Detección de Anomalías en Ataques Epilépticos con Autoencoders

Enrique Ruiz Amakrache

Resumen

En este trabajo se propone realizar un aprendizaje automático para la detección de anomalías en casos de epilepsia a través del análisis de señales EEG. Se realizará un estudio del impacto/beneficio de incorporar la temporalidad en modelos de redes neuronales LSTM, principalmente Autoencoders, capaces de clasificar anomalías.

Su prevención mejoraría la calidad de vida de los pacientes reduciendo el número y gravedad de las lesiones que la epilepsia ocasiona, con el fin de tomar las suficientes medidas preventivas.

Previamente se han aplicado distintos modelos (Random Forest, SVM, KNN,..) que permiten realizar clasificaciones con precisiones entre el 80 % y 98 %. Los resultados de emplear Autoencoders muestran un 85,6 % de precisión en la clasificación de anomalías en señales EEG .

Palabras clave– *Ataques epilépticos, redes neuronales, predicción, Redes Neuronales Recurrentes(RNN), LSTM, Autoencoders, Series Temporales, anomalías...*

Abstract

In this work, it is proposed to carry out machine learning for the detection of anomalies in cases of epilepsy through the analysis of EEG signals. A study of the impact/benefit of incorporating temporality into LSTM neural network models, mainly Autoencoders, capable of classifying anomalies, will be carried out.

Its prevention would improve the quality of life of patients by reducing the number and severity of injuries caused by epilepsy, in order to take sufficient preventive measures.

Different models have been previously applied (Random Forest, SVM, KNN,...) that allow classifications to be made with precision between 80% and 98%. The results of using Autoencoders show an 85.6% accuracy in the classification of anomalies in EEG signals.

Keywords– *Epileptic seizures, neural networks, prediction, Recurrent Neural Networks (RNN), LSTM, Autoencoders, Time Series, anomalies...*

1 Introducción

La epilepsia es un trastorno del sistema nervioso central (neurológico) en el que la actividad cerebral normal se altera, lo que provoca convulsiones o períodos de comportamiento y sensaciones inusuales, y a veces, pérdida de la consciencia.
Estas convulsiones incontroladas podrían provocar daños cerebrales, lesiones graves, o incluso la muerte de sus pacientes en casos de accidentes de tráfico o circunstancias de riesgo.

• E-mail de contacto: enriamak@gmail.com
• Mención: Computación (2021/22)
• Trabajo tutorizado por: Débora Gil

Las crisis epilépticas generalmente duran apenas unos segundos o unos minutos, después de los cuales finalizan y el cerebro vuelve a funcionar con normalidad. La forma de manifestarse una crisis depende de la parte del cerebro afectada y la causa de la epilepsia.

Existen dos grandes tipos de crisis epiléptica: las crisis focales y las generalizadas.

Las crisis focales son más frecuente, comienzan en una parte delimitada del cerebro y pueden manifestarse de diversas formas:

- Crisis parcial simple: se produce una alteración del movimiento, la memoria y las sensaciones, además de

los sentidos de la vista y el oído. La persona no pierde el conocimiento.

- Crisis parcial compleja: la persona pierde el conocimiento y puede aparentar un estado de trance. Puede darse una repetición compulsiva de ciertos movimientos. Aproximadamente, dos tercios de las personas que padecen epilepsia sufren este tipo de crisis.
- Secundariamente generalizada: comienza como una crisis parcial y se extiende al resto del cerebro convirtiéndose en una crisis generalizada.

Por otro lado las crisis generalizadas empiezan simultáneamente en todo el cerebro y provoca la pérdida de conocimiento. También pueden manifestarse de distintas formas:

- Epilepsia menor o pequeño mal, pueden causar un parpadeo rápido o la mirada fija a lo lejos por unos pocos segundos.
- Epilepsia mayor o gran mal, pueden hacer que la persona grite, pierda el conocimiento, se caiga al suelo, tenga rigidez o espasmos musculares.

2 Estado del arte

2.1 Trabajo Previo

Hasta la actualidad, se han empleado técnicas de clasificación de señales EEG con resultados muy positivos
Existen multitud de técnicas dentro del aprendizaje supervisado, de las que destacamos K-NN, el clasificador del Descenso del Gradiente Estocástico(SGD), máquinas de soporte vectorial(SVM), Random Forest y CNN.

- KNN (K-Nearest Neighbor classifier): El clasificador K vecinos más cercanos es un clasificador no paramétrico que predice las clases desconocidas en función de las clases asociadas con las k (número de vecinos) que están más cerca. Utiliza una regla de decisión de mayoría simple (la media de los k valores).

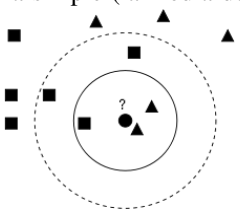

Fig. 1: K-Nearest Neighbor Classifier

- SGD (Stochastic Gradient Descendent): Realiza una Clasificación Lineal (SVM, logistic regression...) utilizando el optimizador del descenso del gradiente estocástico (SGD) para entrenar.
  Es un clasificador muy eficiente con conjuntos de datos grandes, esto se debe a que el algoritmo trata con las instancias de entrenamiento una a una, de manera independiente.
  El modelo aprende el gradiente o la dirección que debe tomar el modelo para reducir los errores (diferencia entre el valor real y el predicho).
- SVM (LinearSVM): Las Máquinas Vector Soporte encuentran la mejor separación posible entre clases. En el caso de las SVM Linear, crea una línea recta que separa de la mejor manera las dos clases.
  En la mayoría de problemas de aprendizaje automático existen múltiples dimensiones, en estos casos, la SVM encuentran el hiperplano que maximiza el margen de separación entre estas clases.

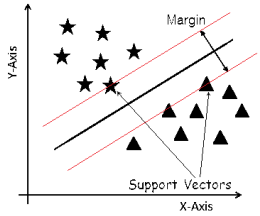

Fig. 2: Support Vector Machines

- Random Forest: Es uno de los algoritmos más potentes que existen actualmente, se basa en Árboles de decisión y métodos de ensamblaje. Es una combinación de árboles predictores en el que cada árbol depende de los valores de un vector aleatorio, probado independientemente y con la misma distribución para cada uno de estos.
  Utilizan el mismo algoritmo de entrenamiento para todos los predictores y los entrenan con subconjuntos distintos aleatorios del conjunto de entrenamiento. Esta técnica se conoce como “bagging”, en lugar de buscar la mejor característica cuando divide un nodo (árboles de decisión) busca la mejor característica de un subconjunto aleatorio de características.

- Convolutional Neural Network (CNN): Es una clase de redes neuronales artificiales (RNA) capaz de establecer relaciones no lineales entre variables de entrada y salida. Cada neurona artificial se somete a una función de activación que es modelada por una combinación lineal de las entradas de dicha neurona y que da como resultado su salida.
  Se basa en el funcionamiento del sistema neuronal humano, un sistema complejo, no lineal y paralelo. Este sistema se conforma de interconexiones entre neuronas formando un punto clave para el procesamiento del conocimiento.
  Tienen aplicaciones en el reconocimiento de imágenes y vídeos , sistemas de recomendación, análisis y clasificación de imágenes, procesamiento de lenguaje natural y series temporales.

2.2 Resultados hasta la actualidad

En 2019, Nandy [5] consiguió mediante un clasificador SVM una precisión del 97,05% en la clasificación de ataques epilépticos.
Un año después, Almustafa [4] utilizó varias técnicas de aprendizaje automático, entre ellas las mencionadas anteriormente, obteniendo la mayor precisión para el Random Forest, con una precisión del 97.08%.
Recientemente, Shekokar [7] propusieron una técnica de clasificación de señales de EEG basado en un modelo LSTM. Este modelo afirma tener una precisión del 98,5 %.

En 2022, Chirasani y Manikandan [6] propusieron integrar un mecanismo de atención jerárquico a un modelo basado en CNN mediante el cual obtuvieron precisiones del 96.34%.
Los resultados de estas técnicas se muestran ordenados en la siguiente tabla:

Modelo	Precisión
LSTM	98,50%
Random Forest	97.08%
SVM	97.05%
CNN	96.34%
KNN	95.23%
SGD	81.92%

2.3 Paradigmas

En este trabajo se pretende mostrar de forma detallada el proceso de clasificación de los casos epilépticos mediante el empleo de redes neuronales recurrentes (RNN), ver Figura 3. Estas redes, caracterizadas por el uso de datos secuenciales, contienen bucles de retroalimentación en las capas recurrentes dotando a la red de una memoria temporal.

Lamentablemente, este tipo de redes no resuelven las dependencias temporales a "largo plazo", como es nuestro caso, debido a que el gradiente de la función de pérdida disminuye exponencialmente con el tiempo (Problema de desvanecimiento de gradiente).

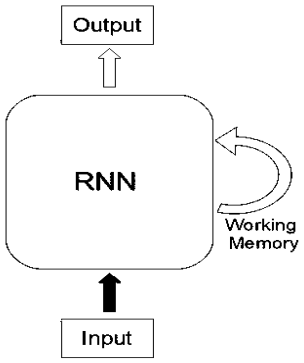

Fig. 3: Red Neuronal Recurrente

Para solventar este problema, existen ciertos posibles paradigmas:

1. **Redes Long short-term memory (LSTM)**, son una extensión de las RNN muy útiles para el aprendizaje profundo sobre datos secuenciales.
   A diferencia de las RNN se caracterizan por el uso de unidades especiales además de las unidades estándar. Estas unidades especiales utilizan un conjunto de puertas para controlar cuándo la información ingresa a la memoria, cuándo sale y cuándo se olvida. Esta arquitectura (ver Figura 4) les permite aprender dependencias a largo plazo. Son muy usadas en el reconocimiento de voz y el reconocimiento de escritura.

2. **Transformers**, se trata de un modelo de Machine Learning que al igual que la redes LSTM están

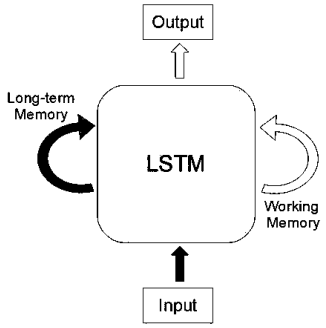

Fig. 4: Long short-term memory (LSTM)

diseñados para manejar datos de entrada secuenciales. A diferencia de las redes LSTM, este modelo mantiene la propiedad de conexiones residuales, esto permite que los gradientes fluyan a través de las redes directamente.
Esta propiedad reduce el olvido de algunas características del conjuntos de datos de entrada durante el entrenamiento.

3. **LSTM con conexiones residuales**, éste es el paradigma curioso, consiste en utilizar las conexiones residuales (ver Figura 5), llamadas también "Skip Connetions", en una red neuronal LSTM con el objetivo de reutilizar características entre los bloques de la red.

   Estas conexiones, empleadas también en los Transformers, omiten algunas de las capas de la red neuronal y alimenta la salida de una capa como entrada a las siguientes capas.
   Gracias a estas conexiones se evita el problema de la degradación, en el cual, el rendimiento del modelo disminuye con el aumento de la profundidad de la arquitectura de la red durante su entreno.

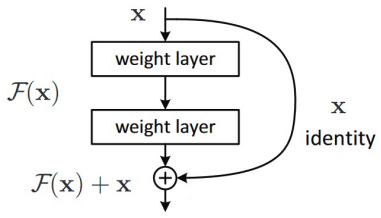

Fig. 5: Bloque con una "Skip Connection"

4. **Fully Convolutional Neural Networks (FCN)**, aunque las CNN (Redes Neuronales Convolucionales) predominan en el uso de conjuntos de datos de imágenes, también pueden ser más útiles que las RNN en el uso de datos temporales, gracias a que presentan una arquitectura (FCN) que se caracterizan por realizar únicamente operaciones de convolución, es decir, sin capas completamente conectadas.

   Las CNN son computacionalmente más baratas debido a que aprenden por lotes mientras que las RNN se entrenan secuencialmente y no pueden usar la paralelización porque debe esperar los cálculos

anteriores.
Pero no todo son ventajas, cuando se trata de que el modelo dependa de un largo historial y maneje diferentes tamaños de entrada y salida, las RNN son más adecuadas para esas tareas.

5. **LSTM Autoencoder**, este modelo busca aprender una representación comprimida de una entrada. Son capaces de aprender la dinámica compleja dentro del ordenamiento temporal de las secuencias de entrada, así como de usar una memoria interna para recordar o usar información a lo largo de largas secuencias de entrada.

   Su arquitectura interna se organiza en un Codificador-Decodificador LSTM (ver Figura 6) que permite usar el modelo para admitir secuencias de entrada de longitud variable y para predecir o generar secuencias de salida de longitud variable.

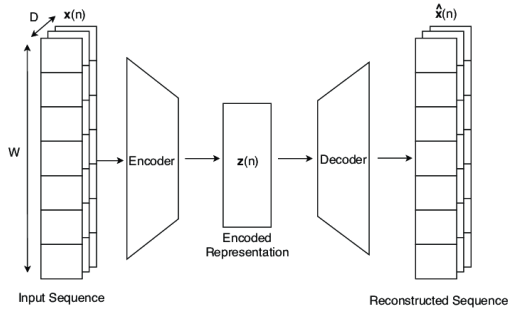

Fig. 6: Arquitectura de un Autoencoder

   La gracia de usar un Autoencoder sobre señales EEG es detectar las posibles anomalías presentes en las señales en función a la capacidad del modelo para recrear la secuencia de entrada.

   El modelo se entrenaría sólo con señales EEG normales permitiendo obtener una señal de salida muy similar a la señal de entrada. Sin embargo, si la señal de entrada es anómala, la representación resultante del Autoencoder diferirá de la entrada que se le ha proporcionado.

   El cálculo de la función de perdida entre la entrada y la salida y la definición de un "Threshold" nos permitirá clasificar las señales como normales o anómalas.

Conocidos los diferentes modelos, este proyecto se enfocará principalmente en el análisis del impacto/beneficio de incorporar la temporalidad en un modelo LSTM Autoencoder, capaz de detectar anomalías y clasificar ataques epilépticos. Compararemos los resultados de emplear un Autoencoder con los modelos actuales de clasificación.

3 OBJETIVOS

1. **Familiarización y estudio de la temática**
   - La base de datos: los datos originales se encuentran bastantes dispersos en cuanto a información se refiere. Hay mucha información que hay que tratar (los pacientes, los canales de las señales, el tamaño de las ventanas temporales,...) para generar una entrada definida.
   - Tipos de redes : investigar y entender los conceptos generales de las redes usadas hasta el momento, incluyendo sus características y diferencias.
   - Integración de los apartados anteriores: estudiar la posibilidad y el método de integrar la base de datos al modelo.
2. **Proyección de canales** Los 21 canales de las señales EEG proyectan diferentes tipos de actividad eléctrica cerebral, como la actividad de origen extra-cerebral, el parpadeo, el movimiento, el sudor, etc.
   Componen una dimensión más de los datos y presentan el problema de cómo estructurarlos para que el entrenamiento sea el correcto debido a que los canales no miden las señales de la misma manera. Esto hace que los resultados difieran en función a la distribución que les asignemos.

   Se han utilizado 3 tipos de disposición de los canales:
   - C3-C4: Consiste en emplear únicamente el canal resultante de la resta de los canales C3 y C4, considerado el canal con mayor importancia para la detección de la epilepsia. Nos permitirá reducir la información redundante de los otros canales.
   - Canales consecutivos: Concatenar los canales con el fin de tenerlos a todos en cuenta y reducir una dimensión en los datos.
   - Media aritmética: Realizar el promedio de todos los canales, considerando, de manera proporcional, la información procedente de todos los canales. Se partirá desde esta configuración.
3. **Disposición de las ventanas** Si las ventanas de las señales no están dispuestas de manera secuencial, no estaríamos aplicando la temporalidad al modelo LSTM. Para ello, hay que transformar las ventanas, muestreadas a 128Hz (128 valores por cada segundo de señal), en una disposición que contenga un conjunto como entrada. Es un parámetro ajustable.

   En el proceso de transformación se ha teniendo en cuenta por igual las señales de todos los sujetos. Lo malo de esto es que podría ocasionar una estratificación de la población (Population Stratification[1]). Pero al tener pocas ventanas de cada paciente y numerosos pacientes, obtendremos mejores resultados tratando los sujetos como uno solo en lugar de individualizarlos dentro del modelo.

[1]La estratificación de la población surge cuando diferentes proporciones de casos son tomados de muestras de sujetos diferentes, lo que provoca que cualquier influencia en los resultados se deba a diferencias entre sujetos en lugar del estudio en sí.

4. **Baseline Model** El primer modelo a desarrollar será una red LSTM Autoencoder con proyección linear que permita reconstruir una ventana suministrada como input y clasificar su salida.

   Como datos de entrada se realizará el promedio de los canales y las ventanas se insertarán en conjuntos de longitud variable para cada entreno. De esta forma comprobaremos cómo responde un Autoencoder frente a la detección de anomalías epilépticas.

5. **Selección del umbral y métricas de evaluación** Una vez entrenada la red, hay que definir una función de pérdida para calcular la desviación entre las predicciones realizadas y los valores obtenidos de todas las muestras. Mediante la representación de las desviaciones podemos establecer un umbral que nos permitirá clasificar basándose en las pérdidas de ventanas anómalas y no anómalas.

   Para evaluar la exactitud y precisión del modelo se emplearán las principales métricas de clasificación binaria: Precision, Recall, F1, Accuracy y la matriz de confusión. Estas métricas están implementadas en la librería "Sklearn.metrics".

6. **Optimización del modelo** La optimización se realizará atendiendo a las métricas de evaluación. Se modificarán los parámetros generales en busca de la mejor configuración de la clasificación. Entre estos parámetros tenemos:

   - Disposición de los canales
   - Número de épocas de entreno
   - Encoding Dimension[2]
   - Learning Rate[3]
   - Número de ventanas y de muestras

7. **Outputs** Capacidad para clasificar ventanas empleando diferentes "outputs". Estas salidas pueden tener diferentes dimensiones, la clasificación se puede obtener sólo de la última ventana de una secuencia, o también de un conjunto de ventanas secuenciales (requiere modificación de las entradas a la red).

4 Planificación del trabajo

El trabajo a realizar se va a dividir en cuatro fases objetivas, a cada fase se le estima una dedicación del tiempo total invertido de todo el trabajo:

4.1 Fase inicial (35% del tiempo total)

La primera fase consiste en entender el funcionamiento de las redes neuronales sobre datos secuenciales, en nuestro caso, las señales EGG tratadas. Además de entender la definición de los datos.
Por otro lado, se ha de estructurar estos datos de entrada al modelo, eligiendo una aproximación y disposición adecuada de las señales EEG en ventanas temporales secuenciales.

[2]Parámetro que establece el ratio de compresión, relacionado al número de neuronas ocultas en cada capa de la red

[3]Parámetro de ajuste en un algoritmo de optimización

Una vez estructurados los datos de manera correcta, pasamos a la elaboración del modelo clasificatorio.

4.2 Fase implementación (30% del tiempo total)

La siguiente fase comienza con la implementación y entrenamiento de la red LSTM Autoencoder. En esta fase se ven los primeros resultados de cómo afecta el uso de la temporalidad en un Autoencoder. Además de usar diferentes disposiciones de canales y ventanas.

4.3 Fase testeo (20% del tiempo total)

Una vez realizado el entrenamiento, pasamos a la fase de validación, en la que comprobamos la salida obtenida con el groundtruth de los datos.
Se cuantificarán los resultados mediante el uso de métricas y se realizarán modificaciones necesarias para optimizar/mejorar la exactitud del modelo.

4.4 Fase final (15% del tiempo total)

La ultima fase consistirá en el análisis y documentación de los resultados obtenidos de la fase previa además de la elaboración del informe final y la presentación.

5 Metodología

5.1 Tratamiento de datos

Este proceso implica eliminar la información redundante, reducir el número de dimensiones realizando combinaciones entre los diferentes canales de las señales EEG y establecer los parámetros iniciales.

Como datos previos al estudio y creación del modelo predictivo partimos de líneas filtradas de actividad eléctrica cerebral. Estas líneas proceden de 22 sujetos (5 hombres de 3 a 22 años; y 17 mujeres de 1.5 a 19 años) que padecen convulsiones intratables y han sido recopiladas en el Hospital de Niños de Boston. Constan de señales multicanales EEG (ver Figura 7) agrupadas en 23 casos.

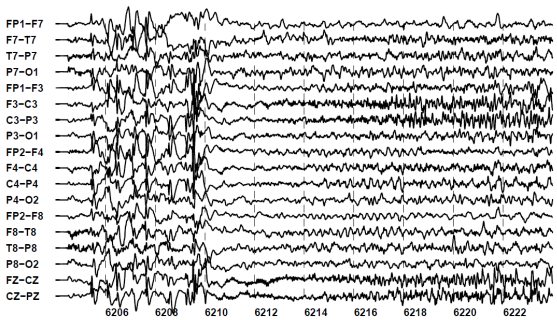

Fig. 7: Señales EEG durante un ataque epiléptico

La base de datos inicial, CHB-MIT, se encuentra publicada en el portal web de PhysioNet [1] y ha sido tratada y

cedida para este trabajo por parte del Centro de Visión por Computador de la UAB [2].

El equipo del CVC ha empleado ciertos umbrales, gracias a los cuales se ha podido realizar la correcta distribución y etiquetación de las señales EEG.
Estos datos se encuentran clasificados en dos conjuntos de muestras no balanceadas, las muestras anómalas solo componen un 2% de los datos.
No realizamos un balanceo ya que como se comentó anteriormente, la gracia de utilizar un Autoencoder es entrenar el modelo atendiendo únicamente a los datos normales. Estas muestras anómalas junto a la partición de los datos normales no vistos por el modelo, componen el conjunto de validación.
El dataset original se ha reducido debido al elevado número de muestras que contenía, esto era uno de los principales problemas por el coste computacional que requiere tratar con muchos datos .
La estructura de las muestras anómalas se han transformado para que sea igual a la de las muestras de entrenamiento de la siguiente manera: [NSamples, LSeq, LSignal] donde:

- NSamples: Número de muestras del conjunto. Cada muestra contiene un número de ventanas consecutiva.
- LSeq: Número de ventanas dentro de una secuencia.
- LSignal: Longitud de la ventana. Como los datos están muestreados a 128hz, cada ventana de un segundo contiene 128 valores de señales.

Inicialmente existía una dimensión más correspondiente al número de canales, pero desaparece ya que la proyección de canales que se va a seguir es la media aritmética.

5.2 Implementación Red

Para esta primera implementación de la red LSTM Autoencoder empleamos 3600 muestras normales, previamente tratadas y distribuidas en 80% para la parte de entreno y el 20% restante para la parte de testeo. Estas particiones son resultantes de hacer un KFOLD 5.
Dimensionamos los datos de la siguiente manera: [NSamples, LSeq, LSignal] siendo NSamples=3600 LSeq=5 y LSignal=128.
El numero de muestras y de ventanas han sido escogidos aleatoriamente para esta primera implementación. Para el batchSize se ha seleccionado la longitud de la ventana(LSignal)

El Autoencoder se compone de 2 conjuntos de capas, un codificador o "Encoder" y un decodificador o "Decoder":

- **Encoder**: compuesto por 2 capas LSTM, se encarga de comprimir la información que recibe como entrada basandose en el ratio de compresión (definido por el parámtro "Encoding Dimension"). Este parámetro se ha probado con el valor 64.
  El codificador transforma una entrada de [1,LSeq,LSignal] en una salida codificada de [1,EncodingDim]. Esta salida es recibida por el decodificador.
- **Decoder**: además de dos capas LSTM (como el Encoder) incluye una capa Linear para reconstruir la estructura de la secuencia inicial. El Decoder transforma una entrada [1,EncodingDim] en una secuencia [1,LSeq,LSignal].

De esta manera hemos implementado un modelo de red neuronal capaz de codificar y decodificar una secuencia. Lo siguiente es establecer una función de perdida y un umbral de separación para que cuando se reciba una señal anómala, la función de pérdida aumentará permitiéndonos detectarlas y clasificarlas a partir del umbral.

Durante el entrenamiento de este modelo observamos que el coste computacional para una CPU era bastante elevado. Gracias a que el CVC[2] me proporcionó una máquina virtual con GPU, el coste del proceso de entrenamiento se redujo casi a la mitad.

5.3 Selección del umbral (Threshold)

Una vez entrenada la red y obtener las perdidas resultantes de aplicar el criterio MSE[4] como función de pérdida, definimos el "Threshold".
Es un factor muy importante debido a que es el valor distintivo entre una señal anómala y otra que no lo es.
Se ha elegido el método OTSU, cuya fórmula es:

$$\sigma_w^2(t) = w_0(t)\sigma_0^2(t) + w_1(t)\sigma_1^2(t)$$

Este algoritmo se utiliza para realizar la umbralización automática de imágenes devolviendo un único umbral de intensidad que separa los píxeles en dos clases, primer plano y fondo.
Aplicando esto a una distribución univariada de todas las pérdidas (ver Figura 8) resultantes del entreno con ventanas normales(sin ataques), podemos representar el umbral de separación:

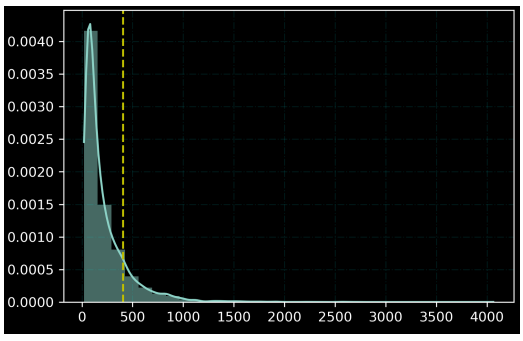

Fig. 8: Threshold OTSU

5.4 Optimización de la red

Para la optimización de la red atendemos a ciertos criterios:

- Entrenar la red con ventanas temporales de diferentes longitudes, para evaluar con qué número de ventanas clasifica mejor.

[4]Error cuadrático medio (MSE) mide el promedio de los errores al cuadrado, es decir, la diferencia entre el estimador(input) y lo que se estima(output)

- Modificar el número de neuronas internas de la red.
- Atender a la función de pérdida (Loss Function[5]), gracias a la cual podemos conocer a partir de qué momento temporal la red deja de aprender.

Se han creado listas de configuraciones de los parámetros mediante las cuales podremos combinar distintos tipos de configuraciones en busca del mejor funcionamiento del modelo.

5.5 Evaluación

Para evaluar la exactitud y precisión del modelo realizamos el testeo con el 20% de los datos no incluidos en la fase de entrenamiento y cuantificamos los resultados utilizando las métricas de clasificación binaria mencionadas en los Objetivos.

Se aplicará un K-Fold o validación cruzada (ver Figura 9), para garantizar que los resultados son independientes de la partición entre datos de entrenamiento y testeo.
Dividimos los datos en k=5 grupos de aproximadamente el mismo tamaño y realizaremos k iteraciones del proceso de entreno.

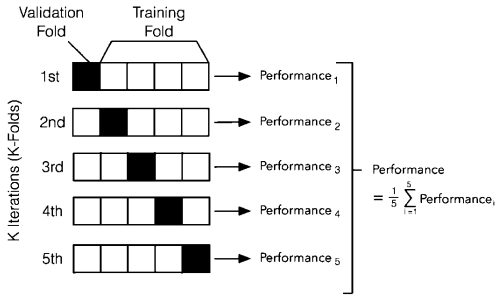

Fig. 9: KFold 5 Datos Entrenamiento

De esta manera utilizamos un grupo distinto como validación en cada iteración y la evaluación será la resultante de la media aritmética de todas las evaluaciones de las iteraciones.

6 Resultados y Comparación

Los resultados de las primeras ejecuciones no fueron del todo buenos, obteniendo precisiones cerca del 46% para los datos de validación. Principalmente se debía un error de estructura en el Encoder, cuya salida no codificaba correctamente la dimensión LSignal(128). Este error se corrigió mediante la activación del parámetro batch_first[6] de la red.

Con esta modificación los resultados mejoraron notoriamente con precisiones cerca del 70% de aciertos con los datos de validación.

El coste computacional del entrenamiento es bastante elevado, la duración del entreno con 3600 muestras de 5 ventanas de 1 segundos y 30 épocas era de casi 9 horas empleando un KFold 5.

A partir de este punto he establecido diferentes combinaciones de configuraciones para eliminar aquellas parametrizaciones que no aportan mejoras al modelo:

- LIST_LEARNING_RATE=[0.01, 0.001, 0.0001]
- LIST_ENCODING_DIM=[2, 7, 64]
- LIST_SEQL=[2, 5, 10]

Los resultados del entreno de estas configuraciones podemos verlos mediante sus funciones de pérdida (ver Figura 11).

Atendiendo a los criterios de convergencia y decrecimiento de la pérdida a medida que pasan las épocas, seleccionamos como mejores resultados las muestras con 2 y 5 ventanas y un LearningRate = 0.001.

Vemos en los resultados que las pérdidas continúan decreciendo sin estabilizarse, es por ello que hay que aumentar el número de épocas para que el modelo se entrene de manera efectiva.
Se repite el proceso de entreno con estas selecciones, pero esta vez aumentando a 150 el número de épocas y a 10.000 el de muestras normales(8.000 para el entreno y 2.000 para la validación).

Los resultados (ver Figura 12) de las funciones de pérdida muestran una mejor convergencia y entrenamiento con estos parámetros. Se observa que la variación del parámetro EncodingDim no afecta en la mayoría de los casos durante el entrenamiento, pero con un EncodingDim = 7 las gráficas convergen algo mejor. Realizamos la clasificación estableciendo como parámetros más óptimos los siguientes:

- Learning_Rate = 0.001
- Encoding_Dim = 7
- SeqL = 2

Los resultados de la clasificación, cuantificados por las métricas, se muestran en la siguiente tabla:

class	precision	recall	f1-score
0	0.86	0.87	0.86
1	0.87	0.86	0.86
accuracy	0.865	0.865	0.86
macro avg	0.865	0.865	0.86
weighted avg	0.865	0.865	0.86

Un 86% en cuanto a la precisión, es un resultado muy positivo para esta clasificación empleando Autoencoders. No llega a ser un valor tan alto como el de los actuales modelos que rondan entre el 95% y 98%, pero es un valor que se le aproxima.

El conjunto de validación está compuesto por 2.000 muestras normales no vistas por el modelo y 2.000 muestras

[5]Función que evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje.

[6]En las RNN, este parámetro nos traspone las dos primeras dimensiones a la hora de seleccionar el batch size

anómalas. Si observamos la matriz de confusión resultante (ver Figura 10) vemos que el prácticamente 3.476 de las 4.000 muestras se han clasificado correctamente, que corresponde con una precisión del 86% de aciertos. En cuanto al tiempo, la clasificación se ha realizado en menos de 3 segundos.

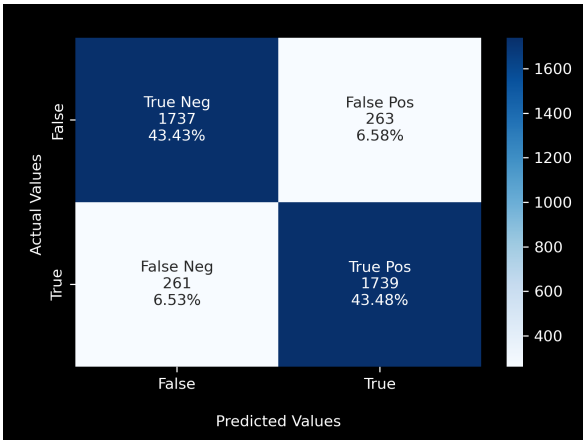

Fig. 10: Matriz de confusión

7 Conclusiones

Mediante este estudio hemos podido concluir que los Autoencoders pueden ser un clasificador bastante efectivo en cuanto a detección de anomalías se refiere. Los resultados han sido mejores de los que esperaba, aproximados a una precisión de casi el 90%. Uno de los inconvenientes que le veo, al igual que ocurre en otros modelos, es el alto coste computacional para la fase de entrenamiento.
Entre las conclusiones que destaco de este trabajo:

- Me ha servido para perfeccionar mis conocimientos en la rama del Aprendizaje Profundo(Deep Learning).
- Me ha parecido curioso la idea de emplear un codificador para la clasificación de anomalías entrenado sólo con muestras de una clase.
- Pienso que ha sido de gran utilidad investigar sobre una temática que actualmente representa un 0,5% de la carga de morbilidad mundial, convirtiendo la epilepsia en uno de los trastornos neurológicos más comunes.

8 Agradecimientos

El agradecimiento del presente Trabajo de Fin de Grado va dirigido a mi tutora y profesora Débora Gil, gracias al aporte de ideas y seguimiento en todo momento.
Mi gratitud al Centro de Visión por Computador[2] por la cesión de los datos de Epilepsia correctamente tratados y etiquetados.
Y finalmente a mi familia y amigos, que gracias al apoyo y confianza pude acabar mi carrera de Ingeniero Informático.

References

[1] Base de datos CHB-MIT

[2] Centro de Visión por Computador (CVC). Campus UAB, Edifici O, s/n, 08193 Cerdanyola del Vallès, Barcelona

[3] Ali Shoeb. Application of Machine Learning to Epileptic Seizure Onset Detection and Treatment. PhD Thesis, Massachusetts Institute of Technology, September 2009.

[4] Almustafa, K. M. (2020). Classification of epileptic seizure dataset using different machine learning algorithms. Informatics in Medicine Unlocked, 21, 100444.

[5] Nandy, A., Alahe, M. A., Nasim Uddin, S. M., Alam, S., Nahid, A. A., Awal, M. A. (2019). Feature Extraction and Classification of EEG Signals for Seizure Detection. 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)

[6] Chirasani, S.K.R., Manikandan, S. A deep neural network for the classification of epileptic seizures using hierarchical attention mechanism. Soft Comput 26, 5389–5397 (2022)

[7] K. Shekokar, S. Dour y G. Ahmad, ”Clasificación de convulsiones epilépticas mediante LSTM”, 8.ª Conferencia internacional sobre procesamiento de señales y redes integradas (SPIN) de 2021, 2021, págs. 591-594, doi: 10.1109/SPIN52536.2021.9566118.

[8] Goldberger, A., Amaral, L., Glass, L., Hausdorff, J., Ivanov, P. C., Mark, R., ... Stanley, H. E. (2000). PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. Circulation [Online]. 101 (23), pp. e215–e220.

[9] José Yauri, Aura Hernández-Sabaté, Pau Folch and Débora Gil. Mental Workload Detection Based on EEG Analysis. Computer Vision Center, Universitat Autònoma de Barcelona, Spain

[10] Lotte, F., Bougrain, L., Cichocki, A., Clerc, M., Congedo, M., Rakotomamonjy, A., Yger, F. (2018). A review of classification algorithms for EEG-based brain–computer interfaces: a 10 year update. Journal of neural engineering, 15(3), 031005.

[11] Blankertz, B., Tomioka, R., Lemm, S., Kawanabe, M., Muller, K. R. (2007). Optimizing spatial filters for robust EEG single-trial analysis. IEEE Signal processing magazine, 25(1), 41-56.

[12] Roy, Y., Banville, H., Albuquerque, I., Gramfort, A., Falk, T. H., Faubert, J. (2019). Deep learning-based electroencephalography analysis: a systematic review. Journal of neural engineering, 16(5), 051001.

[13] Craik, A., He, Y., Contreras-Vidal, J. L. (2019). Deep learning for electroencephalogram (EEG) classification tasks: a review. Journal of neural engineering, 16(3), 031001.

[14] Schirrmeister, R. T., Springenberg, J. T., Fiederer, L. D. J., Glasstetter, M., Eggensperger, K., Tangermann, M., ... Ball, T. (2017). Deep learning with convolutional neural networks for EEG decoding and visualization. Human brain mapping, 38(11), 5391-5420.

[15] Golmohammadi, M., Harati Nejad Torbati, A. H., Lopez de Diego, S., Obeid, I., Picone, J. (2019). Automatic analysis of EEGs using big data and hybrid deep learning architectures. Frontiers in human neuroscience, 13, 76.

[16] Li, X., Zhao, Z., Song, D., Zhang, Y., Pan, J., Wu, L., ... Wang, D. (2020). Latent Factor Decoding of Multi-Channel EEG for Emotion Recognition Through Autoencoder-Like Neural Networks. Frontiers in Neuroscience, 14, 87.

[17] López-Larraz, E., Ibáñez, J., Trincado-Alonso, F., Monge-Pereira, E., Pons, J. L., Montesano, L. (2018). Comparing recalibration strategies for electroencephalography-based decoders of movement intention in neurological patients with motor disability. International journal of neural systems, 28(07), 1750060.

[18] Hartmann, K. G., Schirrmeister, R. T., Ball, T. (2018). EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals. arXiv preprint arXiv:1806.01875.

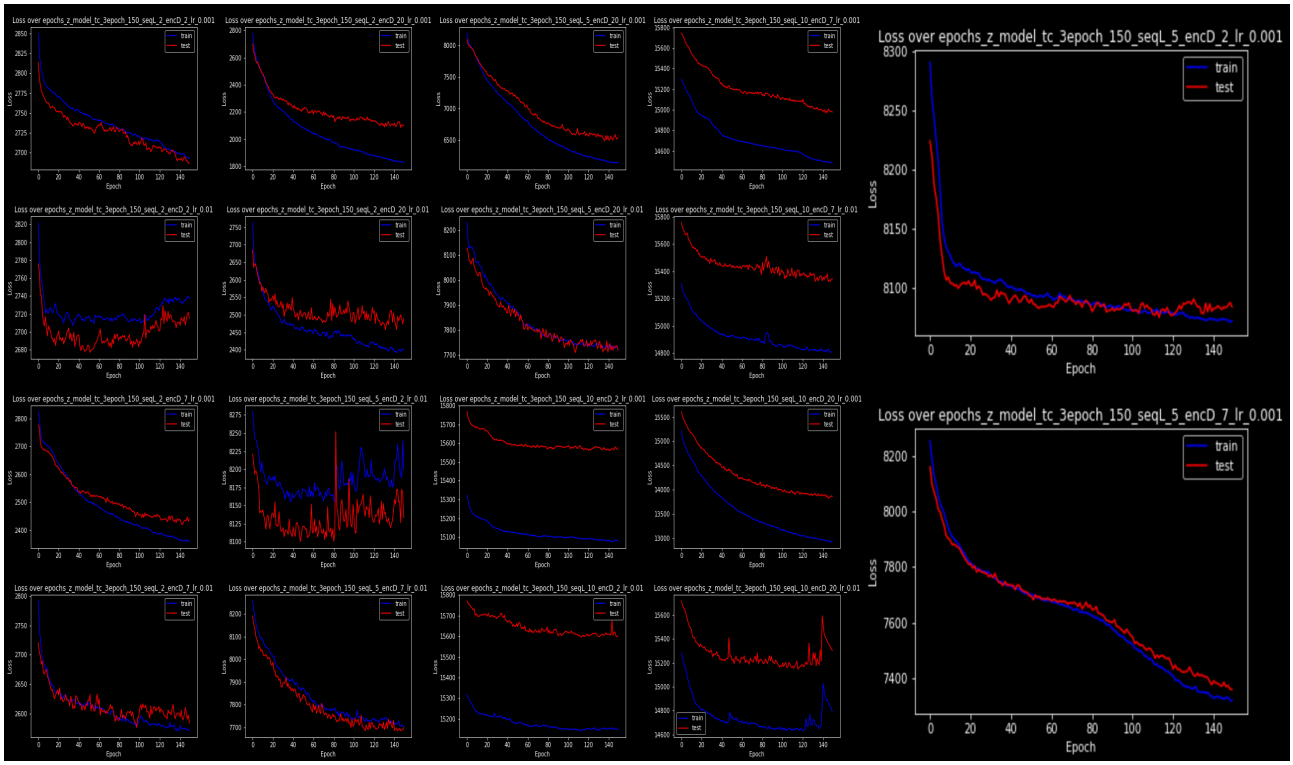

Fig. 11: Funciones de Pérdida por Épocas

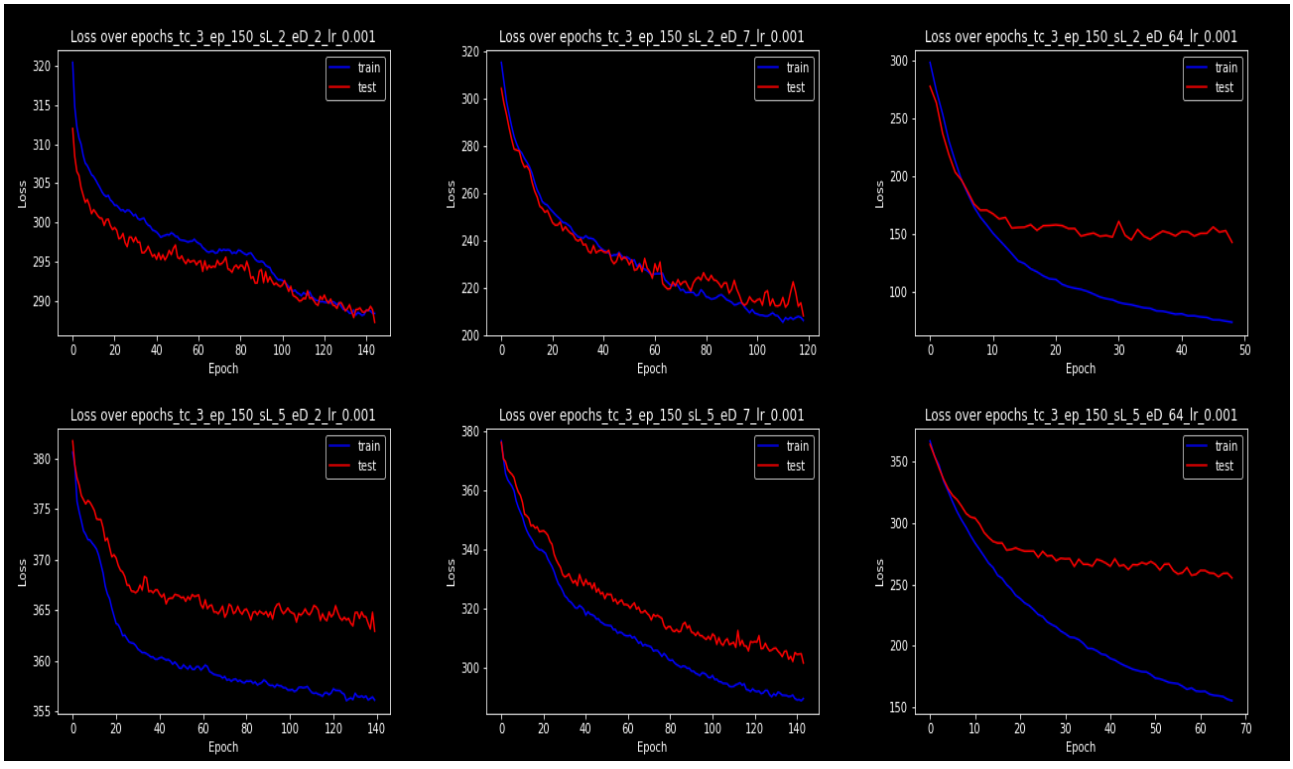

Fig. 12: Funciones de Pérdida por Épocas (Modelo Optimizado)