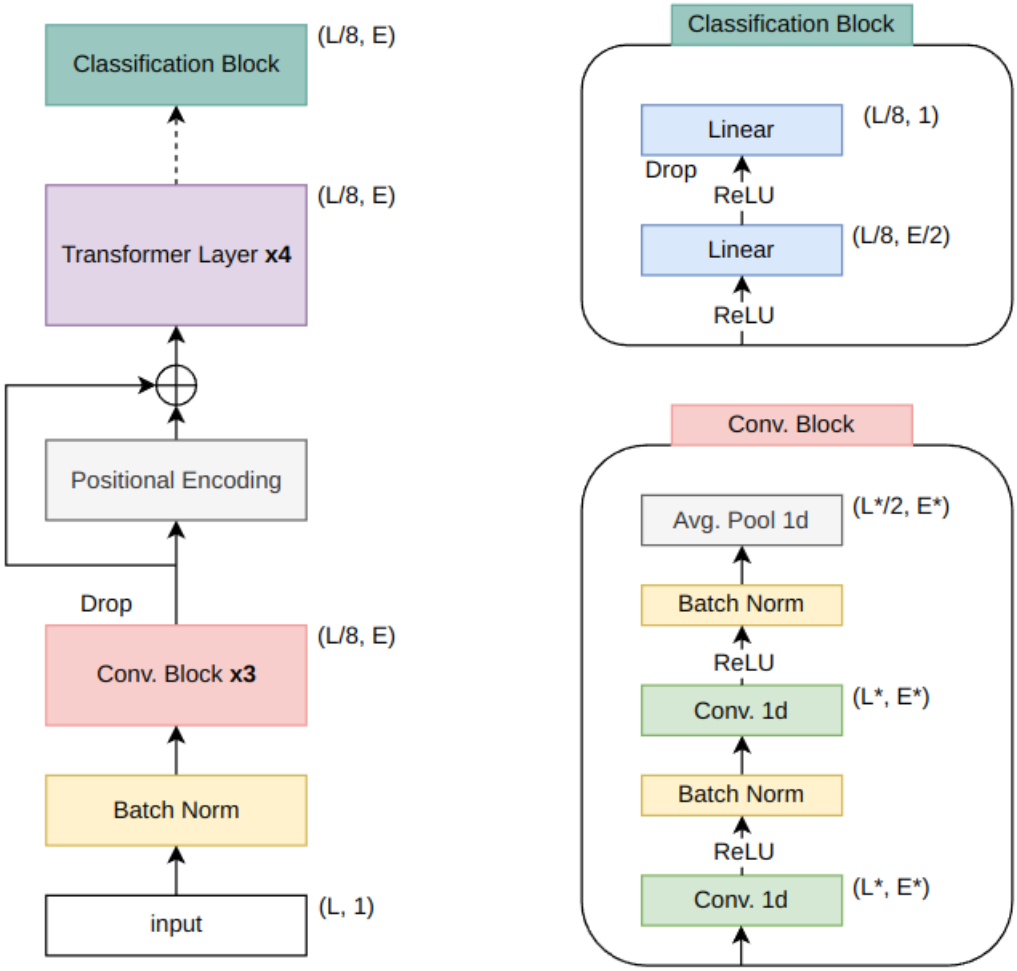

Figura 3.5: *Transformer* propuesto. A la izquierda se encuentra la arquitectura general del modelo propuesto que recibe señales de entrada con 1 solo canal, seguido de una capa de BN y 3 bloques convolucionales, donde cada uno disminuye la resolución a la mitad extrayendo a su vez características. Luego, se incluye la información posicional (*positional encoding*) para finalmente pasar por 4 *transformer layers*. El bloque de clasificación tiene linea punteada porque solo se incluye en el ajuste fino (tarea de detección).

3.2.1. Transformer convolucional

Las capas del *transformer* propuesto, a diferencia de las de BENDR o de Wav2vec 2.0, forma las llamadas *querys*, *keys* y *values* utilizando capas convolucionales, como propuso Li et al. (2019). Sin embargo, existen diferencias en la forma en que se aplican estas convoluciones en comparación a Li et al. (2019). Estas diferencias son que, primero, las *querys* no se forman con ventanas de largo mayor a 1, es decir, se forman con una capa lineal *fully connected*. Segundo, los *values* se forman con una capa convolucional de largo mayor a 1, así pudiendo calcular los valores sobre los que se está poniendo atención de manera local y no puntual. Por último, se aplica un stride de 2 para calcular las *keys* y los *values*, reduciendo así el largo de la secuencia con el fin de calcular una menor cantidad de *scores* de atención, lo que facilita a la red el proceso de poner atención. Considerando que el largo de la secuencia que entra a los cabezales de atención es de 500 puntos de resolución, disminuir la cantidad de

scores a 250 evita que la ponderación disminuya demasiado sus valores, repartiendo la masa de ponderación en más puntos.

En figura 3.6, se ilustra la arquitectura de las capas del *transformer* propuesto. En el bloque de la izquierda de la figura 3.6 se puede ver que, similar a BERT de Devlin et al. (2019), luego de la capa de atención multicabezal (MCA) y también de las capas *fully connected* se aplica una conexión residual y la normalización (*layer normalization*) propuesta por Ba, Kiros, y Hinton (2016). Además, luego de la primera capa lineal en las capas *fully connected*, se utiliza la función de activación GELU propuesta por Hendrycks y Gimpel (2016). Esta función, al ser no lineal en el dominio positivo, facilita la aproximación a no-linealidades. Finalmente se puede ver que el módulo de atención multicabezal que se utiliza, tiene 4 cabezales de atención y cada cabezal de atención tiene una salida con $\frac{E}{2}$ características, las que al concatenarse suman $E * 2$ características. Éstas son reducidas mediante una capa lineal tal como se realiza en los *transformers* de la literatura.

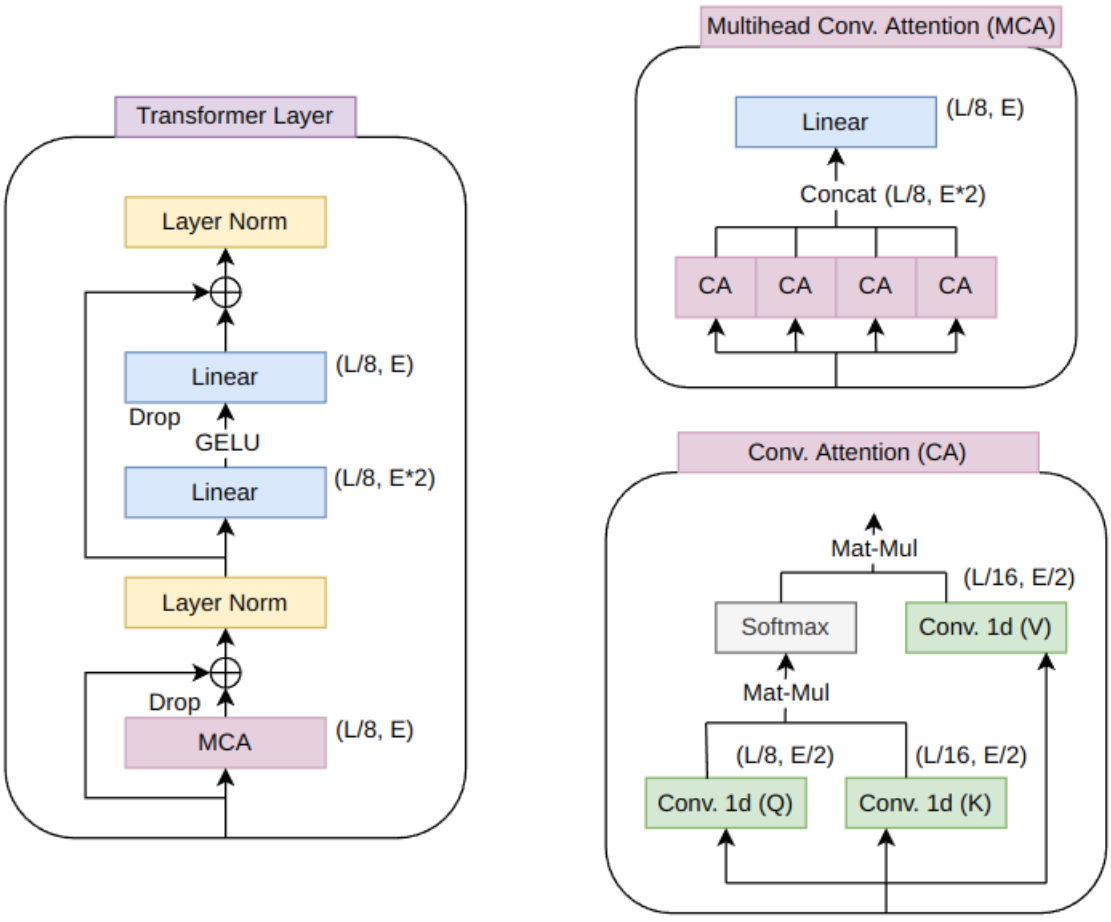

Figura 3.6: Capa del *transformer* convolucional en LeBENDR. A la izquierda se encuentra la arquitectura de la *transformer layer* mencionada en la figura 3.5. A la derecha se encuentra la arquitectura utilizada para procesar múltiples cabezales de atención (MCA) y además se muestra la arquitectura de la atención convolucional que mejora la localidad del *transformer* propuesta en este trabajo.

3.2.2. Transformer de aprendizaje por contraste

La principal razón por la que se adapta el *Transformer* para la detección de eventos transitorios del sueño en EEG, es la capacidad que ha demostrado el *Transformer*, en otras áreas, de aprender representaciones de secuencias. Estas representaciones son aprendidas mediante el pre-entrenamiento del *Transformer* en secuencias de datos no etiquetadas. BERT es un *transformer* pre-entrenado para representar texto (secuencias de palabras). Se pre-entrena aplicando máscaras sobre las secuencias de entrada, de manera que el *transformer* pueda completar las palabras enmascaradas utilizando la información contextual que entregan los cabezales de atención. En el caso de BENDR (*BErt-inspired Neural Data Representations*), este se pre-entrena enmascarando las representaciones locales aprendidas por las capas convolucionales previas al *transformer*. A diferencia de BERT, BENDR no puede completar estos valores enmascarados como un problema de clasificación, donde las palabras son las clases. Esto es porque cuando las capas convolucionales reducen la resolución de la señal, se pierde toda la noción de los valores originales de la señal. Por lo tanto, las representaciones locales enmascaradas se tratan de reconstruir calculando las similitudes coseno entre la representación aprendida por el *transformer*, la representación enmascarada correspondiente (el positivo) y otras 20 representaciones enmascaradas seleccionadas de manera aleatoria (negativos de contraste).

Inicialmente, durante meses, se trató de pre-entrenar un *transformer* como un problema de regresión, utilizando el error cuadrático medio como función de pérdida. Este paradigma seguía, al igual que BENDR, los mismos lineamientos que BERT de enmascaramiento. La diferencia estaba en que el *transformer* regresivo enmascaraba los valores reales de la señal directamente. Esta forma de abordar el pre-entrenamiento se degeneraba a un simple problema de interpolación lineal, ya que al nivel de resolución que se encuentran las señales (4000 puntos en 20 segundos) basta con interpolar los puntos adyacentes al punto enmascarado para reconstruirlo. La consecuencia directa de este fenómeno, es que el *Transformer* no necesita las capas de atención para lograr reconstruir señales, nunca antes vistas, a la perfección. El *Transformer* solamente utilizaba las capas lineales para poder interpolar los valores. Esto finalmente, resultaba en que el *Transformer* no aprendía representaciones con buenas características que sirvieran en la etapa de clasificación de eventos, entrenando en datos etiquetados.

El *transformer* propuesto se pre-entrena utilizando el mismo paradigma contrastivo que utiliza BENDR. Notar que durante el pre-entrenamiento se necesita aprender un vector que representa la máscara de representaciones y además no se necesita el bloque de clasificación de la figura 3.5. El pre-entrenamiento se prueba primero en la base de datos de MASS, con la finalidad de que el *transformer* aprenda a representar las señales de manera general, que posteriormente, aprenderá a etiquetar con eventos en el proceso de ajuste fino, o entrenamiento en las bases de datos etiquetados. Es importante mencionar que un mismo pre-entrenamiento en la base de datos de MASS, puede ser utilizado para diferentes ajustes finos, razón por la cual se construyeron las particiones especificas mostradas en la figura 3.4.

3.2.3. Transferencia de aprendizaje

Además del pre-entrenamiento que se realiza en la base de datos de MASS, se realiza una etapa previa de pre-entrenamiento en las bases de datos de NSRR. Como ya se mencionó anteriormente, las señales EEG están fuertemente correlacionadas dentro de un mismo sujeto.

Esto implica que el hecho de pre-entrenar en la base de datos de MASS ayuda al *transformer* a aprender representaciones de señales EEG que serán similares en la etapa de ajuste fino en las bases de datos etiquetadas. Recordando que las bases de datos etiquetadas MASS-MODA y MASS-SS2 son subconjuntos de la base de datos de MASS. Por lo que un pre-entrenamiento previo al pre-entrenamiento en la base de datos de MASS, permite tener una transferencia de aprendizaje desde lo más general a lo más particular para la representación de señales EEG.

Lo anterior se apoya en que las bases de datos de NSRR contienen señales de EEG de aproximadamente más de 11 mil sujetos, mientras que la base de datos de MASS contiene solamente 200 sujetos. Esto quiere decir que el primer pre-entrenamiento resultaría en el aprendizaje de representaciones con características mucho más generales. Esto implica que además se utilizan muchos más datos no-etiquetados para entrenar el *transformer* y así, que este pueda aprender representaciones con mejores características. Esto retoma la motivación principal de utilizar un *transformer*, el cual ha demostrado poder aprovecharse mejor de los datos no-etiquetados, en otra áreas de estudio, que el resto de los modelos neuronales.

3.2.4. Ajuste fino del modelo de clasificación

El ajuste fino del modelo de clasificación es, el entrenamiento con datos de eventos etiquetados del *transformer* pre-entrenado en datos no etiquetados. Es importante aclarar que las representaciones resultantes de un mismo modelo pre-entrenado, son utilizadas en distintos ajustes finos. Es decir, un modelo pre-entrenado se ajusta (entrena) en la base de datos de MASS-MODA, así como también la base de datos MASS-SS2 con husos de sueño y otro con la base MASS-SS2 con complejos-k, resultando en 3 instancias distintas para ser evaluadas en conjuntos de pruebas distintos. Esto permite evaluar no solo la efectividad del *Transformer* para detectar los eventos en cada tarea específica, sino que también permite evaluar la utilidad del *Transformer* para representar EEG con características generales que son ajustables en datos con etiquetas para resolver problemas particulares.

El ajuste fino en este trabajo tiene un desafío importante que es, el desbalance de clases. La ocurrencia de etiquetas positivas (evento) en segmentos de EEG etiquetados es una anomalía con respecto a la cantidad de etiquetas negativas (no evento). Es por esto que al momento de entrenar en las bases de datos no etiquetados, se le entregan con mayor frecuencia segmentos que contienen eventos al modelo y menos segmentos que no contienen eventos. Esto último permite reducir la cantidad de falsos negativos (no detectar el evento erróneamente) en la evaluación del modelo. Además, para evitar complicaciones de borde, al *transformer* siempre se le entregan eventos completos, es decir, los bordes de las señales (20 segundos) de entrenamiento nunca cortan un evento. Finalmente, durante el entrenamiento, se le entregan *batches* de datos equilibrados. Cada uno de los *batches* de entrenamiento está compuesto de una mitad de señales las cuales están bajo la mediana de suma de puntos de evento y de otra mitad de señales sobre la mediana de suma de puntos de evento. Esto último, ayuda a la red a ser más robusta en distintos entrenamientos bajo las mismas condiciones.

3.2.5. Entrenamiento

El entrenamiento del modelo, en la etapa de pre-entrenamiento y ajuste fino, fue realizado utilizando el optimizador AdamW propuesto por Loshchilov y Hutter (2019). Este optimizador es similar al optimizador Adam propuesto por Kingma y Ba (2015), pero la principal

deferencia es que AdamW implementa la regularización original por *weight decay* de Hanson y Pratt (1988), mientras que Adam utiliza la regularización L_2 en su lugar. Loshchilov y Hutter (2019) demuestran que la regularización L_2 no es equivalente a la regularización por*weight decay* de Hanson y Pratt (1988) en el caso de Adam (si para el optimizador SGD clásico) y, que además, la regularización por*weight decay* tiene mejores resultados. El valor del parámetro λ para aplicar *weight decay*; es decir, en la ecuación $\theta_t \leftarrow \theta_{t-1} - \lambda\theta_{t-1}$ previa a la actualización de parámetros; es de $\lambda = 0.0001$ para ambas etapas del entrenamiento.

Al igual que el optimizador, la tasa de aprendizaje para ambas etapas del entrenamiento también es la misma, con un valor de 0.00001. El *dropout* en las capas del *transformer* también se conserva en ambas etapas del entrenamiento, con una probabilidad del 10 % de eliminar las conexiones neuronales. Por último, se utilizan 25 épocas de entrenamiento para ambas etapas del entrenamiento, donde una época corresponde a todas las iteraciones que sean necesarias por *batches* para entrenar sobre todos los datos de entrenamiento.

Las diferencias que existen entre la etapa de pre-entrenamiento y ajuste fino del modelo, son primero, que en el ajuste fino del modelo, se utiliza un tamaño de batch de 32, versus un batch de 256 en pre-entrenamiento. Segundo, en la etapa de ajuste fino, se utiliza un *scheduler*, el cual divide la tasa de aprendizaje en 2, cada vez que la pérdida en el conjunto de validación no mejore por 2 épocas consecutivas, además de guardar el modelo en la época de entrenamiento con la menor pérdida en el conjunto de validación. En el caso del pre-entrenamiento, mediante una búsqueda exhaustiva de hiperparámetros y dado que el modelo no se sobreajusta en pre-entrenamiento con respecto al conjunto de validación, se encontró que 25 épocas mejora el desempeño general del modelo. Esto porque la extracción de características se concentra en los cabezales de atención y no en las capas *fully connected*. Por último, en el ajuste fino se utiliza un balanceo aleatorio de los *batches* de entrenamiento, lo cual no se realiza en pre-entrenamiento debido a que no hay etiquetas que balancear.

En cuanto a la inicialización, todos los pesos del modelo en pre-entrenamiento se inicializan por defecto mediante una distribución uniforme $\mathcal{U}(-\sqrt{k}, \sqrt{k})$. Donde, en el caso de las capas lineales, $k = (in\_features)^{-1}$ y en el caso de las capas convolucionales 1d, $k = (in\_features * kernel\_size)^{-1}$, teniendo que $in\_features = 256$ es la dimensionalidad elegida de los *embeddings* del modelo propuesto. Como en la etapa de ajuste fino no se deben inicializar los pesos del modelo, ya que se cargan los pesos del modelo pre-entrenado, solamente se inicializan los pesos del bloque de clasificación. Las capas lineales del bloque de clasificación se inicializan según la inicialización de Xavier (Glorot y Bengio, 2010) con distribución uniforme $\mathcal{U}(-a, a)$. Donde $a = g\sqrt{\frac{6}{in\_features+out\_features}}$. En el caso de la primera capa, la ganancia g es igual a $\sqrt{2}$ debido a la función de activación ReLU presente y la ganancia en la neurona de salida es igual a 1. Además de lo anterior, los sesgos de la primera capa lineal se inicializan en cero y el sesgo de la neurona de salida se inicializa en $log(\frac{p_1}{1-p_1})$, con $p_1 = 0.1$. Esto último implica que al comienzo del entrenamiento la probabilidad de la clase positiva será $\mathbb{P}(evento) \sim p_1 = 0.1$ evitando el dominio de la clase predominante (no-evento) al comienzo del entrenamiento, añadiendo estabilidad según lo recomendado por Lin, Goyal, Girshick, He, y Dollar (2020).

Por último, en los entrenamientos con las bases de datos etiquetadas se utiliza un criterio de *early stopping* (Gençay y Qi, 2001). Para los entrenamientos de cada uno de los modelos

utilizados en el presente trabajo, se utilizan los parámetros del modelo que obtuvo el menor valor en la función de pérdida para el conjunto de validación. En la figura 3.7 se observa la evolución de la función de pérdida en un entrenamiento de LeBENDR en la base de datos SS2-E1 de husos de sueño. En este caso, el modelo seleccionado para evaluar el desempeño en el conjunto de prueba sería el modelo obtenido al final de la época 10 de entrenamiento.

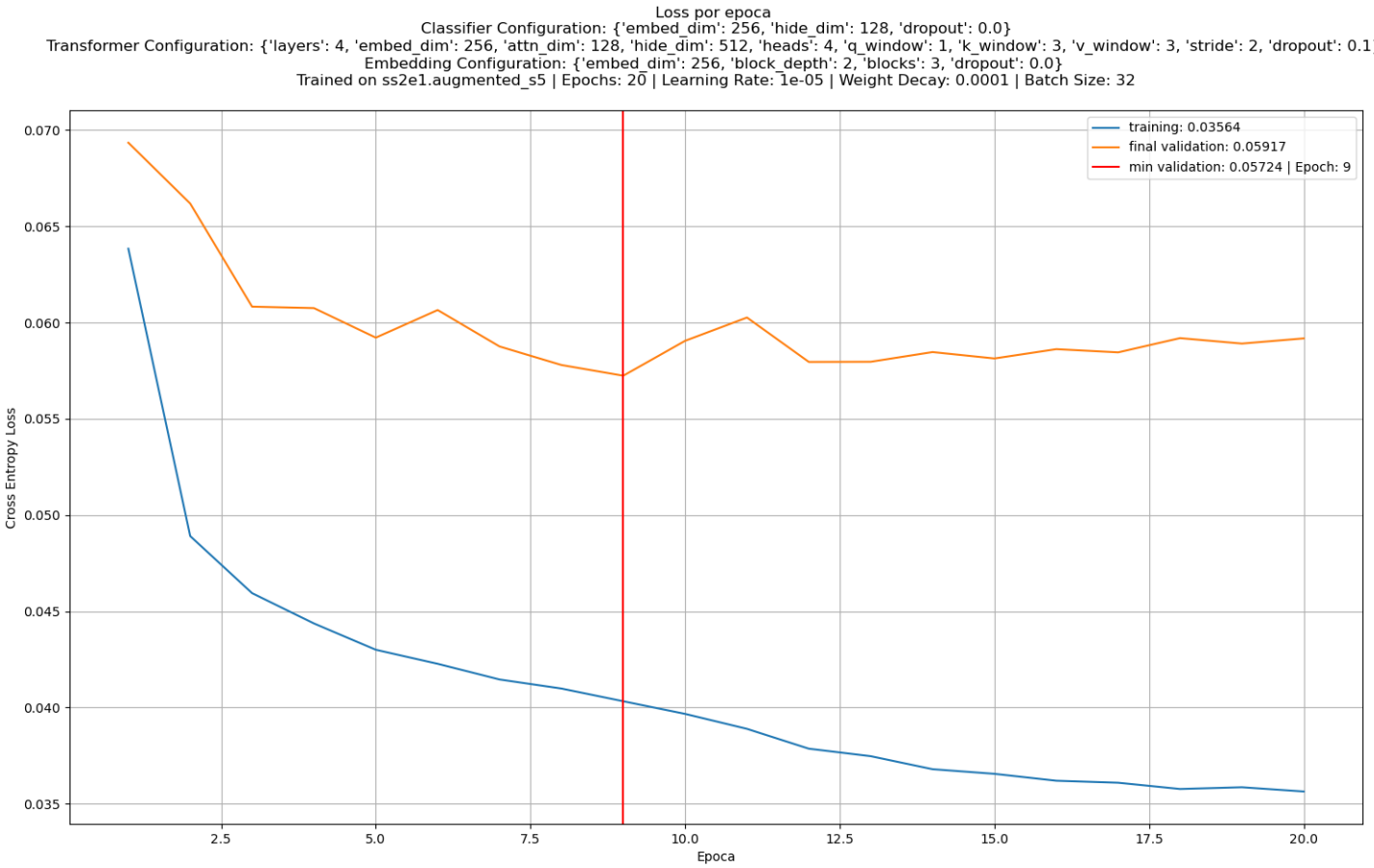

Figura 3.7: Función de pérdida en un entrenamiento de LeBENDR en una de las validaciones cruzadas. La linea roja representa la época donde se guardan los pesos del modelo (selección del modelo) al tener un valor mínimo para la función de perdida en el conjunto de validación.

3.2.6. Inferencia de las detecciones

Como ya se mencionó anteriormente, el LeBENDR está diseñado y entrenado para procesar segmentos de 20 segundos de duración, muestreadas a 200 Hz. Por lo tanto, al igual que en el entrenamiento, en inferencia para realizar las detecciones se procesan señales de 20 segundos. A diferencia con el entrenamiento en inferencia, se quiere formar una señal de probabilidad de evento completa. Para esto el modelo procesa señales completas de EEG, usando segmentos de 20 segundos y con un paso de 10 segundos. Cada inferencia de 20 segundos del modelo representa un señal de probabilidad positiva utilizando la función sigmoidea. Dentro de cada inferencia de 20 segundos, se conservan los 10 segundos centrales, evitando

así efectos de borde. Esto es, porque en inferencia se debe calcular la probabilidad de evento en cada uno de los puntos de una señal completa. Por lo que, al recorrer una señal completa con un paso de 10 segundos, puede ocurrir que en los bordes de los segmentos de 20 segundos hayan eventos recortados. Al ir completando todas las probabilidades de evento con solo 10 segundos (centrales) de los 20 segundos que procesa el LeBENDR, se evita este efecto de borde.

A medida que se construye la señal de probabilidad de evento, los 10 segundos centrales de salida del *transformer* se sobre-muestrean en un factor de 8, repitiendo cada probabilidad de salida 8 veces, para recuperar el largo de señal EEG de entrada al *transformer* dado que los bloques convolucionales reducen la resolución de la señal en un factor de 8. Una vez que se obtiene la señal de probabilidad de evento para una señal de EEG completa, y basándose en detectores tradicionales, se definen dos umbrales con respecto a la probabilidad de evento. Estos umbrales son, el umbral de detección, que determina la existencia de un evento, el cual se deja en 0.5 (50 % de probabilidad de evento como mínimo) y el umbral de duración, que determina la extensión de un posible evento detectado, el cual se fija en 0.425 (85 % de 0.5) al igual que en el trabajo de Tapia y Estevez (2020). Finalmente, en base a estos dos umbrales se obtienen dos señales con las etiquetas 0 o 1, determinadas por los dos umbrales distintos.

3.3. Evaluación del modelo

Como se mencionó anteriormente, las métricas de evaluación se calculan mediante validaciones cruzadas. Estas evaluaciones se realizan sobre los conjuntos de prueba de cada una de las 3 validaciones cruzadas. Para evaluar las detecciones del modelo propuesto, es necesario definir los inicios y comienzos del evento que predice el modelo. El problema con lo anterior, es que el modelo solo conoce etiquetar segmentos que abarcan 8 puntos de resolución o de 40ms. Esto introduce la necesidad de un procesamiento de las predicciones del modelo que permita definir, bajo ciertos estándares, cuando se detecta un evento, definir su comienzo y su final.

3.3.1. Procesamiento de detecciones

Como se mencionó anteriormente, es necesario procesar las detecciones realizadas por el modelo. Recordar que el modelo propuesto entrega 500 predicciones en 20 segundos (4000 puntos) de segmento muestreados a 200 Hz. Esto, debido a la reducción de resolución de las capas convolucionales. Por lo que antes de realizar cualquier procesamiento, se sobre-muestrean las probabilidades de evento en un factor de 8 para volver a la resolución de las etiquetas originales y se generan 2 etiquetas predichas por el modelo según 2 umbrales distintos, de detección y de duración. Lo anterior se realiza para ambos tipos de eventos transitorios del sueño.

Detecciones de husos de sueño

Siguiendo los procedimientos propuestos por Berry et al. (2018), se fusionan las muestras clasificadas por el umbral de duración, que estén a una distancia de menos de 0.3 segundos. Luego de este proceso, las muestras consecutivas clasificadas con más de 5 segundos de duración, se recortan a los 5 segundos centrales. Esto porque la duración de los husos de sueño está acotada según Purcell et al. (2017). Finalmente, luego de estos procesamientos, se eliminan todas las muestras consecutivas clasificadas por el umbral de duración que no contengan muestras clasificadas por el umbral de detección.

Detecciones de complejos-k

El procesamiento de las detecciones de los complejos-k, siguen el mismo procedimiento que las detecciones de husos de sueño. Sin embargo, luego de aplicar este procesamiento se aplica un procesamiento de separación de detecciones. Este algoritmo de separación de detecciones fue propuesto por Tapia y Estevez (2020) y se basa en la detección de picos negativos de detectores tradicionales (sin redes neuronales) como el propuesto por Lajnef et al. (2017). El proceso de separación de detecciones es necesario en el caso de los complejos-k, dado que, a diferencia de los husos de sueño, los complejos-k no tienen una separación mínima, por lo que el modelo muchas veces junta detecciones de eventos que normalmente son disjuntos. El proceso de separación de detecciones consiste en lo siguiente. Dentro de cada una de los eventos detectados, se aplica un filtro pasa bajos tipo *Butterworth* de orden 3, con una frecuencia de corte en 4 Hz (banda delta) para detectar los picos negativos. Dentro de los picos detectados, se ignora cualquier pico que esté más cerca que 0.05 segundos del comienzo del evento detectado y más cerca que 0.2 segundos del final. Los picos restantes son agrupados en sus puntos medios en caso de que hayan picos consecutivos sin un cruce por cero. Si luego de este paso aún queda más de un pico, el evento detectado se divide en el punto medio entre cada par de picos restantes consecutivos.

3.3.2. Métricas de desempeño

El análisis para evaluar los eventos detectados contra los eventos etiquetados, sigue el procedimiento de Warby et al. (2014). Para comparar un evento detectado y un evento etiquetado, se utiliza la razón entre la intersección y la unión de estos. Es importante que en una señal, cada evento detectado solo se puede comparar con un evento etiquetado y vice versa. Sean A y B un conjunto de muestras que definen un evento etiquetado y un evento detectado respectivamente, los cuales tienen una intersección de muestras no nula. Se tiene que el IoU (*intersection over union*) está dado por

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.1}$$

donde IoU $\in (0, 1]$, notando que si $A = B$, entonces $\text{IoU}(A, B) = 1$. De manera algo evidente, dentro de una señal con eventos detectados y eventos etiquetados, cada evento detectado se aparea con el evento detectado más próximo o con IoU > 0 (con traslape) de ser posible. Luego, se definen los verdaderos positivos o *true positive* (TP) a aquellos apareamientos donde IoU $> \mu$, con μ cierto umbral de exigencia dado donde $\mu \in (0, 1)$. En caso de que el apareamiento de dos eventos no cumpla con IoU $> \mu$, entonces eso implica la existencia de un falso negativo, o *false negative* (FN) y la existencia de un falso positivo, o *false positive* (FP). Esto significa directamente que todos los eventos detectados que no tienen intersección con eventos etiquetados son falsos positivos (FP) y todos los eventos etiquetados que no tienen intersección con eventos detectados son fasos negativos (FN). Finalmente, las métricas que se utilizan para evaluar y comparar el desempeño del modelo son, el F1-score y el mIoU, dando prioridad al F1-score. Esto vuelve muy importante definir para que umbral se calcula el F1-score, el cual, basándose en la literatura, se fija en un valor de $0.2 = \mu$.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.3}$$

$$\text{F1-score} = 2\frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}} \tag{3.4}$$

$$\text{mIoU} = \frac{1}{N}\sum_{i=1}^{N} \text{IoU}_i \tag{3.5}$$

Para calcular el F1-score y el mIoU para un conjunto de prueba de una partición de una las 3 validaciones cruzadas, se puede calcular a nivel de sujetos o a nivel del conjunto de prueba total, a esto se le llama el cálculo de métricas por macro-promedio y micro-promedio respectivamente. El cálculo de las métricas por macro-promedio es el calculo de las métricas por sujeto dentro de un conjunto de prueba y luego promediar el valor de las métricas por sujeto. Por otro lado, el calculo por micro promedio calcula las métricas directamente en el conjunto de prueba. Esto implica que, por micro-promedio, cada evento detectado o etiquetado tiene la misma influencia sobre la métrica, mientras que, por macro-promedio, los eventos dentro de los sujetos con mayor cantidad de eventos, serán menos influyentes en las métricas. En este caso, se utiliza el micro-promedio para el calculo general de métricas y así tener el mismo tipo de calculo de desempeños en todas las bases de datos etiquetadas, ya que, en el caso de la base de datos de MODA se tienen pocos ejemplos por sujeto, por lo que algunas métricas podrían resultar distorsionadas.

En la figura 3.8, se observa un ejemplo de las métricas y resultados de inferencia en el conjunto de prueba en una de las particiones de validación cruzada de la base de datos SS2-E1 de husos de sueño. En el gráfico de la izquierda se puede ver el F1-Score promedio (micro-promedio) para el umbral de IoU en 0.2, además de la cantidad de falsos positivos "absolutos" (detecciones que no intersectan con ninguna etiqueta) y de falsos negativos "absolutos" (etiquetas que no intersectan con ninguna detección). En el gráfico central de la figura 3.8, se puede ver como la *precision* y el *recall* disminuyen a medida que aumenta el umbral del IoU. Esto es normal ya que al aumentar el umbral, se van agregando un falso positivo y un falso negativo por cada verdadero positivo que se pierde por el aumento del umbral. Finalmente, en el gráfico de la derecha se puede ver el histograma de los IoUs y el IoU promedio (micro-promedio.)

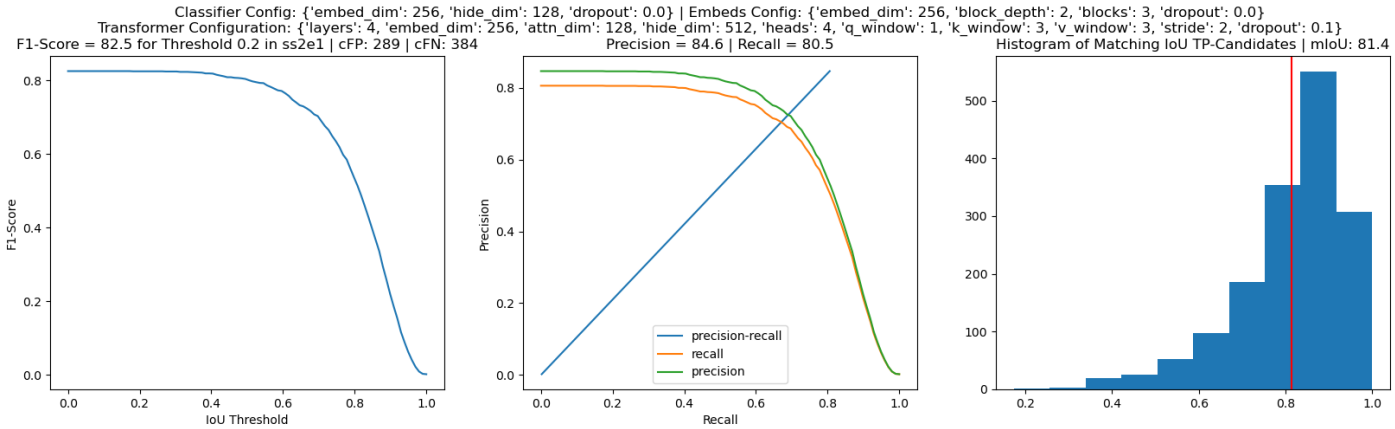

Figura 3.8: Resultados en inferencia de LeBENDR para el conjunto de prueba en alguna partición de la validación cruzada de SS2-E1

Capítulo 4

Resultados y discusiones

En este capítulo se presentan los resultados y discusiones de los distintos modelos mencionados en este trabajo de tesis. Cada modelo se evalúa en las 3 bases de datos con eventos etiquetados enunciadas anteriormente. Es importante destacar, que la principal métrica para comparar los modelos es el F1-Score, dado que en la literatura es la más usada, pero además porque se considera que es más importante el hecho de contabilizar cuando hay una intersección en la detección, más que el hecho de que estás intersecciones sean perfectas. Los modelos a comparar, diferenciados por su arquitectura, son los siguientes.

LeBENDR

Este es el nombre que se utiliza en los resultados para referirse al modelo con la arquitectura propuesta. Este modelo se destaca principalmente por tener un *transformer* como contextualizador y donde el mecanismo de atención incluye capas convolucionales en el cómputo de *keys* y *values*, con un paso para reducir el cómputo de ponderadores de atención.

LeBENDR - LE original

Este es el nombre que se utiliza en los resultados para referirse al modelo que tiene una arquitectura similar a la propuesta pero con la diferencia de que el mecanismo de atención se basa netamente en la atención propuesta por Li et al. (2019). Esto implica que las capas convolucionales con ventanas mayores que uno, solamente se utilizan para el cálculo de los vectores de consulta y *keys*, por lo que, no se reduce la cantidad de ponderadores de atención. Es importante destacar que este modelo es equivalente al modelo propuesto en todo el resto de sus ámbitos a nivel de arquitectura.

LeBENDR - MHA original

Este es el nombre que se utiliza para referirse al modelo con una arquitectura donde el contextualizador también es un *transformer*, pero con la diferencia que no se utilizan convoluciones en el cómputo de los ponderadores de atención. El mecanismo de atención utilizado por este modelo es equivalente al mecanismo de atención multi-cabezal del *transformer* original propuesto por (Vaswani et al., 2017). Es importante destacar que este modelo también es equivalente al modelo propuesto en todo el resto de sus ámbitos a nivel de arquitectura.

RED-Time

Este es el nombre que se utiliza en los resultados para referirse al modelo del estado del arte (Tapia y Estevez, 2020). La principal diferencia de este modelo radica en el contextualizador, el cual se compone de capas Bi-LSTM. Sin embargo, este modelo comparte la misma arquitectura en el extractor de características y en el clasificador con el resto de los modelos analizados en esta sección.

4.1. Tiempos de cómputo

Los tiempos de cómputo de los modelos se enuncian en la tabla 4.1. Todos estos tiempos se miden en una misma GPU en el mismo servidor facilitado por el laboratorio de inteligencia computacional de la Facultad de Ciencias Físicas y Matemáticas de la Universidad de Chile. El modelo de esta GPU es una **NVIDIA GTX 1080Ti**, la cual tiene aproximadamente 11Gb de memoria VRAM.

Los resultados de los tiempos de cómputo fueron medidos utilizando un lote (*batch*) de 32 ejemplos para computar un procesamiento sin propagación de gradientes (*forward*) y un procesamiento con propagación de gradiente (*forward backward*). Estos resultados muestran el promedio y la desviación estándar de 1000 cómputos en cada caso, lo que permite tener una buena aproximación de cuanto se demora el cómputo de un solo lote en cada caso, considerando que un entrenamiento completo tendrá $\frac{N}{B}$ cómputos, donde N es la cantidad de ejemplos totales y B, la cantidad de ejemplos por lote. Por último, además de los tiempos de cómputo, se muestra la cantidad aproximada de parámetros en millones que tiene cada contextualizador de cada modelo.

Tabla 4.1: Tiempos de cómputo de inferencia (*Forward*) y entrenamiento (*Forward Backward*) de los modelos

Modelo	Forward	Forward Backward	Parámetros
RED-Time	46.7 ± 1.7 [ms]	98.2 ± 2.7 [ms]	2.63M
LeBENDR	45.8 ± 1.1 [ms]	199.9 ± 7.7 [ms]	5.26M
LeBENDR - LE original	60.8 ± 0.7 [ms]	181.1 ± 7.2 [ms]	5.26M
LeBENDR - MHA original	54.2 ± 0.9 [ms]	172.1 ± 7.3 [ms]	3.16M

En la tabla 4.1 se observa que el modelo más rápido en tiempo de inferencia (*forward*) es el modelo con el mecanismo de atención propuesta. Esto se puede afirmar sin necesidad de realizar un test de hipótesis estadístico de diferencia de medias (t-test de Student, 1908), ya que la cantidad de cómputos promediados es suficientemente grande (1000) para que el estimador de la varianza de la diferencia de medias tienda a cero. Si bien el modelo propuesto posee el doble de parámetros que el modelo del estado del arte que utiliza Bi-LSTM como contextualizador, se puede ver que su tiempo es ligeramente menor, lo que comprueba la gran capacidad de paralelización de cómputo del *transformer*. Además, el mecanismo de atención propuesta es mejor en tiempos de inferencia que los otros dos mecanismos de atención con

los que se está comparando. Esto es principalmente debido a la cantidad de ponderadores de atención, ya que, en el mecanismo de atención, la cantidad de operaciones es proporcional al número de representaciones de las *keys* y los *values*, y como ya se mencionó anteriormente, el modelo con el mecanismo de atención propuesto disminuye este número a la mitad, lo que a su vez se traduce en la mitad de ponderadores de atención.

Sin embargo, cuando los tiempos de cómputo incluyen la propagación de gradientes, se ve que el modelo propuesto es el que tiene el mayor costo. Las principales causas de este fenómeno son difíciles de identificar o teorizar, dado que los flujos de gradiente por los cuales se generan los cómputos de los mismos, son difíciles de representar y entender. Sin embargo, independientemente de la causa, este tiempo es menos relevante que el anterior si se trata de comparar estos modelos en un ambiente productivo, ya que en un ambiente productivo se trabaja la mayoría del tiempo en inferencia, y los entrenamientos (que es cuando se computa la propagación de gradiente) ocurren un número de veces limitado y definido.

4.2. Desempeño de modelos

A continuación se muestran los resultados de *F1-Score*, *Recall* y *Precision*, obtenidos para las distintas arquitecturas a comparar en las 3 bases de datos con etiquetas de eventos transitorios del sueño. Además se comparan distintos métodos de entrenamiento incluyendo varias bases de datos de pre-entrenamiento, además de variaciones en la tarea de pre-entrenamiento. Estos resultados se obtienen mediante una validación cruzada de 5 *folds*, con 3 combinaciones distintas de sujetos. Esto último quiere decir que los promedios y las desviaciones estándar mostradas en las siguientes sub-secciones son obtenidas mediante 15 evaluaciones de conjuntos de prueba distintos.

En las siguientes sub-secciones se comparan distintos desempeños, principalmente en base a la métrica *F1-Score*. Esta es la métrica más adecuada para comparar los desempeños dado que combina a la vez el comportamiento de los falsos positivos y los falsos negativos, los cuales son valores que se quieren minimizar. Esto dado que los falsos positivos representan la detección errónea de un evento con un $IoU \geqslant 0.2$, que los falsos negativos son una errónea falta de detección con un $IoU < 0.2$ y que el *F1-Score* es una media armónica entre *Precision* y *Recall*.

Notar que las comparaciones de los desempeños vienen acompañadas con un análisis estadístico respecto a la confianza en la significancia de las diferencias de desempeño. Esto se realiza mediante un proceso similar a un test de hipótesis estadístico de diferencia de medias (t-test de Student, 1908), pero con la diferencia en que no se define un nivel de significancia arbitrario para rechazar o no la hipótesis de existencia de diferencia, sino que se calcula el complemento del p-valor ($1 - p_{value}$), lo que en palabras más generales representa la probabilidad o confianza con la que esta diferencia **no** fue una "casualidad". Este análisis resulta más útil que rechazar o no una hipótesis de existencia de diferencia significativa, ya que no es necesaria la determinación de un nivel de significancia que, al fin y al cabo, es un valor de uso convencional más bien "arbitrario" (1 %, 5 %, 10 %, etc). Además, esto permite prescindir de la decisión de testear la diferencia o testear la hipótesis de que un valor sea mayor que otro, ya que la masa de probabilidad complementaria al p-valor es la misma en ambos casos de test de hipótesis.

4.2.1. Modelos sin pre-entrenamiento

En esta primera subsección se muestran los resultados obtenidos del modelo con el mecanismo de atención propuesto y el modelo del estado del arte cuyo contextualizador está basado en redes recurrentes Bi-LSTM. Estos modelos fueron primero evaluados sin pasar por un pre-entrenamiento previo. Sin embargo, estos modelos sí fueron entrenados en cada una de las bases de datos etiquetadas enunciadas anteriormente, partiendo de pesos aleatorios según el método de inicialización por defecto en las capas lineales, convolucionales y Bi-LSTM que pone a disposición la librería *PyTorch*.

En la tabla 4.2 se muestran los resultados en la base de datos de husos de sueño MODA. En esta tabla se puede ver claramente que el modelo RED-Time tiene un mejor desempeño que el modelo propuesto basado en atención neuronal. La diferencia entre ambos desempeños, considerando su desviación estándar, es significativa con un nivel de confianza que permite rechazar completamente que la diferencia de promedios haya sido un evento azaroso.

Tabla 4.2: Desempeño de detección de husos de sueño en MODA

Modelo	**F1-Score**	Precision	Recall
LeBENDR	81.33 ± 1.11	81.57 ± 3.59	81.41 ± 3.98
RED-Time	$\mathbf{82.38} \pm 1.02$	83.46 ± 2.05	81.45 ± 2.81

Diferencia significativa con 99.7 % de confianza

En la tabla 4.3 se muestran los resultados en la base de datos de husos de sueño SS2-E1. En esta tabla se puede ver que ambos modelos tuvieron prácticamente el mismo desempeño y que la pequeña diferencia entre ambos promedios se debe casi de forma segura a un evento azaroso. Sin embargo, se puede ver que el modelo RED-Time fue ligeramente más consistente, pero no se realizó un análisis estadístico que permita asegurar que RED-Time tiene menos variabilidad en su desempeño.

Tabla 4.3: Desempeño de detección de husos de sueño con modelos sin pre-entrenamiento en SS2-E1

Modelo	**F1-Score**	Precision	Recall
LeBENDR	81.07 ± 2.26	78.13 ± 6.92	85.12 ± 4.78
RED-Time	$\mathbf{81.08} \pm 1.83$	81.17 ± 5.94	81.69 ± 5.26

Diferencia significativa con 50.5 % de confianza

En la tabla 4.4 se muestran los resultados en la base de datos de complejos-k SS2-KC. En esta tabla se puede ver que nuevamente el modelo RED-Time tiene un mejor desempeño que LeBENDR. Además, la diferencia entre ambos desempeños es significativa con un nivel de confianza que permite rechazar completamente que la diferencia de promedios haya sido un evento azaroso.

Tabla 4.4: Desempeño de detección de complejos-k con modelos sin pre-entrenamiento en SS2-KC

Modelo	**F1-Score**	Precision	Recall
LeBENDR	80.77 ± 1.73	80.67 ± 3.16	81.06 ± 3.36
RED-Time	$\mathbf{82.47} \pm 1.54$	82.64 ± 3.41	82.47 ± 2.60

Diferencia significativa con 99.8 % de confianza

A modo de discusión, se puede decir que entre los dos modelos comparados, RED-Time tiene un mejor desempeño que LeBENDR. Es necesario destacar que si bien este primer resultado no es alentador para el modelo propuesto, este resultado no considera un pre-entrenamiento de los modelos, que es una de las razones por las que se propone usar un *transformer* como contextualizador. Además, si bien la diferencia es significativa en 2 de las bases de datos, esta diferencia no es tan grande, lo que permite considerar a LeBENDR como una alternativa viable a RED-Time.

4.2.2. Modelos pre-entrenados

En esta subsección se muestran los desempeños del modelo con el mecanismo de atención propuesta (LeBENDR) y RED-Time. Sin embargo, en este caso, ambos modelos fueron pre-entrenados en las bases de datos no-etiquetadas de NSRR. Este pre-entrenamiento fue realizado como se describió en el capítulo anterior, por lo que, para el entrenamiento y evaluación en las bases de datos etiquetadas, solamente se tiene inicialización aleatoria de pesos en el bloque de clasificación como también se describió anteriormente. Cabe destacar que es posible pre-entrenar RED-Time exactamente igual que como se pre-entrena LeBENDR ya que RED-Time tiene una arquitectura equivalente a la de LeBENDR donde solo se diferencian por el contextualizador, que en el caso de RED-Time es un conjunto de capas Bi-LSTM y en el caso de LeBENDR es un conjunto de *transformer layers*, pero ambos generan una secuencia de representaciones del EEG previas a la clasificación.

En la tabla 4.5 se muestran los resultados en la base de datos de husos de sueño etiquetados MODA, para los modelos pre-entrenados en NSRR. Los resultados son bastante similares entre ambos modelos, donde la diferencia en el desempeño entre ellos es con absoluta probabilidad un evento azaroso. Además, se puede ver que en comparación con los resultados sin pre-entrenamiento de la tabla 4.2, existe un gran aumento en el desempeño de LeBENDR, a diferencia de RED-Time donde ocurre el efecto contrario, ya que el desempeño parece disminuir cuando este modelo pasa por una etapa de pre-entrenamiento. Por último, se puede ver que la variabilidad de LeBENDR disminuyó considerablemente con el pre-entrenamiento, llegando a ser menor que la de RED-Time, la cual no varió con respecto al desempeño sin pre-entrenamiento.

Tabla 4.5: Desempeño de detección de husos de sueño con modelos pre-entrenados (NSRR) en MODA

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	**82.17** $\pm$ 0.92	83.73 $\pm$ 2.07	80.75 $\pm$ 2.10]
RED-Time (NSRR)	82.14 $\pm$ 1.02	84.16 $\pm$ 2.13	80.29 $\pm$ 2.25

Diferencia significativa con 53.4 % de confianza

En la tabla 4.6 se muestran los resultados en la base de datos de husos de sueño etiquetados SS2-E1, para los modelos en cuestión pre-entrenados en NSRR. En esta tabla se aprecia que LeBENDR mejora su desempeño con respecto a su resultado sin pre-entrenamiento, mientras que, RED-Time mantiene su desempeño pero aumentando ligeramente su variabilidad con respecto a los resultados sin pre-entrenamiento de la tabla 4.3. Sin embargo, la diferencia no es suficientemente significativa como para descartar la posibilidad de que sea un evento azaroso, pero si puede decirse que es probable que no lo sea. Por último, al igual que en los resultados de MODA, se puede ver que la variabilidad de LeBENDR disminuyó considerablemente con el pre-entrenamiento, llegando a ser menor que la de RED-Time sin pre-entrenamiento mientras que la variabilidad de RED-Time aumentó con pre-entrenamiento.

Tabla 4.6: Desempeño de detección de husos del sueño con modelos pre-entrenados (NSRR) en SS2-E1

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	**81.45** $\pm$ 1.67	79.56 $\pm$ 5.51	84.06 $\pm$ 4.68
RED-Time (NSRR)	81.04 $\pm$ 2.07	81.95 $\pm$ 5.75	80.87 $\pm$ 5.69

Diferencia significativa con 72.5 % de confianza

En la tabla 4.7 se muestran los resultados en la base de datos de complejos-k etiquetados, SS2-KC para los modelos en cuestión pre-entrenados en NSRR. En esta tabla se puede ver que esta base de datos es la única en la que RED-Time mantiene un mejor desempeño que LeBENDR, aunque esta diferencia no sea significativa en general. Sin embargo, sí se observa la misma tendencia que en las otras bases de datos, donde LeBNDR aumenta su desempeño cuando pasa por una etapa de pre-entrenamiento y RED-Time incluso disminuye su desempeño.

Tabla 4.7: Desempeño de detección de complejos-k con modelos pre-entrenados en SS2-KC

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	82.04 $\pm$ 1.69	81.81 $\pm$ 2.72	82.39 $\pm$ 2.90
RED-Time (NSRR)	**82.21** $\pm$ 1.78	82.60 $\pm$ 3.37	82.01 $\pm$ 3.27

Diferencia significativa con 60.6 % de confianza

Como se puede ver en las tres tablas enunciadas en esta subsección, se puede afirmar con cierta confianza que LeBENDR tiene un mejor desempeño para identificar husos de sueño

que RED-Time cuando se pasa por una etapa de pre-entrenamiento. Además, la variabilidad de LeBENDR disminuyó considerablemente, mientras que, la variabilidad de RED-Time con pre-entrenamiento aumento en SS2-E1. En el caso de los complejos-k, RED-Time sigue teniendo un mejor desempeño pero en ambos casos se puede ver que LeBNDR aumenta considerablemente su desempeño al pasar por un pre-entrenamiento, mientras que RED-Time incluso disminuye su desempeño cuando se somete a un pre-entrenamiento. Estos resultados respaldan la hipótesis de que un modelo con un *transformer* en su contextualizador puede aprovecharse mejor de las bases de datos de EEG no etiquetadas, las cuales son abundantes en la literatura.

4.2.3. Transferencia de aprendizaje

En esta subsección se muestran los resultados de distintas instancias del modelo propuesto, con distintas combinaciones de bases de datos usadas para pre-entrenamiento, con el fin de analizar el efecto de la transferencia de aprendizaje. En las siguientes tablas de resultados se muestra el desempeño del modelo propuesto pre-entrenado en NSRR (mismo resultado mostrado en la subsección anterior), otro pre-entrenado en MASS y por último el modelo pre-entrenado en NSRR y luego pre-entrenado en MASS (NSRR → MASS), recordando que MODA y SS2 son subconjuntos de MASS (pre-entrenamiento de lo más general a lo más específico). Al tener tres desempeños a comparar, el cálculo de diferencias estadísticamente significativas se realiza con respecto al modelo mostrado en la subsección anterior (pre-entrenado en NSRR), el cual se utiliza como referencia. Finalmente, al igual que en la subsección anterior, para el entrenamiento en las bases de datos con eventos transitorios etiquetados se inicializaron los pesos de la capa de clasificación de la forma en que se describió en el capítulo anterior.

En la tabla 4.8 se muestran los resultados en la base de datos MODA de husos de sueño etiquetados. En esta tabla se puede ver que LeBENDR pre-entrenado en MASS tiene un mejor desempeño que LeBENDR pre-entrenado en NSRR. No obstante, la diferencia en los desempeños es significativa con un 63.5 % de confianza, lo cual nos dice que es probable que la diferencia pueda deberse a factores aleatorios. Por otro lado, el doble pre-entrenamiento (NSRR → MASS) obtiene el mejor desempeño y además tiene una diferencia que es significativa con un 74.6 % de confianza con respecto a LeBENDR pre-entrenado solo en NSRR. Esto último nos dice que, si bien no se puede descartar la posibilidad de que las diferencias se deban a factores aleatorios, existe cierto nivel de confianza en que tener más pre-entrenamiento tiene un efecto positivo en el desempeño del modelo.

Tabla 4.8: Desempeño en MODA de modelos pre-entrenados en distintas bases de datos

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	82.17 ± 0.92	83.73 ± 2.07	80.75 ± 2.10
LeBENDR (MASS) [a]	82.29 ± 0.99	83.53 ± 2.00	81.16 ± 2.27
LeBENDR (NSRR → MASS) [b]	$\mathbf{82.40} \pm 0.98$	83.50 ± 1.91	81.39 ± 1.80

[a] Diferencia significativa con 63.5 % de confianza
[b] Diferencia significativa con 74.6 % de confianza

En la tabla 4.9 se muestran los resultados en la base de datos SS2-E1 con husos de sueño etiquetados. En esta tabla se puede ver que al igual que en el caso de MODA, los desempeños de LeBENDR pre-entrenados en NSRR y en MASS por separado son muy similares, la diferencia entre ellos es con gran probabilidad un evento aleatorio. Esta similitud con los resultados en MODA también se puede observar para el caso de LeBENDR con doble pre-entrenamiento, el cual tiene un mayor desempeño que LeBENDR pre-entrenado solamente con NSRR. Esta última diferencia, al igual que los resultados en MODA, tampoco permite rechazar la posibilidad de que existe una diferencia a causa de la aleatoriedad, pero si se puede afirmar con cierto nivel de confianza que con más pre-entrenamiento se obtiene un mejor desempeño.

Tabla 4.9: Desempeño en SS2-E1 de modelos pre-entrenados en distintas bases de datos

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	81.45 ± 1.67	79.56 ± 5.51	84.06 ± 4.68
LeBENDR (MASS) [a]	81.42 ± 1.93	79.02 ± 6.23	84.74 ± 4.71
LeBENDR (NSRR → MASS) [b]	$\mathbf{81.85} \pm 1.77$	79.31 ± 5.82	84.53 ± 4.36

[a] Diferencia significativa con 51.8 % de confianza
[b] Diferencia significativa con 73.8 % de confianza

En la tabla 4.10 se muestran los resultados en la base de datos SS2-KC con complejos-k etiquetados. En esta tabla se puede ver que LeBENDR pre-entrenado en NSRR tiene un mejor desempeño que LeBENDR pre-entrenado en MASS, pero la diferencia no es estadísticamente muy significativa. LeBENDR pre-entrenado en NSRR y MASS, al igual que en las otras bases de datos etiquetadas, presenta un mejor desempeño que LeBENDR pre-entrenado en NSRR, pero esta diferencia se debe muy probablemente a factores aleatorios.

Tabla 4.10: Desempeño en SS2-KC de modelos pre-entrenados en distintas bases de datos

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	82.04 ± 1.69	81.81 ± 2.72	82.39 ± 2.90
LeBENDR (MASS) [a]	81.77 ± 1.58	82.16 ± 3.05	81.51 ± 2.50
LeBENDR (NSRR → MASS) [b]	$\mathbf{82.13} \pm 1.52$	82.10 ± 3.41	82.32 ± 2.42

[a] Diferencia significativa con 67.4 % de confianza
[b] Diferencia significativa con 56.1 % de confianza

Los resultados de las tres tablas anteriores muestran que, a pesar de que las diferencias no son estadísticamente muy significativas, consistentemente se obtienen mejoras en el desempeño cuando se pre-entrena por segunda vez en una segunda base de datos. Esto permite reafirmar la hipótesis de la utilidad del pre-entrenamiento cuando el contextualizador del modelo está basado en un *transformer*. Sin embargo, esta afirmación tiene mayor solidez cuando se trata de husos de sueño, dado que las diferencias en la base de datos con complejos-k son menos significativas con respecto a LeBENDR pre-entrenado por segunda vez.

4.2.4. Variaciones del mecanismo de atención

En esta subsección se muestran los resultados de tres arquitecturas cuyo contextualizador es un *transformer*. Estos tres modelos se diferencian netamente en el mecanismo de atención que implementan las capas del *transformer* respectivo. El mecanismo de atención que implementa cada una de las arquitecturas corresponde a las enunciadas al comienzo de la sección 4, LeBENDR (propuesta), LeBENDR - LE original y LeBENDR - MHA original. Al igual que en la subsección anterior, que muestra los resultados de la transferencia de aprendizaje, se calcula la significancia estadística de la diferencia en torno al modelo propuesto (LeBENDR). Por último, cabe destacar que todos los desempeños de las arquitecturas en esta subsección fueron obtenidos con un pre-entrenamiento en NSRR, previo al entrenamiento en cada una de las bases de datos con eventos transitorios etiquetados.

En la tabla 4.11 se muestran los resultados en la base de datos MODA de husos de sueño etiquetados. En esta tabla se puede ver que el modelo con la atención propuesta tiene un desempeño claramente superior al mismo modelo pero con la atención de Li et al. (2019) y con la atención del *Transformer* original sin capas convolucionales. Esta diferencia en los desempeños de los modelos es significativa por lo que se puede rechazar totalmente la posibilidad de que esta diferencia exista por factores aleatorios. Por último, sumado a lo anterior, se puede ver además que el modelo es más consistente con respecto a sus resultados ya que su variabilidad también es menor.

Tabla 4.11: Desempeño en MODA de modelos con distintos mecanismos de atención

Modelo	**F1-Score**	Precision	Recall
LeBENDR - propuesto (NSRR)	**82.17** $\pm$ 0.92	83.73 $\pm$ 2.07	80.75 $\pm$ 2.10
LeBENDR - LE original (NSRR) [a]	80.45 $\pm$ 1.32	79.39 $\pm$ 2.78	81.68 $\pm$ 2.51
LeBENDR - MHA original (NSRR) [b]	80.85 $\pm$ 1.54	81.64 $\pm$ 3.69	80.42 $\pm$ 4.29

[a] Diferencia significativa con 99.9 % de confianza
[b] Diferencia significativa con 99.8 % de confianza

En la tabla 4.12 se muestran los resultados en la base de datos SS2-E1 de husos de sueño etiquetados. En esta tabla se puede ver que LeBENDR con la atención propuesta tiene un mejor desempeño que los otros dos modelos de atención. Sin embargo, esta diferencia en el desempeño es significativa solamente con respecto al modelo con la atención propuesta por Li et al. (2019) y la diferencia con respecto al modelo con el mecanismo de atención original no es significativa y probablemente se deba a factores azarosos. A pesar de esto último, LeBENDR con el mecanismo de atención propuesta tiene resultados más consistentes (menos variables) que LeBENDR con el mecanismo de atención original.

Tabla 4.12: Desempeño en SS2-E1 de modelos con distintos mecanismos de atención

Modelo	**F1-Score**	Precision	Recall
LeBENDR - propuesto (NSRR)	**81.45** $\pm$ 1.67	79.56 $\pm$ 5.51	84.06 $\pm$ 4.68
LeBENDR - LE original (NSRR) [a]	80.54 $\pm$ 1.81	78.44 $\pm$ 5.78	83.41 $\pm$ 4.58
LeBENDR - MHA original (NSRR) [b]	81.32 $\pm$ 2.12	78.43 $\pm$ 6.52	85.21 $\pm$ 4.55

[a] Diferencia significativa con 92.4 % de confianza
[b] Diferencia significativa con 57.4 % de confianza

En la tabla 4.13 se muestran los resultados en la base de datos SS2-KC de complejos-k etiquetados. En esta tabla se puede ver que LeBENDR con el modelo de atención propuesto tiene un mejor desempeño que los otros dos modelos con distintos mecanismos de atención. Al igual que para los resultados en SS2-E1, la diferencia es significativa solamente con respecto al modelo con la atención propuesta por Li et al. (2019) y la diferencia con respecto al modelo con el mecanismo de atención original no es significativa y probablemente se deba a factores azarosos. No obstante, en esta ocasión también se tiene que LeBENDR con el mecanismo de atención propuesto tiene menor variabilidad.

Tabla 4.13: Desempeño en SS2-KC de modelos con distintos mecanismos de atención

Modelo	**F1-Score**	Precision	Recall
LeBENDR - Propuesto (NSRR)	**82.04** $\pm$ 1.69	81.81 $\pm$ 2.72	82.39 $\pm$ 2.90
LeBENDR - LE original (NSRR) [a]	80.15 $\pm$ 1.79	80.39 $\pm$ 2.45	80.02 $\pm$ 3.06
LeBENDR - MHA original (NSRR) [b]	81.88 $\pm$ 2.13	81.17 $\pm$ 3.06	81.72 $\pm$ 3.20

[a] Diferencia significativa con 99.9 % de confianza
[b] Diferencia significativa con 59.1 % de confianza

Los resultados descritos en esta subsección, permiten afirmar que el mecanismo de atención propuesto presenta una mejora general en el desempeño para la detección de eventos transitorios. No solamente aumenta el desempeño en promedio en varios casos con una mayor confianza, sino que también la variabilidad es menor. Esto respalda la hipótesis que disminuir la cantidad de ponderadores de atención y generar nuevas representaciones locales entre las representaciones sobre las cuales se está poniendo atención (y no sobre las que están poniendo atención), permite tener un modelo más robusto, eficaz y eficiente. Estos resultados son probablemente la contribución más importante de este trabajo de tesis, ya que no se ha encontrado en la literatura un mecanismo de atención que preste atención sobre la cantidad de ponderadores disponibles, junto con comprimir representaciones de manera local en las *keys* y *values* de la forma en la que se propone en este trabajo de tesis.

4.2.5. Entropía en pre-entrenamiento

En esta última subsección se muestran los resultados de distintos pre-entrenamientos con respecto a la función de pérdida en la tarea contrastiva. Los dos modelos que se comparan se basan en el mecanismo de atención propuesto y están pre-entrenados en las bases de datos

no etiquetadas de NSRR. La diferencia entre estas instancias radica netamente en la función de pérdida utilizada en el pre-entrenamiento. La función de pérdida que utiliza LeBENDR (propuesto) tiene un componente que busca maximizar la entropía de las representaciones aprendidas por el extractor de características, tal como se describe en el capítulo 3. Sin embargo, esta componente en la función de pérdida no está presente en el modelo de BENDR de Kostas et al. (2021) pero sí lo está en el modelo de Wav2Vec 2.0 de (Baevski et al., 2020).

Existen dos razones principales por las que se propone utilizar esta componente en la función de pérdida. Primero, maximizar la entropía de la salida del extractor de características (entrada al contextualizador) significa que cada representación, vista como una distribución, esté distribuida de manera más uniforme. Esto evita que las representaciones aprendidas se concentren mucho en algunos puntos debido a que el contextualizador está tratando de reconstruir representaciones contrastándolas (en ángulo) unas con otras. Segundo, el gradiente puede fluir con mayor facilidad hasta las capas iniciales si existe una inyección de flujo de gradiente entre el extractor de características y el contextualizador. Esto último permite disminuir cualquier efecto que pueda existir en relación al desvanecimiento de gradiente, lo que facilita el entrenamiento. Finalmente, además de las razones mencionadas, la inclusión de esta componente tiene un costo computacional despreciable.

En la tabla 4.14 se muestran los resultados en la base de datos MODA de husos de sueño etiquetados. En esta tabla se puede ver que el modelo pre-entrenado considerando la componente de entropía tiene un desempeño muy similar que el modelo pre-entrenado sin considerar entropía, con una pequeña diferencia en favor del modelo que considera la entropía. Si bien esta diferencia no es estadísticamente significativa, es decir, que la deferencia que existe entre los desempeños se debe a un factor netamente aleatorio, se puede ver que el modelo que maximiza la entropía en pre-entrenamiento tiene resultados ligeramente más consistentes.

Tabla 4.14: Desempeño en MODA de modelos pre-entrenados con distintas funciones de perdida

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	**82.17** $\pm$ 0.92	83.73 $\pm$ 2.07	80.75 $\pm$ 2.10
LeBENDR - Sin entropía (NSRR)	82.15 $\pm$ 1.01	83.73 $\pm$ 2.05	80.68 $\pm$ 1.68

Diferencia significativa con 52.3 % de confianza

En la tabla 4.15 se muestran los resultados en la base de datos SS2-E1 de husos de sueño etiquetados. En esta tabla se puede ver que el modelo pre-entrenado considerando la componente de entropía también tiene aparentemente un mejor desempeño. Sin embargo, aunque la diferencia en este caso sea más significativa, la probabilidad de que ésta sea un evento aleatorio sigue siendo alta. Además se puede ver nuevamente que los resultados del modelo que considera la componente de maximización de entropía en el pre-entrenamiento, son escasamente menos variables. No obstante, esta diferencia en la variabilidad es bastante pequeña y menor que en los resultados enunciados para MODA.

Tabla 4.15: Desempeño en SS2-E1 de modelos pre-entrenados con distintas funciones de perdida

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	**81.45** $\pm$ 1.67	79.56 $\pm$ 5.51	84.06 $\pm$ 4.68
LeBENDR - Sin entropía (NSRR)	81.23 $\pm$ 1.72	79.94 $\pm$ 6.11	83.30 $\pm$ 4.94

Diferencia significativa con 63.9 % de confianza

En la tabla 4.16 se muestran los resultados para la base de datos SS2-KC de complejos-k etiquetados. En esta tabla se puede ver que, al igual que en las otras dos bases de datos, el desempeño del modelo que considera la entropía en pre-entrenamiento es levemente mejor. A diferencia de las otras dos bases de datos, la diferencia que existe en los desempeños en este caso es más significativa y la probabilidad de que esta diferencia exista por factores aleatorios no es tan alta. Sin embargo, en este caso la diferencia de variabilidades es prácticamente nula, a diferencia de los resultados para MODA y SS2-E1 (husos de sueño).

Tabla 4.16: Desempeño en SS2-KC de modelos pre-entrenados con distintas funciones de perdida

Modelo	**F1-Score**	Precision	Recall
LeBENDR (NSRR)	**82.04** $\pm$ 1.69	81.81 $\pm$ 2.72	82.39 $\pm$ 2.90
LeBENDR - Sin entropía (NSRR)	81.69 $\pm$ 1.68	82.66 $\pm$ 2.94	80.88 $\pm$ 2.98

Diferencia significativa con 71.5 % de confianza

Los resultados que se muestran en esta sección permiten concluir que es preferible incluir la componente de maximización de entropía en la función de pérdida en pre-entrenamiento. Esto porque se ve que hay una ligera mejora en el desempeño al incluir esta componente y una ligera mejora en la variabilidad de los resultados, lo que corrobora en parte las razones por las cuales se quiere considerar esta componente. Si bien estas mejoras no son significativas, el costo computacional que implica incluir esta componente es despreciable como ya se mencionó anteriormente. Finalmente, cabe destacar que esta es la única característica que se rescata directamente de Wav2Vec 2.0 y que fue modificada en BENDR, considerando que el modelo propuesto (LeBENDR) está principalmente inspirado en BENDR.

4.3. Análisis de detecciones y atención neuronal

En esta sección se muestran múltiples ejemplos de visualizaciones de los ponderadores de atención, los cuales, cobran gran importancia en el contexto de la explicabilidad de la decisión de modelos neuronales. Este punto tiene una gran relevancia, ya que los modelos neuronales carecen generalmente de la capacidad de explicar las decisiones que toman. Esto es lo que en la literatura llaman, una "caja negra" o *black box*, dado que lo que ocurre al interior del modelo no es algo que pueda visualizarse ni entenderse fácilmente. Recordar que los ponderadores de atención representan cuanta atención se le está otorgando a cada una de las representaciones que conforman la secuencia que, en este caso, son pequeños segmentos de la señal EEG. La ecuación 4.1 muestra el cálculo de los ponderadores de atención de un

solo cabezal y de una sola representación, que abarca la información de 8 muestras debido a la pérdida de resolución que ocurre en el extractor de características. Notar que existen $\hat{l} \in [1, \frac{L}{16}] = [1, 250]$ ponderadores de atención y que la multiplicación del vector *query* (q) con todas las *keys* (K) se realiza en la dimensión de las características (e), con $e \in [1, d]$.

$$a_{\hat{l}} = \underset{\hat{l}}{softmax} \left(\frac{q_e \cdot K_{\hat{l}e}}{\sqrt{d}} \right) \tag{4.1}$$

El mecanismo de atención permite a los *transformers* aumentar la explicabilidad de las decisiones tomadas. Los ponderadores de atención dan una idea de los lugares de la secuencia donde el modelo está extrayendo más información o "poniendo más atención". Sin embargo, es importante aclarar que estos ponderadores solo aumentan la explicabilidad del modelo, y esto no significa que sea la causa directa de las decisiones tomadas. Esto último quiere decir que los ponderadores de atención permiten tener una idea más clara del porqué el modelo está tomando ciertas decisiones, pero estos no representan necesariamente la única causa de las decisiones tomadas por el modelo. No obstante, si bien los ponderadores de atención no son el único factor de explicabilidad, éstos siguen siendo muy importantes y bastante superiores a la explicabilidad que pueden otorgar las arquitecturas convencionales de modelos neuronales.

La capacidad de explicar las decisiones es un punto particularmente favorable para utilizar un *transformer* en el contextualizador, ya que la neurociencia tiene una fuerte conexión con la medicina, lo que implica que la explicabilidad es una parte fundamental si el objetivo es tener un modelo que pueda ser aceptado y puesto en marcha en un ambiente productivo. A continuación, en las siguientes subsecciones, se muestran varios ejemplos con detecciones en las distintas bases de datos etiquetadas. Estos ejemplos fueron procesados de una partición aleatoria de la validación cruzada, de tal forma que las detecciones mostradas corresponden a muestras aleatorias del conjunto de prueba correspondiente a la partición. Además, todos los ejemplos fueron tomados de detecciones realizadas por LeBENDR con la atención neuronal propuesta, pre-entrenado solamente en NSRR.

Los ejemplos seleccionados para mostrar los ponderadores de atención, incluyen todo tipo de detecciones. En todos los ejemplos hay presencia de falsos positivos absolutos (evento detectado sin intersección con un evento etiquetado), falsos negativos absolutos (evento etiquetado sin intersección con un evento detectado) e intersecciones parciales entre eventos etiquetados y detectados. Las figuras que ilustran estos ejemplos en las subsecciones a continuación contienen 3 gráficos cada una. El primer gráfico muestra las detecciones realizadas por el modelo, incluyendo la evolución de la probabilidad de presencia de evento inferida por el modelo, junto con el umbral de duración (0.425) y el umbral de detección (0.5). El segundo gráfico muestra los ponderadores de atención que realiza una representación local que no pertenece a un evento etiquetado, y por último, el tercer gráfico muestra los ponderadores de atención de una representación local que sí pertenece a un evento etiquetado.

4.3.1. Detecciones y atención en ejemplo de MODA

En la figura 4.1 se muestran las detecciones de un ejemplo que pertenece al conjunto de prueba en la base de datos de MODA de husos de sueño etiquetados. En el primer gráfico se puede observar la presencia de dos falsos negativos absolutos (IoU igual a 0) seguido de un falso positivo absoluto (Iou igual a -1) y de tres eventos correctamente detectados con un IoU mayor que 0.2, el cual es el umbral estándar de la literatura. En este gráfico se puede ver que el primer falso negativo está, de hecho, presente dentro de una detección. Un requisito en el estándar de evaluación de la literatura es que un evento detectado solo se puede conectar con un evento etiquetado y viceversa. Además de esto, se puede observar que el falso positivo absoluto está sobre una zona que a simple vista pareciera tener una mayor frecuencia, lo que permite entender en parte la causa del error.

En la figura 4.1 b) se pueden observar los ponderadores de atención provenientes de una representación local elegida de manera aleatoria, que no pertenece a un evento etiquetado. En este gráfico se muestran ver los ponderadores de las 4 capas del *transformer* con 4 cabezales de atención cada una. En la primera capa del *transformer* se puede ver que los cabezales tienen la atención relativamente distribuida por todo el segmento de EEG, por lo que no es particularmente interesante desde el punto de vista de la interpretabilidad. En la segunda capa, se puede ver claramente como los cabezales concentran la atención alrededor del pequeño segmento en rojo que es la zona que está poniendo atención. En la tercera capa del *transformer*, se puede notar que en general la atención está bastante distribuida, pero se puede ver que pone atención en todas las zonas que tienen algún tipo de *peak* (punto de inflexión), ya sea positivo o negativo, de alta o de baja amplitud. Por último, en la cuarta capa del *transformer* se puede ver que los cabezales concentran su atención en todos los subsegmentos del EEG con mayor frecuencia.

En la figura 4.1 c) se observan los ponderadores de atención provenientes de una representación local aleatoria, que pertenece a un evento etiquetado. En este gráfico, se muestran los ponderadores de atención de las 4 capas con sus 4 cabezales cada una. En la primera capa se puede ver nuevamente que los cabezales tienen la atención distribuida casi de manera uniforme sobre el segmento de EEG. Luego, en la segunda capa se observa nuevamente, y se verifica que los cabezales concentran su atención alrededor de las muestras locales donde se está poniendo atención (pequeño segmento destacado en color rojo). En la tercera capa se puede observar que los cabezales de atención, otra vez, concentran su atención sobre zonas que tienen *peaks* de amplitud, pero además pone atención sobre zonas de mayor frecuencia. Por último, en la cuarta capa, los cabezales concentran su atención en las zonas donde la probabilidad de evento de salida del modelo es mayor a cero.

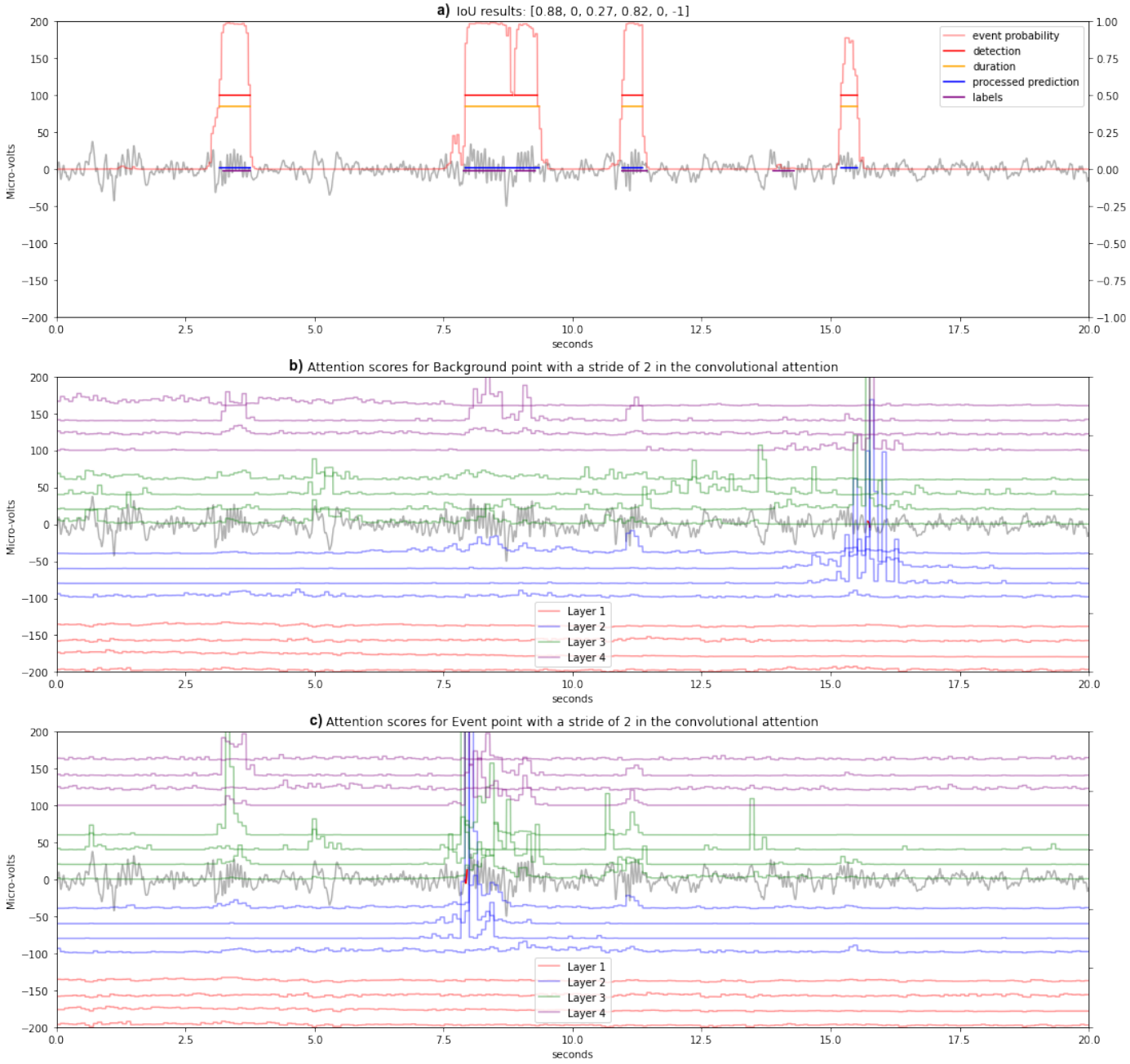

Figura 4.1: Ejemplo de prueba en MODA. a) detecciones realizadas por el modelo y probabilidades de evento. b) ponderaciones de atención ejercidas por sección roja de EEG ubicada fuera de un evento. c) ponderaciones de atención ejercidas por sección roja de EEG ubicada dentro de un evento.

De la figura 4.1 se observa que el modelo, al ser entrenado en esta base de datos, aprendió ciertas características particulares, relativamente definidas e identificables. La primera capa parece no tener un rol muy fundamental ya que la atención está distribuida casi de manera uniforme y se puede decir que extrae información de todas las zonas del EEG por igual, teniendo una suerte de promedio de las representaciones. La segunda capa por su parte, sirve para identificar las zonas adyacentes a la representación local donde se está poniendo atención. La tercera capa pareciera ser una de las más importantes, ya que identifica los

cambios de pendiente (*peaks*) y las zonas con mayor frecuencia y, por último, la cuarta capa pone atención sobre las zonas donde el modelo determina que tienen cierta probabilidad mayor que cero de pertenecer a un evento.

4.3.2. Detecciones y atención en ejemplo de SS2-E1

En la figura 4.2 se muestran las detecciones de un ejemplo que pertenece al conjunto de prueba en la base de datos SS2-E1 de husos de sueño etiquetados. En la figura 4.2 a) se observa la presencia de dos falsos negativos absolutos al comienzo, seguido por un falso positivo absoluto y finalmente una intersección que determina una correcta detección de un evento etiquetado. Con respecto a los falsos negativos, si bien no hubo detección en esas zonas, en la figura se observa que el modelo determina que existe una probabilidad de evento en las zonas etiquetadas, lo cual es una buena señal del comportamiento del modelo, aunque la probabilidad esté por debajo del umbral de detección. Luego, con respecto al falso positivo, la zona identificada como evento efectivamente tiene una mayor probabilidad asignada por el modelo y se puede observar que el evento detectado sí se parece a un huso de sueño, lo cual también es una buena señal con respecto a los errores que el modelo comete en esta base de datos.

En la figura 4.2 b) se observan los ponderadores de atención provenientes de una representación local elegida de manera aleatoria, que no pertenece a un evento etiquetado. En la primera capa del *transfomer* se puede observar que los cabezales distribuyen su atención por todo el segmento de EEG. En la segunda capa, los cabezales vuelven a concentrar su atención en las zonas adyacentes a la representación local que está ejerciendo atención sobre el EEG. La tercera capa pareciera identificar las zonas del "fondo" del EEG (zonas que no tienen ni mayor amplitud ni mayor frecuencia). Por último, la cuarta capa concentra los cabezales en identificar las zonas que tienen mayor probabilidad de ser un evento detectado.

En la figura 4.2 c) se observan los ponderadores de atención provenientes de una representación local elegida de manera aleatoria, que pertenece a un evento etiquetado. En la primera capa del *transformer* se puede ver, al igual que en todos los casos anteriores, que los cabezales distribuyen su atención de manera más bien uniforme. En la segunda capa del *transformer* los cabezales vuelven a concentrarse alrededor de la representación en la cual se está haciendo el análisis de atención neuronal. En la tercera capa, la atención se concentra en zonas de alta frecuencia que están cerca de las muestras cuya representación está ejerciendo atención. Finalmente, la cuarta capa del *transformer* vuelve a concentrar la atención de sus cabezales en las zonas donde finalmente el modelo identifica que hay una mayor probabilidad de detección de huso de sueño.

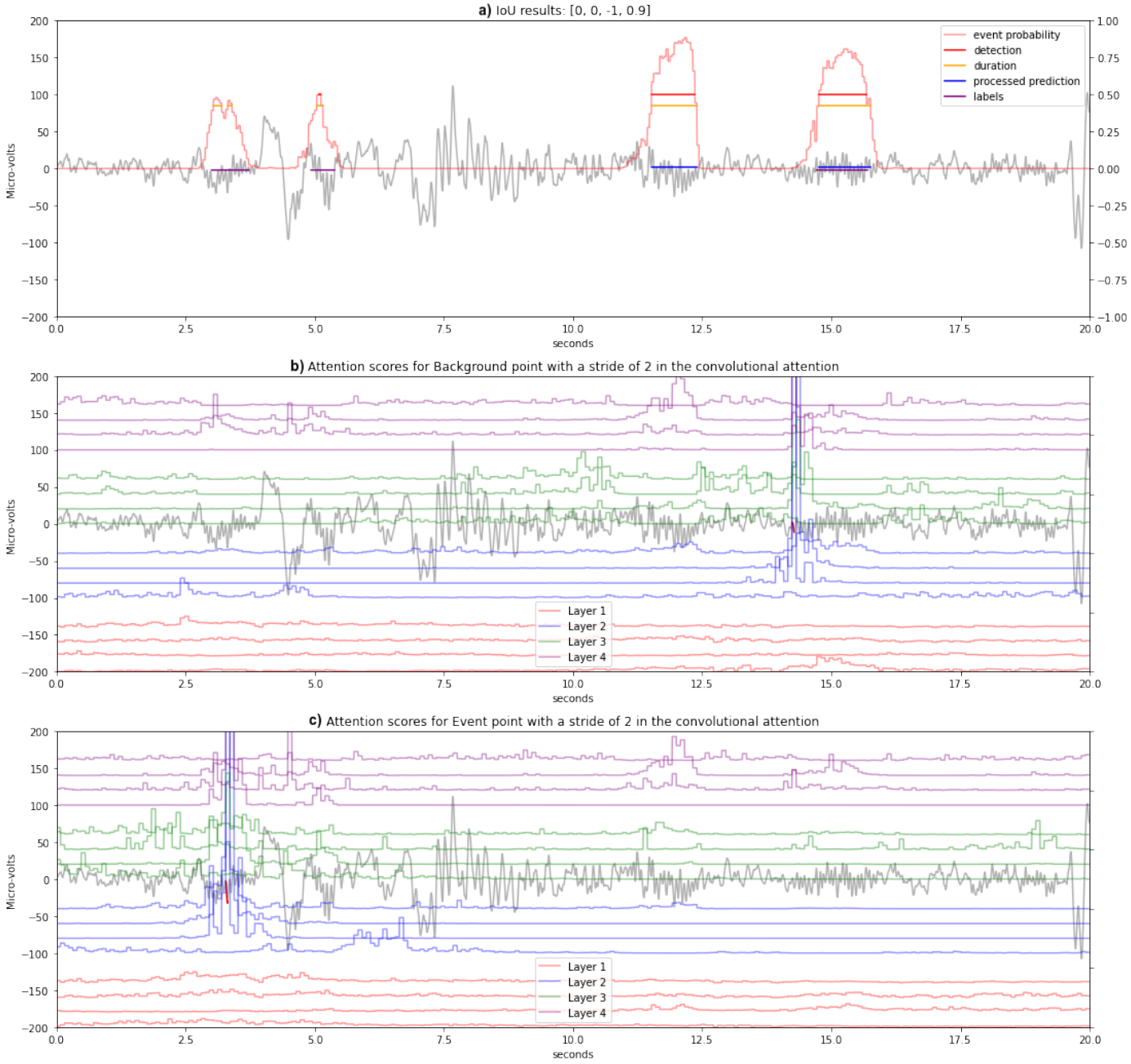

Figura 4.2: Ejemplo de prueba en SS2-E1. a) detecciones realizadas por el modelo y probabilidades de evento. b) ponderaciones de atención ejercidas por sección roja de EEG ubicada fuera de un evento. c) ponderaciones de atención ejercidas por sección roja de EEG ubicada dentro de un evento.

A partir del análisis de lo observado en la figura 4.2, se puede observar que el modelo propuesto al ser entrenado en una base de husos de sueño, aprende características relativamente bien definidas en cada una de las capas del contextualizador. La primera capa aprende a extraer información uniforme de todo el segmento de EEG en cuestión. La segunda capa muestra que el *transformer* aprendió que es importante localizar la zona donde se está poniendo atención. La tercera capa muestra ser más dinámica y que dependiendo del segmento, se enfoca en patrones de distintos tipos que determinan en lo que se está fijando el *trans-*

former para tomar una decisión, y por último, la cuarta capa aprende a determinar donde están las zonas que deberían tener mayor probabilidad de detección de evento. Todo esto se correlaciona muy bien con el caso de MODA, en el cual se ven los mismos aprendizajes y funciones por capa.

4.3.3. Detecciones y atención en ejemplo de SS2-KC

En la figura 4.3 se muestran las detecciones de un ejemplo que pertenece al conjunto de prueba en la base de datos SS2-KC de complejos-k etiquetados. En la figura 4.3 a) se observa la presencia de un falso negativo absoluto, seguido de una intersección correcta entre una detección y un evento etiquetado, y finalmente un falso positivo absoluto. En el caso del falso negativo absoluto, nuevamente se observa que el modelo determina que hay cierta probabilidad de detección de evento, pero el post-procesamiento de la salida del modelo no marca una detección de evento. Luego, en el caso del falso positivo absoluto a la derecha, se puede ver que la zona marcada podría confundirse con un complejo-k, ya que los complejos-k se caracterizan por tener un *peak* inicialmente negativo, seguido de un *peak* positivo con gran amplitud. Por lo tanto, a pesar de estos dos errores, se puede ver que el modelo si tiene una buena noción de como son los eventos y como ya se mencionó anteriormente, las etiquetas tienen cierto grado de relatividad, por lo que también existe la posibilidad de que la etiqueta sea la que esté incorrecta y el modelo sea el que esté detectando correctamente en ciertos casos de falsos positivos absolutos.

En la figura 4.3 b) se observan los ponderadores de atención provenientes de una representación local elegida de manera aleatoria, que no pertenece a un evento etiquetado. En la primera capa se puede ver que al igual que en todos los casos anteriores, los cabezales distribuyen su atención de manera uniforme en el segmento de EEG. En la segunda capa se vuelve a corroborar que los cabezales se encargan de poner atención sobre las zonas adyacentes a la representación local donde se está poniendo atención. Los cabezales de la tercera capa, en este caso concentran su atención en valles del EEG, y por último, los cabezales de la cuarta capa parecen prestar más atención a las zonas donde se determina que existe mayor probabilidad de evento.

En la figura 4.3 c) se observan los ponderadores de atención provenientes de una representación local elegida de manera aleatoria, que pertenece a un evento etiquetado. En la primera capa nuevamente se observa que los cabezales distribuyen su atención de manera más bien uniforme. En la segunda capa los cabezales nuevamente concentran su atención en las zonas adyacentes a la representación que está prestando atención. Los cabezales de la tercera capa, en esta ocasión, pareciera que concentran su atención en zonas de *peaks* negativos, y finalmente, los cabezales de la cuarta capa concentran su atención en donde el modelo determina que hay probabilidad de detección de un evento.

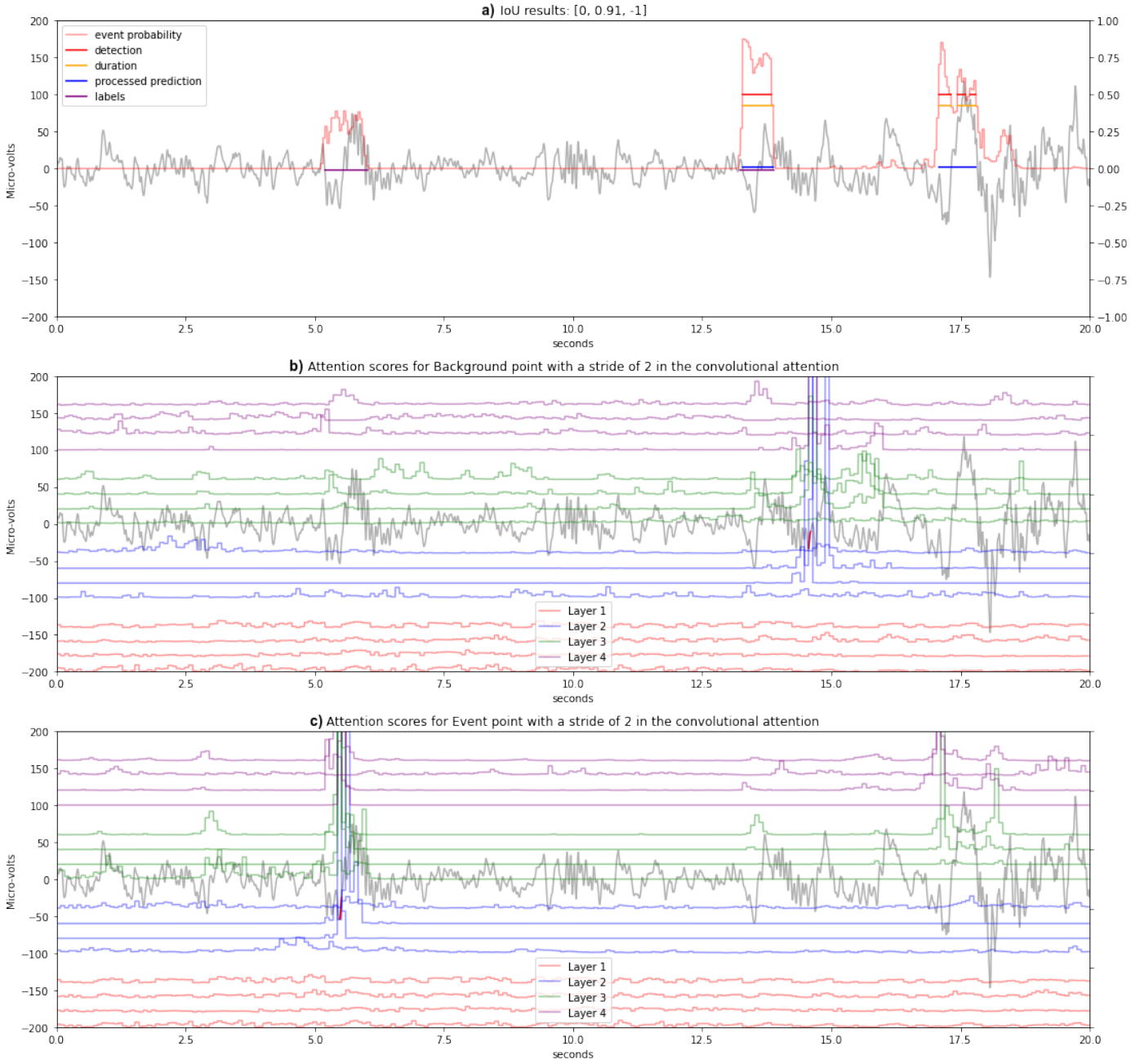

Figura 4.3: Ejemplo de prueba en SS2-KC. a) detecciones realizadas por el modelo y probabilidades de evento. b) ponderaciones de atención ejercidas por sección roja de EEG ubicada fuera de un evento. c) ponderaciones de atención ejercidas por sección roja de EEG ubicada dentro de un evento.

A partir del análisis de los observado en la figura 4.3, complementado a los análisis de atención neuronal anteriores, se observa que LeBENDR aprende a reconocer ciertas características particulares, las cuales ayudan a comprender las decisiones que toma el modelo. Se corrobora que la primera capa del *transformer* contextualizador de LeBENDR extrae información de manera general en el EEG. También se corrobora que la segunda capa ayuda a localizar la zona que se está evaluando. La tercera capa es la más particular de todas porque pareciera que identifica ciertos patrones en el EEG, dependiendo de la zona que se está

evaluando y del evento transitorio que se está tratando de detectar. Luego, para terminar, se corrobora que la cuarta capa aprende a reconocer las zonas que pueden ser las que tengan alguna probabilidad mayor que cero de ser detectadas como evento.

Los análisis realizados en esta sección son útiles desde el punto de vista de la interpretabilidad y explicabilidad de un modelo y las decisiones que este toma. Este punto a favor de LeBENDR en comparación con otros modelos, independiente de si tiene un mejor o peor desempeño en ciertos casos, lo vuelven una alternativa mucho más viable si se trata de aterrizar alguno de estos modelos a un despliegue en un ambiente productivo real. Es importante recordar que, el problema que pretenden resolver el modelo propuesto y los modelos con los que se pueda comparar, tiene mucha relación con la salud, la cual es una área que naturalmente es más cautelosa en cuanto a implementar nuevas tecnologías de automatización.

Capítulo 5

Conclusiones

A partir de los resultados obtenidos, se concluye que el *Transformer* de tipo *encoder* se puede adaptar muy bien para la representación de EEG y que este modelo puede alcanzar, y a veces superar, el estado del arte en la detección de ciertos eventos transitorios del sueño. Esto implica que se logró el objetivo general de este trabajo de tesis, el cual consiste en proponer un modelo con muy buen desempeño, y además con mejor interpretabilidad y explicabilidad que sus predecesores. Además del objetivo general, se cumple con los objetivos específicos dado que se evalúan distintas variaciones de la arquitectura propuesta en distintas bases de datos y se compara con un modelo del estado del arte (RED-Time). Además, se interpretan las detecciones resultantes mediante el análisis de las capas de atención neuronal.

Dado los resultados de detección en husos de sueño y complejos-k en las bases de datos etiquetadas utilizadas, se puede concluir que el hecho de reducir el tamaño de *scores* a calcular en el proceso de atención neuronal, facilita el mecanismo del mismo. Además, a diferencia de la atención neuronal con mejora de localidad, el tamaño de las convoluciones para formar las *querys* son de largo 1. Esto permite que cada muestra por sí sola "consulte" otras localidades de muestras y que no se pierda información relevante a cada representación aprendida. Esto último tiene sentido ya que al final del proceso de atención, cada representación de manera puntual será sumada con el resultado de la atención neuronal y además será clasificada como un evento o un no-evento. Los resultados evidencian que el *transformer* tipo BENDR como arquitectura para la representación de EEG mediante un pre-entrenamiento contrastivo es eficaz para la etapa de ajuste fino en la detección de eventos transitorios del sueño. Sin embargo, la mejora en el desempeño al pre-entrenar LeBENDR con grandes bases de datos no fue tan grande como se esperaba, lo que da cabida a futuras investigaciones sobre otras posibles tareas de pre-entrenamiento que puedan ser más adecuadas o beneficiosas.

Limitaciones del modelo propuesto

Si bien el *Transformer* tiene muy alta capacidad de paralelización, este necesita de una gran cantidad de datos etiquetados y de aún más datos no etiquetados para alcanzar altos desempeños. Si bien el *Transformer* sin la etapa de pre-entrenamiento en datos no-etiquetados tiene un muy buen desempeño, es gracias a la etapa de pre-entrenamiento con la que puede alcanzar y en ocasiones superar el desempeño de otros métodos del estado del arte en la detección de eventos transitorios del sueño. Sumado al costo en cantidad de datos, se necesita de un *hardware* muy potente (GPU) para aprovechar la capacidad de paralelización que tiene el *Transformer* y poder soportar así la gran cantidad de parámetros que posee

este modelo en comparación al resto de los modelos basados en redes neuronales. Además de esto, la aceleración con GPU también es necesaria para lograr tiempos de entrenamiento más razonables, recordando que, para la etapa de pre-entrenamiento se necesita entrenar con un cantidad enorme de datos (del orden del millón de ejemplos).

Aprendizajes principales

El trabajo realizado da cuenta de varios puntos a destacar. Primero, el pre-procesamiento de datos es una etapa fundamental en el desarrollo de modelos de aprendizaje de máquinas, en particular, de redes neuronales. El hecho de preparar y de limpiar los datos evita que el modelo aprenda particularidades indeseables (artefactos como movimientos oculares o musculares) en los datos, y facilita el aprendizaje de la tarea en sí. Segundo, para los *transformers*, la etapa de pre-entrenamiento es una etapa muy beneficiosa y que permite alcanzar mejores desempeños en el entrenamiento de tareas particulares (ajuste fino). El balanceo de clases mediante el aumento de datos o mediante la estratificación de los *batches* de entrenamiento, tienen efectos notables en el desempeño del modelo, el cual tiende a sobre entrenarse en clases predominantes con mucha facilidad. Tercero, muchos eventos etiquetados en EEG son muy difíciles de detectar hasta para un ojo humano experto, por lo que dependiendo de la base de datos en la cual se entrena el modelo, siempre existirá un grado de subjetividad en los datos, y por lo tanto, en los resultados entregados por el modelo. Cuarto, el aprendizaje contrastivo como tarea de pre-entrenamiento no evidencia nunca señales de sobreajuste. Sin embargo, la influencia que tienen los cabezales de atención parece disminuir con muchas épocas de entrenamiento, lo cual es bastante interesante y abre un tema de investigación alrededor de este efecto. Quinto, es computacionalmente más eficiente utilizar una sola neurona de salida y la entropía cruzada binaria como función de pérdida para entrenar el clasificador, frente a utilizar 2 neuronas de salida y la entropía cruzada como función de pérdida. Además tener una sola neurona de salida evita matemáticamente la redundancia en la variación del valor de los parámetros aprendidos por la segunda neurona. Por último, los procesamientos de las probabilidades de evento son fundamentales para poder concretizar mejor las detecciones, pudiendo atenuar falencias del modelo como excesos de falsos positivos y/o falsos negativos.

Trabajo futuro

A continuación, se listan posibles trabajos propuestos a futuras reproducciones, extensiones o continuaciones de este trabajo. Estos son:

- Mejorar la tarea contrastiva de pre-entrenamiento, diferenciando mejor los negativos en la función de pérdida contrastiva, por ejemplo, diferenciando las muestras por banda como la banda sigma y la banda delta.
- Cambiar la tarea de pre-entrenamiento por completo, o añadir una segunda tarea de pre-entrenamiento que pueda ser más específica al problema particular de detección de eventos transitorios del sueño. Esta podría ser, por ejemplo, una tarea de enmascaramiento semi-supervisado de clasificación con etiquetas sintéticas, generadas a partir de la pertenencia a la banda sigma, a la banda delta o a ninguna de las dos.
- Probar variaciones en la arquitectura del bloque convolucional de extracción de características, como por ejemplo, conexiones residuales tipo ResNet (He et al., 2016) o

conexiones residuales tipo DenseNet (Huang, Liu, Van Der Maaten, y Weinberger, 2017) que le den más expresividad a la primera etapa de extracción de características.

- Desarrollar métricas objetivas de interpretabilidad en los ponderadores de atención. Esto permitiría que las detecciones del modelo y la inferencia de probabilidades de evento tengan mejor explicabilidad, aumentando la capacidad análisis de resultados para casos de uso en ambientes productivos.

- Representar otros tipos de series de tiempo además de los EEG. El *transformer* propuesto podría resolver una gama amplia de problemas en torno a las series de tiempo. Cabe destacar que, en general las series de tiempo son mucho más largas y con comportamientos más complejos o arbitrarios que las secuencias de palabras para las cuales fueron pensados originalmente los *transformers*. Por lo que, los problemas con series de tiempo no suelen ser resueltos utilizando este tipo de modelos, ni redes neuronales en general, siendo este modelo una buena alternativa a probar dependiendo del caso de uso.

Bibliografía

[1] Ba, J. L., Kiros, J. R., y Hinton, G. E. (2016). Layer Normalization. , 1–14. Descargado de http://arxiv.org/abs/1607.06450

[2] Baevski, A., Zhou, H., Mohamed, A., y Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, *2020-Decem*, 1–19.

[3] Bahdanau, D., Cho, K. H., y Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015*, 1–15.

[4] Berry, R. B., Albertario, C. L., Harding, S. M., Lloyd, R. M., y Plante, D. T. (2018). *The AASM Manual for the scoring of sleep and associated events : rules, terminology and technical specifications*. American Academy of Sleep Medicine.

[5] Blackwell, T., Yaffe, K., Ancoli-Israel, S., Redline, S., Ensrud, K. E., Stefanick, M. L., ... Stone, K. L. (2011, dec). Associations between sleep architecture and sleep-disordered breathing and cognition in older community-dwelling men: the Osteoporotic Fractures in Men Sleep Study. *Journal of the American Geriatrics Society*, *59*(12), 2217–2225. doi: 10.1111/j.1532-5415.2011.03731.x

[6] Butterworth, S. (1930). On the Theory of Filter Amplifiers. *Experimental Wireless & the Wireless Engineer*, *7*, 536–541.

[7] Carskadon, M. A., y Dement, W. C. (2005). Chapter 2 - Normal Human Sleep : An Overview..

[8] Chambon, S., Thorey, V., Arnal, P. J., Mignot, E., y Gramfort, A. (2019). DOSED: A deep learning approach to detect multiple sleep micro-events in EEG signal. *Journal of Neuroscience Methods*, *321*, 64–78. doi: https://doi.org/10.1016/j.jneumeth.2019.03.017

[9] Christian, O., Gosselin, N., Carrier, J., y Nielsen, T. (2014). Montreal Archive of Sleep Studies: an open-access resource for instrument benchmarking and exploratory research. *Journal of Sleep Research*, *23*(6), 628–635. doi: https://doi.org/10.1111/jsr.12169

[10] Clawson, B. C., Durkin, J., y Aton, S. J. (2016). Form and Function of Sleep Spindles across the Lifespan. *Neural plasticity*, *2016*, 6936381. doi: 10.1155/2016/6936381

[11] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., y Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. En *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255). doi: 10.1109/CVPR.2009.5206848

[12] Devlin, J., Chang, M. W., Lee, K., y Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference, *1*(Mlm), 4171–4186.

[13] El Helou, J., Navarro, V., Depienne, C., Fedirko, E., LeGuern, E., Baulac, M., ... Adam, C. (2008, oct). K-complex-induced seizures in autosomal dominant nocturnal frontal lobe epilepsy. *Clinical neurophysiology : official journal of the International Federation of Clinical Neurophysiology*, *119*(10), 2201–2204. doi: 10.1016/j.clinph.2008.07.212

[14] Fernandez, L. M. J., y Lüthi, A. (2020, apr). Sleep Spindles: Mechanisms and Functions. *Physiological reviews*, *100*(2), 805–868. doi: 10.1152/physrev.00042.2018

[15] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics*, *36*, 193–202.

[16] Gençay, R., y Qi, M. (2001). Pricing and hedging derivative securities with neural networks: Bayesian regularization, early stopping, and bagging. *Neural Networks, IEEE Transactions on*, *12*, 726–734. doi: 10.1109/72.935086

[17] Glorot, X., y Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Journal of Machine Learning Research*, *9*, 249–256.

[18] Hanson, S. J., y Pratt, L. Y. (1988). Comparing Biases for Minimal Network Construction with Back-Propagation. En *Proceedings of the 1st international conference on neural information processing systems* (pp. 177–185). MIT Press.

[19] He, K., Zhang, X., Ren, S., y Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-Decem*, 770–778. doi: 10.1109/CVPR.2016.90

[20] Hendrycks, D., y Gimpel, K. (2016). Gaussian Error Linear Units (GELUs). , 1–6. Descargado de http://arxiv.org/abs/1606.08415

[21] Hochreiter, S., y Schmidhuber, J. (1997, nov). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. doi: 10.1162/neco.1997.9.8.1735

[22] Huang, G., Liu, Z., Van Der Maaten, L., y Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 2261–2269. doi: 10.1109/CVPR.2017.243

[23] Hubel, D. H., y Wiesel, T. N. (1959). Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, *148*(3), 574–591.

[24] Ioffe, S., y Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *32nd International Conference on Machine Learning, ICML 2015*, *1*, 448–456.

[25] Jiang, D., Ma, Y., y Wang, Y. (2021, mar). A robust two-stage sleep spindle detection approach using single-channel {EEG}. *Journal of Neural Engineering*, *18*(2), 26026. doi: 10.1088/1741-2552/abd463

[26] Kingma, D. P., y Ba, J. L. (2015). Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 1–15.

[27] Kostas, D., Aroca-Ouellette, S., y Rudzicz, F. (2021). BENDR: Using Transformers and a Contrastive Self-Supervised Learning Task to Learn From Massive Amounts of EEG Data. *Frontiers in Human Neuroscience*, *15*. doi: 10.3389/fnhum.2021.653659

[28] Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. En *Proceedings of the 25th international conference on neural information processing systems - volume 1* (pp. 1097–1105).

[29] Kulkarni, P. M., Xiao, Z., Robinson, E. J., Jami, A. S., Zhang, J., Zhou, H., ... Chen, Z. (2019, jun). A deep learning approach for real-time detection of sleep spindles. *Journal of neural engineering*, *16*(3), 36004. doi: 10.1088/1741-2552/ab0933

[30] Lacourse, K., Delfrate, J., Beaudry, J., Peppard, P., y Warby, S. C. (2019, mar). A sleep spindle detection algorithm that emulates human expert spindle scoring. *Journal of neuroscience methods*, *316*, 3–11. doi: 10.1016/j.jneumeth.2018.08.014

[31] Lacourse, K., Yetton, B., Mednick, S., y Warby, S. C. (2020). Massive online data annotation, crowdsourcing to generate high quality sleep spindle annotations from EEG data. *Scientific Data*, *7*(1), 1–14. doi: 10.1038/s41597-020-0533-4

[32] Lajnef, T., O'Reilly, C., Combrisson, E., Chaibi, S., Eichenlaub, J. B., Ruby, P. M., ... Jerbi, K. (2017). Meet spinky: An open-source spindle and K-complex detection toolbox validated on the open-access montreal archive of sleep studies (MASS). *Frontiers in Neuroinformatics*, *11*(March). doi: 10.3389/fninf.2017.00015

[33] Lecun, Y., Bottou, L., Bengio, Y., y Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*, 2278–2324.

[34] Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y. X., y Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in Neural Information Processing Systems*, *32*(NeurIPS).

[35] Lin, T.-Y., Goyal, P., Girshick, R., He, K., y Dollar, P. (2020, feb). Focal Loss for Dense Object Detection. *IEEE transactions on pattern analysis and machine intelligence*, *42*(2), 318–327. doi: 10.1109/TPAMI.2018.2858826

[36] Loshchilov, I., y Hutter, F. (2019). Decoupled weight decay regularization. *7th International Conference on Learning Representations, ICLR 2019*.

[37] Marcus, C. L., Moore, R. H., Rosen, C. L., Giordani, B., Garetz, S. L., Taylor, H. G., ... Redline, S. (2013, jun). A randomized trial of adenotonsillectomy for childhood sleep apnea. *The New England journal of medicine*, *368*(25), 2366–2376. doi: 10.1056/NEJMoa1215881

[38] Pérez, J., Marinković, J., y Barceló, P. (2019). On the turing completeness of modern neural network architectures. *7th International Conference on Learning Representations, ICLR 2019*, 1–36.

[39] Purcell, S. M., Manoach, D. S., Demanuele, C., Cade, B. E., Mariani, S., Cox, R., ... Stickgold, R. (2017). Characterizing sleep spindles in 11,630 individuals from the National Sleep Research Resource. *Nature Communications*, *8*(1), 15930. doi: 10.1038/ncomms15930

[40] Quan, S. F., Howard, B. V., Iber, C., Kiley, J. P., Nieto, F. J., O'Connor, G. T., ... Wahl, P. W. (1997, dec). The Sleep Heart Health Study: design, rationale, and methods. *Sleep*, *20*(12), 1077–1085.

[41] Redline, S., Tishler, P. V., Tosteson, T. D., Williamson, J., Kump, K., Browner, I., ... Krejci, P. (1995, mar). The familial aggregation of obstructive sleep apnea. *American journal of respiratory and critical care medicine*, *151*(3 Pt 1), 682–687. doi: 10.1164/ajrccm/151.3_Pt_1.682

[42] Rosen, C. L., Larkin, E. K., Kirchner, H. L., Emancipator, J. L., Bivins, S. F., Surovec, S. A., ... Redline, S. (2003, apr). Prevalence and risk factors for sleep-disordered breathing in 8- to 11-year-old children: association with race and prematurity. *The Journal of pediatrics*, *142*(4), 383–389. doi: 10.1067/mpd.2003.28

[43] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65(6)*, 386–408.

[44] Rumelhart, D. E., y McClelland, J. L. (1985). Learning Internal Representations by Error Propagation. En *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations* (pp. 318–362).

[45] Shinomiya, S., Nagata, K., Takahashi, K., y Masumura, T. (1999, apr). Development of sleep spindles in young children and adolescents. *Clinical EEG (electroencephalography)*, *30*(2), 39–43. doi: 10.1177/155005949903000203

[46] Spira, A. P., Blackwell, T., Stone, K. L., Redline, S., Cauley, J. A., Ancoli-Israel, S., y Yaffe, K. (2008, jan). Sleep-disordered breathing and cognition in older women. *Journal of the American Geriatrics Society*, *56*(1), 45–50. doi: 10.1111/j.1532-5415.2007.01506.x

[47] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., y Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

[48] Stern, J. M. (2005). *Atlas of EEG patterns*. Lippincott Williams & Wilkins.

[49] Student. (1908). The probable error of a mean. *Biometrica*, 1–25.

[50] Tapia, N. I., y Estevez, P. A. (2020). RED: Deep Recurrent Neural Networks for Sleep EEG Event Detection. *Proceedings of the International Joint Conference on Neural Networks*. doi: 10.1109/IJCNN48605.2020.9207719

[51] Ulloa, S., Estevez, P. A., Huijse, P., Held, C. M., Perez, C. A., Chamorro, R., ... Peirano, P. (2016, aug). Sleep-spindle identification on EEG signals from polysomnographie recordings using correntropy. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, *2016*, 3736–3739. doi: 10.1109/EMBC.2016.7591540

[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

[53] Warby, S. C., Wendt, S. L., Welinder, P., Munk, E. G., Carrillo, O., Sorensen, H. B., ... Mignot, E. (2014). Sleep-spindle detection: Crowdsourcing and evaluating performance of experts, non-experts and automated methods. *Nature Methods*, *11*(4), 385–392. doi: 10.1038/nmeth.2855

[54] Wolpert, E. A. (1969). A Manual of Standardized Terminology, Techniques and Scoring System for Sleep Stages of Human Subjects. *Archives of General Psychiatry*, *20*(2), 246–247. doi: 10.1001/archpsyc.1969.01740140118016

[55] Zhang, G.-Q., Cui, L., Mueller, R., Tao, S., Kim, M., Rueschman, M., ... Redline, S. (2018, oct). The National Sleep Research Resource: towards a sleep data commons. *Journal of the American Medical Informatics Association : JAMIA*, *25*(10), 1351–1358. doi: 10.1093/jamia/ocy064