UNIVERSIDAD AUTÓNOMA DE CHIHAUHUA

FACULTAD DE INGENIERÍA

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

CLASIFICACIÓN DE VISUALIZACIONES MOTORAS UTILIZANDO SEÑALES DE UN EEG, BASADO EN ALGORITMOS DE APRENDIZAJE PROFUNDO

POR:

EVELYN JANETH GONZÁLEZ WONG

TESIS PRESENTADA COMO EQUIVALENTE PARA OBTENER EL GRADO DE

Maestra en Ingeniería en Computación

CHIHUAHUA, CHIH., MÉXICO **MARZO 2019**

Clasificación de Visualizaciones Motoras, utilizando señales de un EEG, basado en algoritmos de Aprendizaje Profundo. Tesis presentada por Ing. Evelyn Janeth González Wong, como requisito parcial para obtener el grado de Maestría en Ingeniería en Computación, ha sido aprobada y aceptada por

M.I. Javier González Cantú
Director de la Facultad de Ingeniería

Dr. Alejandro Villalobos Aragón
Secretario de Investigación y Posgrado

M.S.I. Karina Rocío Requena Yáñez
Coordinadora Académica

Graciela Ramírez A.

Dra. Graciela María de Jesús Ramírez Alonso
Director de Tesis

Fecha

Comité:

Dra. Graciela María de Jesús Ramírez Alonso
Dr. Alain Manzo Martínez
Dr. Luis Carlos González Gurrola
MI. Jesús Roberto López Santillan

Evelyn Janeth González Wong
C División del Norte 14105
Col Nuevo Triunfo 31140
Chihuahua, Chih. México

Febrero 2019

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

28 de febrero de 2019

ISC EVELYN JANETH GONZÁLEZ WONG

Presente

En atención a su solicitud relativa al trabajo de tesis para obtener el grado de Maestro en Ingeniería en Computación, nos es grato transcribirle el tema aprobado por esta Dirección, propuesto y dirigido por el director **Dr. Graciela María de Jesús Ramírez Alonso** para que lo desarrolle como tesis, con el título: **"CLASIFICACIÓN DE VISUALIZACIONES MOTORAS, UTILIZANDO SEÑALES DE UN EEG, BASADO EN ALGORITMOS DE APRENDIZAJE PROFUNDO".**

ÍNDICE

Índice de contenido

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

UNIVERSIDAD AUTÓNOMA DE
CHIHUAHUA

4. Resultados

4.1. Aprendizaje Máquina

4.2. Aprendizaje Profundo

5. Conclusiones

Referencias

A. Autoencoders Motor Imagery 4-class

A.1 Matriz de confusión de cada uno de los sujetos clasificados

A.2 Pesos del primer y segundo autoencoder de cada uno de los sujetos clasificados.

B. Autoencoders Motor Imagery 2-class

B.1 Matriz de confusión de cada uno de los sujetos clasificados

B.2 Pesos del primer y segundo autoencoder de cada uno de los sujetos clasificados

Solicitamos a Usted tomar nota de que el título del trabajo se imprima en lugar visible de los ejemplares de las tesis.

A T E N T A M E N T E
"Naturam subiecit aliis"

EL DIRECTOR

M.I. JAVIER GONZÁLEZ CANTÚ

FACULTAD DE INGENIERÍA U.A.CH.

DIRECCIÓN

EL SECRETARIO DE INVESTIGACIÓN Y POSGRADO

DR. ALEJANDRO VILLALOBOS ARAGÓN

FACULTAD DE INGENIERÍA
Circuito No.1, Campus Universitario 2
Chihuahua, Chih., México. C.P. 31125
Tel. (614) 442-95-00
www.fing.uach.mx

Índice de figuras

Índice de tablas

1

Introducción

Según cifras del INEGI en el año 2015, del 6 % de la población en México que presenta alguna discapacidad, el 41.3 % cuenta con una discapacidad motriz a causa de alguna enfermedad, el 8.8 % por accidente, y el 33.1 % por edad avanzada [2]. Con el fin de ayudar a esta población vulnerable, algunos investigadores se han enfocado al desarrollo de diferentes herramientas que apoyen el avance de su autonomía [3]. Tal es el caso del control de prótesis, sillas de ruedas, o algún dispositivo externo de acuerdo con alguna acción de control. Gracias a los avances en la tecnología y a la reducción de costos en la misma, las interfases cerebro computadora (sistemas BCI) han tenido un gran auge en el desarrollo de sistemas de control [4]. Una de las técnicas que se utiliza en sistemas BCI es la visualización motora la cual es la representación mental de un movimiento, pero sin realizar ningún gesto físico. De esta manera, una persona con discapacidad física puede imaginar el movimiento y el sistema BCI lo identificará y podrá realizar una acción de control en apoyo a la persona con discapacidad.

En este apartado se describen los antecedentes necesarios para el análisis y desarrollo de un sistema de visualización motora con relación a la fuente biológica de las señales, su adquisición, tratamiento, y trabajos previos en algoritmos de aprendizaje convencionales que se enfocan a la clasificación del movimiento o visualización motora. También se define la justificación y objetivos para realizar este trabajo.

1.1. Antecedentes

El cerebro humano generalmente se encuentra dividido en dos partes principales que son la corteza cerebral y las regiones subcorticales. Las regiones subcorticales son aquellas áreas que controlan las funciones básicas y vitales como la frecuencia cardíaca, la respiración, la temperatura corporal y las respuestas emocionales como el miedo, el aprendizaje y la memoria [5]. La corteza cerebral se divide en dos hemisferios (véase Figura 1.1), en la cual cada hemisferio está dividido en cuatro lóbulos que son: frontal, parietal, occipital y temporal. El lóbulo frontal se encuentra vinculado con el pensamiento, la organización, planificación, memoria, control emocional y la generación de emociones. El lóbulo parietal es responsable de varias funciones como es la ortografía, conocimiento numérico, percepción, manipulación de objetos y conciencia espacial. En el lóbulo occipital se relaciona con la interpretación de estímulos visuales y reconocimiento espacial. El lóbulo temporal procesa la información auditiva, el lenguaje, reconocimiento de caras, generación de emociones, equilibrio y coordinación[6].

La corteza cerebral es el enfoque en la mayor parte de las investigaciones con tecnología de interfaces cerebro computadora[4].

1.1.1. BCI

Usualmente se describe por su nombre en inglés *Brain Computer Interface* (BCI), es un medio de comunicación basado en la actividad neuronal generada por el cerebro y es independiente de sus vías de salida a nervios periféricos y músculos. De las funciones más básicas del BCI es registrar la actividad del cerebro y procesarla para obtener las características de interés de la señal, luego que ya se ha obtenido la señal, utilizarla para realizar interacciones con el entorno de la forma que el usuario desee [4]. Es posible monitorizar múltiples tipos de actividades cerebrales con distinta procedencia como pueden ser señales eléctricas, magnéticas o de origen metabólico. Para ello, hay una etapa en la que se registra la actividad en forma de señales eléctricas que pueden ser tratadas con más facilidad en las etapas de procesado[7].

Un sistema BCI básico que analiza señales EEG consta de cuatro etapas: adquisición de señales, preprocesamiento de señales, extracción de características y clasificación (Véase Figura 1.2).

- Adquisición de señales. El componente de adquisición de señal es responsable de registrar

las ondas cerebrales y enviarlas al componente de preprocesamiento para mejorar la señal y reducir el ruido. Dicha adquisición se realiza mediante electrodos para después amplificar, filtrar y digitalizar la señal.

- Pre-procesamiento. La mayoría de las señales biomédicas se combinan con muchas otras de diversos orígenes a las cuales se les puede denominar como una interferencia, artefacto o simplemente ruido. Las fuentes de ruido pueden ser siológicas, causadas por los dispositivos utilizados para su adquisición o el medio ambiente del experimento [8]. Con el fin de reducir este ruido, se pueden utilizar diferentes filtros y algoritmos para eliminar las señales no deseadas que pueden dificultar una buena interpretación de la señal. Este proceso se debe hacer con el cuidado de no sustraer las señales esenciales de la actividad cerebral y así preparar la señal para su posterior procesamiento.

- Extracción de características. Este componente genera características discriminativas a partir

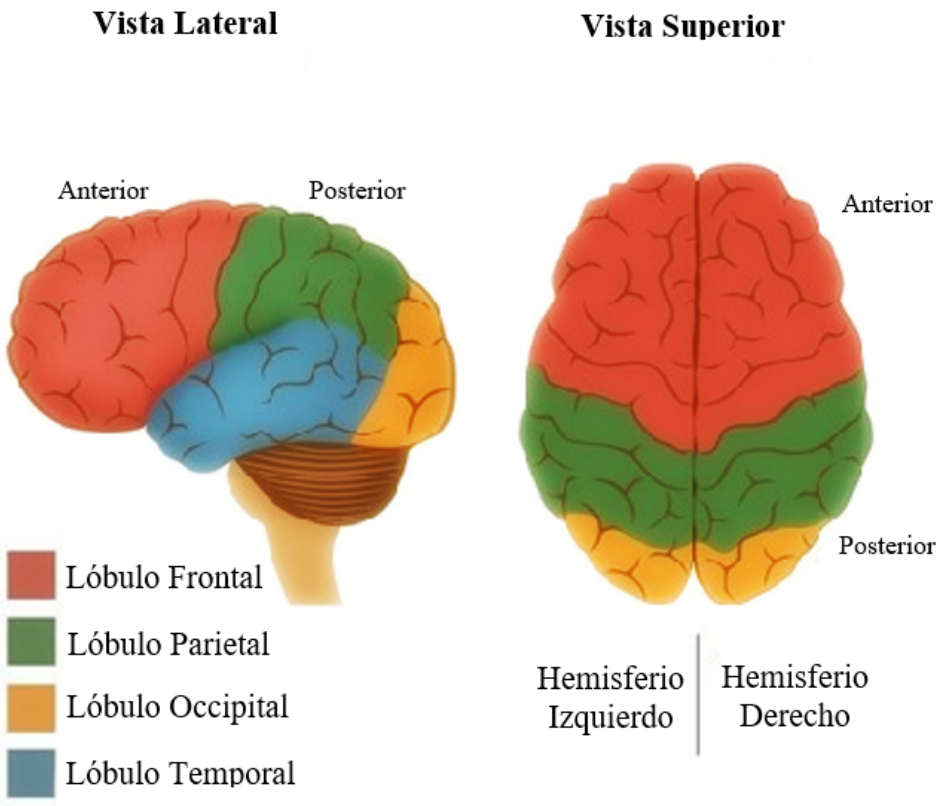


Figura 1.1: Corteza cerebral que se divide en dos hemisferios, cada uno se reparte en cuatro lóbulos llamados, Frontal, Parietal, Occipital y Temporal.

de la señal mejorada, disminuyendo el tamaño de los datos que serán la entrada al algoritmo de clasificación.

- Clasificación. Los clasificadores traducen las funciones producidas en comandos de dispositivo.

1.1.2. Electroencefalograma

El electroencefalograma o EEG es una prueba que permite el registro de la actividad cerebral. Se considera como el método más empleado para la adquisición de las señales del cerebro ya que tiene una alta resolución temporal, fácil de usar y seguro. Registra la actividad eléctrica en un corto periodo de tiempo a través de múltiples electrodos (activos o pasivos) situados en el cuero cabelludo directamente en la corteza. Uno de los problemas del EEG es que para reducir la impedancia del contacto del electrodo se necesita gel o líquido salino. En la Figura 1.3 se muestra la colocación de

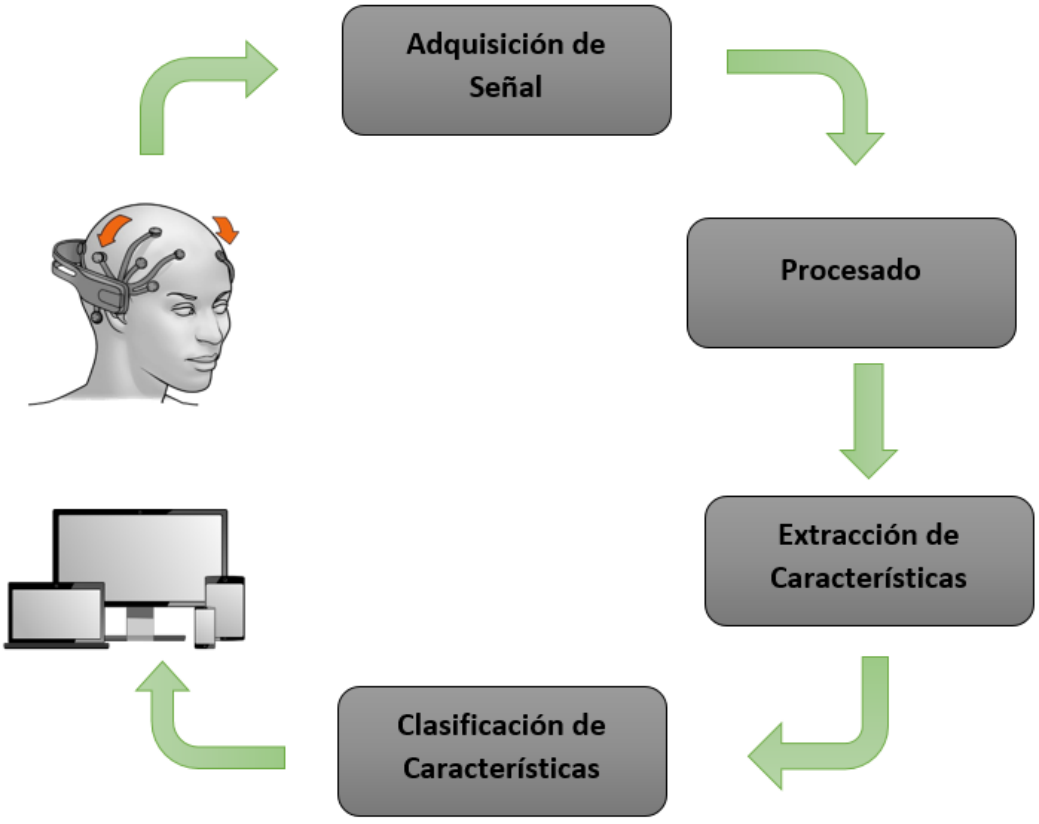


Figura 1.2: Diagrama de bloques de Componentes del Sistema BCI

sensores para un protocolo estándar para el intercambio de información de encaminamiento entre sistemas autónomos.

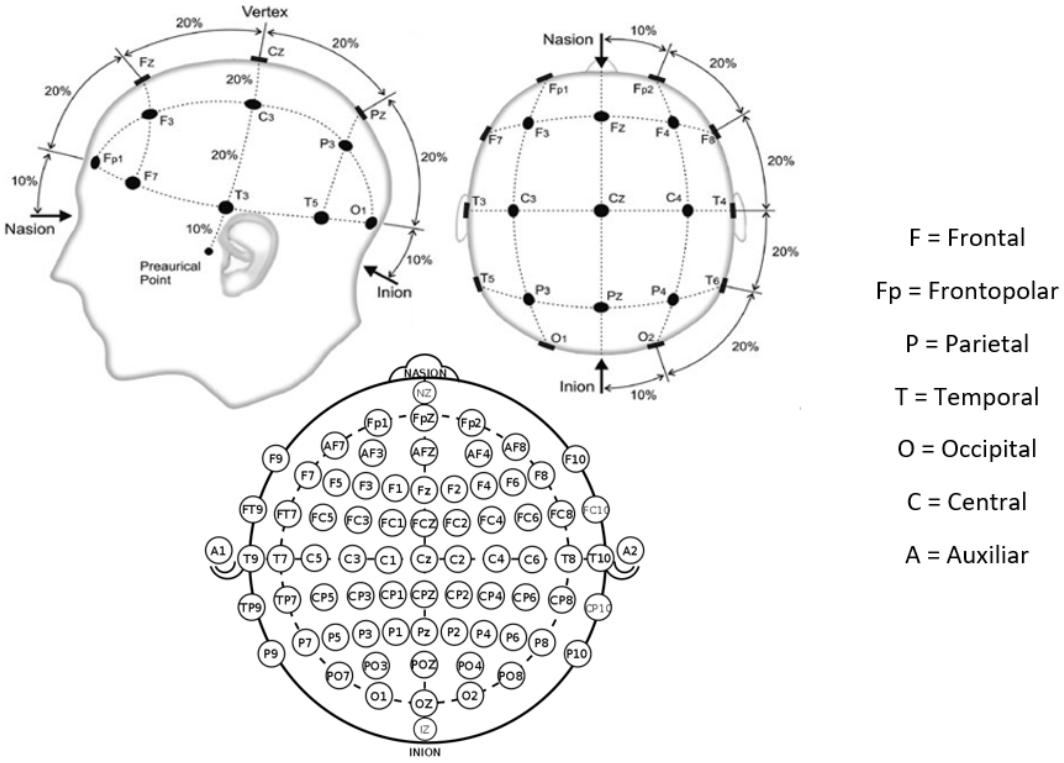


Figura 1.3: Colocación de sensores para un protocolo estándar para el intercambio de información de encaminamiento entre sistemas autónomos (EGP).

Dentro de las investigaciones BCI, se pueden encontrar aquellas enfocadas al análisis de señales EEG para realizar visualización motora *(Motor Imagery)*. En estos sistemas el usuario tiene solamente la tarea de pensar en controlar un dispositivo o en realizar alguna actividad como es el movimiento de alguna extremidad (brazo, pierna, mano, dedos, etc.) pero sin realizar dicha acción.

1.1.3. Trabajos Relacionados

En los trabajos realizados por Kai Keng Ang, Liu Guangquan, Wei Song, Damien Coyle y Jin Wu, que se describen a continuación, se implementan sistemas BCI enfocados a visualizaciónes motoras, en donde el conjunto de datos o *dataset* EEG es proporcionado por el *Institute for Infocomm Research*. Este dataset comprende 4 clases de registros de EEG de 9 sujetos, a saber, mano izquierda, mano derecha, ambos pies y lengua. La adquisición de la señal se llevó a cabo en 2 sesiones en diferentes días con cada sujeto. Cada sesión comprendió 6 *sets* separados por descansos

cortos. Cada *set* consiste en 12 pruebas de cada una de las clases mencionadas, para obtener un total de 48 pruebas por sujeto y un total de 288 pruebas por sesión.

En el trabajo realizado por Kai Keng Ang [9], se implementa un algoritmo con múltiples clasificadores para detectar las características de cada una de las clases. Cada clasificador selecciona las características discriminatorias utilizando un patrón espacial común (CSP por sus siglas en inglés, *Common Spatial Pattern*) para cada clasificador de componentes. La clasificación de las características CSP seleccionadas se realiza utilizando el clasificador *Naïve Bayes*. El resultado de clasificación final se determina a partir de todos los resultados de probabilidad de cada uno de los clasificadores de componentes. Como el cálculo de cada punto de tiempo de los datos de evaluación es computacionalmente extenso, la clasificación se realiza en cada décima muestra alternativa y se utiliza la retención de orden cero para hacer un mapa de cada vez que se toman muestras. El rendimiento se evaluó en términos de la medida del coeficiente *kappa*, los resultados mostraron que produjo el mejor promedio del coeficiente *kappa* con valor medio de 0.572.

Liu Guangquan en [10], aplica un filtro pasa-banda para eliminar los artefactos EOG (Electroocolugrama) de la señal. Para cada una de las clases, se aplica CSP para encontrar los filtros espaciales. Las varianzas de registros de los 8 mejores componentes se usaron como características. Lo que resulta en un vector de 48 características dimensionales para cada prueba en cada punto de tiempo. El periodo de tiempo para el entrenamiento del CSP se determinó mediante una validación cruzada de diez veces en el conjunto de datos de entrenamiento. El análisis discriminante lineal de Fisher (LDA por sus siglas en inglés, *Linear Discriminant Analysis*) se utilizó para reducir la dimensión de 48 a 3. Finalmente, se implementó un clasificador bayesiano para catalogar las características tridimensionales. Para cada ensayo de prueba, sólo el período de tiempo entre el segundo 3-6 se proporcionó con etiquetas estimadas. El rendimiento se evaluó en términos de la medida del coeficiente *kappa*. Los resultados mostraron que produjo el mejor promedio del coeficiente *kappa* con valor medio de 0.52.

En el trabajo realizado por Wei Song [11], se aplica regresión lineal para eliminar los artefactos EOG antes del procesamiento posterior de los datos, filtrar las señales y utilizar un algoritmo

basado en la eliminación de canales recursivos para seleccionar los canales. Wei Song divide los datos en 3 subconjuntos, implementa un CSP para la extracción de características y un clasificador de máquinas de vectores de soporte (SVM por sus siglas en inglés, *Suppor Vector Machine*). El rendimiento se evaluó en términos de la medida del coeficiente *kappa*, los resultados mostraron que produjo el mejor promedio del coeficiente *kappa* con valor medio de 0.31.

Damien Coyle implementa en [12] un algoritmo de predicción de series de tiempo basado en una red neuronal difusa (NTSPP por sus siglas en inglés, *Neural Time Series Prediction Preprocessing*). En algunos casos el rendimiento de sujetos es ligeramente mejor sin NTSPP sin embargo, para la mayoría de los sujetos incrementa significativamente en la precisión promedio. Para la extracción de características, se utilizó el modelo CSP. La varianza logarítmica de cada canal de salida del modelo CSP se calcula dentro de una ventana deslizante de un segundo en todos los puntos temporales y estos datos se utilizan como características. Coyle implementó varios clasificadores incluyendo 3 diferentes variantes de LDA, SVM para conjuntos de datos multiclase.En la mayoría de los casos, un solo clasificador funcionó mejor, aunque en algunos casos el conjunto de técnicas mejoró el rendimiento. El desempeño se evaluó en términos de la medida del coeficiente *kappa*, los resultados mostraron que produjo el mejor promedio del coeficiente *kappa* con valor medio de 0.30.

En el trabajo realizado por Jin Wu [13] se analizó la señal de los electrodos (canales) alrededor de C3 y C4 para luego filtrarlas. Para la eliminación de señales EOG se implementó un método de regresión automática. Se diseñó un clasificador de clases múltiples de dos jerarquías mediante el refuerzo del método de aprendizaje por conjuntos. El algoritmo de CSP se utilizó para la extracción de características y se empleó la primera jerarquía SVM como clasificador de dos contra dos (clase 1 y 2, contra clase 3 y 4), diseñando dos SVMs de jerarquía de segundo orden como clasificadores uno contra uno. Todos los parámetros fueron probados con validación cruzada doble. El rendimiento se evaluó en términos de la medida del coeficiente *kappa*, los resultados mostraron que produjo el mejor promedio del coeficiente *kappa* con valor medio de 0.29.

Diferentes enfoques basados en aprendizaje profundo se han utilizado exitosamente en recientes estudios y los resultados que han reportado han superado a aquellos modelos basados en clasi-

ficadores como SVM, LDA, o probabilísticos. Una de sus principales ventajas es que la parte de extracción de características la realizan dentro del mismo modelo de aprendizaje profundo, evitando así una selección manual de características.

En el trabajo realizado por Lijuan Duan, Menghu Bao, Jun Miao, Yanhui Xu y Juncheng Chen [14], se aplicó un sistema de clasificación basado en aprendizaje máquina extrema multicapa (ML-ELM). Para la reducción del vector de características se utiliza una combinación de PCA y LDA, y para la clasificación de la señal EEG el modelo propuesto de ML-ELM. Los autores demostraron que este método mejoró el rendimiento de clasificación para los datos EEG de visualización motora en contraste con el estado del arte. Los resultados reportados lograron un *accuracy* de un 94 % con ML-ELM en comparación con otros métodos donde varía del 88 al 93 %.

Xianlun Tang, Na Zhang, Jialin Zhou y Qing Liu [15] utilizaron el conjunto de datos de BCI *competition* IV-2a [16] y propusieron un método llamado *optimized hidden-layer visible deep stacking network* de PSO (PHVDSN) para la extracción de características y el reconocimiento de señales de EEG de visualización motora. Los autores proponen que el conocimiento previo en la capa intermedia de la red de apilamiento profundo (DSN) y los nodos ocultos se expandan mediante el entrenamiento no supervisado de la máquina restringida de Boltzmann (RBM). Luego, se aplica el algoritmo de optimización del enjambre de partícula (PSO) a los pesos de entrada. El rendimiento del método propuesto se evalúa con señales EEG reales de diferentes sujetos. Los resultados experimentales muestran que la precisión de reconocimiento de PHVDSN es de 81.35 %, siendo superior a algunos algoritmos que obtuvieron desempeños entre el 66 y el 79 % de precisión de reconocimiento.

En el trabajo por Ming-ai Li, Xing-yong Luo, y Jin-fu Yang [17], se sugiere una nueva técnica en la investigación de visualización y clasificación de señales EEG. Esta técnica se emplea para la reducción de la dimensionalidad no lineal no supervisada, que es denominada *t-distributed stochastic neighbor embedding* (t-SNE) y un clasificador para la extracción de características *Discrete Wavelet Transform Power* (DWTP). Esta técnica es adecuada, especialmente cuando el conjunto de datos contiene poco ruido, la cual se aplica a componentes de onda seleccionados para obtener

las características no lineales. Posteriormente, se hace una combinación en serie para construir el vector de las características. Las pruebas experimentales muestran que las características no lineales tienen un gran rendimiento de visualización con una distribución obvia de agrupamiento y el método de extracción de características indica un excelente rendimiento de clasificación evaluado por un SVM. Los resultados que reportan son del 94.10 % de clasificación utilizando la extracción de características DWT-P.t-SNE y el clasificador SVM. Los autores realizan una comparación con otros métodos que logran desempeños entre 84 y 92 %.

En el estudio realizado por Yousef Rezaei Tabar y Ugur Halici [18] utilizando el mismo conjunto de datos de Kai Keng Ang de BCI competition [9], se analizaron las redes neuronales convolucionales (CNN) y autocificadores apilados (SAE) para clasificar las señales EEG de visualización motora. Se propuso una nueva red profunda que es una combinación de CNN y SAE, así las características que se extraen en CNN se clasifican a través de la red profunda SAE. Los autores reportan el resultado de valor *kappa* de 0.547, observando claramente que se produce una mejora del 9 % con respecto al trabajo de Kai Keng Ang [9].

Zhichuan Tang, Chao Li y Shouqian Sun proponen en [15] un nuevo método basado en CNN para disminuir los errores en clasificación en un solo ensayo de visualización motora del EEG. Los autores proponen un modelo de CNN de 5 capas para clasificar las tareas de movimiento de la mano izquierda y de la mano derecha. El modelo CNN experimental se aplica en el conjunto de datos recogido de diferentes sujetos, y se compara la clasificación con otros tres métodos convencionales (potencia de la señal + SVM, CSP + SVM, AR + SVM). Los resultados demuestran que la CNN puede mejorar aún más la clasificación ya que reporta porcentajes de 9.24 %, 3.80 % y 5.16 % superior a los otros modelos.

Como se puede observar, los resultados que se reportaron en investigaciones donde el clasificador está basado en algoritmos de aprendizaje profundo superan a aquellos en donde se utilizan técnicas comunes de aprendizaje máquina como algoritmos de SVM, LDA, probabilísticos, entre otros.

1.2. Justificación

Los avances en la investigación y desarrollo de sistemas BCI dependen principalmente de la identificación y clasificación correcta de señales cerebrales para seleccionar los mejores algoritmos para traducir estas señales en comandos y así lograr la mejor clasificación de características utilizando un hardware adecuado para su realización.

En los antecedentes que se presentan en este documento se observa como los resultados que reportan investigaciones de BCI basadas en métodos de aprendizaje profundo logran una mejor clasificación de las señales en comparación con otros enfoques. Una de las razones por las cuales el uso de estas técnicas se ha incrementado en la comunidad científica es por el hecho de que no necesitan de una etapa de preprocesamiento en donde se definen las características que utilizará el clasificador. La arquitectura del modelo de aprendizaje profundo realiza este procesamiento. Por tal motivo resulta de gran interés el análisis de estas técnicas de aprendizaje profundo identificando las diferentes arquitecturas que se han implementado, resumiendo los resultados a los que se han llegado y así definir una nueva arquitectura que clasifique diferentes visualizaciones motoras.

1.3. Objetivos

1.3.1. Objetivo General

Implementar un modelo computacional basado en aprendizaje profundo que sea capaz de clasificar cuatro diferentes visualizaciones motoras: mano derecha, mano izquierda, ambos pies y movimientos con la lengua.

1.3.2. Objetivos Específicos

- Analizar los diferentes conjuntos de datos de competencias BCI enfocadas a tareas de visualización motora.
- Realizar un estudio de las diferentes técnicas basadas en aprendizaje profundo utilizadas para la clasificación de señales EEG y que se enfocan a tareas de visualización motora con el fin de encontrar un clasificador que proporcione buenos resultados

- Implementar en software las técnicas que resulten ser más estables para la clasificación de visualizaciones motoras.
- Comparar los resultados obtenidos con los reportados en publicaciones recientes.

1.4. Hipótesis

El uso de modelos basados en aprendizaje profundo para el reconocimiento y clasificación de señales de visualización motora producirá mejores desempeños en comparación con algoritmos tradicionales de aprendizaje máquina.

2

Marco Teórico

En la literatura se han desarrollado diferentes modelos computacionales de aprendizaje máquina, dentro de los que destacan los modelos de perceptrón multi capa (por sus siglas en inglés MLP, *Multi-layer Perceptrón*, las redes neuronales artificiales (por sus siglas en inglés ANN, *Artificial Neural Networks*), redes neuronales convolucionales (por sus siglas en inglés CNN, *Convolutional Neural Network*), redes neuronales recurrentes (por sus siglas en inglés RNN, *Recurrent Neural Network*), por mencionar algunas, que se han aplicado a campos como visión artificial, reconocimiento de voz, procesamiento de lenguaje natural, reconocimiento de audio, filtrado de redes sociales, traducción automática y bioinformática [9] [19]. En este apartado se dará una explicación general de este tipo de modelos poniendo énfasis en los algoritmos que se utilizan para analizar señales EEG.

2.1. Aprendizaje Máquina

El aprendizaje automático consiste en programar computadoras para optimizar un criterio de rendimiento utilizando datos de ejemplo o experiencias pasadas. Se tiene un modelo definido de acuerdo con algunos parámetros, y el aprendizaje es la ejecución de un programa de computadora para optimizar los parámetros del modelo utilizando los datos de entrenamiento o la experiencia pasada. El modelo puede ser predictivo, descriptivo o ambos. El predictivo es para hacer predicciones en el futuro y el descriptivo para obtener conocimiento de los datos [20].

El aprendizaje automático utiliza la teoría de las estadísticas en la construcción de modelos matemáticos, porque la tarea principal es hacer una inferencia a partir de una muestra. El rol de las ciencias computacionales es doble:

Primero, en el entrenamiento, se necesitan algoritmos eficientes para resolver el problema de optimización, así como para almacenar y procesar la enorme cantidad de datos que generalmente se tienen.
En segundo lugar, una vez que se aprende un modelo, su representación y solución algorítmica para la inferencia también debe ser eficiente[20].
En ciertas aplicaciones, la eficiencia del algoritmo de aprendizaje o de su complejidad de espacio y tiempo, puede ser tan importante como su precisión predictiva.

El aprendizaje automático se puede clasifican en:

- Aprendizaje supervisado: Permite buscar patrones en el histórico de datos generando una relación entre los campos y un campo objetivo.
- Aprendizaje no supervisado: A diferencia del aprendizaje supervisado, en este caso se utiliza el histórico de datos que no se encuentran etiquetados con el fin de encontrar alguna estructura o forma de organizarlos.

2.2. Aprendizaje Profundo

El aprendizaje profundo es una técnica de aprendizaje automático que enseña a una computadora a simular el enfoque de aprendizaje que utilizan los seres humanos. Este modelo aprende a realizar tareas de clasificación que pueden llegar a obtener mejor precisión que el rendimiento humano. Los modelos se entrenan mediante conjuntos etiquetados y arquitecturas de redes neuronales que contienen varias capas. A diferencia del aprendizaje máquina, este método no tiene la necesidad de una extracción manual de características.

2.3. Red Neural Artificial

La neurona artificial se encuentra conceptualmente inspirado en la neurona biológica [21]. Esta es una red de elementos simples, que se forma conectando la salida de ciertas neuronas en la entrada de otras, formando así un grafo dirigido [22] (véase Figura 2.1).

Dentro de una red de neuronas se localiza un peso o fuerza sináptica que se encarga de determinar la fuerza de conexión entre cada neurona. Para evaluar la neurona se calcula el conjunto de

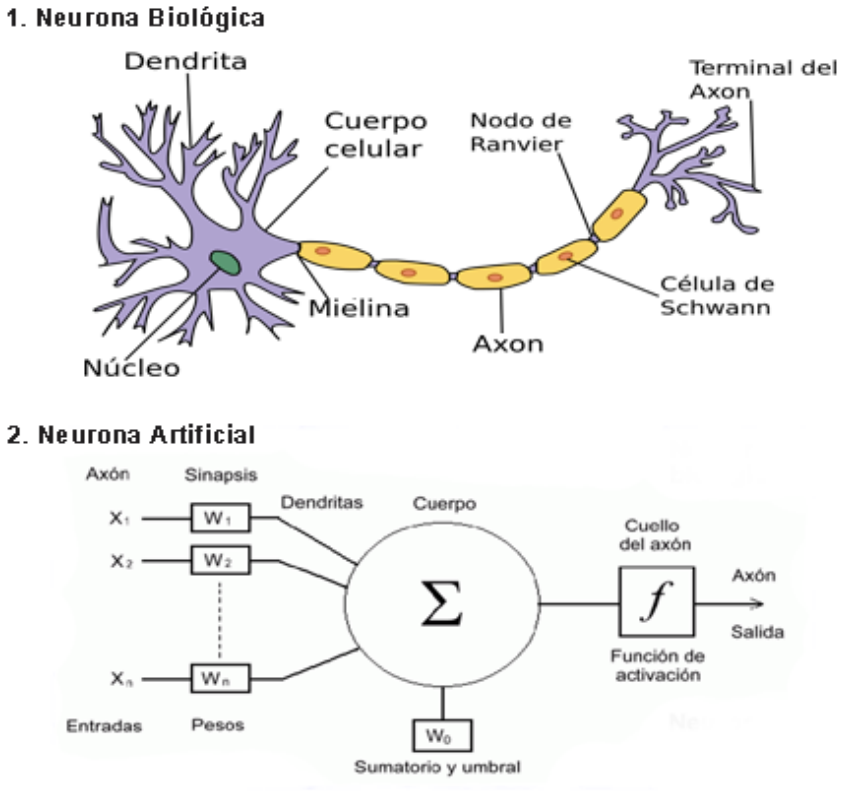


Figura 2.1: Representación de una Neurona Biológica y una Neurona Artificial

todas las fuerzas de la red (NET) que se reciben por sus entradas. Una vez que se realiza dicho cálculo se aplica la función de activación para determinar el valor del estado interno de la neurona para así convertirse en el valor de la salida [23]. La combinación de las señales que recibe una neurona se puede calcular como se muestra en la siguiente ecuación:

$$NET_i(t) = \sum_{j=1}^{N-1} [W_{ij} \cdot O_j \cdot (t-1)] \tag{2.1}$$

donde W_{ij} representa el peso de la conexión que hay entre las neuronas *j* e *i*.

Una neurona artificial consiste en los siguientes componentes:

- Una activación, dependiendo de un parámetro de tiempo discreto.
- Puede haber un umbral que si no es cambiado por una función de activación pertenecerá fijo.

- La Función de activación f que es representada por la ecuación:

$$A_i(t) = f(A_i(t-1), NET_i(t-1)) \tag{2.2}$$

calcula la nueva activación en un momento dado y la entrada neta dando lugar a la relación. Algunas de las funciones de activación pueden ser tipo escalón o sigmoidea (véase Figura 2.2).

- Y la función de salida

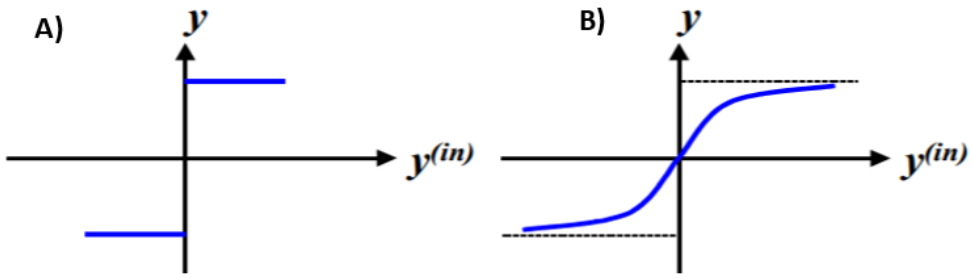


Figura 2.2: Algunas Funciones de Activación típicas. A) Función Escalón. B) Función Sigmoidea.

2.4. Redes Neuronales Recurrentes

Las redes recurrentes, también conocidas como redes re-alimentadas, son redes en las que se permiten autoconexiones o lazos de retroalimentación en las neuronas y conexiones hacia atrás en las capas (véase Figura 2.3). Para entrenar este tipo de redes se siguen principalmente la aproximación al *Backpropagation*[24].

Backpropagation consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados. Primero se aplica un patrón de entrada como estímulo a las neuronas de la primera capa de la red, esta entrada se va propagando a través de todas las capas hasta generar una salida. Luego, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada una de ellas. A continuación, estos errores se propagan hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total y con ello se calcula una actualización en todas ellas [25].

2.5. *Multi-layer Perceptron (MLP)*

El perceptrón multicapa (por sus siglas en inglés MLP) consiste en un sistema simple de neuronas o nodos interconectados que representa un mapeo no lineal entre un vector de entrada y un vector de salida [26]. En la Figura 2.4 se muestra este modelo. Los nodos están conectados por pesos y señales de salida que son una función de la suma de las entradas al nodo modificado por una función de transferencia no lineal o activación [27].

La arquitectura de un perceptrón multicapa es variable, pero en general consiste en varias capas de neuronas, puede tener una o más capas ocultas y finalmente una capa de salida.

El perceptrón multicapa se ha aplicado a una amplia variedad de tareas, todas las cuales se pueden clasificar como predicción, aproximación de funciones o clasificación de patrones.

La predicción implica la anticipación de tendencias futuras en una serie temporal de datos dadas las condiciones actuales y anteriores. La aproximación de funciones se refiere a modelar la relación entre variables. La clasificación de patrones implica catalogar datos en clases discretas [27].

El entrenamiento del MLP consta de dos partes, calcular cuáles son las salidas para las entradas

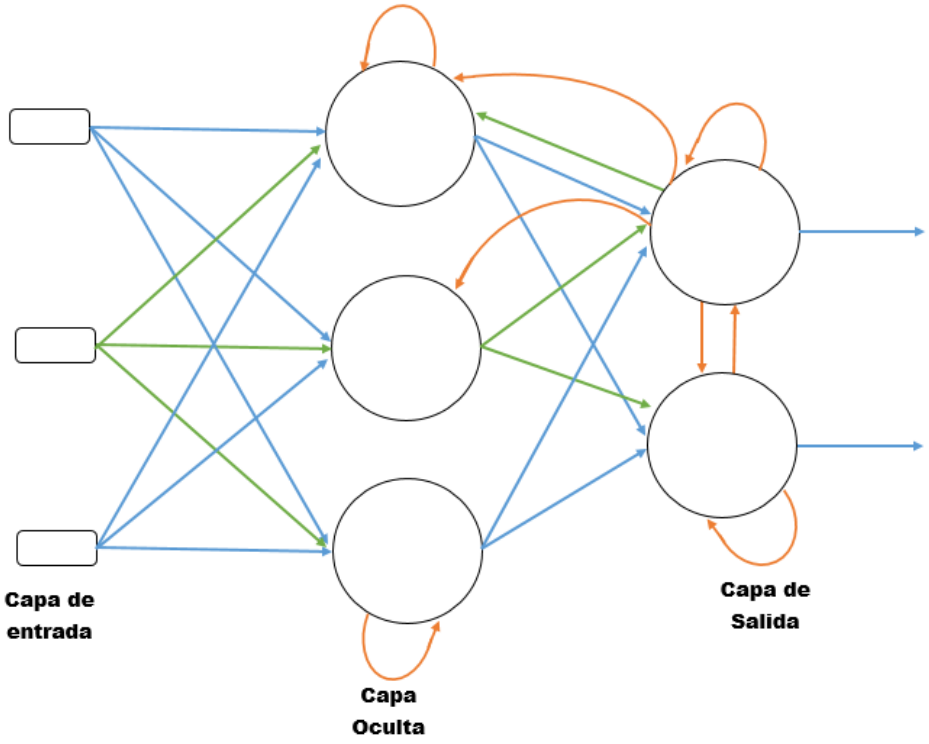


Figura 2.3: Representación de una red recurrente.

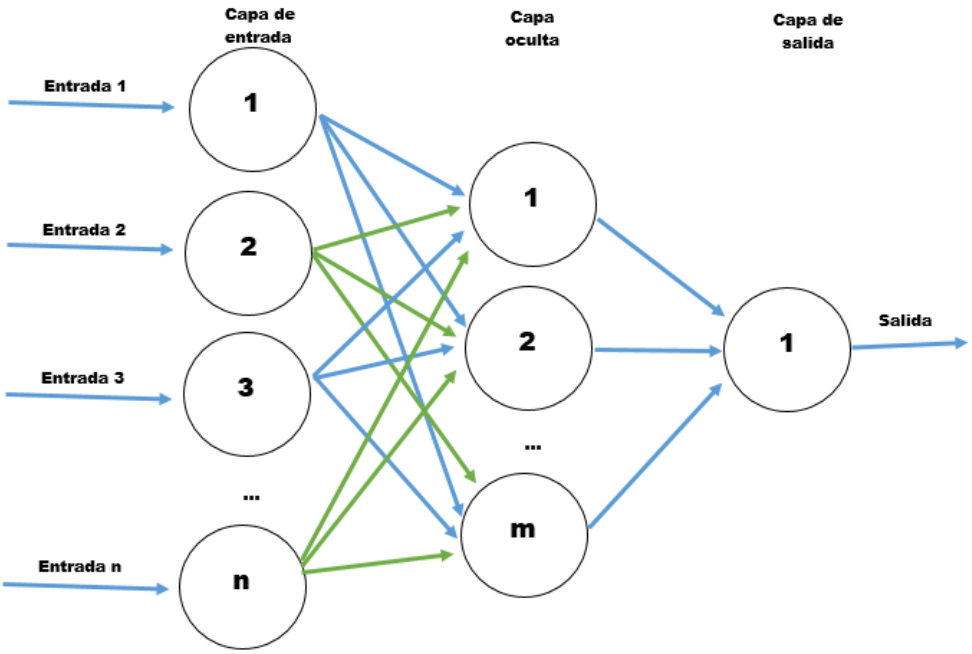


Figura 2.4: Arquitectura MLP.

dadas y las ponderaciones actuales, y luego actualizar las ponderaciones según el error, que es una función de la diferencia entre las salidas y los objetivos. Generalmente se conocen como *forward* y *backward* a través de la red [28].

A continuación se describirá el algoritmo que se lleva a cabo en un modelo *Multi-Layer Perceptrón* (MLP). Los índices *i, j, k* describen a los nodos en cada capa, mientras que las letras griegas correspondientes a (ι, ζ, κ) representan índices fijos.

- **Inicialización.**
  - Se inicializan todos los pesos de las neuronas con valores aleatorios pequeños (positivos y negativos).
- **Entrenamiento.**
  - Repetir

    Fase *Forward*

    Calcular la activación de cada neurona *j* en la(s) capa(s) oculta(s) donde la entrada y los pesos de la primera capa se identifican con la etiqueta υ y la función de activación con g(·). La salida de dichas neuronas y los pesos de la segunda capa se etiquetan con ω.

$$h_{\zeta i}(t) = \sum_{j=0}^{L} x_j \upsilon_{j\zeta} \tag{2.3}$$

$$a_\zeta = g(h_\zeta) = \frac{1}{1 + e^{-\beta h_\zeta}} \tag{2.4}$$

Trabajar a través de la red hasta que llegue a las neuronas de la capa de salida, que estén activas.

$$h_k = \sum_j a_j \omega_{jk} \tag{2.5}$$

$$y_k = g(h_k) = \frac{1}{1 + e^{-\beta h_k}} \tag{2.6}$$

Fase *Backward*

Calcular el error de salida usando:

$$\delta_0(k) = (y_k - t_k) y_k (1 - y_k) \tag{2.7}$$

Calcular el error en las capas ocultas usando:

$$\delta_0(\zeta) = a_\zeta (1 - a_\zeta) \sum_{k=1}^{N} \omega_\zeta \delta_0(k) \tag{2.8}$$

Actualizar los pesos de la capa de salida usando:

$$\omega_{\zeta k} \leftarrow \omega_{\zeta k} - \eta \delta_0(k) a_\delta^{hidden} \tag{2.9}$$

Actualizar los pesos de la capa oculta usando:

$$\nu_\iota \leftarrow \nu_\iota - \eta \delta_h(k) x_\iota \tag{2.10}$$

- **Repetir**
  - Desde la fase *Forward* en la sección de entrenamiento anterior hasta que el error sea mínimo o se llegue a un determinado número de épocas.

2.6. *Support Vector Machine (SVM)*

Las Máquinas Vectoriales de Soporte (por sus siglas en inglés SVM) son un conjunto de métodos de aprendizaje automático supervisado utilizados principalmente para problemas de clasificación, regresión y detección atípica [29]. La idea de este algoritmo es hacer uso de una función para representar el conjunto de datos y encontrar el mejor hiperplano que definirá de forma óptima si un elemento pertenece a una clase o a otra. Para encontrar la mejor solución medirá el margen de hiperplano buscando un punto máximo (véase Figura 2.5). El margen es la distancia entre el hiperplano con el patrón más cercano de cada clase. Los patrones más cercanos se llaman vectores de soporte [30].

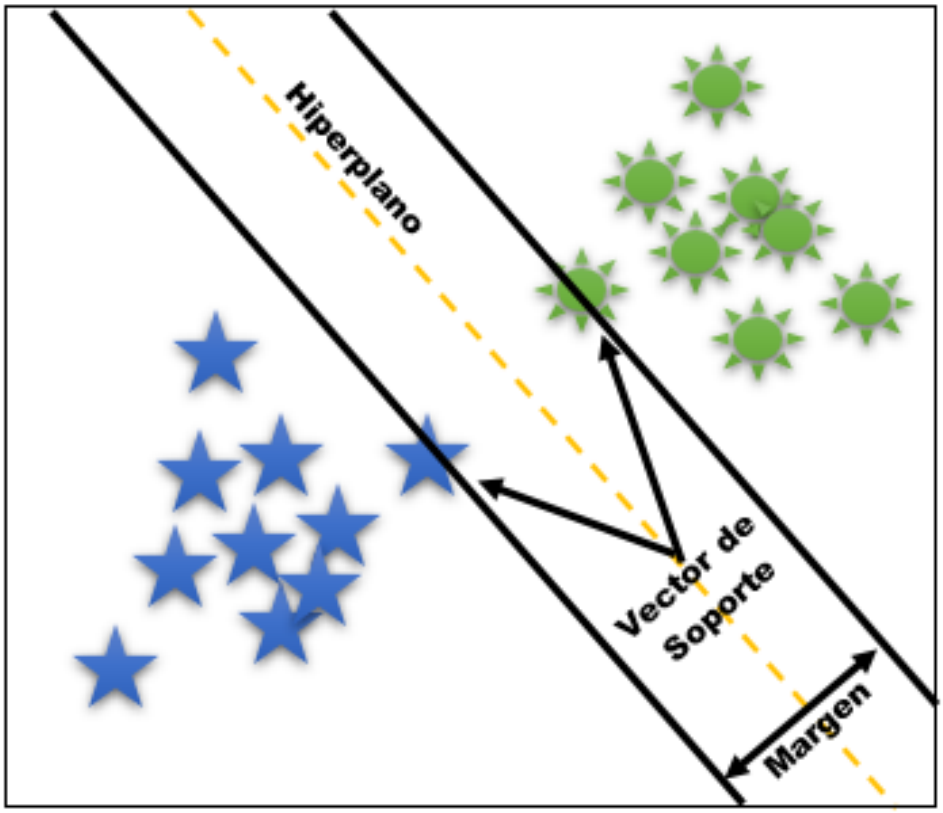


Figura 2.5: Representación de SVM

El SVM puede ser utilizado para problemas lineales y no lineales, es decir, cuando el problema es lineal (Véase Figura 2.6), el conjunto de datos puede ser separado por un hiperplano dejando los datos de una clase en un lado y los datos de la otra clase en el otro. De lo contrario cuando no son linealmente separables (Véase Figura 2.6), no existe un hiperplano que cumpla correctamente

dicha separación, por lo que se debe hacer el uso de una función *Kernel* (Lineal, Sigmoide, Radial Gaussiana) que se encargue de proyectar la información a un espacio de características de mayor dimensión.

A continuación, se mencionan diferentes tipos de *kernels* utilizados en modelos de SVM:
Polinómico: Representa las muestras de entrenamiento en un espacio funcional sobre polinomios de las variables originales, permitiendo el aprendizaje de modelos lineales. Para polinomios de grado *d* donde *x,y* son vectores en el espacio de entrada y *c* es un parámetro libre y se define como:

$$Ke(x, y) = (c + x^T y)^d \tag{2.11}$$

Sigmoidea: el kernel de función sigmoidea se representa por los elementos de los vectores de entrada x_k y los parámetros *k* y δ

$$Ke(x, y) = tanh(kx^T y - \delta) \tag{2.12}$$

Radial: En este kernel se tiene a la expansión radial de la función base de los elementos x_k de los vectores de entrada, con un parámetro σ

$$Ke(x, y) = exp(-(x - y)^2/2\sigma^2) \tag{2.13}$$

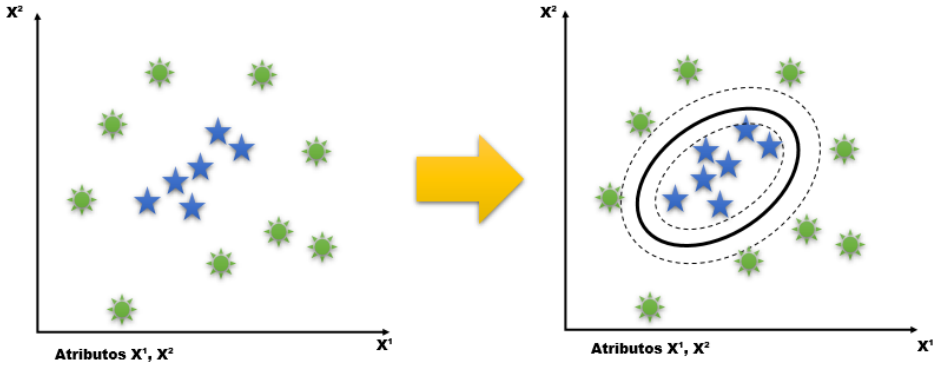


Figura 2.6: Representación de separación lineal de un SVM

A continuación, se da una explicación de cómo funciona el modelo de *Support Vector Machine* (SVM).

- **Inicialización**

  - Para especificar el Kernel y los parámetros del mismo, se deben calcular las distancias entre los puntos de datos.
    - El trabajo principal es calculado con $Ke = XX^T$
    - Para el *Kernel* Lineal, regresar ***Ke***, para el polinomio de grado ***d*** regresa $\frac{1}{d}\ Ke^d$
    - Para el *Kernel* radial, calcular $Ke = exp(-x - x')^2/2\sigma^2$

  - **Entrenamiento**
    - Ensamblar el conjunto de restricciones como matrices a resolver:

$$min_x \frac{1}{2} x^T t_i t_j Kex + q^x \quad sujeto\ a \quad Gx \geq h, Ax = b \tag{2.14}$$

      donde *x* es la variable por resolver, *q* es un vector columna
    - Resolver las matrices
    - Identificar los vectores de soporte que están dentro de una distancia especificada del punto más cercano y disponer del resto de los datos de entrenamiento.
    - Calcular *b** utilizando la siguiente ecuación:

$$b* = \frac{1}{N_s} \sum_{supportvectors j} (t_j - \sum_{i=1}^{n} \lambda_i t_i X_t^T X_j) \tag{2.15}$$

2.7. *K-Nearest Neighbors (KNN)*

K-vecinos más cercanos (por sus siglas en inglés, KNN), es un clasificador donde se encuentran los patrones más cercanos a un patrón objetivo *x*, para los cuales se busca la etiqueta. KNN asigna la etiqueta de clase de la mayoría de los patrones *K* más cercanos en el espacio de datos. Sea $(x_1, y_1), ..., (x_N, y_N)$ el conjunto de observaciones de patrones *q-dimensionales* $X = \{x\}_{i=1}^N \subset R^q$ y un conjunto correspondiente de etiquetas $Y = \{y_i\}_{i=1}^N \subset R^d$. Por este motivo, se debe definir una medida de similitud en el espacio de datos. En R^q, es razonable emplear la métrica de Minkowski (norma p) definida en la ecuación 2.16 [31].

$$||x' - x_j||^p = (\sum_{i=1}^{q} |(x_j)' - (x_i)_j|^p)^{\frac{1}{p}} \tag{2.16}$$

que corresponde a la distancia euclidiana para *p=2*. En otros espacios de datos, se deben elegir funciones de distancia adecuadas, por ejemplo, la distancia de Hamming en B^q. En el caso de la clasificación binaria, se usa el conjunto de etiquetas *y=1,-1*, y KNN se define como:

$$fKNN(x') = \begin{cases} 1 \; if \sum_{i \in Nk(x')} & y_i \geq 0 \\ \text{-1} \; if \sum_{i \in Nk(x')} & y_i \geq 0 \end{cases} \tag{2.17}$$

con tamaño de espacio *K* y con el conjunto de índices $N_k(x')$ de los patrones *K* más cercanos. La elección de *K* define el lugar de KNN. Para *K=1*, pequeños barrios surgen en regiones, donde los patrones de diferentes clases están dispersos. Para tamaños de vecindarios más grandes, p. *K=3*, los patrones con etiquetas en la minoría son ignorados (Véase Figura 2.7).

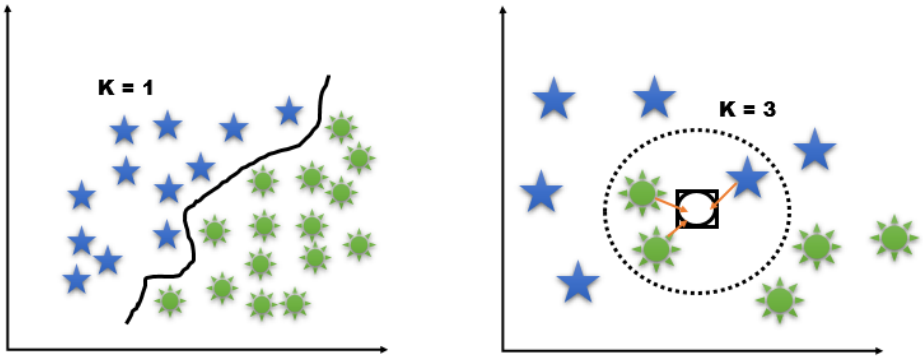


Figura 2.7: Representación de un KNN

2.8. Redes Neuronales Convolucionales

Una red convolucional se caracteriza por contar con capas de convolución en donde la señal de entrada se convoluciona con diferentes filtros (*kernels*) y se produce una salida (véase figura 2.8). Esta salida está relacionada con qué tan parecidas es la señal de entrada con el filtro. En una red convolucional se debe de definir el tamaño del filtro en cada una de las capas, así como también el tamaño de paso (*stride*) y el parámetro de rellenado de ceros (*padding*). El parámetro del tamaño de paso se refiere a la cantidad de pixeles que se va a desplazar el filtro para realizar el cálculo de la convolución. El rellenado de ceros es para que la operación de convolución pueda realizarse desde que comienza y termina la señal.

La topología de la red convolucional aplicada al análisis de señales EEG se describe a continuación [32]:

- L1: Capa de entrada.
- C2: Capa convolucional (primera capa oculta). Esta capa convoluciona el canal individual a lo largo del tiempo y se obtienen mapas de características después de la convolución.
- C3: Capa convolucional (segunda capa oculta). Esta capa tiene como objetivo filtrar las señales de EEG en el dominio del espacio. En esta capa se utilizan filtros espaciales y se obtienen mapas de características después de la operación de convolución. Se utiliza un max-pooling para producir un mapa de características.
- C4: Capa convolucional (tercera capa oculta). Esta capa tiene como objetivo convolucionar y submuestrear las señales de EEG. Para cada mapa de características en C3, se utilizan 2 filtros y el resultado es de ciertos mapas de características en total. Se utiliza un max-pooling para producir un mapa de características.
- C5: Capa convolucional (cuarta capa oculta). Esta capa pretende transformar el mapa de características en un tamaño menor. Para cada mapa de características en C4, se utilizan 2 filtros, cada uno con un tamaño. Se utiliza un max-pooling para producir un mapa de características.
- C6: Capa convolucional (quinta capa oculta). Esta capa pretende transformar el mapa de características.
- F7: Capa totalmente conectada. Cada neurona de F6 es Conectado a cada neurona de C5. C5 y F6 están completamente conectado.
- O8: Capa de salida. Esta capa tiene solo dos neuronas, que representan las dos clases del problema (estado de atención y de no atención). Una operación softmax se aplica a los dos valores para proporcionar el resultado final de la clasificación.

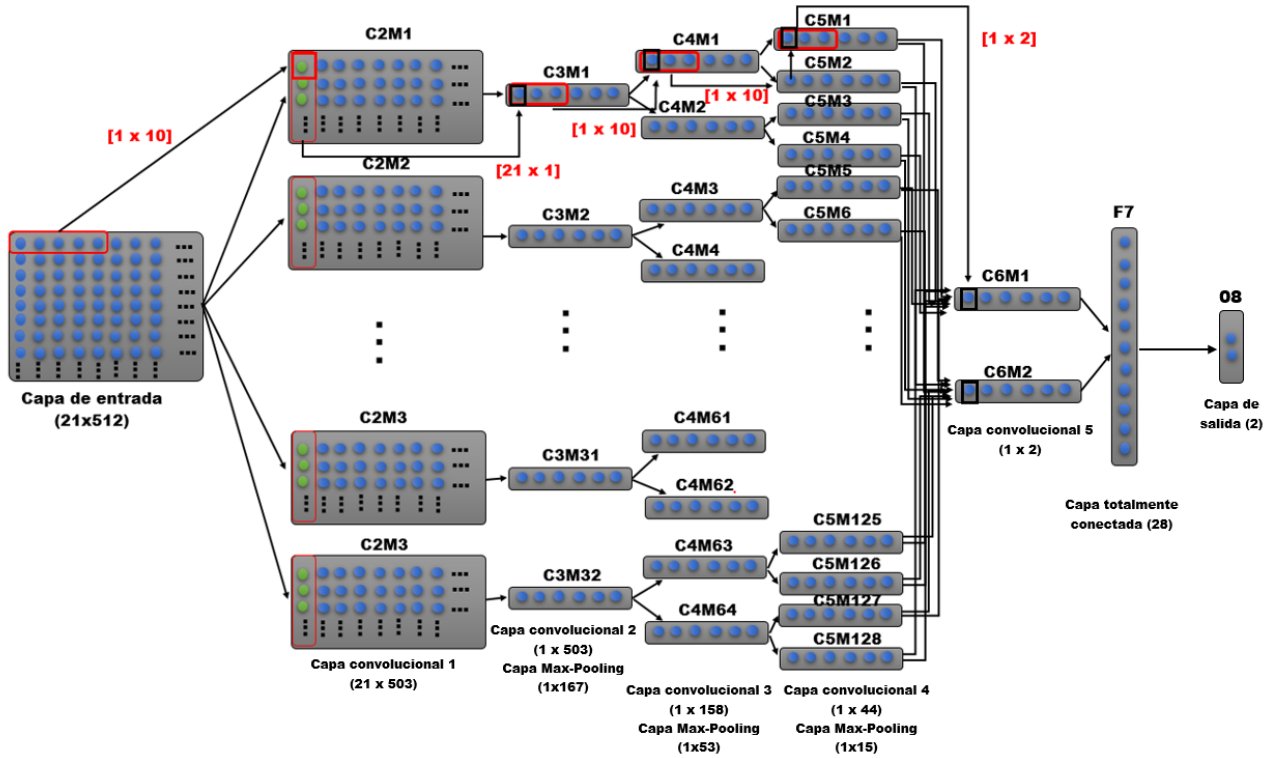


Figura 2.8: Arquitectura de una red neuronal convolucional.

2.9. Autoencoders

El uso de las redes neuronales son una forma de simular ciertas características que son particulares de los humanos, por ejemplo, la capacidad de memorizar y de relacionar sucesos [33]. Por ejemplo, un autoencoder es una red neuronal que está entrenada para intentar copiar su entrada a su salida (véase Figura 2.9). Internamente, tiene una capa oculta que describe una aplicación para representar la entrada [34]. Está formado por la combinación de un codificador (encoder) que convierte la entrada a una representación diferente y un decodificador (decoder) que regresa dicha representación a su formato original.

- Red de codificación: traduce la entrada original de alta dimensión al código latente de baja dimensión. El tamaño de entrada es más grande que el tamaño de salida.
- Red de decodificadores: recupera los datos del código, probablemente con capas de salida cada vez más grandes.

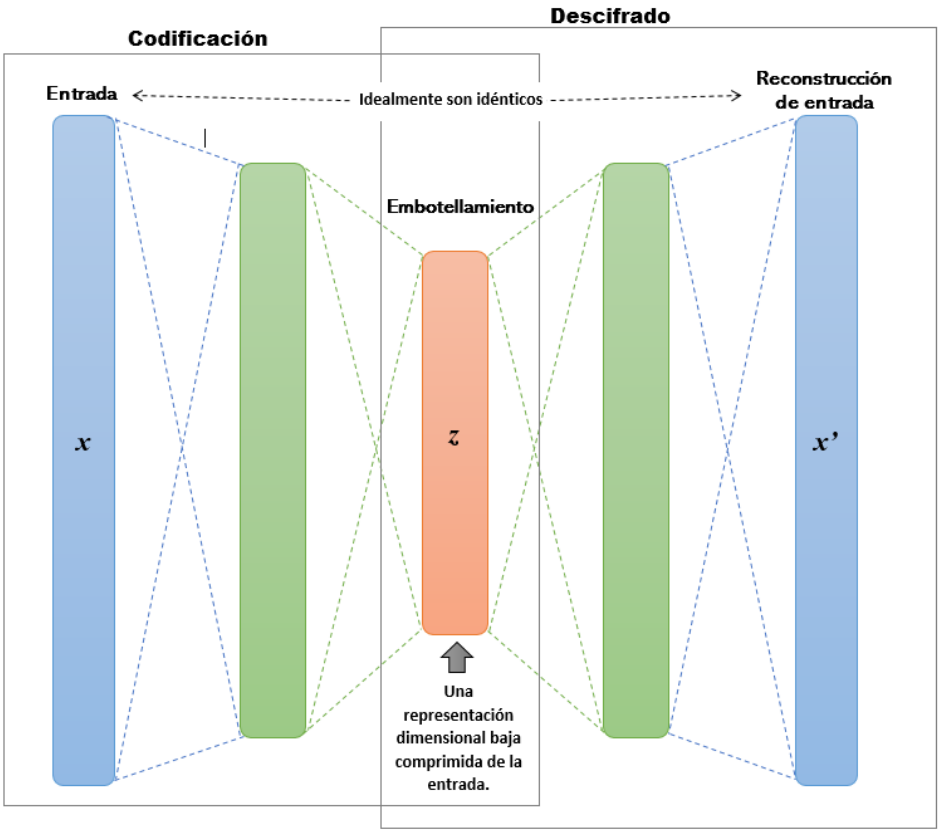


Figura 2.9: Representación de la arquitectura de autoencoder.

2.10. *Long Short Term Memory* (LSTM)

Memoria a corto y largo plazo (por sus siglas en inglés LSTM, *Long Short Term Memory*) es una red de tipo RNN que puede aprender dependencias a largo plazo entre medidas de tiempo de datos de la secuencia. Fue propuesto por Hochreiter y Schmidhuber en [35]. LSTM es capaz de superar los problemas previamente inherentes de las RNN, es decir, no sufre un error de desaparición o crecimiento exponencial a través de datos de series de largo plazo, a diferencia de otro tipo de RNN [36].

2.10.1. Arquitectura LSTM

El diagrama de bloques de la Figura 2.10 ilustra la arquitectura de una red LSTM simple para la clasificación. La red comienza con una capa de entrada de secuencia seguida de una capa LSTM. Una capa completamente conectada, una de softmax y una capa de salida se incluyen en esta arquitectura para predecir las etiquetas de la clase.

Figura 2.10: Diagrama de bloques de la arquitectura de red LSTM.

La secuencia de capa de entrada es una capa de datos de la secuencia a una red y la capa LSTM aprende las dependencias a largo plazo entre los pasos de tiempo de los datos de secuencia.

Arquitectura en capas LSTM

El LSTM contiene unidades especiales llamadas bloques de memoria en la capa oculta recurrente. Los bloques de memoria contienen celdas de memoria con auto-conexiones (véase Figura 2.11) que almacenan el estado temporal de la red además de unidades multiplicativas especiales llamadas puertas para controlar el flujo de información. Cada bloque de memoria en la arquitectura original contiene una puerta de entrada y una puerta de salida. La puerta de entrada controla el flujo de activaciones de entrada en la celda de memoria. La puerta de salida controla el flujo de salida de las activaciones de celda en el resto de la red. Después, se agrega la "puerta de olvido" al bloque de memoria [37]. Esto soluciona una debilidad de los modelos LSTM que les impide procesar flujos de entrada continuos que no están segmentados en subsecuencias. La puerta de olvido escala el estado interno de la celda antes de agregarla como entrada a la celda a través de la conexión auto-recurrente de la celda, por lo tanto, olvida adaptativamente o restablece la memoria de la celda. LSTM puede propagar el error constantemente utilizando un tipo especial de neurona llamada Constant Error Carousel (CEC). Además, la arquitectura LSTM moderna contiene conexiones de mirilla desde sus celdas internas a las puertas en la misma celda para aprender la sincronización precisa de las salidas [38].

2.10.2. Arquitectura DLSTM

La memoria profunda a corto plazo (por sus siglas en inglés DLSTM, *Deep Long Short Term Memory*) es una versión modificada de LSTM que se forma al apilar varias capas de LSTM. Cada una de las capas recibe la salida de la capa LSTM anterior como su entrada. La primera capa recibe

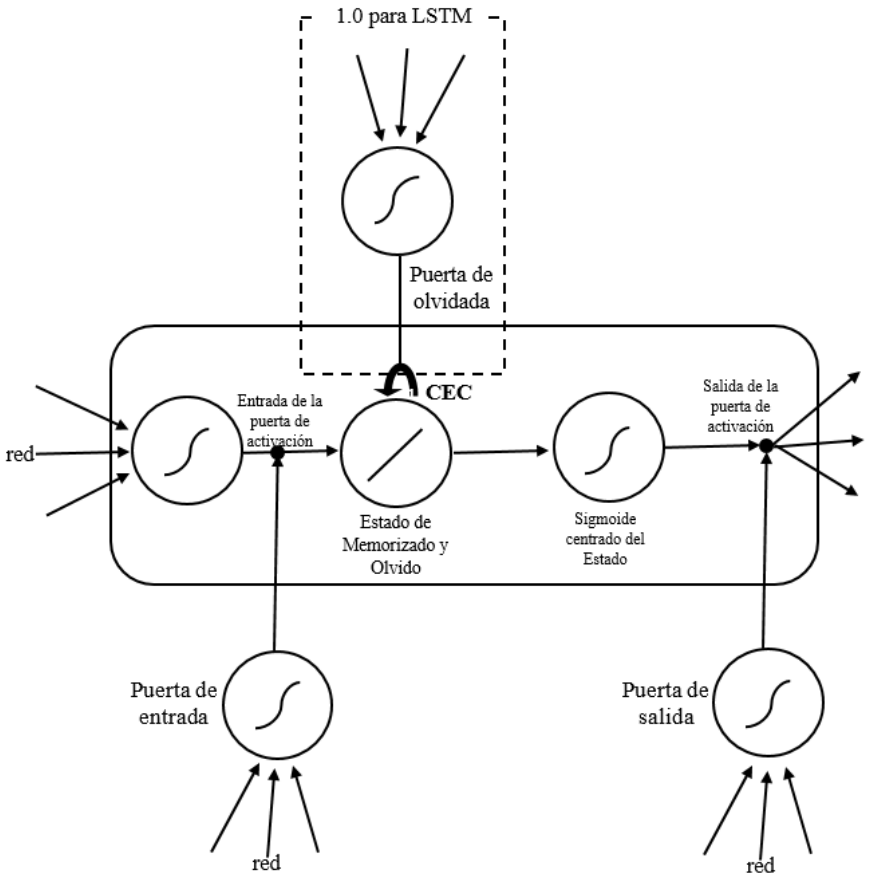


Figura 2.11: Representación de una celda de memoria

datos en bruto de series de tiempo como su entrada. La Figura 2.12 ilustra una capa simple de LSTM y las pilas de tres LSTM que forman un DLSTM.

Pseudocódigo del proceso de celda de memoria:

1. Calcular la activación de la puerta de entrada

$$y_{in_j}(t) = f\left(W_{in_j} x(t-1)\right) \tag{2.18}$$

2. Calcular el estado interno del Carrusel de error constante (CEC)

$$s_{c_j}(t) = s_{c_j}(t-1) + y_{in_j} g(W c_j x(t-1)); s_{c_j}(0) = 0 \tag{2.19}$$

3. Calcular la activación de la puerta de salida.

$$y_{out_j}(t) = f\left(W_{out_j} x(t-1)\right) \tag{2.20}$$

4. Calcular la salida de la celda de memoria

$$y_{c_j}(t) = y_{out_j} h(s_{c_j}(t)) \tag{2.21}$$

En donde *f, g y h* son funciones de activación que son usualmente las funciones sigmoideas. Wc_j, W_{in_j} y W_{out_j} son pesos que se pueden actualizar a través del algoritmo de descenso de gradiente para el aprendizaje. $x(t)$ se ingresa a la celda de memoria en el tiempo *t*.

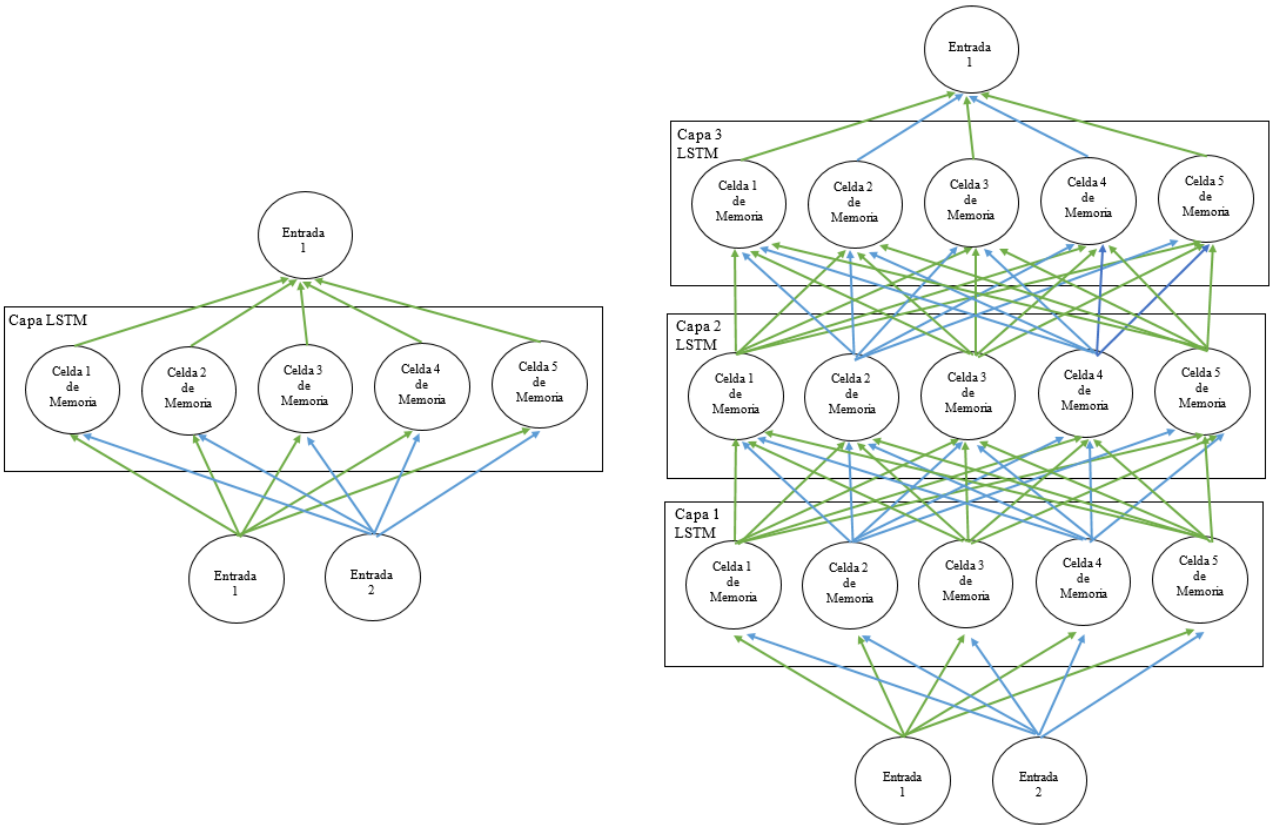


Figura 2.12: La imágen del lado izquierdo representa una arquitectura de una capa simple de LSTM. Y del lado derecho se representa la arquitectura Deep LSTM

3

Experimentacion

3.1. Descripción de los conjuntos de datos

Los conjuntos de datos 2A y 2B, de "*Berlin Brain Computer Interface*"proporcionados por "*The Institute for Knowledge Discovery"* son señales de EEG relacionadas con visualizaciones motoras, y se describen a continuación:

3.1.1. *Dataset II-A Motor Imagery* 4 clases

Paradigma experimental

Este conjunto de datos consiste en lecturas de electroencefalograma (EEG) de 9 sujetos. El paradigma BCI basado en señal consistió en cuatro diferentes tareas de imágenes motoras, a saber, la visualización motora de la mano izquierda (clase 1), mano derecha (clase 2), ambos pies (clase 3) y lengua (clase 4). Se registraron dos sesiones en diferentes días para cada tema. Cada sesión consta de 6 sets separados por pausas cortas. Una prueba consiste en 48 ensayos (12 para cada una de las cuatro clases posibles), dando un total de 288 ensayos por sesión. Al comienzo de cada sesión, se realizó una grabación de aproximadamente 5 minutos para estimar la influencia del Electrooculograma (EOG). La grabación se dividió en 3 bloques:

1. Dos minutos con los ojos abiertos (mirando una cruz de fijación en la pantalla)
2. Un minuto con los ojos cerrados
3. Un minuto con los movimientos oculares.

Los sujetos estaban sentados en un sillón frente a una pantalla de ordenador. Al comienzo de un ensayo $(t = 0s)$, aparece una cruz de fijación en la pantalla negra. Además, se presenta un breve tono acústico de advertencia.

Para el registro de la señal EEG se utilizaron 25 electrodos (véase Figura 3.1) con una distancia de 3.5cm entre cada uno, de los cuales 3 electrodos son utilizados para registrar el electrooculograma (EOG). La información se almacenó en un formato general de datos de señales biomédicas (GDF), un archivo por tema y sesión.

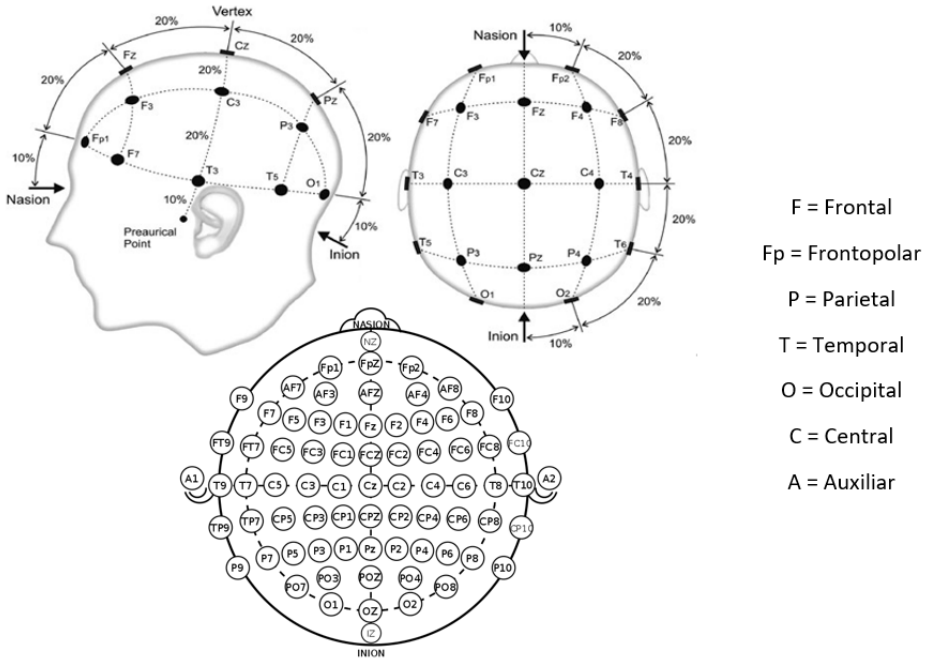


Figura 3.1: Colocación de sensores para un protocolo estándar para el intercambio de información de encaminamiento entre sistemas autónomos (EGP).

3.1.2. *Dataset II-B Motor Imagery* 2 clases

Paradigma experimental

Este conjunto de datos consiste en lecturas EEG de 9 sujetos los cuales eran diestros, tenían una visión normal o corregida a la normal y se les pagaba por participar en los experimentos. Todos los sujetos estaban sentados en un sillón, viendo un monitor de pantalla plana situado a aproximadamente 1m de distancia a nivel de los ojos. Para cada tema se proporcionan 5 sesiones, en las que las dos primeras sesiones contienen datos de entrenamiento sin retroalimentación *(screening)*, y las

últimas tres sesiones fueron grabadas con retroalimentación. El paradigma consistió en dos clases, a saber, la visualización motora (MI) de la mano izquierda (clase 1) y la mano derecha (clase 2). Cada sujeto participó en dos sesiones sin retroalimentación registrada en dos días diferentes dentro de dos semanas. Cada sesión consistió en seis pruebas con diez ensayos cada una y dos clases de imágenes. Esto dio como resultado 20 ensayos por prueba y 120 ensayos por sesión. Los datos de 120 repeticiones de cada clase de MI estaban disponibles para cada persona en total. Antes de la primera formación de visualizaciones motoras, el sujeto ejecutó e imaginó diferentes movimientos para cada parte del cuerpo y seleccionó el que mejor imaginaban (por ejemplo, apretar una pelota o tirar de un freno). Cada sesión consta de varias ejecuciones. Al comienzo de cada sesión, se realizó una grabación de aproximadamente 5 minutos para estimar la influencia del electrooculograma (EOG). La grabación se dividió en 3 bloques:

1. Dos minutos con los ojos abiertos (mirando una cruz de fijación en la pantalla).

2. Un minuto con los ojos cerrados.

3. Un minuto con los movimientos oculares.

El bloque de artefactos, en donde el paciente debía de realizar movimientos de parpadeo, balanceo, arriba o abajo, se dividió en cuatro secciones (artefactos de 15 segundos con 5 segundos de descanso). Al principio y al final de cada tarea se presentó un tono de advertencia bajo y alto, respectivamente. El registro de los datos se realizó mediante tres grabaciones bipolares (C3, Cz y C4, véase Figura 3.2). La colocación de los electrodos fue ligeramente diferente para cada sujeto, utilizando el electrodo Fz como tierra. Además de los canales EEG se utilizaron 3 electrodos monopolares con el mismo ajuste para registrar el electrooculograma (EOG). Almacenándose toda la información en un formato general de datos para señales biomédicas (GDF), un archivo por tema y sesión.

3.2. Procesamiento de los conjuntos de datos

A los 2 conjuntos de datos mencionados se les aplicó un filtro pasa banda Butterworth de cuarto orden con las frecuencias de 8 a 13 Hz y de 13 a 45 Hz, desarrollando un nuevo modelo de

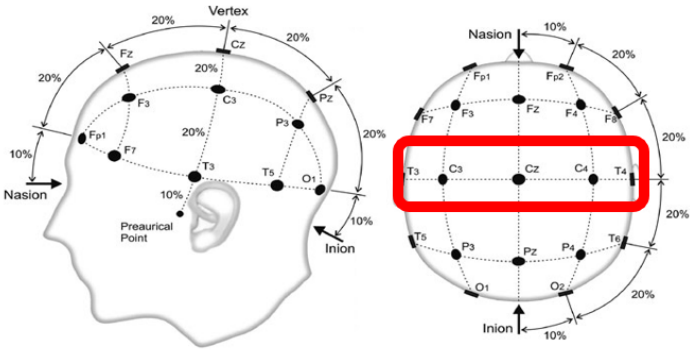


Figura 3.2: Ubicación de los electrodos C_3, C_z y C_4

normalización para cada canal EEG, minimizando la influencia de los artefactos EOG y mejorando las características, esto tomando como base el documento en [39].

Posteriormente, mediante el algoritmo Hjorth [39] se extrajeron características de actividad (potencia media), movilidad (frecuencia media) y complejidad (ancho de banda). Dichas características se calcularon como se indica a continuación:

$$m_0 = \sigma_y^2 \tag{3.1}$$

$$m_2 = \frac{\sigma_d}{\sigma_y} \tag{3.2}$$

$$m_4 = \frac{\sigma_{dd}}{\sigma_d} / \frac{\sigma_d}{\sigma_y} \tag{3.3}$$

donde σ_y^2 es la varianza de la señal EEG x, σ_d es la desviación estándar de la primera derivada de x, σ_{dd} es la desviación estándar de la segunda derivada de x. La primera y segunda derivadas se estiman con el método de diferencia finita. Finalmente, a estas señales se les aplicaron diferentes clasificadores tales como MLP, SVM y KNN calculando la precisión, *Recall, F1-score, Accuracy y Kappa* para medir su desempeño.

Matriz de Confusión

	1	2
1	True Positive	False Positive
2	True Positive	False Positive

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3.4}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{3.5}$$

$$F - Score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{3.6}$$

$$Accuracy = \frac{TruePositive + TrueNegative}{TruePositive + TrueNegative + FalsePositive + FalseNegative} \tag{3.7}$$

$$Kappa = \frac{\eta \sum_{i=1}^{c} x_{ii} - \sum_{i=1}^{c} x_{i+} x_{+i}}{\eta^2 - \sum_{i=1}^{c} x_{i+} x_{+i}} \tag{3.8}$$

En dónde:

Accuracy es la medida de rendimiento más intuitiva y es simplemente una relación entre la observación correctamente predicha y las observaciones totales.

Precision es la relación entre las observaciones positivas predichas correctamente y las observaciones positivas predichas totales.

Recall es la relación entre las observaciones positivas correctamente predichas y todas las observaciones en la clase real.

F1 Score es el promedio ponderado de precisión y recuperación.

Kappa refleja la concordancia inter-observador y puede ser calculado en tablas de cualquier dimensión, siempre y cuando se contrasten dos observaciones, para más de 2 observaciones se utiliza el coeficiente Kappa de Fleiss [40].

El coeficiente Kappa utiliza valores entre -1 y 1. Cuando el valor se acerca más a '1' la concordancia es mayor. Cuando el valor es '0' la concordancia que se observa es la que se espera a causa exclusivamente del azar.

En la métrica Kappa, $\sum_{i=1}^{c} x_{ii}$ es la sumatoria de la matriz de confusión, n es el número total de las muestras colectadas, c es el número total de las clases, x_{i+} es la suma de línea i de la matriz de confusión y x_{+i} es la suma de columna i de la matriz de confusión [26].

3.3. Clasificación

Antes de empezar a utilizar los siguientes clasificadores, los conjuntos de datos fueron divididos en 70 % para entrenamiento y 30 % para pruebas, quedando como se muestra en la Figura 3.3.

Figura 3.3: División de los conjuntos de datos: A) *Motor Imagery 4-class*. B) *Motor Imagery 2-class*

Multi-Layer Perceptron

En la prueba de clasificación MLP se probaron diferentes combinaciones de funciones de entrenamiento, activación y número de neuronas que se indican a continuación:

F. Entrenamiento	F. Activación	Neuronas
Levenberg-Marquardt (trainlm)	*Log-sigmoid* (logsig)	25
Bayesian regularization (trainbr)	*Linear transfer* (purelin)	50
Gradient descent with momentum and adaptive learning rate (traingdx)	*Hyperbolic tangent sigmoid* (tansig)	100
		300
		500

Mediante varias combinaciones se obtuvieron los resultados que se pueden observar en la Tabla 4.1, para el conjunto de datos 2A *Motor Imagery 4-class* y la Tabla 4.2 para el conjunto de datos 2B *Motor Imagery 2-class*. Dichos resultados se encuentran representados con valores *Kappa* y *Accuracy*.

Suport Vector Machine

Para la siguiente prueba se aplicó SVM al conjunto de datos en el cual se utilizó la función de matlab 'fitcsvm' (*cross-validates a support vector machine*), el cual hace un mapeo de los datos del predictor utilizando las funciones del *Kernel* para la minimización de la función objetivo, obteniendo los siguientes resultados en Tabla 4.3 para el conjunto de datos 2A *Motor Imagery 4-class* y los resultados del conjunto de datos 2B *Motor Imagery 2-class* se muestran en la Tabla 4.4.

k-Nearest Neighbors

Para realizar esta prueba de KNN también fue utilizada una función de matlab con nombre 'fitcknn' (*Fit k-nearest neighbor classifier*), el cual devuelve un modelo de clasificación de vecinos más cercano, basado en las variables de entrada (también conocidas como predictores, características o atributos) obteniendo los mejores resultados. Véase Tabla 4.5 y Tabla 4.6 para observar los resultados de los conjuntos de datos.

Convolutional Neural Networks

La arquitectura CNN utilizada, véase Tabla 3.1, consta de una capa convolucional, una capa de agrupación (*pooling*) y una capa totalmente conectada (*fully connected*). Se utilizaron 4 filtros en la capa de convolución. La salida de la capa aplica una función de unidad lineal rectificada (ReLU). Y un optimizador 'Adam'. En la Tabla 3.2 se muestran las características de la arquitectura.

Los resultados obtenidos para el conjunto de datos *Motor Imagery 4-class* se encuentran representados en la Tabla 4.12.

Tabla 3.1: Modelo CNN utilizado

Capa (tipo)	Forma de Salida	Parámetros
Capa entrada	(None, 22, 1)	0
Convolución 1D	(None, 10, 4)	16
Pool1 (Maxpooling 1D)	(None, 5, 4)	0
Features (Flatten)	(None, 20)	0
Dense1 (Dense)	(None, 64)	1344
Salida	(None, 4)	260

Tabla 3.2: Características del modelo CNN utilizado

Capa (tipo)	Características
Convolución 1D	activación = ReLU kernel_size = 3 strides = 2 padding = valid
Pool1 (Maxpooling 1D)	pool_size = 1 strides = 2
Dense	activación = ReLU
Salida	Out_dims = 4 activación = Softmax

Primeramente, se agregó la capa convolucional 1D para procesar la señal. Para especificar la longitud de la ventada de convolución 1D *kernel* es de 3, *strides = 2* y la función de activación es una unidad lineal rectificada (ReLU). A continuación, se agregó una capa *maxpooling* 1D. Una vez construidas las capas convolucionales hay que aplanar la salida de éstas para ingresar a las capas totalmente conectadas (*Fully Connected*). Ya desarrollada la arquitectura, se especifica la función de pérdida (*categorical_crossentropy*), el optimizador (*Adam*) y finalmente, la métrica (*Accuracy*) para calcular la evaluación en el modelo.

Long Short Term Memory

La arquitectura utilizada en estas pruebas, véase Figura 3.4, fue secuencial de 3 capas de LSTM véase Tabla 3.3 , en las cuales se utilizaron 64 nodos ocultos, al igual que un tamaño de lote de 64, en una dimensión. Se empleó una función de activación 'Softmax' con un optimizador 'adam' y mil epocas. Los resultados obtenidos se pueden apreciar en la Tabla 4.11.

Tabla 3.3: Modelo LSTM utilizado

Capa (tipo)	Forma de Salida	# Parámetros
LSTM-1	(64, 22, 64)	16896
LSTM-2	(64, 22, 64)	33024
LSTM-3	(64, 64)	33024
Dense-1	(64, 4)	260

En la primera capa de la red se especifica la longitud de la entrada. Para la siguiente se proporciona el número de nodos en las capas ocultas dentro de la celda LSTM. La forma de salida de cada capa LSTM es (batch_size, num_steps, hidden_size). La activación de estas capas está configurada para ser softmax en la capa final de nuestro modelo LSTM. Una vez realizado el modelo, se compiló utilizando la perdida para el entrenamiento *categorical_crossentropy*, que se aplica a la entropía cruzada en los casos en que hay muchas clases o categorías de las cuales solo una es verdadera. Aquí mismo, se utilizó el optimizador 'Adam'. Y finalmente se especificó la métrica *Accuracy*, que nos permite ver cómo mejora la precisión durante el entrenamiento. Todo esto, fue realizado con cada uno de los 9 sujetos por separado. El modelo se ejecutó con mil épocas en un portátil Asus GL552VX utilizando solo el CPU y teniendo una duración aproximada de 3 días por sujeto.

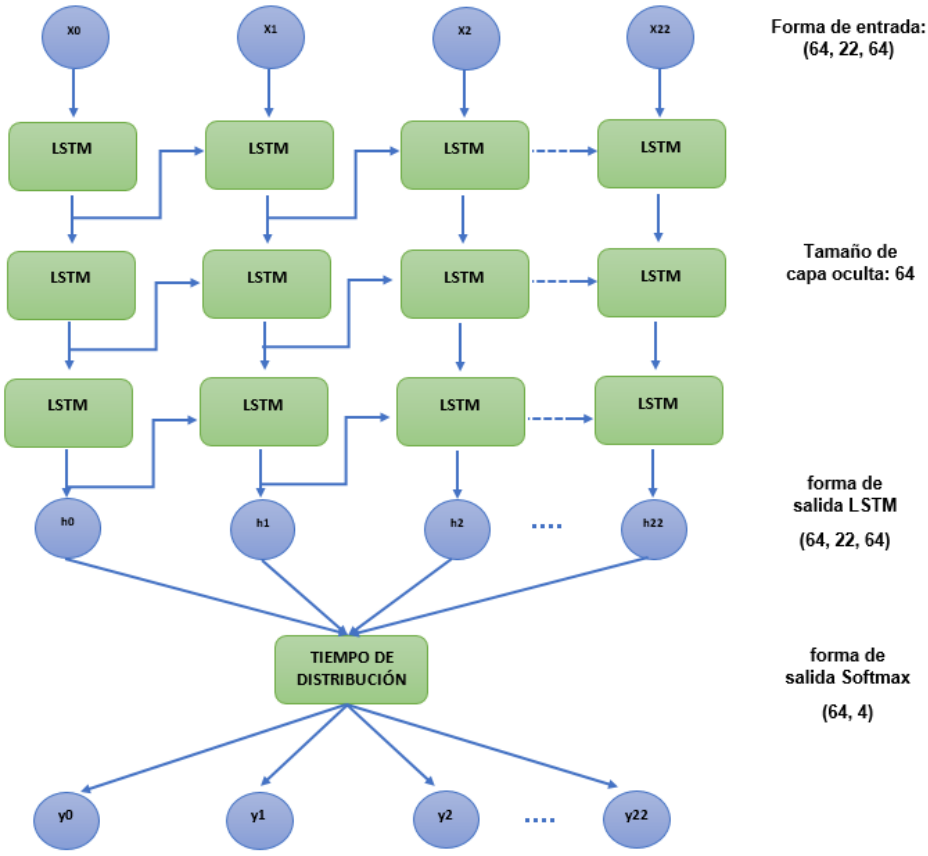


Figura 3.4: Arquitectura LSTM

Autoencoders

En esta arquitectura se utilizaron 2 capas de encoder y una capa softmax como se describe en la Tabla 3.4 y se puede observar en la Figura 3.5.

Tabla 3.4: Arquitectura autoencoer

	Primer encoder:	
Capa oculta	Epocas	Regularizador
25	300 500	peso L2 = 0.001 dispersado = 4
	Segundo encoder:	
50	300 500	peso L2 = 0.002 dispersado = 4

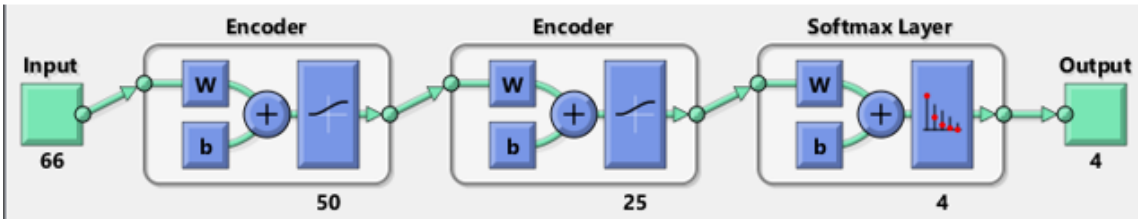


Figura 3.5: Autoencoder utilizado para las pruebas

4

Resultados

En esta sección se muestran los resultados obtenidos de los diferentes clasificadores utilizados después de aplicar los distintos métodos de extracción de características.

4.1. Aprendizaje Máquina

Los resultados de los clasificadores MLP, SVM y KNN se representa con el coeficiente *kappa* con la finalidad de poder compararse con los resultados que se encuentran en la literatura. Los resultados se muestran en las siguientes tablas para cada clasificador. En la Tabla 4.7 y la Tabla 4.8 se puede hacer la comparación entre los resultados obtenidos y los publicados en la literatura para el conjunto de datos 2A-*Motor Imagery 4-class*. Y en la Tabla 4.9 y la Tabla 4.10 se puede visualizar la comparación entre los datos obtenidos y los publicados en la literatura del conjunto de datos 2B-*Motor Imagery 2-class* Como se mencionó en secciones anteriores, el coeficiente *Kappa* utiliza valores entre -1 y 1. Cuando el valor se acerca más a '1' la concordancia es mayor. Cuando el valor es '0' la concordancia que se observa es la que se espera a causa exclusivamente del azar.

Como se puede observar en la Tabla 4.1, los mejores resultados que se pudieron obtener para la red MLP fue en su mayoría utilizando la combinación de la función de entrenamiento *Levenber-Marquardt* (trainlm), función de activación *Hyperbolic tangent sigmoid* (tansig) y *Log-sigmoid*(logsig), con 100 Neuronas y 100 Épocas.

Al igual que en el conjunto de datos *Motor Imagery 4-class* los mejores resultados que se obtuvieron para el conjunto de datos *Motor Imagery 2-class* se muestran en la Tabla 4.2 y son con las

combinaciones de la función de entrenamiento *Levenber-Marquardt* (trainlm), función de activación *Hyperbolic tangent sigmoid* (tansig) y *Log-sigmoid*(logsig), con 100 Neuronas y 100 Épocas.

En las tablas 4.3 y 4.4 observamos los resultados que obtuvieron el mejor desempeño para los conjuntos de datos mencionados. El clasificador utilizado en estas pruebas es el *Support Vector Machine (SVM)* en el cual los resultados son representados con la métrica *Kappa* para cada uno de los 9 sujetos. Este método no muestra resultados favorables haciendo comparación con los resultados de la literatura.

Las pruebas realizadas con KNN muestra los resultados con mejor desempeño representado con la métrica *Kappa* en las tablas 4.5 y 4.6 para cada uno de los sujetos de los conjuntos de datos correspondientes. También se muestran la cantidad de vecinos y distancia utilizada para llegar al ya mencionado desempeño.

Tabla 4.1: Resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A-*Motor Imagery 4-class*, utilizando el filtro Butterworth y la red MLP

Sujeto	KAPPA	F. ENTRENAMIENTO	F. ACTIVACION	NEURONAS	EPOCAS
A01	0.1250	Trainlm	Logsig/Tansig	100	100
A02	0.2917	Trainlm	Purelin/Tansig	25	500
A03	0.6250	Trainlm	Logsig/Tansig	25	100
A04	-0.0046	Trainlm	Logsig/Tansig	100	50
A05	0.0787	Trainlm	Logsig/Tansig	500	300
A06	0.3704	Trainlm	Logsig/Tansig	100	300
A07	0.4583	Trainlm	Logsig/Lansig	100	100
A09	0.2130	Trainlm	Logsig/Tansig	100	100

Tabla 4.2: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2B-*Motor Imagery 2-class*, utilizando el filtro Butterworth y la red MLP

Sujeto	KAPPA	F. ENTRENAMIENTO	F. ACTIVACION	NEURONAS	EPOCAS
B01	0.3083	Trainlm	Logsig/Tansig	100	100
B02	0.1208	Trainlm	Purelin/Tansig	25	500
B03	0.2125	Trainlm	Logsig/Tansig	25	100
B04	0.1833	Trainlm	Logsig/Tansig	100	50
B05	0.2130	Trainlm	Logsig/Tansig	500	300
B07	0.1670	Trainlm	Logsig/Tansig	100	100
B08	0.2830	Trainlm	Logsig/Tansig	500	100
B09	0.2710	Trainlm	Logsig/Tansig	100	100

Tabla 4.3: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A *Motor Imagery 4-class*, utilizando el filtro Butterworth y SVM.

Sujeto	Kappa
A01	0.3056
A02	0.1435
A03	0.3981
A04	0.2315
A05	0.0436
A06	0.1019
A07	0.5417
A08	0.4167
A09	0.3935

Tabla 4.4: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2B *Motor Imagery 2-class*, utilizando el filtro Butterworth y SVM.

Sujeto	Kappa
B01	-0.0250
B02	-0.0429
B03	0.1062
B04	0.0000
B05	0.0125
B06	-0.0813
B07	0.0750
B08	-0.1563
B09	0.2437

Tabla 4.5: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A- *Motor Imagery 4-class*, utilizando el filtro Butterworth y KNN

Sujeto	KAPPA	NEIGHBORS	DISTANCIA
A01	0.1574	144	Mahalanobis
A02	0.1065	7	Correlation
A03	0.2222	27	Cosine
A04	0.1342	77	Mahalanobis
A05	0.0278	50	Mahalanobis
A06	0.1410	29	Cityblock
A07	0.1296	6	Spearman
A08	0.1898	22	Spearman
A09	0.3704	42	Seucledian

Tabla 4.6: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2B, utilizando el filtro Butterworth y KNN

Sujeto	KAPPA	NEIGHBORS	DISTANCIA
B01	0.1875	237	Correlation
B02	-0.0071	7	Correlation
B03	0.0313	1	Chebychev
B04	0.0813	4	Mahalanobis
B05	-0.0500	27	Mahalanobis
B06	-0.0813	209	Spearman
B07	0.0500	1	Spearman
B08	0.0750	1	Cosine
B09	0.0813	3	Cityblock

Tabla 4.7: Resumen de resultados de aprendizaje máquina del conjuntos de datos 2A-*Motor Imagery 4-class*

Sujeto	MLP-MI	SVM-MI	KNN-MI
A01	0.1250	0.3056	0.1574
A02	0.2917	0.1435	0.1065
A03	0.6250	0.3981	0.2222
A04	-0.0046	0.2315	0.1342
A05	0.0787	0.0436	0.0278
A06	0.3704	0.1019	0.1410
A07	0.4583	0.5417	0.1296
A08	0.3108	0.4167	0.1898
A09	0.2130	0.3935	0.3704
Promedio	0.2743	0.2865	0.1643

Tabla 4.8: Resultados publicados en la literatura del conjuntos de datos 2A-*Motor Imagery 4-class*

Sujeto	Kai Keng Ang	Liu Guangquan	Wei Song	Damien Coyle	Jin Wu
A01	0.6800	0.6900	0.3800	0.4600	0.4100
A02	0.4200	0.3400	0.1800	0.2500	0.1700
A03	0.7500	0.7100	0.4800	0.6500	0.3900
A04	0.4800	0.4400	0.3300	0.3100	0.2500
A05	0.4000	0.1600	0.0700	0.1200	0.0600
A06	0.2700	0.2100	0.1400	0.0700	0.1600
A07	0.7700	0.6600	0.2900	0.0000	0.3400
A08	0.7500	0.7300	0.4900	0.4600	0.4500
A09	0.6100	0.6900	0.4400	0.4200	0.3700
Promedio	0.5700	0.5200	0.3100	0.3000	0.2900

Tabla 4.9: Resumen de resultados de aprendizaje máquina del conjuntos de datos 2B-*Motor Imagery 2-class*

Sujeto	MLP-MI	SVM-MI	KNN-MI
B01	0.3083	-0.0250	0.1875
B02	0.1208	-0.0429	-0.0071
B03	0.2125	0.1062	0.0313
B04	0.1833	0.0000	0.0813
B05	0.2130	0.0125	-0.0550
B06	0.1985	-0.0813	-0.0813
B07	0.1670	0.0750	0.0550
B08	0.2830	-0.1563	0.0750
B09	0.2710	0.2437	0.0813
Promedio	0.2174	0.0147	0.0409

Tabla 4.10: Resultados publicados en la literatura del conjuntos de datos 2B-*Motor Imagery 2-class*

Sujeto	Zheng Yang	Huang Gan	Damien Coyle	Shaun Lodder	Jaime Fernando	Yang Ping
B01	0.4000	0.4200	0.1900	0.2300	0.2000	0.0200
B02	0.2100	0.2100	0.1200	0.3100	0.1600	0.0900
B03	0.2200	0.1400	0.1200	0.0700	0.1600	0.0700
B04	0.9500	0.9400	0.7700	0.9100	0.7300	0.4300
B05	0.8600	0.7100	0.5700	0.2400	0.2100	0.2500
B06	0.6100	0.6200	0.4900	0.4200	0.1900	0.0000
B07	0.5600	0.6100	0.3800	0.4100	0.3900	0.1400
B08	0.8500	0.8400	0.8500	0.7400	0.8600	0.7600
B09	0.7400	0.7800	0.6100	0.5300	0.4400	0.4700
Promedio	0.6000	0.5800	0.4600	0.4300	0.3700	0.2500

Los mejores resultados obtenidos en estas pruebas de aprendizaje máquina son mediante el clasificador SVM, pero aun quedando por debajo de los registrados en la literatura. Esto posiblemente se debió a la complejidad de los conjuntos de datos y el uso de diferentes técnicas de preprocesamiento que manejaron los diferentes autores con los cuales se compararon.

A diferencia del método empleado para el preprocesamiento en estas pruebas, los autores de la literatura utilizaron diferentes filtros pasabandas entre 8 y 30 HZ, y bancos de filtros como el *Common Spatial Partner* (CSP). Como clasificadores utilizaron Naive Bayes, Bayesian, Chebyshev, LDA y SVM. Tanto para el conjunto de datos *Motor Imagery 4-class* como para el conjunto de datos *Motor Imagery 2-class*.

El algoritmo y el cálculo para estas pruebas se realizó utilizando Matlab R2018a con Windows 10 pro en un equipo ASUS GL552VX. Como se mencionó en capítulos anteriores, los conjuntos de datos se encuentran en formato GDF, por lo que se utilizó la herramienta 'BIOSIG' para Matlab.

4.2. Aprendizaje Profundo

Los resultados de los modelos basados en aprendizaje profundo que se proponen en este trabajo: CNN, Autoencoder y LSTM, se reportan con la métrica *Accuracy* con la finalidad de poder compararse con los resultados que se encuentran en la literatura basados también en modelos de deep learning. Dichos resultados se muestran en las siguientes tablas.

En la Tabla 4.11 se muestran los mejores resultados obtenidos utilizando la arquitectura propuesta en este trabajo de tesis con el modelo de LSTM. Estos resultados quedan solamente con un 8 % por debajo de los publicados por Ping Wang y Aimin Jiang en [1].

En la Tabla 4.12 se muestran los resultados obtenidos utilizando el modelo propuesto de CNN con el conjunto de datos *Motor Imagery 4-Class* y se hace comparación con los resultados publicados por Huijuan Yang [41], Siavash Sakhavi [42] y Hyeon Kyu Lee [43].

En la Tabla 4.13 y 4.14 se muestran los mejores resultados obtenidos utilizando diferentes combinaciones de Epocas con los modelos de Autoencoders.

Tabla 4.11: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A-*Motor Imagery 4-class*, utilizando LSTM y comparandolos con Ping Wang [1].

Sujeto	ACC LSTM-MI	ACC Ping Wang
A01	72.85 %	55.90 %
A02	58.50 %	63.20 %
A03	52.78 %	64.50 %
A04	58.43 %	66.00 %
A05	62.84 %	79.90 %
A06	61.76 %	63.50 %
A07	61.86 %	84.30 %
A08	64.47 %	78.90 %
A09	62.54 %	71.00 %
Promedio	61.78 %	69.69 %

Tabla 4.12: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A-*Motor Imagery 4-class*, utilizando CNN.

Sujeto	CNN-MI	Hyeon Kyu Lee	Siavash Sakhavi	Huijuan Yang
A01	78.60 %	83.50 %	78.82 %	79.16 %
A02	75.30 %	59.50 %	53.47 %	52.08 %
A03	75.32 %	66.75 %	82.64 %	83.33 %
A04	75.10 %	95.50 %	60.76 %	62.15 %
A05	75.45 %	80.00 %	59.03 %	54.51 %
A06	75.24 %	75.60 %	43.75 %	39.24 %
A07	75.27 %	77.25 %	82.64 %	83.33 %
A08	76.12 %	85.78 %	83.68 %	82.64 %
A09	75.21 %	86.50 %	81.25 %	66.67 %
Promedio	75.73 %	78.93 %	69.56 %	67.01 %

Tabla 4.13: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2A-*Motor Imagery 4-class*, utilizando Autoencoder.

Sujeto	ACC AutoEnc-MI	Epocas
A01	45.80 %	300
A02	34.00 %	500
A03	52.20 %	300
A04	40.30 %	300
A05	40.30 %	300
A06	44.40 %	300
A07	42.00 %	300
A08	55.40 %	300
A09	46.90 %	300

Tabla 4.14: Mejores resultados obtenidos para cada uno de los sujetos del conjunto de datos 2B, utilizando Autoencoder.

Sujeto	ACC EEG-MI	Epocas
B01	58.30 %	200
B02	53.10 %	35
B03	51.20 %	45
B04	53.30 %	40
B05	51.20 %	45
B06	52.70 %	30
B07	52.90 %	40
B08	52.90 %	40
B09	51.70 %	40

Se puede observar que los mejores resultados que se obtuvieron en las pruebas de aprendizaje profundo son utilizando CNN, quedando un 3 % por debajo de Hyeon Kyu Lee, y arriba de Siavash Sakhavi con un 6 % y un 8 % de Huijuan Yang. El modelo de CNN propuesto en este trabajo obtiene un desempeño constante un poco superior al 75 % con todos los sujetos de prueba. Los otros autores reportan desempeños que van desde el 59 al 95 % o desde 43 al 83 %.

5

Conclusiones

El presente trabajo tuvo como objetivo realizar un análisis de trabajos que utilizan técnicas basadas en aprendizaje máquina y aprendizaje profundo con la finalidad de compararlos e identificar cuáles son los mejores modelos para el análisis de señales EEG, particularmente para tareas de visualizaciones motoras.

Primero se realizó una búsqueda en la literatura de publicaciones recientes que utilizaron particularmente dos bases de datos de señales EEG, BCI Competition IV dataset 2A y 2B. Luego, se implementaron diferentes modelos basados en RNA, SVM y KNN. Los resultados que se obtuvieron muestran que el desempeño con estos algoritmos depende en gran medida de la calidad de los datos de entrada. En este caso en particular, no se lograron mejores resultados que los reportados en la literatura con la métrica kappa.

Posteriormente, se implementaron modelos basados en algoritmos de aprendizaje profundo, particularmente CNN, LSTM y Autoencoders. Los resultados obtenidos con modelos de Redes Convolucionales, lograron en algunos casos superar publicaciones recientes, lo que demuestra que éstos resultan ser una mejor opción para el análisis y clasificación de señales EEG para tareas de visualizaciones motoras.

En trabajos futuros se pretende analizar diferentes conjuntos de datos de señales EEG con modelos basados en aprendizaje profundo y proponer un modelo que independientemente de la señal logre identificar a que clase pertenece, es decir, cuál es la intención del usuario.

Referencias

[1] P. Wang, A. Jiang, X. Liu, J. Shang, and L. Zhang, "Lstm-based eeg classification in motor imagery tasks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. PP, pp. 1–1, 10 2018.

[2] I. N. de Estadística y Geografía, *Estadísticas a propósito del día Internacional de las personas con discapacidad*, pp. 1–17. INEGI, 2015.

[3] J. N. Mak and J. R. Wolpaw, "Clinical applications of brain-computer interfaces: current state and future prospects," *IEEE reviews in biomedical engineering*, vol. 2, p. 187, 2009.

[4] J. Minguez, "Tecnologıa de interfaz cerebro-computador," 2008.

[5] R. A. Ramadan and A. V. Vasilakos, "Brain computer interface: control signals review," *Neurocomputing*, vol. 223, pp. 26–44, 2017.

[6] J. Allen, J. Bruss, and H. Damasio, "Estructura del cerebro humano," *Investigación y ciencia*, vol. 340, 2005.

[7] N. Alonso, L. Fernando, *et al.*, "Clasificación de características de electroencefalogramas en sistemas brain computer interface basados en ritmos sensoriomotores," 2012.

[8] R. M. Rangayyan, "Biomedical signal analysis," *press New York*, 2002.

[9] K. K. Ang, "Bci competition iv dataset 2a, institute for infocomm research, agency for science, technology and research singapore."

[10] L. Guangquan, "Bci competition iv dataset 2a, school of mechanical engineeing, shanghai jiao tong university, china."

[11] W. Song, "Bci competition iv dataset 2a, college of information science and technology, beijing normal university, china and national key laboratory for cognitive neuroscience and learning, beijing normal university, china."

[12] D. Coyle, "Bci competition iv dataset 2a, intelligent systems research centre, school of computing and intelligent systems, faculty of computing and engineering, magee campus, university of ulster, uk."

[13] J. Wu, "Bci competition iv dataset 2a, national key laboratory for cognitive neuroscience and learning, beijing normal university, china and college of information science and technology, beijing normal university, china."

[14] L. Duan, M. Bao, J. Miao, Y. Xu, and J. Chen, "Classification based on multilayer extreme learning machine for motor imagery task from eeg signals," *Procedia Computer Science*, vol. 88, pp. 176–184, 2016.

[15] X. Tang, N. Zhang, J. Zhou, and Q. Liu, "Hidden-layer visible deep stacking network optimized by pso for motor imagery eeg recognition," *Neurocomputing*, vol. 234, pp. 1–10, 2017.

[16] B. Competition, "Bci competition iv - final results."

[17] M.-a. Li, X.-y. Luo, and J.-f. Yang, "Extracting the nonlinear features of motor imagery eeg using parametric t-sne," *Neurocomputing*, vol. 218, pp. 371–381, 2016.

[18] Y. R. Tabar and U. Halici, "A novel deep learning approach for classification of eeg motor imagery signals," *Journal of neural engineering*, vol. 14, no. 1, p. 016003, 2016.

[19] Z. Tang, C. Li, and S. Sun, "Single-trial eeg classification of motor imagery using deep convolutional neural networks," *Optik-International Journal for Light and Electron Optics*, vol. 130, pp. 11–18, 2017.

[20] E. Alpaydin, *Introduction to machine learning*. MIT press, 2009.

[21] M. van Gerven and S. Bohte, *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[22] A. Zell, *Simulation neuronaler netze*, vol. 1. Addison-Wesley Bonn, 1994.

[23] M. Pose, "Introducción a las redes de neuronas artificiales," *Departamento de Tecnologías de la Información y las Comunicaciones. Universidad da Coruña*, 2009.

[24] A. J. Serrano, E. Soria, and J. D. Martin, "Redes neuronales artificiales," *Universidad de Valencia (Escuela Técnica Superior Ingeniería, Departamento de Ingeniería Electrónica): Valencia, España*, 2009.

[25] M. A. Valencia Reyes, "Algoritmo backpropagation para redes neuronales: conceptos y aplicaciones," tech. rep., Instituto Politécnico Nacional. Centro de Investigación en Computación, 2007.

[26] G. Figueiredo and C. Vieira, "Estudo do comportamento dos índices de exatidão global, kappa e tau, comumente usados para avaliar a classificação de imagens do sensoriamento remoto," *Simpósio Brasileiro de Sensoriamento Remoto*, vol. 13, pp. 5755–62, 2007.

[27] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[28] S. Marsland, *Machine learning: an algorithmic perspective*. Chapman and Hall/CRC, 2011.

[29] A. Shmilovici, "Support vector machines," in *Data mining and knowledge discovery handbook*, pp. 231–247, Springer, 2009.

[30] A. S. Nugroho, A. B. Witarto, and D. Handoko, "Support vector machine," *Teori dan Aplikasinya dalam Bioinformatika, Ilmu Komputer. com, Indonesia*, 2003.

[31] O. Kramer, "K-nearest neighbors," in *Dimensionality reduction with unsupervised nearest neighbors*, pp. 13–23, Springer, 2013.

[32] S.-H. Liew, Y. F. Low, K. Lim, Y.-H. Choo, and M. Farghaly, "Performance evaluation of convolutional neural network in classification of eeg signals based on attention task," *ARPN Journal of Engineering and Applied Sciences*, vol. 13, pp. 3400–3404, 05 2018.

[33] D. J. Matich, "Redes neuronales: Conceptos básicos y aplicaciones," *Cátedra de Informática Aplicada a la Ingeniería de Procesos–Orientación I*, 2001.

[34] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning (adaptive computation and machine learning series)," *Adaptive Computation and Machine Learning series*, p. 800, 2016.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[37] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[38] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[39] L. Sun, Z. Feng, B. Chen, and N. Lu, "A contralateral channel guided model for eeg based motor imagery classification," *Biomedical Signal Processing and Control*, vol. 41, pp. 1–9, 2018.

[40] J. Cerda and L. VILLARROEL DEL, "Evaluación de la concordancia inter-observador en investigación pediátrica: Coeficiente de kappa," *Revista chilena de pediatría*, vol. 79, no. 1, pp. 54–58, 2008.

[41] H. K. Lee and Y.-S. Choi, "A convolution neural networks scheme for classification of motor imagery eeg based on wavelet time-frequecy image," in *Information Networking (ICOIN), 2018 International Conference on*, pp. 906–909, IEEE, 2018.

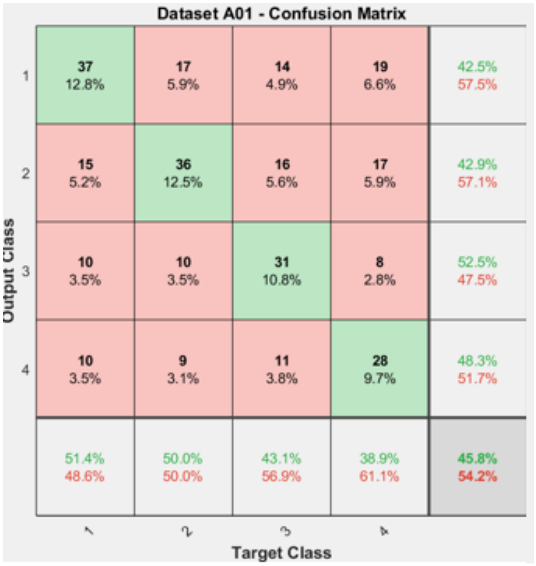
[42] S. Sakhavi, C. Guan, and S. Yan, "Learning temporal information for brain-computer interface using convolutional neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2018.

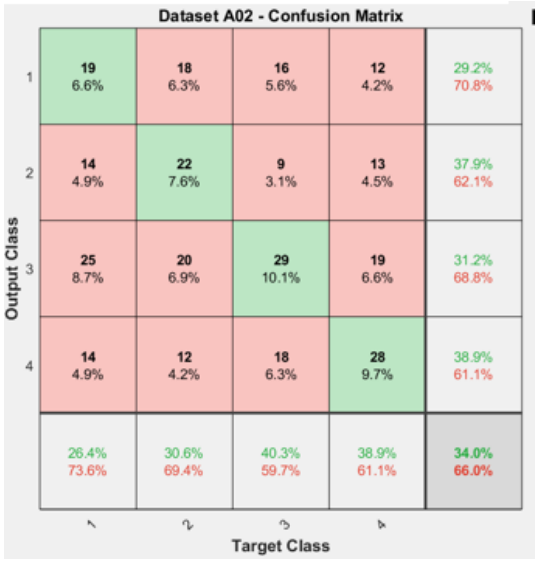
[43] H. Yang, S. Sakhavi, K. K. Ang, and C. Guan, "On the use of convolutional neural networks and augmented csp features for multi-class motor imagery of eeg signals classification," in *Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE*, pp. 2620–2623, IEEE, 2015.

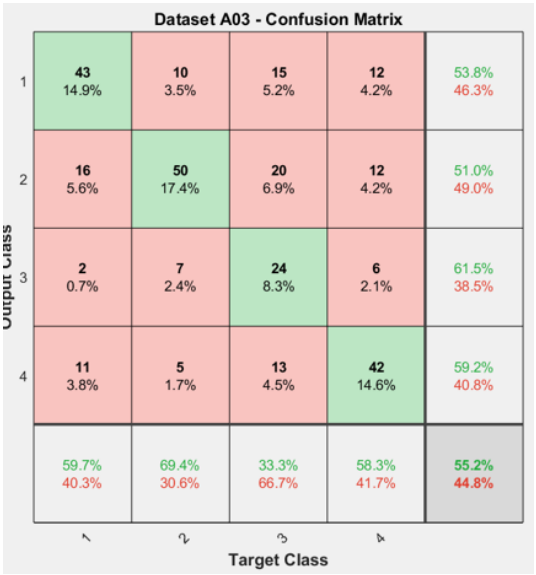
Apéndice A

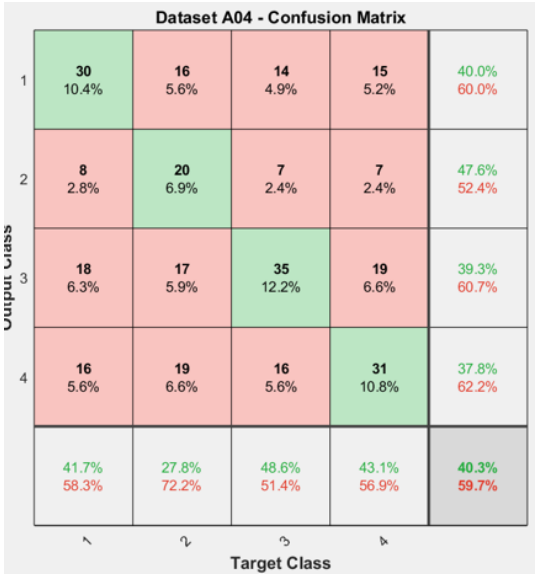
Autoencoders Motor Imagery 4-class

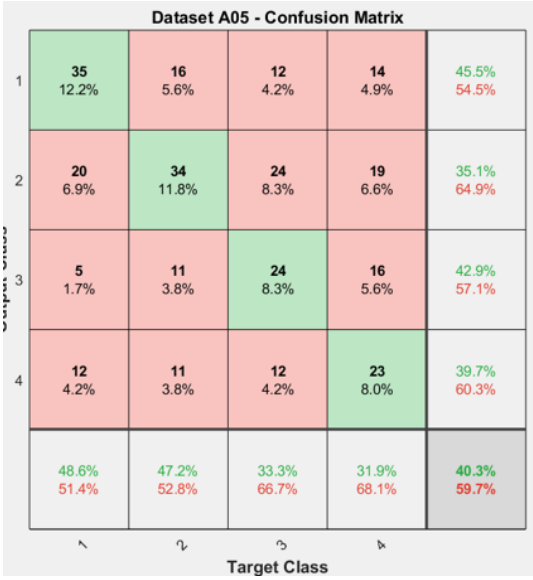
A.1. Matriz de Confusión de cada uno de los sujetos clasificados.

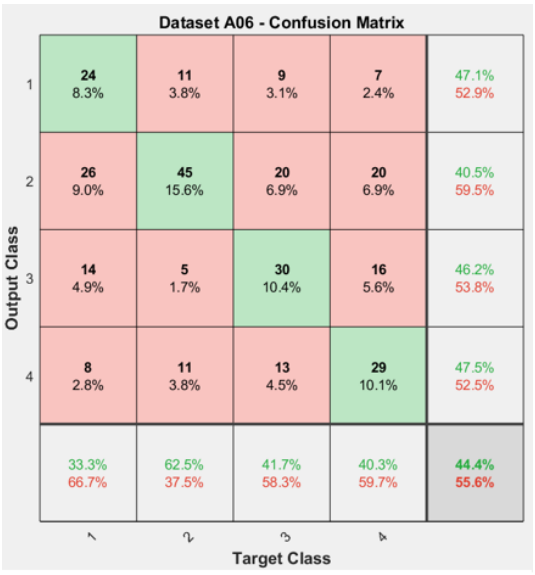


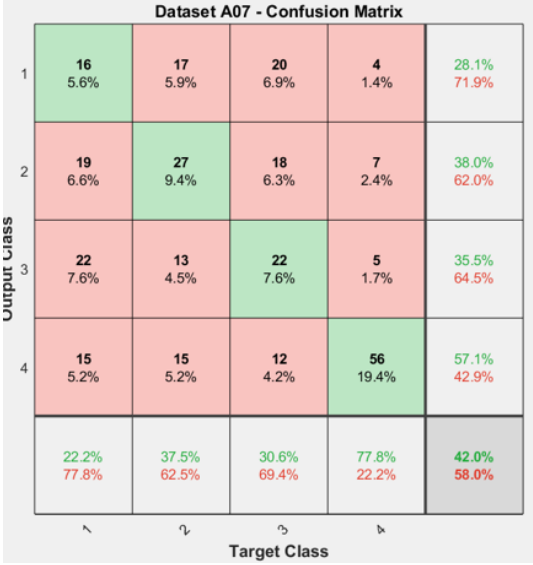


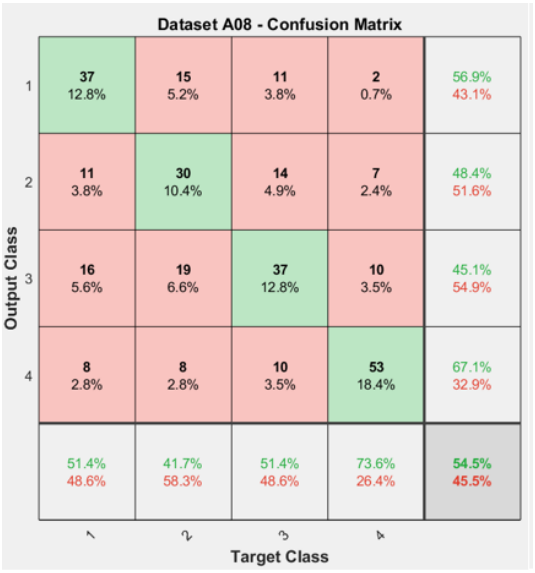


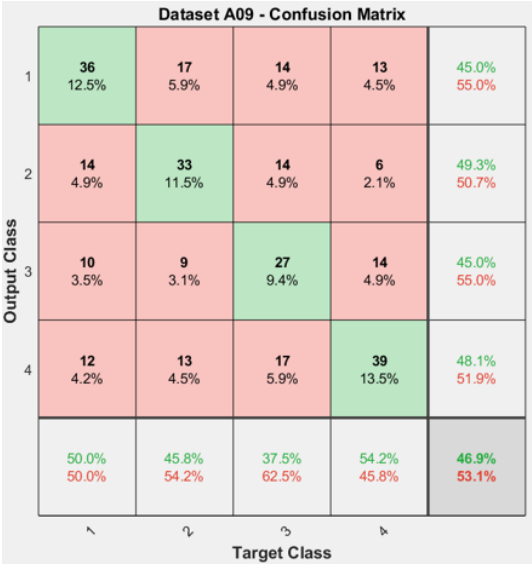



Dataset A05 - Confusion Matrix
35 12.2%
16 5.6%
12 4.2%
14 4.9%
45.5% 54.5%
20 6.9%
34 11.8%
24 8.3%
19 6.6%
35.1% 64.9%
5 1.7%
11 3.8%
24 8.3%
16 5.6%
42.9% 57.1%
12 4.2%
11 3.8%
12 4.2%
23 8.0%
39.7% 60.3%
48.6% 51.4%
47.2% 52.8%
33.3% 66.7%
31.9% 68.1%
40.3% 59.7%
Target Class

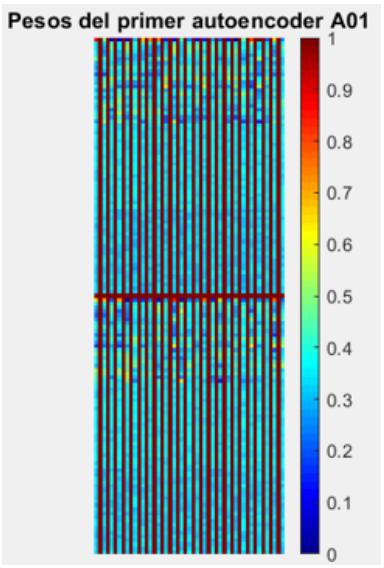

Dataset A06 - Confusion Matrix
Output Class
24 8.3%
11 3.8%
9 3.1%
7 2.4%
47.1% 52.9%
26 9.0%
45 15.6%
20 6.9%
20 6.9%
40.5% 59.5%
14 4.9%
5 1.7%
30 10.4%
16 5.6%
46.2% 53.8%
8 2.8%
11 3.8%
13 4.5%
29 10.1%
47.5% 52.5%
33.3% 66.7%
62.5% 37.5%
41.7% 58.3%
40.3% 59.7%
44.4% 55.6%
Target Class

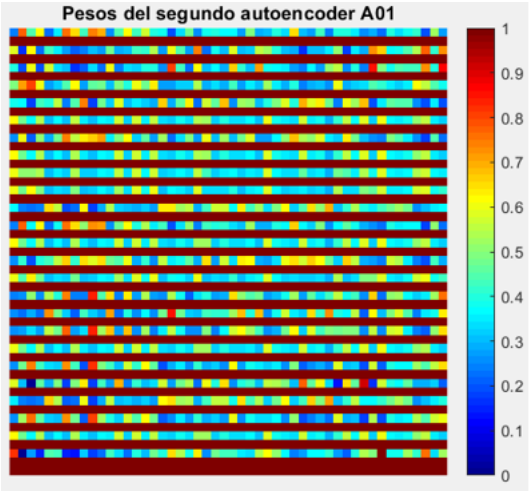

Dataset A07 - Confusion Matrix
Output Class
16 5.6%
17 5.9%
20 6.9%
4 1.4%
28.1% 71.9%
19 6.6%
27 9.4%
18 6.3%
7 2.4%
38.0% 62.0%
22 7.6%
13 4.5%
22 7.6%
5 1.7%
35.5% 64.5%
15 5.2%
15 5.2%
12 4.2%
56 19.4%
57.1% 42.9%
22.2% 77.8%
37.5% 62.5%
30.6% 69.4%
77.8% 22.2%
42.0% 58.0%
Target Class

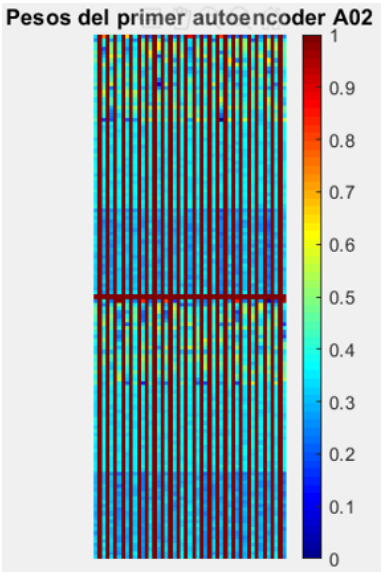

Dataset A08 - Confusion Matrix
Output Class
37 12.8%
15 5.2%
11 3.8%
2 0.7%
56.9% 43.1%
11 3.8%
30 10.4%
14 4.9%
7 2.4%
48.4% 51.6%
16 5.6%
19 6.6%
37 12.8%
10 3.5%
45.1% 54.9%
8 2.8%
8 2.8%
10 3.5%
53 18.4%
67.1% 32.9%
51.4% 48.6%
41.7% 58.3%
51.4% 48.6%
73.6% 26.4%
54.5% 45.5%
Target Class

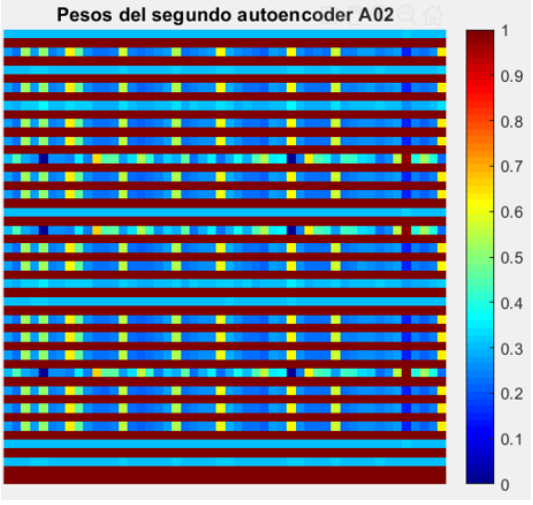

Dataset A09 - Confusion Matrix
Output Class
36 12.5%
17 5.9%
14 4.9%
13 4.5%
45.0% 55.0%
14 4.9%
33 11.5%
14 4.9%
6 2.1%
49.3% 50.7%
10 3.5%
9 3.1%
27 9.4%
14 4.9%
45.0% 55.0%
12 4.2%
13 4.5%
17 5.9%
39 13.5%
48.1% 51.9%
50.0% 50.0%
45.8% 54.2%
37.5% 62.5%
54.2% 45.8%
46.9% 53.1%
Target Class

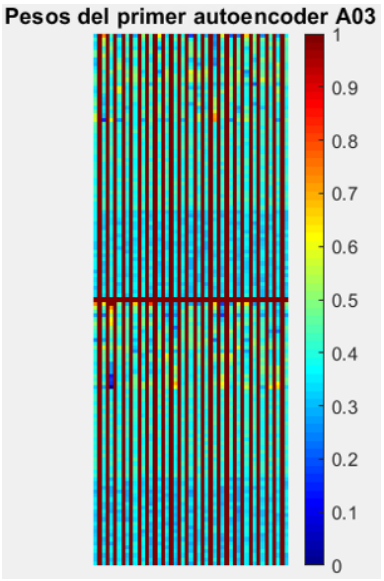
A.2. Pesos del primer y segundo autoencoder de cada uno de los sujetos clasificados

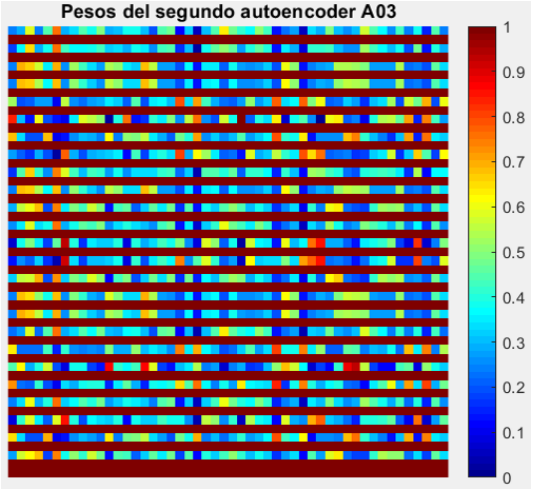


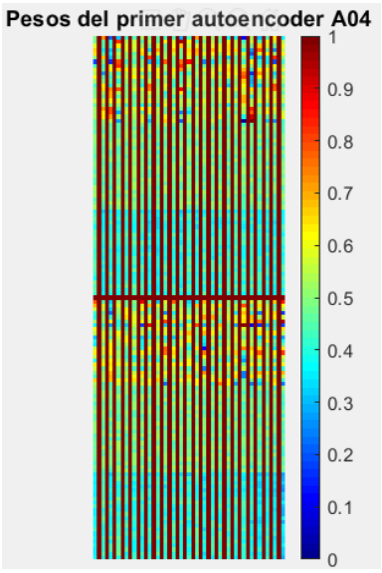


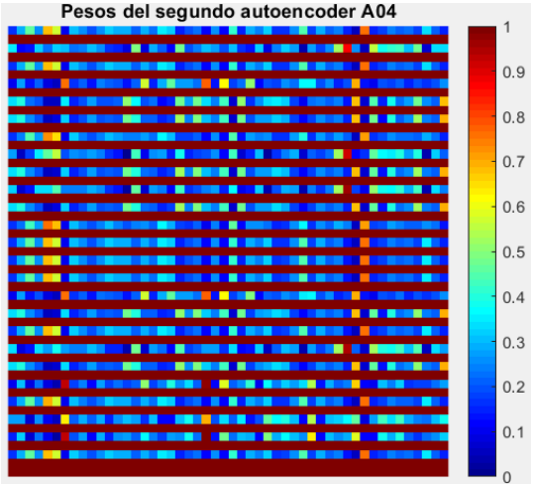


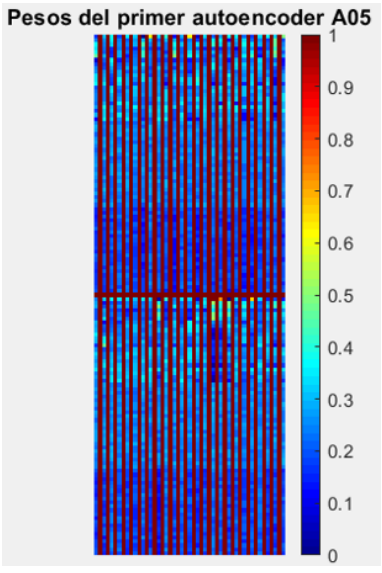


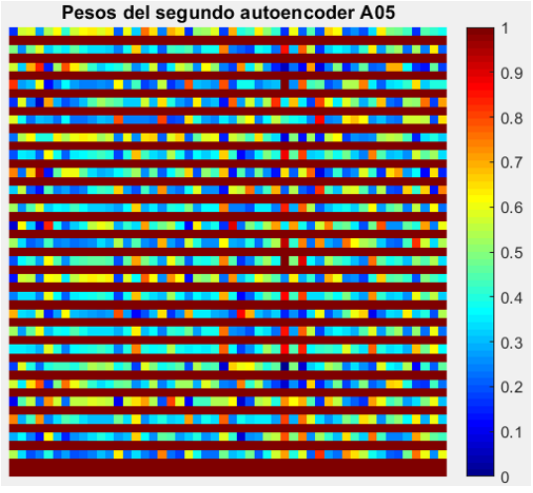


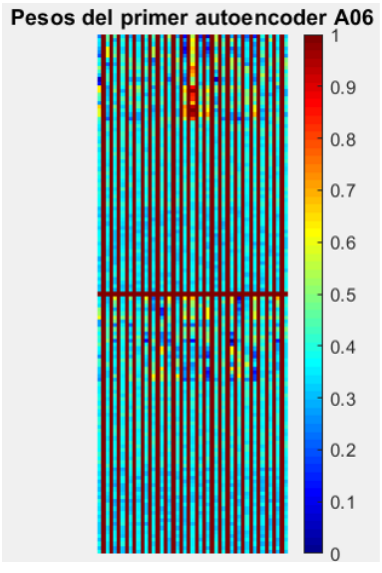


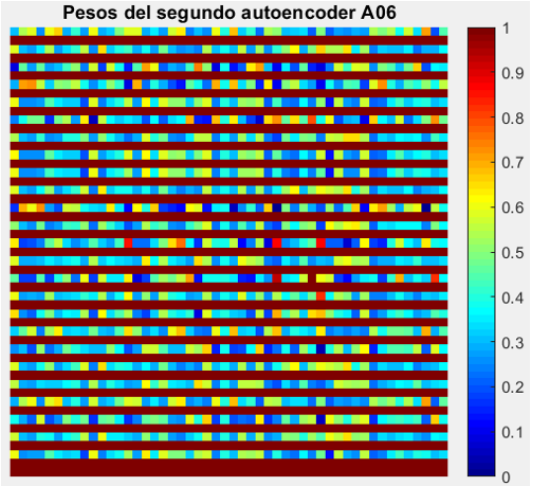

Pesos del primer autoencoder A04
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

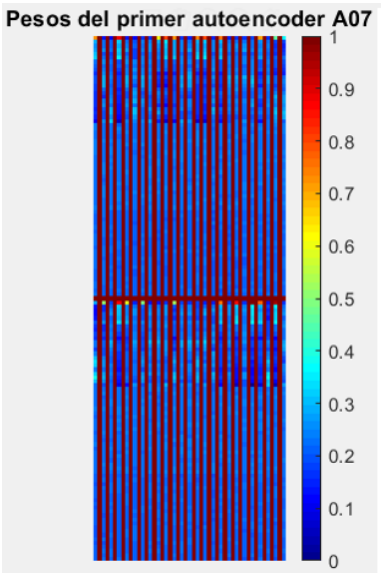

Pesos del segundo autoencoder A04
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

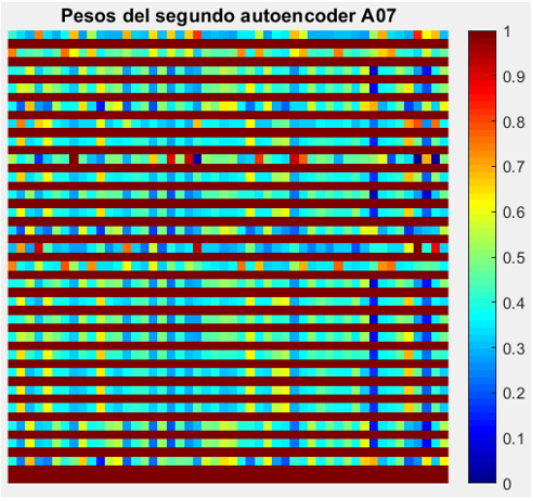

Pesos del primer autoencoder A05
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

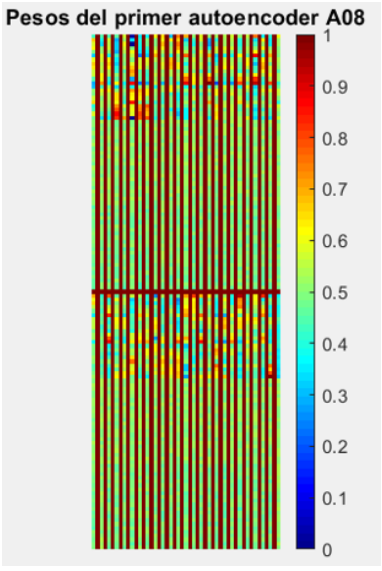

Pesos del segundo autoencoder A05
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

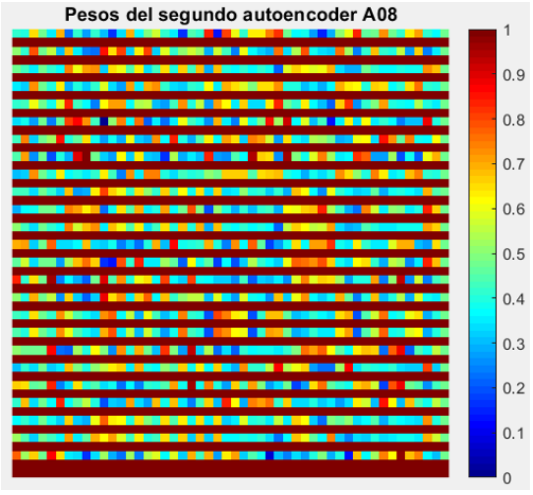

Pesos del primer autoencoder A06
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

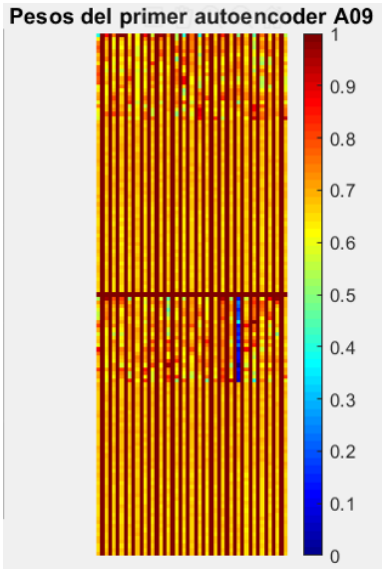

Pesos del segundo autoencoder A06
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

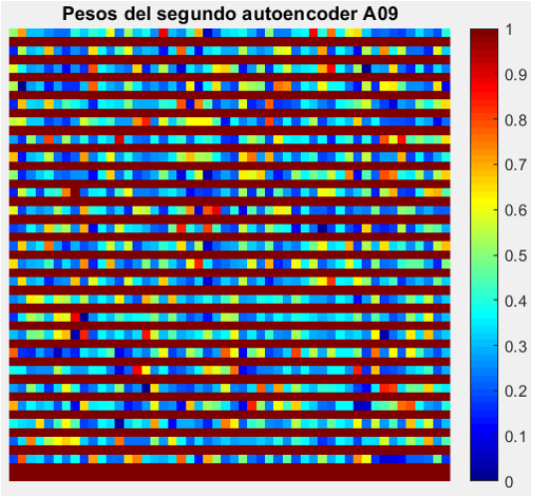

Pesos del primer autoencoder A07
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del segundo autoencoder A07
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del primer autoencoder A08
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del segundo autoencoder A08
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

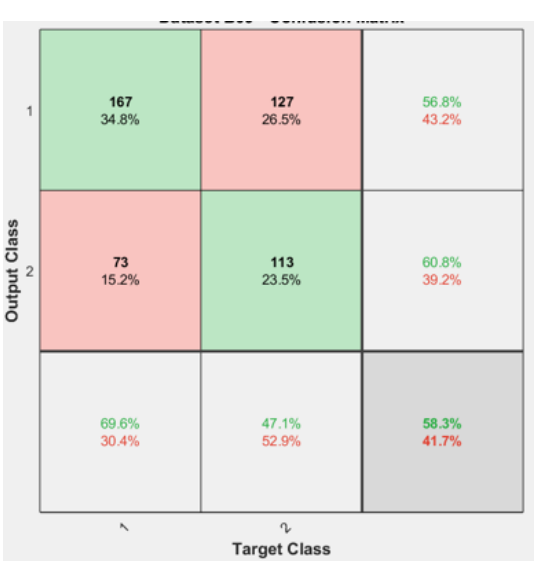

Pesos del primer autoencoder A09
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

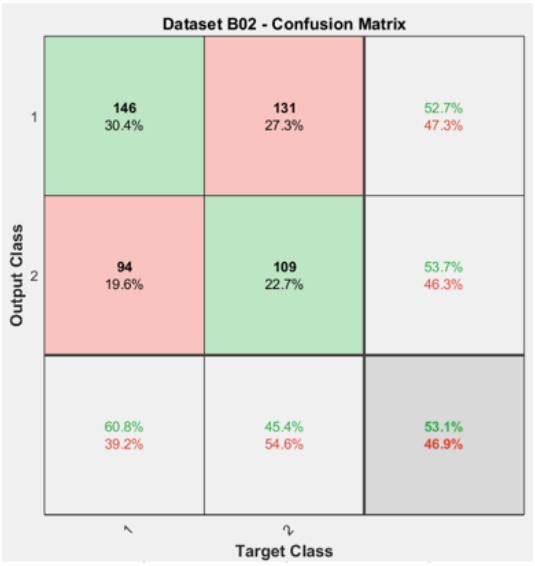

Pesos del segundo autoencoder A09
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

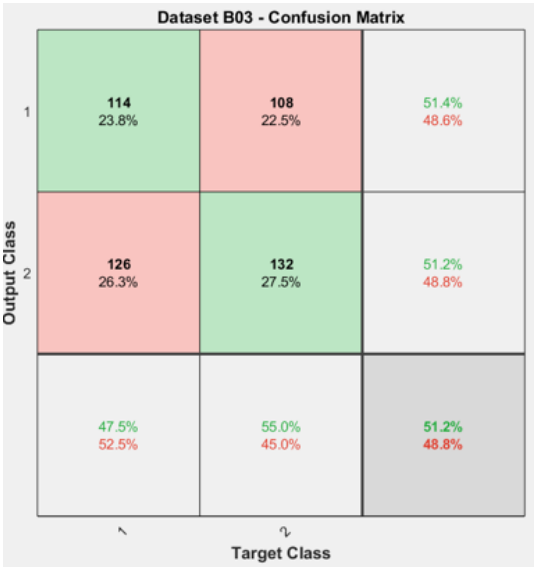
Apéndice B

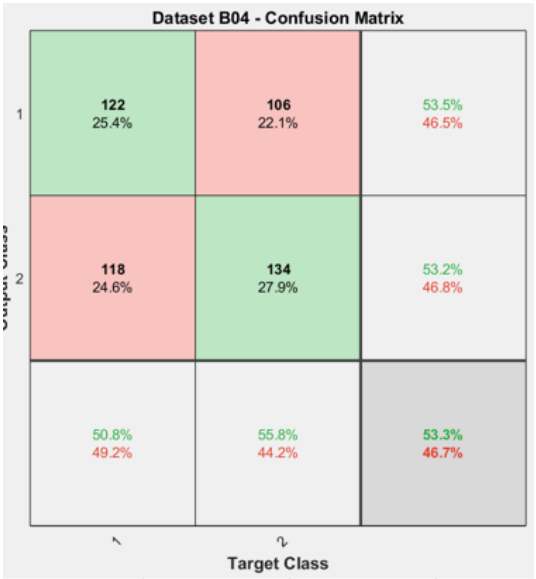
Autoencoders Motor Imagery 2-class

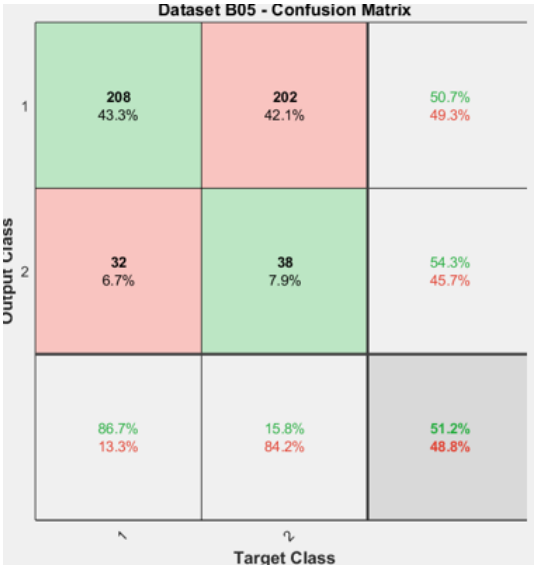
B.1. Matriz de Confusión de cada uno de los sujetos clasificados.

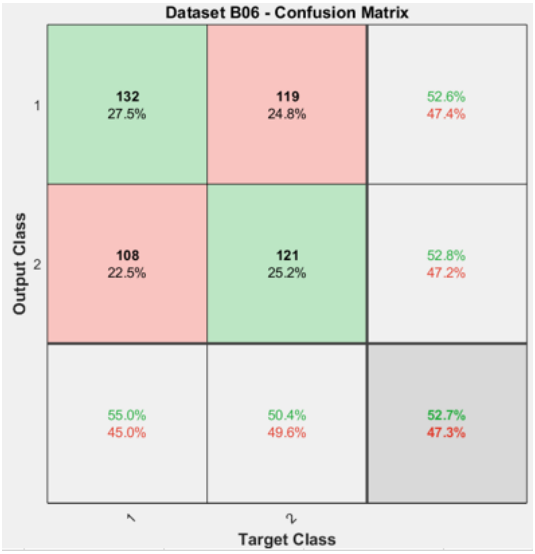


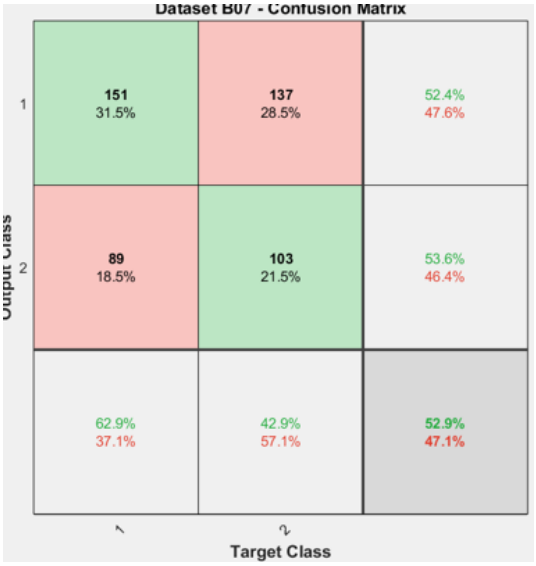


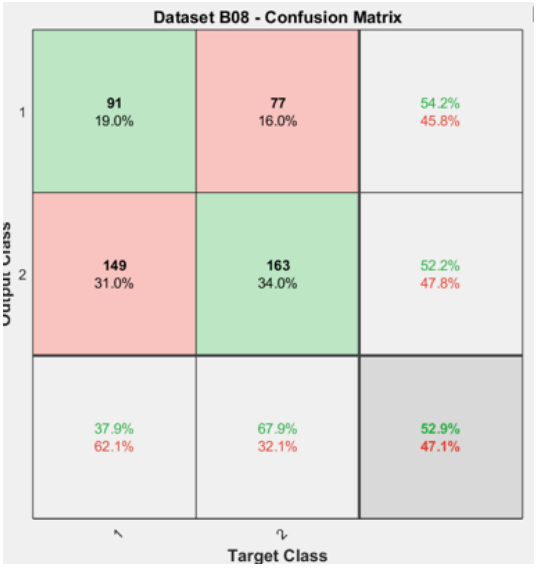


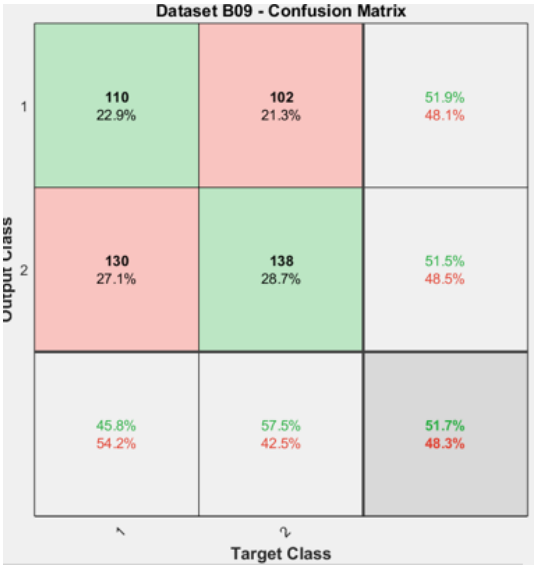



Dataset B05 - Confusion Matrix
208
43.3%
202
42.1%
50.7%
49.3%
32
6.7%
38
7.9%
54.3%
45.7%
86.7%
13.3%
15.8%
84.2%
51.2%
48.8%
Output Class
Target Class

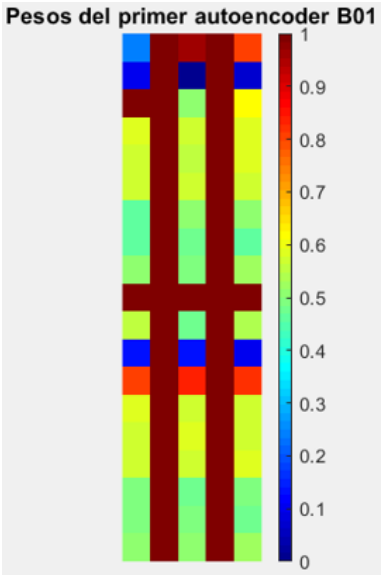

Dataset B06 - Confusion Matrix
132
27.5%
119
24.8%
52.6%
47.4%
108
22.5%
121
25.2%
52.8%
47.2%
55.0%
45.0%
50.4%
49.6%
52.7%
47.3%
Output Class
Target Class

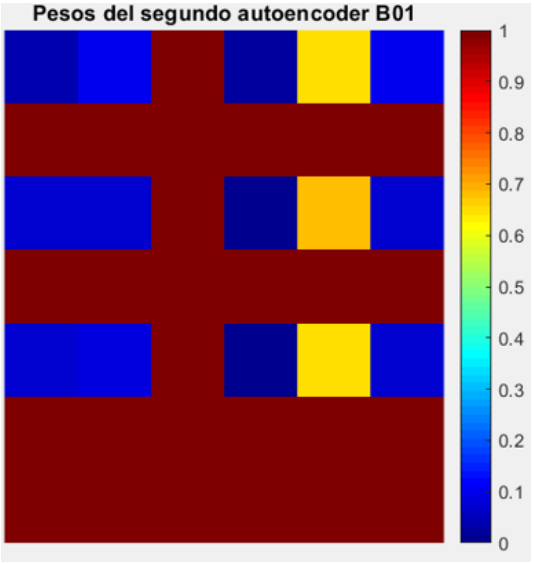

Dataset B07 - Confusion Matrix
151
31.5%
137
28.5%
52.4%
47.6%
89
18.5%
103
21.5%
53.6%
46.4%
62.9%
37.1%
42.9%
57.1%
52.9%
47.1%
Output Class
Target Class

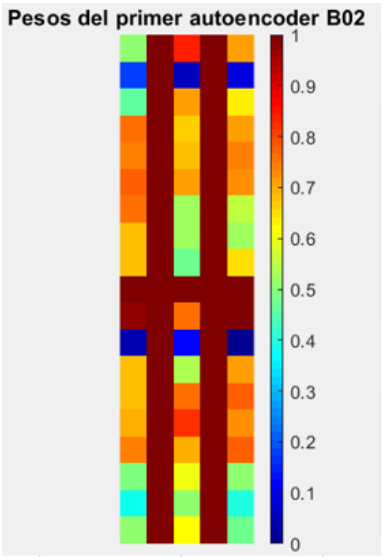

Dataset B08 - Confusion Matrix
91
19.0%
77
16.0%
54.2%
45.8%
149
31.0%
163
34.0%
52.2%
47.8%
37.9%
62.1%
67.9%
32.1%
52.9%
47.1%
Output Class
Target Class

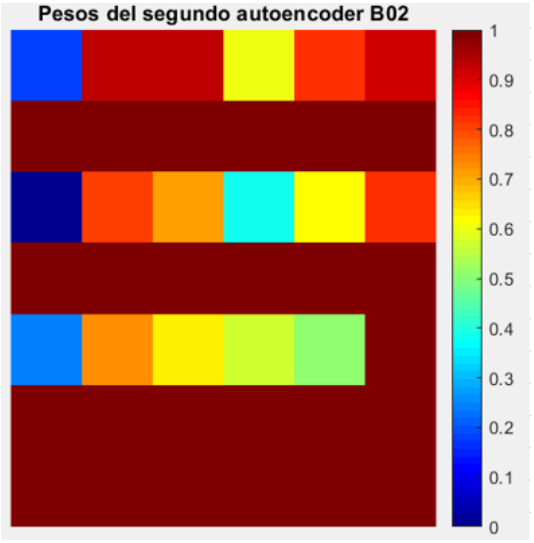

Dataset B09 - Confusion Matrix
110
22.9%
102
21.3%
51.9%
48.1%
130
27.1%
138
28.7%
51.5%
48.5%
45.8%
54.2%
57.5%
42.5%
51.7%
48.3%
Output Class
Target Class

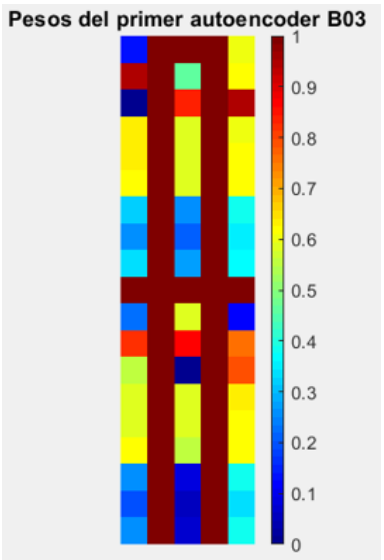
B.2. Pesos del primer y segundo autoencoder de cada uno de los sujetos clasificados

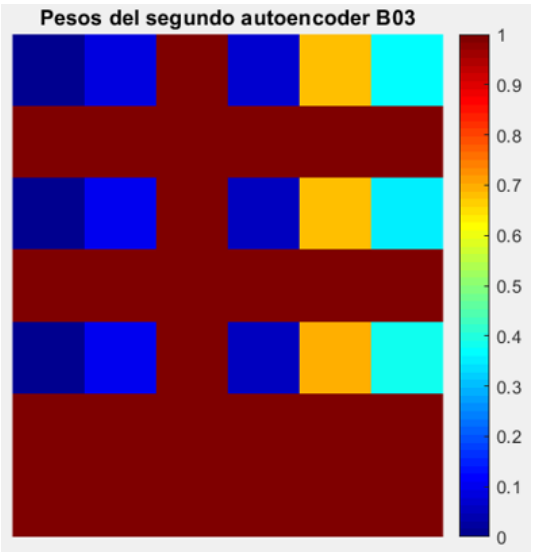


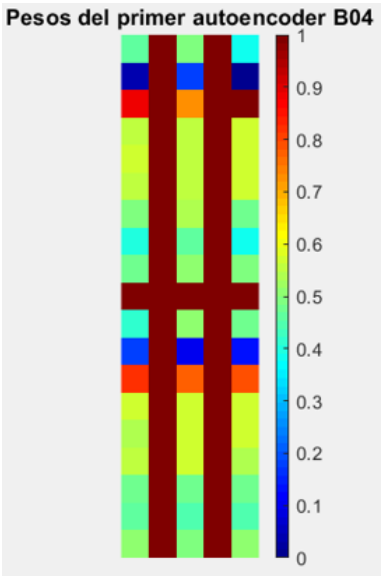


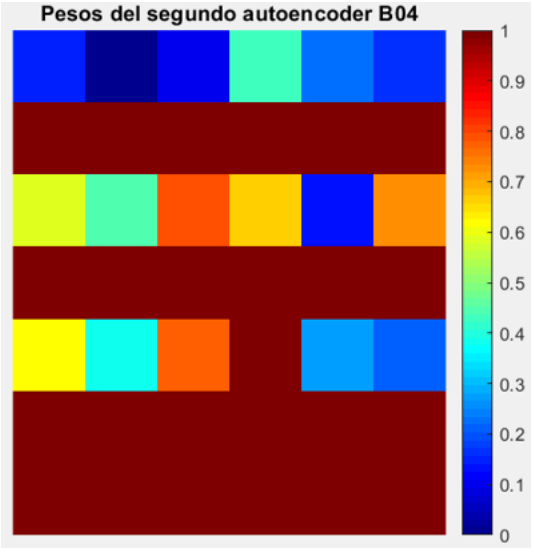


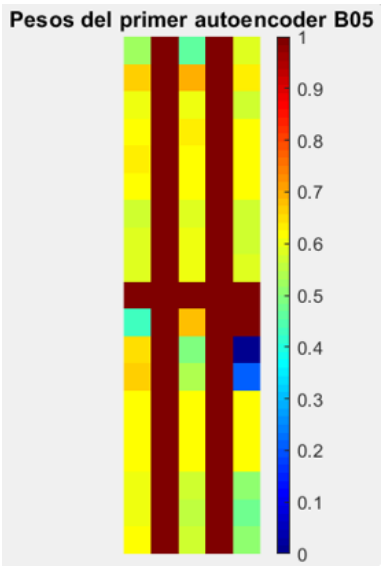


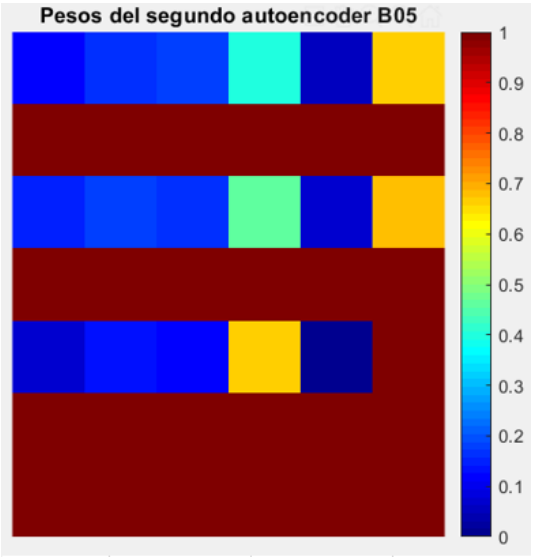


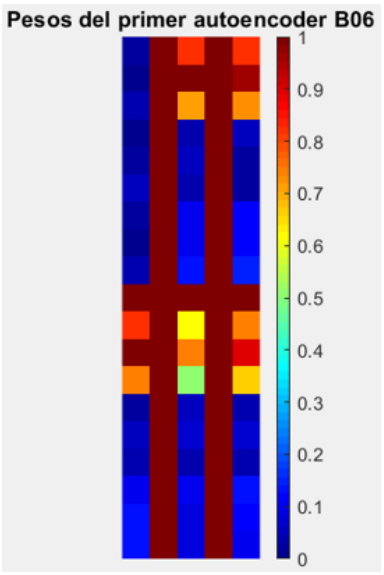


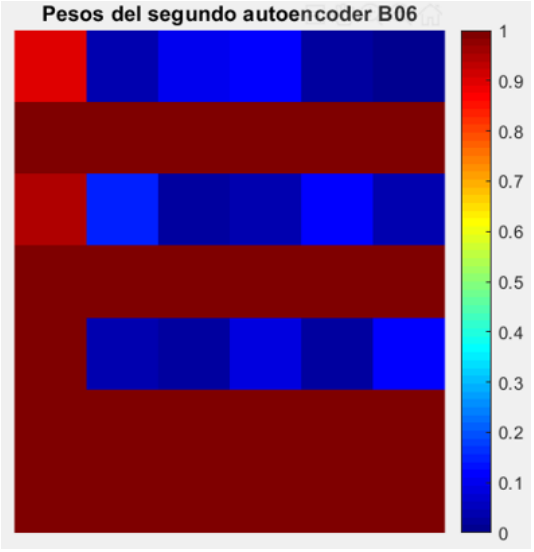

Pesos del primer autoencoder B04
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

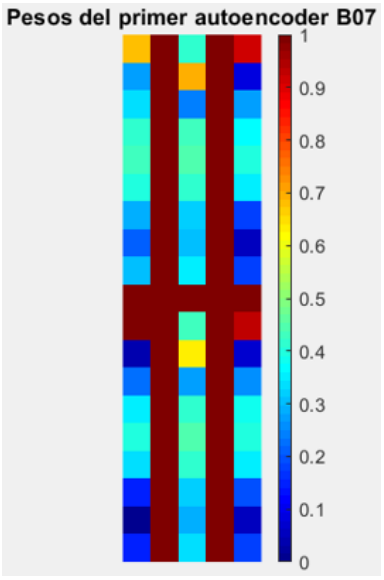

Pesos del segundo autoencoder B04
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

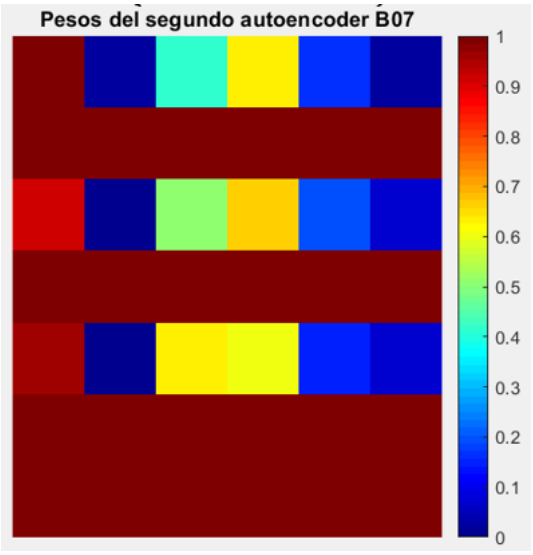

Pesos del primer autoencoder B05
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

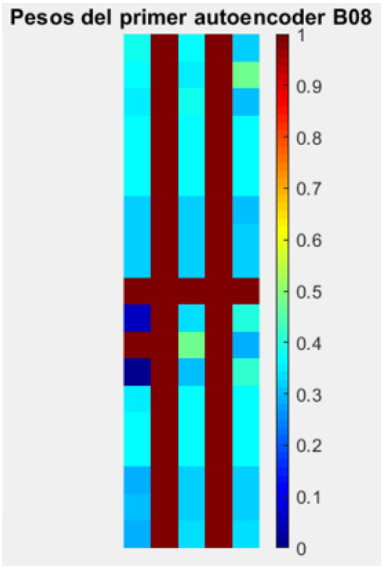

Pesos del segundo autoencoder B05
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

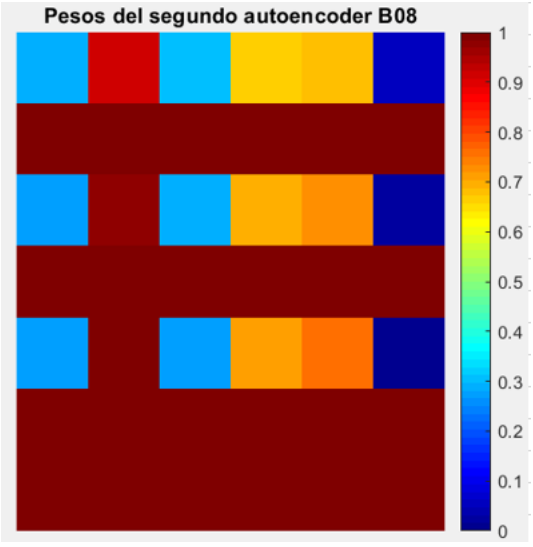

Pesos del primer autoencoder B06
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

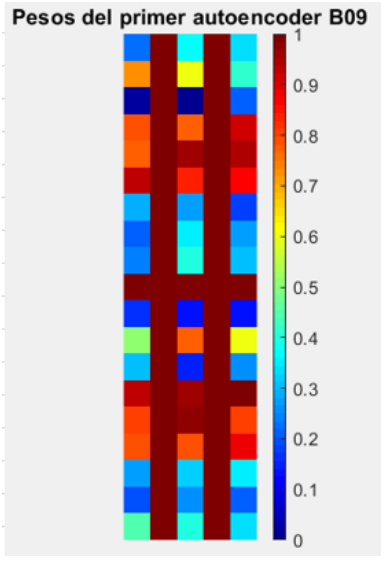

Pesos del segundo autoencoder B06
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0

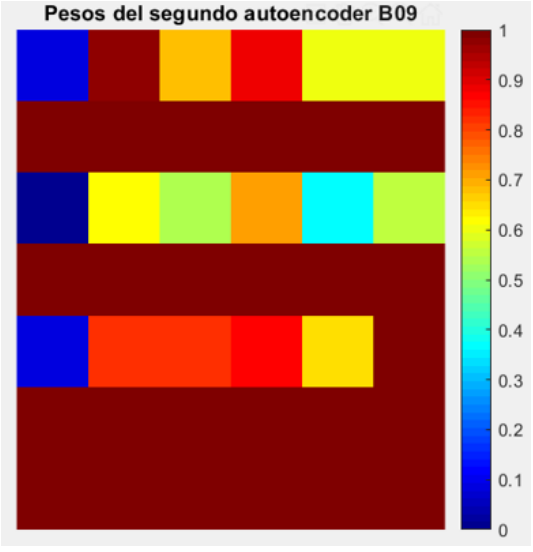

Pesos del primer autoencoder B07
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del segundo autoencoder B07
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del primer autoencoder B08
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del segundo autoencoder B08
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del primer autoencoder B09
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0


Pesos del segundo autoencoder B09
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0