Escuela
Politécnica
Superior

Eliminación de ruido en señales de espectroscopía infrarroja usando redes neuronales

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Antonio Bri Pérez

Tutores:
Juan Ramón Rico Juan
Victor José Climent Payá

Junio 2022

Universitat d'Alacant
Universidad de Alicante

Eliminación de ruido en señales de espectroscopía infrarroja usando redes neuronales

Autor

Antonio Bri Pérez

Tutores

Juan Ramón Rico Juan
Dpto. Lenguajes y Sistemas Informáticos

Victor José Climent Payá
Dpto. Química Física

Trabajo final de grado en Ingeniería Informática

Alicante (España) - Junio 2022

If a machine is expected to be infallible, it cannot also be intelligent.

Alan Turing

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia por haber creído en mí, apoyarme incondicionalmente y transmitirme los valores que hoy me permiten cumplir mis metas. Por otro lado, quiero agradecer a mis amigos, quienes me han visto crecer, que nunca han dudado de mí y de los que nunca me faltó el apoyo y cariño.

También, me gustaría agradecer a mis tutores Juan Ramón y Víctor por contar conmigo para este trabajo y su inestimable ayuda, así como al profesor Antonio Rodes por la extracción de los datos empleados.

Muchas gracias a todos.

Resumen

La espectroscopía es una técnica analítica basada en la medida de la absorción o trasmisión de radiación electromagnética por una determinada muestra en función de su longitud de onda o frecuencia. El gráfico de la intensidad de la radiación a cada longitud de onda se denomina *espectro* y es característico de la composición de la muestra (Díaz et al., 2010). Por ello, esta técnica se usa ampliamente en análisis químico tanto para detectar la presencia de una determinada sustancia química como para cuantificar su concentración. Existe una amplia variedad de técnicas espectroscópicas que pueden aplicarse a distintos tipos de muestras sólidas, líquidas o gaseosas. Con frecuencia, la muestra está constituida por una disolución de una o varias sustancias en un disolvente adecuado. Las muestras que se tratarán en esta memoria emplean como disolvente H_2O (agua).

Sin embargo, uno de los problemas principales de esta aproximación es la presencia del ruido. Este ruido es una señal no deseada que se superpone siempre a la respuesta ideal o verdadera de la muestra y que limita la capacidad de detección de las sustancias, especialmente en aquellos casos en los que los rangos de ambas son similares. El origen de dicho ruido puede ser muy variado, desde interferencias presentes en la muestra o en el camino óptico de la radiación, hasta ruido eléctrico inherente a la electrónica del sistema de medida.

Durante muchos años, se ha tratado de encontrar una forma eficaz de poder reducir este ruido. Una posibilidad es el promediado de múltiples medidas, lo que sirve para eliminar el ruido aleatorio y separarlo de la señal verdadera. Sin embargo, esto no evita el ruido debido a interferencias que no sean aleatorias, además de ser costoso en términos de tiempo de adquisición. Otra posibilidad es tratar la señal mediante algoritmos matemáticos que traten de separar la señal del ruido. Uno de los métodos más aceptados y cuya aplicabilidad ha sido demostrada en este sentido es el de la Transformada Wavelet

(TW). Otras alternativas a la TW, pueden encontrarse en filtros tradicionales, como por ejemplo los filtros gaussianos, Wiener o las Wavelets ortogonales (Vargas Cañas, 2011) En este estudio, se presenta una aproximación alternativa y novedosa basada en redes neuronales profundas (Deep Learning) para intentar extraer la información presente en una señal ruidosa mediante el aprendizaje basado en modelos especializados en el filtrado de ruido como son los Autoencoders Convolucionales (Fuchs et al., 2020).

Índice general

Índice de figuras

Lista de abreviaturas

AE Autoencoder

CNN Convolutional Neural Network

DL Deep Learning

IA Inteligencia Artificial

IR Infrarroja

LSTM Long Short-Term Memory

MA Moving Average

MLP Multi Layer Perceptron

PLN Procesamiento del Lenguaje Natural

RMSE Root Mean Squared Error

RN Red Neuronal

TW Transformada Wavelet

UV Ultravioleta

Lista de tablas

Capítulo 1

Introducción

Existen diferentes aproximaciones para detectar la presencia de una sustancia química en una muestra y/o cuantificar su concentración. Una de las más importantes y eficaces es la espectroscopía, por su rapidez y sensibilidad química. Primero, es fundamental saber qué es un espectro para comprender cómo funciona esta técnica. Un espectro es la representación gráfica de la relación entre la radiación electromagnética absorbida por una muestra y la frecuencia de dicha radicación. Para obtenerlo, se irradia la muestra con radiación electromagnética de distinta longitud de onda o frecuencia y se registra la fracción de intensidad que se trasmite (o, alternativamente, la fracción que se absorbe). Dependiendo del tipo de radiación utilizada para irradiar la muestra, podemos clasificar la espectroscopía en distintos tipos, siendo la espectroscopía Infrarroja (IR) y la Ultravioleta (UV) - visible las más comunes. (Ordóñez, 2012).

Sabiendo ya que es un espectro, la espectroscopía se define como un método que consiste en estudiar la absorción o emisión de energía originada por la interacción entre el material de estudio y la radiación electromagnética (Piqué and Vázquez, 2012). El espectro electromagnético se puede dividir en diferentes regiones según la longitud de onda, la frecuencia y la energía de la radicación. Algunas de estas regiones son la de los rayos gamma, rayos X, luz UV, visible, infrarroja, microondas y ondas de radio, entre otras (Rojas Monsalvo et al., 2013). Según la energía de la radiación, se produce un tipo u otro de interacción con las moléculas de la muestra. Así, la radiación UV visible produce transiciones entre los niveles electrónicos de los átomos que son los responsables de que las sustancias parezcan coloreadas, la radiación IR produce vibraciones en los enlaces entre átomos y la radiación de microondas produce rotaciones de las moléculas

en su conjunto.

Este estudio se ha centrado en el tratamiento numérico de espectros obtenidos mediante espectroscopía IR. Este tipo de espectroscopía es muy sensible, rápida y robusta y cuyo coste de implementación es asumible (Vargas Cañas, 2011). Los espectros están compuestos por un conjunto de bandas representativas de distintos modos vibracionales de las moléculas. Para moléculas orgánicas relativamente complejas, el espectro puede poseer multitud de bandas cuya posición y magnitud es muy característica de la estructura molecular. Se podría decir que el espectro constituye algo similar a una huella digital de la molécula, permitiendo identificar su presencia en una muestra y cuantificar su concentración.

El conjunto de datos empleado en este estudio ha sido obtenido mediante un espectrofotómetro de IR con transformada de Fourier. La transformada de Fourier tiene muchas aplicaciones en el mundo de la ingeniería y la física, por ejemplo, para pasar una señal del dominio temporal al dominio de frecuencia o el cálculo de ecuaciones diferenciales (González, 1997). En el caso concreto de la espectroscopía IR el uso de la transformada rápida de Fourier supuso un hito importante al permitir adquirir los espectros de forma más rápida. En esta modalidad, en lugar de usar radiación de longitud de onda variable e ir registrando consecutivamente la intensidad de la radiación absorbida a cada radiación, se irradia la muestra con una luz blanca que contiene todas las longitudes de onda. Se utiliza un dispositivo, denominado interferómetro de Michelson, para extraer la información correspondiente a cada longitud de onda. Este dispositivo utiliza un conjunto de espejos que, a medida que se mueven, permiten obtener la información relevante. Un interferograma representa la intensidad de la señal trasmitida en función de la posición de los espejos. Este dispositivo está diseñado de tal manera que el interferograma puede transformarse en un espectro mediante una transformada de Fourier.

Algunas de las ventajas de emplear la espectroscopía con transformada de Fourier es que ofrece una mejora de la relación señal/ruido para una resolución dada, mayor frecuencia, velocidad y posibilidades para incorporar procesado de datos. Otra de las ventajas de la espectroscopía con la transformada de Fourier es que, al utilizar al mismo tiempo una radiación con todas las longitudes de onda, en lugar de usar un monocromador para separar las diferentes longitudes de onda, es posible registrar un

interferograma en una fracción de segundo, lo que permite repetir la adquisición y promediar distintos interferogramas para mejorar la relación señal ruido (Harris, 942). El espectrofotómetro utilizado en este estudio permite adquirir un interferograma en 0,2 segundos, permitiendo acumular aproximadamente 100 interferogramas en 20 segundos.

Para filtrar el ruido de los datos extraídos del espectrofotómetro, se han estudiado diferentes arquitecturas y clases de redes neuronales profundas. Sin embargo, hay que tener en cuenta que las medidas experimentales poseen una contribución de ruido debido a varios factores: ruido electrónico de los diferentes componentes tales como la lámpara o el detector, variabilidades en la muestra o ruido ambiental proveniente de la red eléctrica o radiación IR externa. Para conseguir reducir esta contribución de ruido se suele prolongar el tiempo de adquisición o promediar diferentes medidas, por tanto, conviene buscar un compromiso entre mínimo tiempo de adquisición y nivel de ruido aceptable. No obstante, puede ser necesario un tiempo de adquisición bajo para poder determinar sustancias con tiempos de vida reducidos como sustancias intermedias en el transcurso de una reacción química.

Poder evaluar el comportamiento de las redes neuronales ha supuesto todo un reto, ya que a diferencia de los problemas clásicos de filtrado de ruido, de regresión o clasificación, no se ha dispuesto de la salida o de los valores de referencia que queremos que la red neuronal aprenda. Por ello, se proponen un par de aproximaciones en este sentido que se detallan en la sección 4 Metodología.

1.1. Motivación

Los inicios del aprendizaje automático o Deep Learning (DL) se remontan a 1943 con la creación de un modelo computacional basado en las conexiones neuronales del cerebro humano en el que se usaban una combinación de algoritmos y reglas que llamaron "lógica de umbral"para emular el proceso del pensamiento humano (Caiafa and Lew, 2020).

Muchos años más tarde, en 2012, el gran aumento de la velocidad de las tarjetas gráficas, hicieron posible el entrenamiento y desarrollo de la red neuronal convolucional o Convolutional Neural Network (CNN), que será el tipo de red que cobre protagonismo a lo largo del trabajo (Ongsulee, 2017).

En análisis químico, los expertos de la materia realizan la interpretación del espectro siguiendo su juicio experto o bien aplicando una serie de algoritmos matemáticos para el filtrado del ruido de las señales. El objetivo principal del presente trabajo es estudiar y analizar las diferentes aproximaciones basadas en métodos clásicos de eliminación de ruido, comparándolas con nuevas alternativas basadas en el empleo de redes neuronales, para eliminar el ruido en una señal registrada mediante un espectrofotómetro IR con transformada de Fourier.

1.2. Objetivos

Los principales objetivos de este estudio son los siguientes:

- **Objetivo 1**: Realizar una revisión crítica del estado del arte en técnicas de tratamiento de señal y eliminación del ruido.
- **Objetivo 2**: Analizar el resultado de la aplicación de diferentes aproximaciones para la reducción de ruido en señales espectroscópicas reales y medir la calidad de los resultados obtenidos.
- **Objetivo 3**: Valorar el uso de DL para la eliminación de ruido en señales de espectroscopía.

1.3. Estructura del trabajo

Para facilitar la lectura del trabajo, se detalla a continuación la lista de capítulos:

- ***Capítulo 1: Introducción*** ⇒ Se describen conceptos para la comprensión del documento y se encuadran los objetivos principales del estudio.
- ***Capítulo 2: Estado del arte*** ⇒ Se detallan las metodologías empleadas y su funcionamiento.
- ***Capítulo 3: Tecnologías*** ⇒ Se concreta cuál es el lenguaje de programación empleado y las librerías usadas, así como el entorno y las condiciones de desarrollo.
- ***Capítulo 4: Metodología*** ⇒ Se describen los pasos empleados en el estudio, desde la obtención del conjunto de datos empleados, su preprocesado, proce-

dimiento de entrenamiento, validación hasta los detalles más relevantes de la implementación de código.

- ***Capítulo 5: Experimentación*** ⇒ Se detallan los experimentos realizados, los resultados obtenidos y la comparativa entre ellos.
- ***Capítulo 6: Discusión*** ⇒ Se discuten los resultados obtenidos y se ofrece una visión crítica de los mismos.
- ***Capítulo 7: Conclusiones y trabajos futuros*** ⇒ Se reflexiona sobre la metodología empleada y resultados obtenidos y se plantean nuevas líneas de investigación relacionadas al trabajo para el futuro.

Capítulo 2

Estado del arte

En esta sección se describen los principales métodos para la eliminación de ruido en señales y las arquitecturas de las redes neuronales más adecuadas que se han empleado en la experimentación.

2.1. Media móvil

La media móvil o Moving Average (MA) se trata de la técnica más sencilla de todas las empleadas en este estudio y que, además, es ampliamente utilizada por su sencillez y eficacia. Esta técnica es uno de los métodos más comunes de filtrado de señales digitales, principalmente por su facilidad de uso y baja dificultad de comprensión e implementación. Está especialmente indicada para eliminar el ruido aleatorio de una señal (Cupertino and Pereira, 2021).

Este filtro funciona promediando un número de puntos de la señal de entrada para producir cada uno de los puntos de la señal de salida. En la ecuación 2.1 se puede ver la fórmula usada para calcular cada uno de los puntos resultantes de aplicar este filtro.

$$y[k] = \frac{1}{k} \sum_{i=n-k+1}^{n} x_i \tag{2.1}$$

Donde x es señal de entrada, y es la salida de la señal, k es el número de puntos en el promediado o tamaño de la ventana y n es el número de valores observados. El uso de este método es tan simple y eficiente que muchas veces no es necesario emplear ningún método adicional. Sin embargo, el filtro de la MA puede reducir la intensidad de la señal con la consecuente pérdida de las señales más pequeñas adyacentes (Guiñón

et al., 2007).

2.2. Transformada Wavelet

A continuación, se detalla una técnica más compleja y sofisticada de descomposición de señales como son los *wavelets*. Los métodos de filtrado de ruido basados en *wavelets* utilizan los beneficios de la TW que es un tipo particular de transformada que representa una señal en términos de versiones trasladadas y dilatadas de una onda finita. Concretamente, pueden descomponer una señal en diferentes escalas que representan diferentes bandas de frecuencia. En cada una de estas escalas, la posición de las estructuras instantáneas de la señal puede determinarse de forma aproximada. Esta propiedad de la TW puede utilizarse para la eliminación de ruido (Pan et al., 1999).

En este estudio, la *wavelet* empleada es la *Haar Wavelet* (equivalente a la *Daubechies 1*) la cual es uno de los tipos más simples de *wavelet* que existe y es de naturaleza discontinua (Zhang, 2019). Como se puede ver en la figura 2.1 presenta una forma discreta.

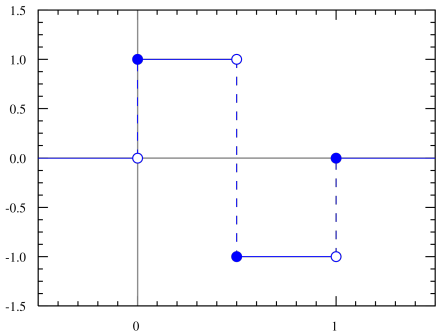

Figura 2.1: Wavelet Haar

En general, la familia de *wavelets Daubechies* es muy amplia y cubre el campo de las *wavelets* ortonormales e incluye desde las *wavelets* más localizadas hasta las más suaves. La diferencia entre estas *wavelets* se debe principalmente a la longitud que existe entre los filtros y las funciones de escalado. Por ejemplo, las *wavelets Daubechies* 4, 12 y 20 tienen coeficientes de filtrado de 4, 12 y 20, respectivamente. Estas características permiten que dichas *wavelets* sean muy útiles para la detección y localización de perturbaciones (Brito et al., 1998).

Por tanto, la transformada *wavelet* de una señal discreta f, puede ser representada

como una transformación lineal tal y como se puede ver en la ecuación (2.2)

$$w = W * f \tag{2.2}$$

Donde w es un vector que contiene los coeficientes de la TW y W es la matriz de los coeficientes de los filtros de la *wavelet* (Pasti et al., 1999).

Estos pocos coeficientes w que se obtienen de la TW y que concentran las características de la señal suelen ser ruido, por lo tanto, estos coeficientes se pueden reducir o eliminar sin afectar a la calidad de la señal. Una vez se han eliminado estos coeficientes, se reconstruye la señal mediante la TW inversa (MATLAB, 2022).

Mientras que la ventaja de esta wavelet es su simplicidad, tiene una desventaja asociada al hecho de ser discontinua, lo que provoca que no sea diferenciable (Zhang, 2019).

2.3. Red neuronal artificial

Para entender correctamente como funciona una Red Neuronal (RN), es fundamental tener presente que su inspiración inicial es modelar la forma de procesamiento de la información de manera similar a la de los sistemas nerviosos biológicos (Izaurieta and Saavedra, 2000).

Este principio les otorga a las redes algunas cualidades muy interesantes como tener una predisposición a adquirir el conocimiento por medio de la experiencia. Esta experiencia se refuerza, o no, mediante valores numéricos en las conexiones interneuronales (Izaurieta and Saavedra, 2000).

Otra cualidad de estas redes, es que son capaces de adaptarse muy bien y cambiar dinámicamente en función de los valores de entrada que se le presenten, así como una alta tolerancia a los fallos (Izaurieta and Saavedra, 2000).

Estas estructuras, como se puede apreciar en la figura 2.2, consisten en una capa de neuronas de entrada, una o más capas ocultas y una capa final de salida de neuronas. Cada una de las conexiones está asociada a un valor numérico al que se le conoce como peso y que se ajustará conforme vaya pasando a las siguientes capas, de forma que cobrará más o menos relevancia en función de los datos de entrada (Wang, 2003).

En una red de este tipo, cada una de las neuronas puede representar diferentes

objetos, tales como características, letras, conceptos o algún tipo de patrón abstracto con significado. En los últimos años, se han hecho grandes avances con estas estructuras en campos como reconocimiento de patrones, inteligencia robótica, control automático, biología, medicina y economía, entre muchos otros (Wu and Feng, 2018).

2.3.1. Perceptrón multicapa

El perceptrón multicapa o Multi Layer Perceptron (MLP) es un tipo de RN artificial dotada de un número variable de capas ocultas en su interior. En esta clase de red neuronal, se parte de un vector o matriz de entrada sobre el que se aplican una serie de transformaciones estructuradas en capas de neuronas para finalmente obtener una salida determinada (Maino, 2013). Uno de los problemas más frecuentes que las redes neuronales resuelven son los de clasificación.

La arquitectura de un MLP, tal y como se aprecia en la figura 2.2 está formada por una capa de entrada, un número variable de capas ocultas y una capa final de salida. Este tipo de RN se diferencia del resto de variantes de redes neuronales en que todas las neuronas están totalmente conectadas unas con otras. Esto significa que una neurona x de la capa n tiene conexión con todas y cada una las neuronas de la capa $n+1$ al mismo tiempo que recibe la entrada de todas y cada una de las neuronas de la capa $n-1$. Esto se cumple siempre para todas las neuronas de la capa oculta (Cichy and Kaiser, 2019).

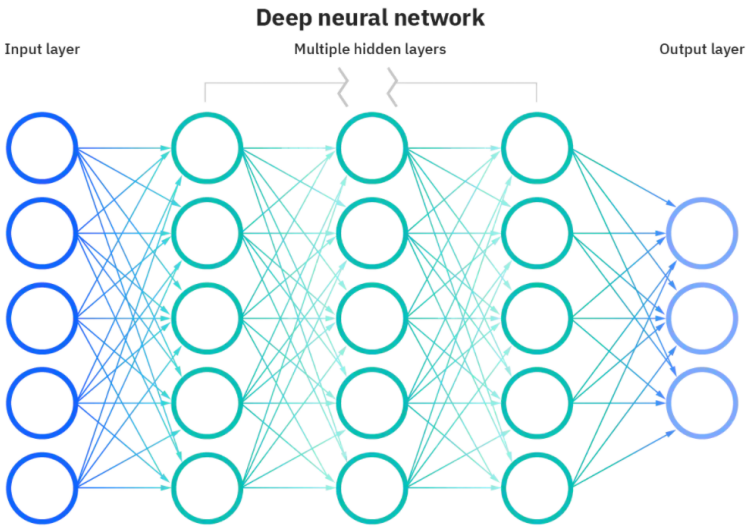

Figura 2.2: Perceptrón multicapa

Este tipo de redes dependen de los datos de entrenamiento en gran medida para aprender y mejorar su precisión[1]. Sin embargo, el funcionamiento teórico de una RN es bastante sencillo. Mediante la figura 2.3 se ilustra su funcionamiento:

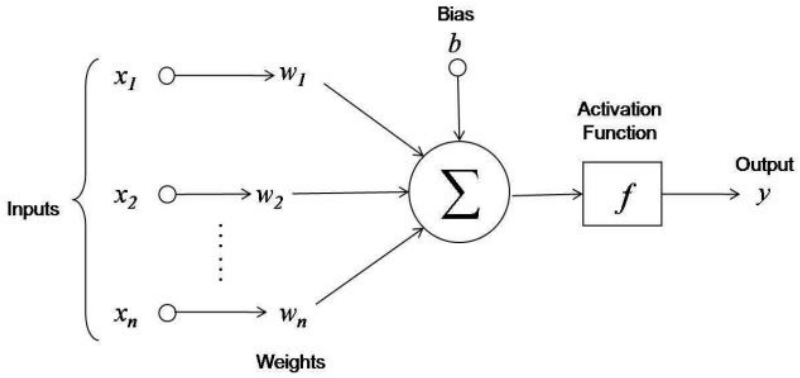

Figura 2.3: Diagrama de red neuronal

Para cada una de las neuronas de entrada x_n se le asigna un valor predeterminado. A estos valores se les llama pesos w_n y representarán la importancia de cualquier variable dada. A continuación, las entradas se multiplican por sus respectivos valores y se suman. Después, se le añade un sesgo o *bias* b, que proporciona un grado de libertad al modelo[2] y la salida producida se introduce a una función de activación, que es necesaria para romper la linealidad natural de las sumas y productos realizados anteriormente. De la figura 2.3 resulta natural obtener la expresión 2.3

$$y = f(\sum_{i=1}^{n} w_i x_i + b) \tag{2.3}$$

Donde f es la función de activación, w el peso, x el valor de entrada y b el sesgo que se le añade a la operación (Izaurieta and Saavedra, 2000).

A continuación, si dicha salida supera un determinado umbral, activa el nodo y pasa los datos a la siguiente capa de la red. Esto resulta en que, al final, las salidas de una capa se convierten en las entradas de la siguiente.

[1]Existen muchas otras métricas que determinan lo bien que se comporta una RN.
[2]Desplaza la función de activación a la izquierda o a la derecha.

2.3.2. Red neuronal recurrente

Hasta el momento, las RN que se han tratado en las secciones Red neuronal artificial 2.3 y Perceptrón multicapa 2.3.1 no tratan con secuencias, sino que reciben valores aislados y trabajan con ellos. En las RN recurrentes se observa un sistema dinámico, el cálculo de una entrada en un paso determinado depende del paso anterior y, en algunos casos, del paso futuro (Cruz et al., 2007).

Un caso de uso muy común para el que se emplean las RN recurrentes son tareas de Procesamiento del Lenguaje Natural (PLN) para realizar minería de opiniones o análisis de sentimientos, con el objetivo de estimar qué puntuación numérica corresponde a una retroalimentación específica (Rico-Juan and Calvo-Zaragoza, 2015). Es decir, estas arquitecturas son especialistas en resolver problemas en los que existen dependencias en el contexto, ya sea en el pasado o en el futuro.

Para resolver estos problemas, las RN recurrentes presentan uno o más ciclos en el grafo definido por las interconexiones de sus unidades de procesamiento, lo que les permite trabajar de forma innata con secuencias temporales (Pérez-Ortiz, 2002). Como se puede apreciar en la figura 2.4, estas redes son similares a las RN de la sección Red neuronal artificial 2.3, pero con bucles recurrentes que les permiten mantener un estado en el tiempo.

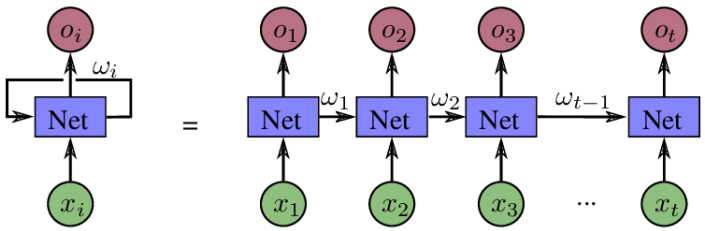

Figura 2.4: Red neuronal recurrente

Por lo tanto, estas neuronas predicen la salida w_i a partir de la entrada en el instante x_i y también a partir de la salida del estado anterior o_{i-1} (Pérez-Ortiz, 2002).

2.3.2.1. Red neuronal LSTM

Este tipo de RN es un tipo especial de RN recurrente y se caracteriza por tener conexiones de retroalimentación. Por lo tanto, puede procesar de forma efectiva secuencias temporales de datos, tales como una serie de imágenes (vídeo) o audio (discursos o melodías). Los mejores resultados con estas redes se han obtenido en campos como el reconocimiento de caracteres manuscritos o reconocimiento del habla (Yu et al., 2019). La estructura de una neurona Long Short-Term Memory (LSTM) (figura 2.5) difiere de la de una neurona recurrente en que incluye internamente una celda de memoria. Esta clase de celdas de RN, se utiliza principalmente en el análisis de secuencias, así como problemas de modelado del lenguaje (Rico-Juan and Calvo-Zaragoza, 2015).

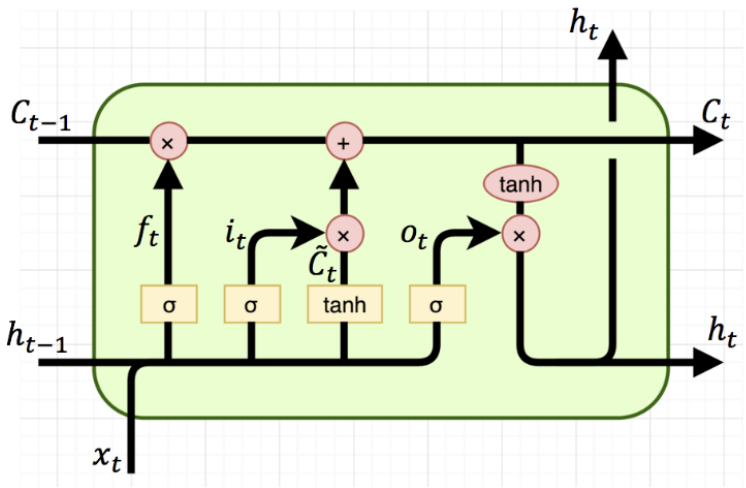

Figura 2.5: Neurona long short-term memory

En estas neuronas se tienen 3 tipos de puertas diferentes:

- Puerta de entrada: controla los valores de entrada que se van a utilizar para actualizar el estado de la memoria.
- Puerta de salida: controla los valores a devolver a partir de la entrada y del contenido de la memoria.
- Puerta de olvido: controla el mantenimiento del contenido de la memoria.

Este tipo de neuronas realizan un muy buen trabajo en tareas que requieren que un conjunto limitado de datos deba ser recordado durante un tiempo largo. Esto se consigue gracias al uso de dichas neuronas. Para ello, las puertas de entrada y salida

al final se encargan de que la información poco relevante entre o salga del bloque. Por otro lado, la puerta de olvido se encarga de ponderar la información dentro de la celda (Staudemeyer and Morris, 2019).

No obstante, estas neuronas requieren que su topología sea fijada al principio, además de que el número de neuronas LSTM no cambia de forma dinámica. Por tanto, la memoria de la red está limitada (Staudemeyer and Morris, 2019).

2.3.3. Red neuronal convolucional

Este tipo de redes presentan arquitecturas multicapas diseñadas para extraer representaciones de alto nivel dada una entrada. Gracias a ellas, se ha mejorado enormemente el estado del arte relativo al tratamiento de imágenes, vídeos, habla, reconocimiento de audio, etc (Gallego et al., 2020).

Cuando las CNN se entrenan para clasificación, son capaces de extraer un conjunto de características relevantes para resolver la tarea para la que se las entrena. Estas características se obtienen introduciendo una entrada sin procesar en la red en la que la última capa aprende las relaciones necesarias entre el dominio de la clasificación y dichas características, de forma que le asigna a cada categoría un porcentaje (Gallego et al., 2020).

Esta clase de redes están dotadas de múltiples capas convolucionales y de reducción alternadas. Al final de la red se ubican las capas de conexión total, al igual que un MLP (sección Perceptrón multicapa 2.3.1).

Tal y como se puede ver en la figura 2.6, en la operación de convolución se realizan operaciones de productos y sumas entre la capa inicial y los *kernels* que generan un mapa de características. Esta operación de convolución viene determinada por el tamaño de la matriz original, el tamaño del filtro y el tamaño del paso o zancada, que suele ser 1 (Albawi et al., 2017a).

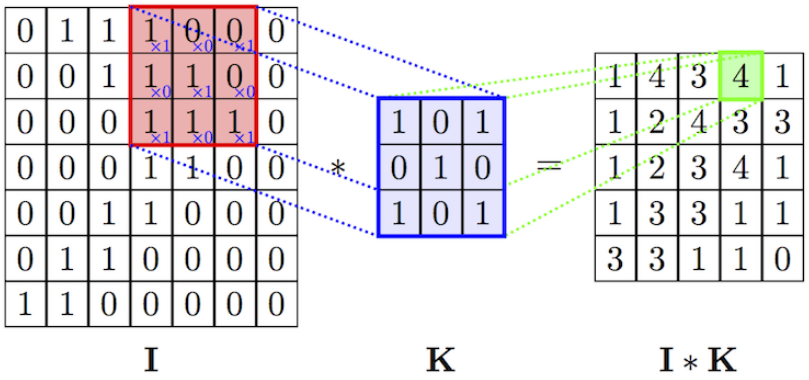

Figura 2.6: Operación de convolución

Por tanto, el tamaño de la matriz resultante vendrá dado por la ecuación que se muestra en la expresión 2.4.

$$O = 1 + \frac{I - K}{S} \tag{2.4}$$

Donde O es el tamaño de la matriz resultante, I el tamaño de la matriz de entrada, K el tamaño del Kernel y S el número de saltos que se avanza en cada iteración del filtro sobre la matriz de entrada.

Tras obtener la capa convolucionada, se aplica una función de activación (frecuentemente ReLU) para romper esa linealidad. Una vez se ha aplicado dicha función de activación, se aplica la capa de *pooling* que consiste en una reducción de la resolución de las características obtenidas anteriormente. Finalmente, se insertan una o varias capas totalmente conectadas antes de producir la salida (Albawi et al., 2017a).

Como se aprecia en la figura 2.7, una CNN se basa en que a partir de una matriz de entrada, se aplican una serie de convoluciones y *pooling* para finalmente resolver el problema que se plantea.

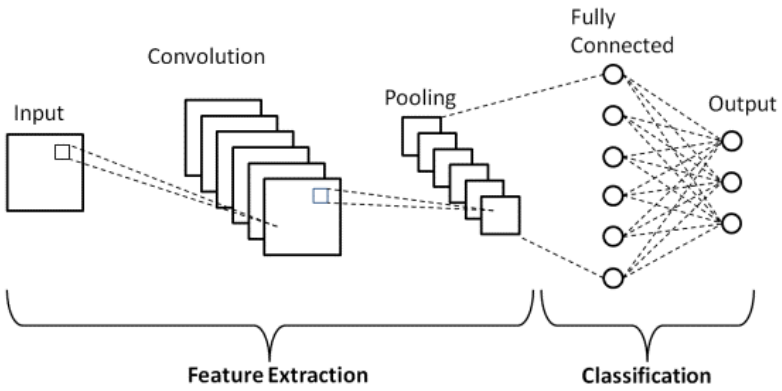

Figura 2.7: Arquitectura de una red neuronal convolucional

La idea del *pooling* es hacer una reducción del muestreo con el objetivo de reducir la complejidad para las capas siguientes. En caso de que se trabaje con imágenes, podría considerarse como una reducción de la resolución. En realidad, existen diferentes tipos de *pooling* entre los que destaca el *maxpooling*, el cual particiona la matriz sobre la que trabaja en subsecciones y obtiene el valor máximo de dicha subsección (Albawi et al., 2017b).

Respecto a la capa totalmente conectada de la parte final de la figura 2.7, se comporta como si fuera una capa más de una RN tradicional, tal y como se ha explicado en la sección Red neuronal artificial 2.3. El mayor inconveniente de estas capas es que incluyen multitud de parámetros que requieren de mucho tiempo y potencia computacional para ser calculados. Por lo tanto, se intenta eliminar algunos de estos nodos y conexiones con técnicas como el *dropout*[3]. Este tipo de técnicas de optimización se ha aplicado con éxito en proyectos como *LeNet* y *AlexNet*, ya que se tratan de redes con millones de parámetros en las que se requiere mantener la complejidad computacional contenida (Albawi et al., 2017b).

2.4. Autoencoders

Los Autoencoder (AE) son las arquitecturas indicadas para resolver problemas de eliminación de ruido (Bengio et al., 2006) y, por tanto, las que más se mencionan a lo largo del trabajo junto a las redes convolucionales, explicadas en la sección Red neuronal convolucional 2.3.3

Partiendo de la premisa anterior, los AE son un tipo de RN cuyo cometido principal es ser capaz de reconstruir una salida a partir de un conjunto de datos comprimido. Esto se traduce en que la salida, idealmente, será la misma que la entrada.

Principalmente, estas estructuras consisten en una RN (sección 2.3) entrenada para reconstruir la entrada. Además, han demostrado una superioridad considerable en aprender la representación elemental de las entradas en muchas aplicaciones diferentes, tales como el reconocimiento de imágenes, visión por computador y reconocimiento del habla (Zhang et al., 2020). Gracias a que las características de estos problemas pueden ser aprendidas, el AE es capaz de reconstruir, con gran solvencia, una imagen ruidosa

[3]El *dropout* también se utiliza para evitar el sobre ajuste una RN a un conjunto de datos.

en una sin ruido (Gondara, 2016).

Para ser capaz de aprender la representación elemental de una entrada dada y poder reconstruirla, un AE consta dos partes bien diferenciadas:

1. **Encoder**: es la primera parte de la red y se encarga de comprimir la entrada en un espacio denominado espacio de variables latentes.

2. **Decoder**: es la segunda parte de la red y su función es la reconstrucción total de la entrada. Para realizar dicha reconstrucción de forma eficaz es necesario haber recolectado información previamente.

Por tanto, la arquitectura de un AE se podría representar mediante la siguiente figura:

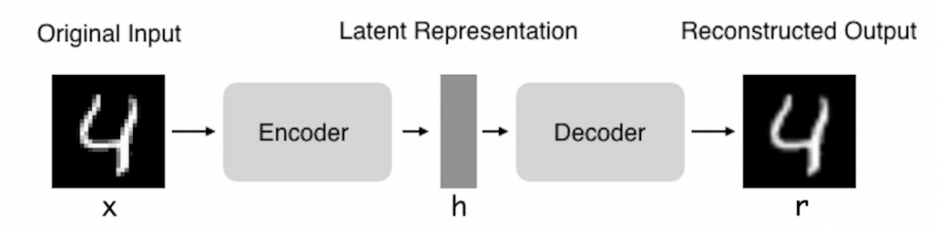

Figura 2.8: Estructura de un autoencoder

Donde x representa la entrada, h representa la función codificación $h = f(x)$ y r representa la función de decodificación $r = g(h)$

Es importante hacer notar que ambas partes del AE (encoder y decoder), son RN independientes y, por tanto, pueden funcionar por separado. Esto significa que es posible entrenar un encoder para ejecutar una tarea específica e incrustarlo en otro AE diferente (Bank et al., 2020).

Existen diferentes tipos de AE, entre los que destacan los AE regularizados, los AE dispersos y los AE variacionales. Estos últimos son modelos generativos que intentan describir la generación de datos mediante modelos probabilísticos (Bank et al., 2020).

2.4.1. Autoencoders Convolucionales

Este tipo de AE combinan los beneficios de las CNN (sección 2.3.3) y del entrenamiento no supervisado de los AE (sección 2.4). En este caso, a diferencia de las capas totalmente conectadas que se observan en los AE tradicionales, el encoder contiene

capas convolucionales y el decoder contiene capas deconvolucionales. En consecuencia, cada capa deconvolucional debe ir seguida de una capa *unpooling* (Seyfioğlu et al., 2018).

Tal y como se puede apreciar en la figura 2.9, el AE convolucional mantiene la forma del AE tradicional (figura 2.8) pero, en la parte del encoder, se comprime la entrada concatenando operaciones de convolución y *maxpooling*, mientras que en la parte del decoder, se aplican operaciones de deconvolución y *upsampling*.

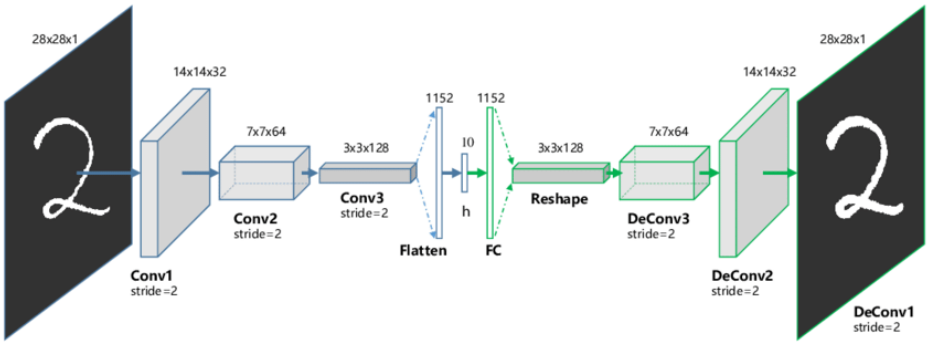

Figura 2.9: Estructura de un autoencoder convolucional

En la parte de la izquierda de la figura 2.9 (encoder) se visualizan las capas convolucionales junto a las funciones de activación y capas de *maxpooling*. En la parte de la derecha (decoder), se observan nuevamente capas convolucionales unidas a las respectivas funciones de activación pero, en este caso, se aplica *upsampling* para intentar reconstruir la representación de la muestra desde el espacio latente (Gondara, 2016).

Cabe destacar que en este estudio se han empleado convoluciones *1-D* debido a la naturaleza de las ondas y señales. Sin embargo, el concepto teórico no cambia, ya que la operación de convolución se aplica de la misma forma, es decir, el kernel se desliza a lo largo de la señal de entrada aplicando el mismo tipo de operación, solo que en vez de ser una matriz bidimensional es un vector unidimensional.

Esta arquitectura de RN, es la que mejores resultados ha ofrecido de todas las que se han probado y la que más se mencionará en el trabajo. Para dar más riqueza a los experimentos realizados, se han probado diferentes agrupaciones de los datos para esta arquitectura y se ha comparado con los métodos tradicionales anteriormente detallados: MA (2.1 Media móvil), la TW (2.2 Transformada Wavelet), MLP (2.3.1 Perceptrón multicapa) y las RN recurrentes (2.3.2 Red neuronal recurrente).

Capítulo 3

Tecnologías

En esta sección, se detallan las principales herramientas, tecnologías y entornos que se han empleado para poder implementar los experimentos que se detallan en la sección 4 Metodología.

3.1. Python

El uso tan extendido del lenguaje de programación Python en Inteligencia Artificial (IA) y DL se debe principalmente a la enorme cantidad de *frameworks* y librerías disponibles para el área de aprendizaje automático y análisis de datos. Algunas de las librerías más potentes y utilizadas en Python para tareas de inteligencia artificial son: NumPy, Pandas, SciPy y SciKit entre muchas otras. Estas librerías son especialmente potentes debido a que las operaciones que se ejecutan por debajo están implementadas en el lenguaje C de forma que se reduce considerablemente el coste temporal de los bucles y las formas de acceso a memoria. Además, la posibilidad de realizar dichos cálculos con tarjeta gráfica otorga una flexibilidad y potencial muy elevados[1]. Estas librerías son la base de frameworks más potentes, como TensorFlow o Pytorch.

3.2. Librerías externas

Para poder hacer uso de algunos de los algoritmos explicados en la sección Estado del arte 2, se han hecho uso de algunas de las librerías mencionadas en la sección

[1]Las tarjetas gráficas son capaces de realizar el procesamiento paralelo de multiplicaciones de matrices (la base de las redes neuronales) de forma muy rápida.

superior 3.1 y que se apoyan sobre otras librerías más elementales. Concretamente, se ha usado:

- **Pandas v1.4.2**: esta librería permite la carga de datos desde ficheros externos y efectuar operaciones y transformaciones sobre los mismos.
- **Matplotlib v3.5.1**: permite la generación de gráficos a partir de listas o *arrays* de NumPy.
- **SciPy v1.8.0**: esta librería ofrece multitud de algoritmos matemáticos para álgebra lineal, optimización o procesamiento de señales. De aquí se ha importado la técnica de la MA (2.1 Media móvil).
- **SciKit Image v0.19.2**: se trata de una librería que contiene una serie de algoritmos para el procesado de imágenes. La TW aplicada (2.2 Transformada Wavelet) se ha importado de aquí.
- **tf.Keras v2.8.0**: permite el desarrollo de redes neuronales profundas (sección 2.3.1) en poco tiempo y de manera modular y extensible. Además, se ejecuta sobre TensorFlow v2.8.0.

3.3. Entorno de desarrollo y ejecución de pruebas

En esta sección, se describe el entorno y las condiciones técnicas sobre las que se ha desarrollado el trabajo. En este caso, se ha utilizado la versión 3.8.13 de Python.

3.3.1. Conda

Para la gestión de las dependencias se ha empleado Conda. Esta herramienta permite la creación de entornos virtuales en Python para gestionar las versiones de las librerías y poder aislarlas de otros proyectos que se tengan en curso.

3.3.2. Máquina de cómputo

La máquina de cómputo se trata de un equipo dotado del siguiente hardware:

- CPU: Intel Core i-7 4770

- RAM: 16GB DDR3
- GPU: GTX 1060 6GB VRAM

Gracias a la tarjeta gráfica, se han podido reducir los costes temporales de entrenamiento de los algoritmos en gran medida.

Capítulo 4

Metodología

4.1. Conjunto de datos

Tal y como se ha comentado en la sección 1 Introducción, para la obtención de los datos experimentales se ha empleado un espectrofotómetro con transformada de Fourier Nicolet iS50 (Thermo Scientific) en configuración de reflexión total atenuada. En esta configuración, se usa una ventana de elevado índice de refracción y se trabaja con un ángulo de incidencia superior al ángulo crítico, de forma que la totalidad de la radiación IR es reflejada en la interfase entre la ventana y la muestra. Una pequeña onda evanescente en dicha interfase interacciona con la muestra, produciendo la variación en la señal objeto del análisis posterior. Esta configuración permite trabajar con muestras acuosas que, de otro modo, presentarían una elevada absorbancia en la región del infrarrojo y obligarían a trabajar con capas muy finas de muestra. En el presente caso, se ha empleado un cristal de seleniuro de cinc hemicilindrico como ventana y se ha ajustado el ángulo de incidencia a 45^{o} mediante un accesorio de reflectancia especular modelo Veemax de Spectra Tech. En este caso, se coloca la muestra entre el interferómetro y el detector y, dado que la muestra absorbe ciertas longitudes de onda de la luz, el interferograma contiene el espectro de la fuente menos el espectro de la muestra.

Se analizaron disoluciones acuosas de dos sustancias: urea y acetato. Estas sustancias se eligieron por ser relativamente comunes, accesibles y poseer bandas intensas en la región del espectro entre 1000 y 1800 cm^{-1}. Se analizaron disoluciones de ambas sustancias a tres concentraciones, 5×10^{-3} M, 50×10^{-3} M=0.05 M y 500×10^{-3} M=0.5 M.

La forma frecuente de operar del espectrofotómetro es acumular múltiples interferogramas para mejorar la relación señal ruido. Habitualmente, se adquiere primero un conjunto de interferogramas para una muestra de referencia (en el caso de disoluciones acuosas, la referencia será simplemente el agua) y después, se adquieren los interferogramas correspondientes a la muestra. El software del espectrofotómetro transforma automáticamente ambas señales en un espectro, que es la representación de la intensidad de radiación trasmitida en función de la frecuencia o longitud de onda de la señal[1].

Para adquirir los datos, que se han usado en el presente estudio, se registraron primero 108 interferogramas usando agua como referencia. Estos interferogramas se transformaron automáticamente en espectros y se almacenaron en formato de texto. A continuación, se reemplazó la muestra por la disolución de la sustancia elegida y se registraron de nuevo otros 108 interferogramas que se almacenaron como nuevos 108 espectros. La transmitancia se calcula como fracción de radiación trasmitida mediante la ecuación que se muestra en la expresión 4.1:

$$Transmitancia = \frac{I_s}{I_{ref}} \tag{4.1}$$

Alternativamente, es habitual trabajar con absorbancias, calculadas como se puede apreciar en la expresión 4.2:

$$\begin{aligned} Absorbancia &= -\log(Transmitancia) = \\ &= -\log\left(\frac{I_s}{I_{ref}}\right) = -\log\left(1 - \frac{I_{ref} - I_s}{I_{ref}}\right) \\ &\simeq \frac{I_{ref} - I_s}{I_{ref}} = 1 - Transmitancia \end{aligned} \tag{4.2}$$

De esta forma, si a una longitud de onda la muestra no absorbe nada de la radiación, la transmitancia será la unidad y la absorbancia será cero. Por el contrario, si la muestra absorbe parte de la radiación, la transmitancia será menor que la unidad y la absorbancia será mayor que cero. En lo que sigue los espectros se muestran en el modo

[1]Un interferograma representa la intensidad de la señal en función de la posición del espejo en el interferómetro (cm), mientras que el espectro representa la intensidad en función de la frecuencia (cm^{-1}). La transformación de una a otra señal se realiza mediante la transformada de Fourier y es realizada automáticamente por el software del espectrofotómetro

de absorbancia.

La adquisición de cada espectro (interferograma) llevó un tiempo de aprox. 0.2 s, mientras que la adquisición de los 108 espectros llevó un tiempo de 20 s.

4.1.1. Características principales

Los datos en crudo se componen de un conjunto de ficheros correspondientes a dos sustancias: acetato y urea. Por cada una de las sustancias se tienen 3 concentraciones diferentes y, para cada una de las concentraciones, se posee el espectro de referencia y su respectivo espectro de muestra; son 108 parejas de espectros muestra-referencia por cada concentración de cada sustancia. En la figura 4.1 se puede ver un desglose de la estructura de los datos de la urea. Para el caso del acetato sería exactamente igual.

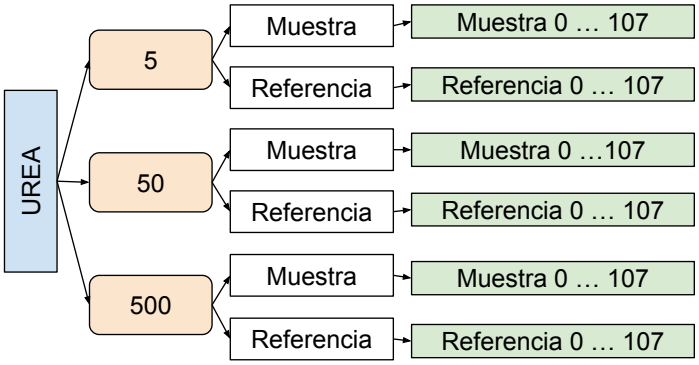

Figura 4.1: Estructura de la urea

Es importante hacer notar que, por ejemplo, la muestra 0 de la concentración 5 de la urea corresponde con la referencia 0 de la concentración 5 de la urea y cada fichero de referencia y de muestra contiene dos columnas: la frecuencia y la intensidad de la señal.

4.1.2. Preprocesado de los datos

Tal y como vienen los datos no es posible cargarlos directamente en la RN, es necesario aplicar una serie de procesos sobre los mismos para eliminar ciertas frecuencias y emparejarlos de manera que sea sencillo trabajar con ellos.

En primer lugar, se cogen los 800 valores que se sitúan en el intervalo de frecuencia

$[794, 3876]$[2] para cada uno de los ficheros de forma que, se crea un único fichero que contiene todos los valores de la señal para cada una de las referencias y para cada una de las muestras de cada concentración y de cada elemento en el rango de frecuencias anterior. La tabla muestra un posible ejemplo de una fila del conjunto de datos preprocesado.

ID	Elemento	Concentracion mN	Tipo	Muestra
a2_H2O0060.CSV	urea	50	references	[29.1455, 29.51303 ...]

Tabla 4.1: Fila de ejemplo del conjunto de datos original ya preprocesado.

Por tanto, este nuevo fichero y que será sobre el que se apliquen las transformaciones necesarias y que se explican en la sección 4.3, contendrá tantas filas como ficheros totales de muestras y referencias existan.

En la figura 4.2 se observa un ejemplo de la sustancia acetato en concentración 500 mM.

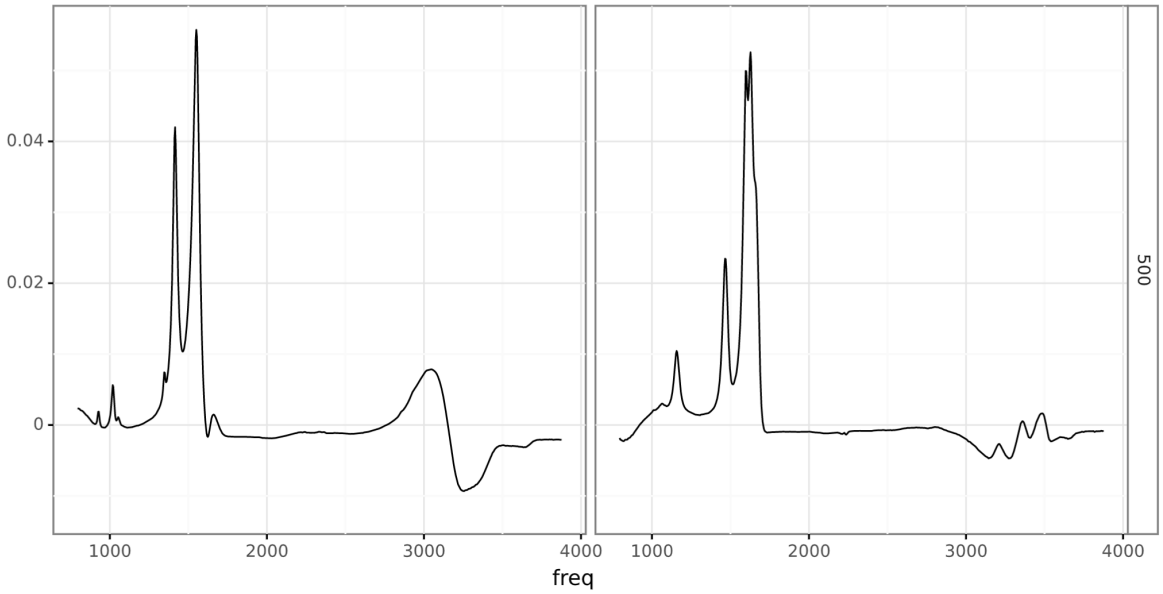

Figura 4.2: Señal de acetato (izquierda) y urea (derecha) con concentración 500 mM

Donde el eje de abscisas indica la frecuencia o longitud recíproca[3] en cm^{-1} y el eje de ordenadas indica la intensidad de la señal. A continuación, se detalla como se realiza la carga de los datos a la RN, que estructura presentan las redes y la forma en la que devuelven los resultados.

[2]El resto de las frecuencias no aportan valor a la muestra.

[3]El centímetro recíproco, cm^{-1} es una unidad de energía igual a la energía de un fotón con una longitud de onda de 1 cm.

4.2. Implementación

Como se ha comentado en la sección 4.1.2 Preprocesado de los datos, los datos no pueden ser cargados a las redes neuronales directamente, hay que aplicar una serie de transformaciones para que la red pueda trabajar con ellos. Antes de explicar las transformaciones aplicadas, se detalla la estructura de las redes que se han empleado.

4.2.1. Red neuronal

Como se ha comentado en la sección 2.3.1 Perceptrón multicapa, en este caso se ha empleado una RN profunda. Esta red posee 5 capas: la capa de entrada, 4 capas ocultas y la capa de salida que se conecta directamente con la última de las capas ocultas. En el diagrama de la figura 4.3 se puede ver el desglose de las capas de las que está dotada la red.

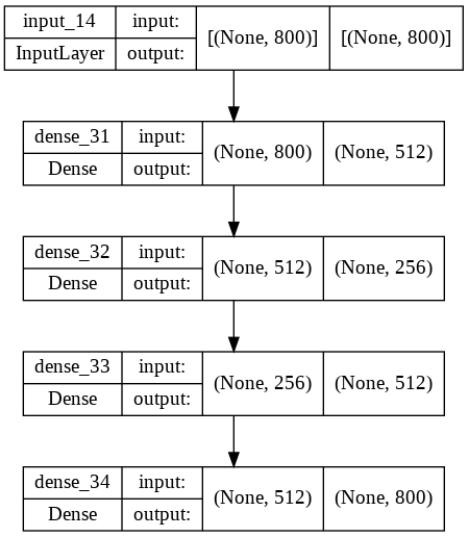

Figura 4.3: Red neuronal profunda empleada

Como se puede apreciar en la figura 4.3, la red tiene una entrada de un vector de longitud 800, exactamente la misma que se referencia en la sección 4.1.2 que son las mismas componentes que tiene cada señal. De la misma forma, la capa de salida tiene 800 neuronas, una para cada una de las componentes del vector.

Por otro lado, en cuanto a las funciones de activación, se ha hecho uso la función ReLu (ecuación 4.3). Esta ecuación suele emplearse en redes convolucionales y redes

neuronales profundas.

$$f(x) = max(0, x) \tag{4.3}$$

Esta RN produnda es una red genérica que no ha dado buenos resultados y se ha empleado únicamente con fines comparativos con otras redes más complejas que se detallan a continuación.

4.2.2. Autoencoder LSTM

En este caso se presenta una combinación de una RN LSTM (sección 2.3.2.1) y un AE (sección 2.4). La finalidad de este tipo de RN es ver si los datos presentaban alguna característica que hiciera predecible el ruido. Es decir, comprobar si existía un orden temporal en las muestras tomadas y de esa forma provocar que la red fuera capaz de aprender dichas características. Esta características de las redes LSTM junto con la capacidad de los AE de reconstruir una salida a partir de un conjunto de datos comprimido, lo hacía un candidato a obtener resultados interesantes.

En el caso del AE LSTM, se lee la señal paso a paso y la representación interna almacenada se intenta reconstruir en la parte del decoder. Tal y como se puede ver en la figura 4.4, la estructura de la red consta 1 capa de entrada y una de salida, así como 3 capas para la parte del encoder y 2 capas para la parte del decoder. Además, cuenta con una capa de repetición que repite el vector de entrada n veces.

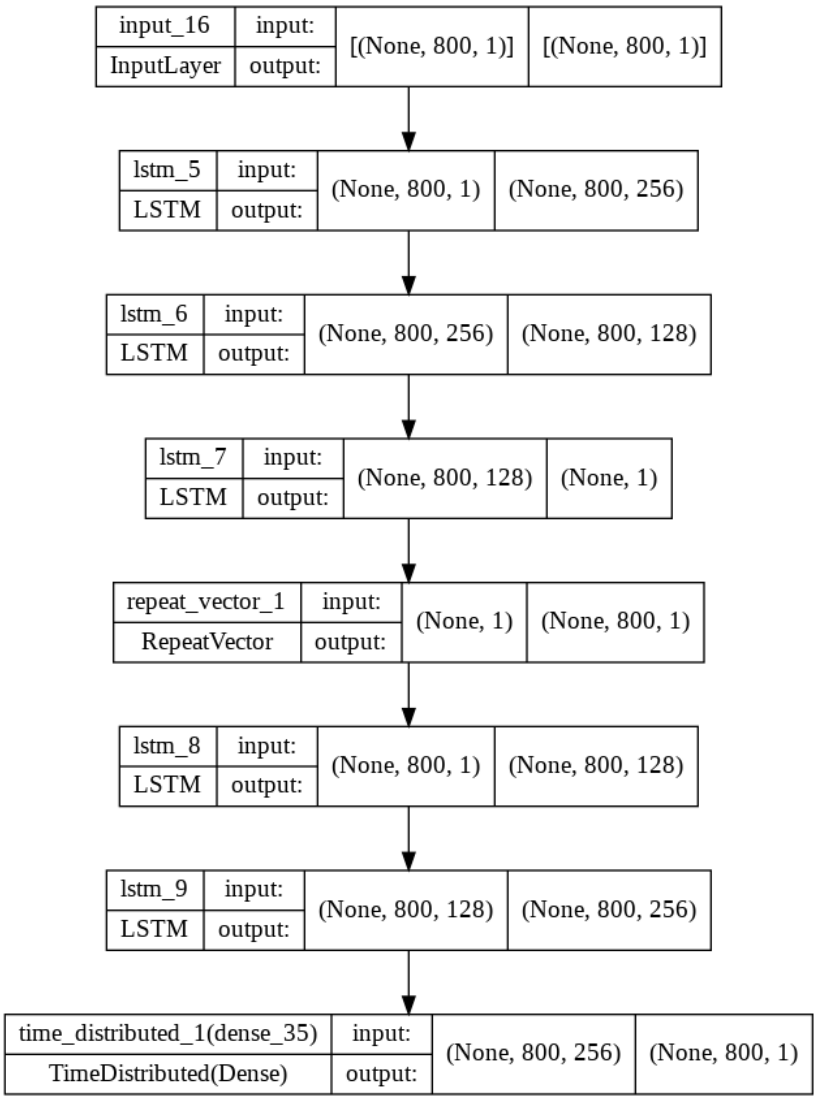

Figura 4.4: Autoencoder long short-term memory empleado

El AE LSTM no ha ofrecido buenos resultados debido a que el ruido presente en las muestras de urea y acetato no presentan una naturaleza temporal que pueda ser predicha por una RN LSTM. Como se ha mencionado en el resumen y en la introducción (1 Introducción), el ruido del conjunto de datos es inherente a la propia muestra y también es sensible a la propia electrónica del instrumento con el que se mide, de modo que la red podría llegar a aprender esas variables y no generalizar bien en entornos con diferente casuística o sustancias.

A continuación, se detalla la implementación de la estructura de RN que mejores resultados ha dado, sobre la que se han probado diferentes agrupaciones de los datos y cuyas métricas se analizan y detallan en las secciones 5 y 6, respectivamente.

4.2.3. Autoencoder Convolucional

En la figura 4.5 se presenta la estructura del AE convolucional empleado. Tal y como se ha explicado en la sección 2.4.1, un AE convolucional no es más que un AE convencional (sección 2.4) en el que en la parte tanto del encoder como del decoder se tiene una CNN (sección 2.3.3)

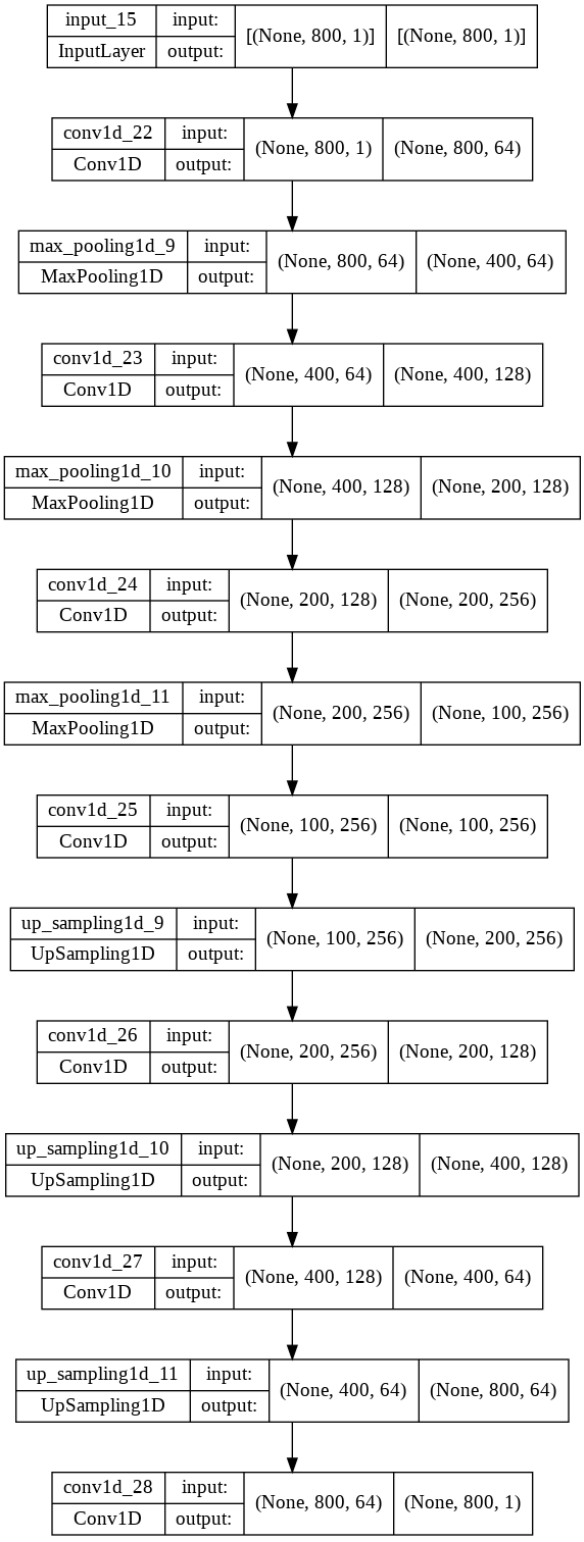

Figura 4.5: Autoencoder convolucional

Tal y como se puede apreciar en la figura 4.5, el AE convolucional consta de la capa de entrada con 800 neuronas, mismo número que valores presentes en cada una de las señales de entrada. A continuación, se presenta la primera CNN (encoder) compuesta por 3 capas convolucionales unidimensionales intercaladas con 3 capas *maxpooling*

unidimensionales. Las capas convolucionales unidimensionales se emplean para poder procesar la señal en la que el kernel de dicha convolución se mueve a lo largo del eje X. Por tanto, las capas convolucionales unidimensionales, así como las capas *maxpooling* unidimensionales necesitan unos datos de entrada bidimensionales.

Seguidamente, en la capa del decoder, se observan nuevamente 3 capas convolucionales intercaladas con 3 capas de *upsampling*, todas unidimensionales también. En la última capa del decoder se tiene una capa convolucional unidimensional con un tamaño de salida igual que en la entrada: $(800, 1)$. Finalmente, con ambas partes independientes (encoder y decoder), se construye el modelo formado por ambos trozos.

Tal y como se ha mencionado, los mejores resultados en la fase previa a la experimentación se han obtenido con esta clase de RN. Es decir, los resultados fueron notablemente superiores al resto de arquitecturas mencionadas en la sección 2. En consecuencia, esta arquitectura con dos agrupaciones diferentes de datos, es la que ha sido comparado contra los dos métodos clásicos analizados en la sección del estado del arte: MA (sección 2.1) y TW (sección 2.2)

A continuación, se va a explicar como se ha realizado la carga de los datos y los dos tipos de agrupación empleadas para entrenar y probar el AE convolucional.

4.3. Carga y agrupación de los datos

Para poder alimentar la RN, se debe ser capaz de poder transferir los datos de nuestra tabla ya preprocesada (4.1) a la RN. Para ello, el primer paso es conseguir cargar esos datos en memoria para, posteriormente, transformarlos e introducirlos en la red. Para poder realizar esta carga de los datos en memoria a partir de un fichero .csv, se ha empleado la librería Pandas. La estructura de este fichero .csv es la que se detalla en la sección 4.1.2 por lo tanto, lo más relevante es el grupo elemento-concentración al que pertenece cada una de las muestras así como la propia muestra. Por tanto, ahora el conjunto de datos se agrupa en 6 categorías: 5-acetato, 50-acetato, 500-acetato y 5-urea, 50-urea y 500-urea.

Por consiguiente, para generar estos grupos se ha de recorrer el conjunto de datos por elemento y por concentración de forma que se seleccionan aquellos índices del conjunto de datos que comparten el mismo grupo elemento-concentración. La finalidad

de realizar esta agrupación concentración-elemento no es otra que evitar que señales de otras concentraciones u otros elementos interfieran en el proceso de entrenamiento o de test. Es decir, que cuando se le presenta a la red una muestra de urea-50 a su vez se le enseñe como es una muestra de urea-50 filtrada, no una muestra de urea-5 o urea-500 ya que presentan morfologías diferentes. Por ejemplo, en la figura 4.6 se puede ver una de las señales de acetato-5 (izquierda) contra una de las señales de acetato 50 (derecha).

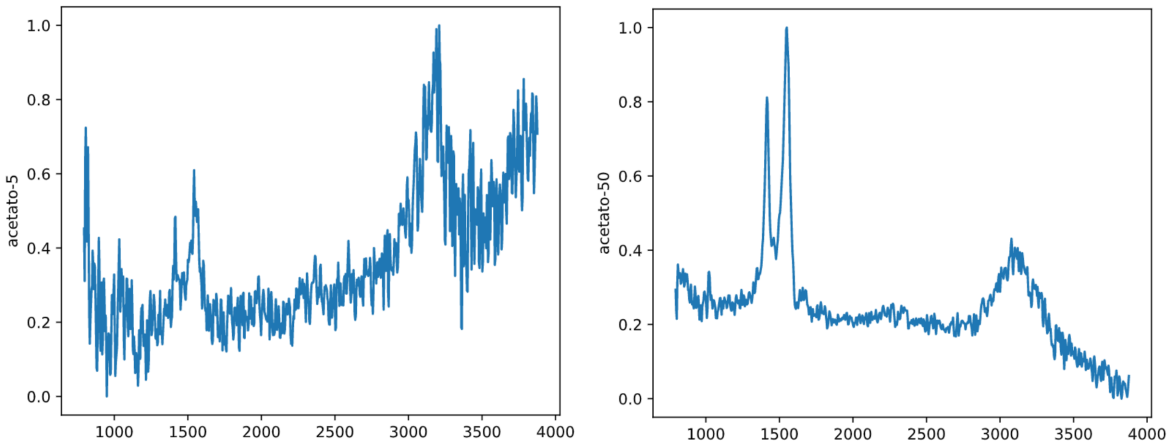

Figura 4.6: Acetato-5 vs acetato-50

Una vez que ya se tiene el conjunto de datos de entrada agrupado para poder realizar el entrenamiento correctamente, se le debe asignar la salida esperada a cada entrada. Tal y como se ha comentado en la introducción (sección 1), en este estudio no se dispone de una señal verdadera con la que entrenar a la red. Por lo tanto, se ha tenido que sintetizar una salida ejemplo para la red. Para realizar dicha síntesis, se ha aplicado la media de todo el conjunto global de señales por elemento-concentración. Es decir, los datos se agrupan por elemento-concentración y se aplica la media al conjunto de muestras que pertenecen a un grupo determinado. Es por ello que, en el conjunto de datos, se almacenan 6 salidas de referencia en total (2 elementos con 3 concentraciones cada uno). En la tabla 4.2 se puede ver un ejemplo de una de las filas de entrenamiento de la tabla resultante de la agrupación concentración-elemento.

ID	Elemento	Concentracion mM	Muestra	Grupo	Min	Max	Salida
0	urea	5	[0.889, 1.0, 0.95...]	urea-5	-0.01	-0.006	[0.87, 0.86, 0.84 ...]

Tabla 4.2: Fila de ejemplo del conjunto de entrenamiento.

Donde las columnas *Min* y *Max* son el valor más pequeño y más grande dentro

de la muestra original, respectivamente. En estas condiciones, ya se tiene el conjunto de datos preparado para entrenar las redes neuronales en los que se explican los dos métodos aplicados, pero antes, se detalla el normalizado en el rango 0-1 que se ha aplicado debido a la notoria mejora de los resultados y que se discute en la sección 5.4.1

4.3.1. Normalizado 0-1

Debido a la propia naturaleza de las redes neuronales, los datos suelen ser normalizados antes de entrenarlas. Esto permite un entrenamiento y una convergencia más rápida. En el caso contrario, cuando se utilizan datos sin normalizar las funciones de activación pueden quedarse fijas en zonas planas del dominio de búsqueda de resultados provocando que la red no converja o que converja de manera errática y lenta (Bhanja and Das, 2018).

En este caso, se ha elegido normalizar las señales en el rango $[0-1]$ ya que con este rango la convergencia es más rápida y el entrenamiento más eficiente. Normalizar en el rango $[0-1]$ implica que el valor máximo de la muestra valdrá 1 y el valor mínimo valdrá 0. La ecuación 4.4 muestra la forma en la que se ha calculado esta normalización.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4.4}$$

Donde la x representa el valor a normalizar, $min(x)$ el valor mínimo del rango y $max(x)$ el valor máximo del rango.

Dicha normalización $[0-1]$ se aplica para cada uno de los valores de cada una de las señales de las que se dispone y se calcula en el momento en el que se procesan los datos. No obstante, también se conserva la muestra original para poder realizar la comparación de resultados. La comparativa de los resultados obtenidos de normalizar las muestras de entrada se discute también en la sección 5 Experimentación.

4.3.2. Agrupación según la media

En este punto, ya se poseen los conjuntos de entrenamiento y test. A continuación, se va a explicar como se han introducido estos conjuntos a la red según la media.

Para realizar el agrupado según la media, los datos se cargan a la red de la misma

forma que se han procesado anteriormente, es decir, se evalúan los modelos con los 6 promedios obtenidos para cada concentración de cada elemento, pero se le aplica un redimensionado a los mismos antes de introducirlos. Dicho redimensionado se aplica porque todos los datos de entrenamiento y de test contienen todas las muestras relativas al grupo que se pretende entrenar, lo que implica que tanto las entradas como las salidas del entrenamiento presentan una forma de $(28422, 800)$ y, como se ha comentado en la sección 4.2.3 Autoencoder Convolucional, la entrada de la red deben ser vectores de tamaño $(800, 1)$.

En la figura 4.7 se aprecia la estructura del entrenamiento del AE convolucional según la media de los datos. En primera instancia, se dividen por sustancia y por concentración de modo que para cada una de las concentraciones se genera la pareja formada por la muestra de la sustancia y la sustancia de referencia. Esta muestra y la sustancia de referencia se juntan de manera que cada una de las muestras de los espectros se combina con cada una de las muestras de referencia y pasa al conjunto de datos. Para realizar dicha combinación se aplica la ecuación 4.5.

$$c = (y - x)/y \tag{4.5}$$

Donde y es el valor del espectro de referencia (H_2O) en una frecuencia determinada y x el valor del espectro de muestra en la misma frecuencia. Dicha combinación se realiza por cada valor de muestra por cada valor de referencia.

A continuación, se unifican los datos obtenidos resultantes de aplicar los procesos de redimensionado y combinación en los conjuntos $Datos_1 Datos_2 ... Datos_n$ hasta completar todas las sustancias de las que se dispongan, en este caso solo urea y acetato.

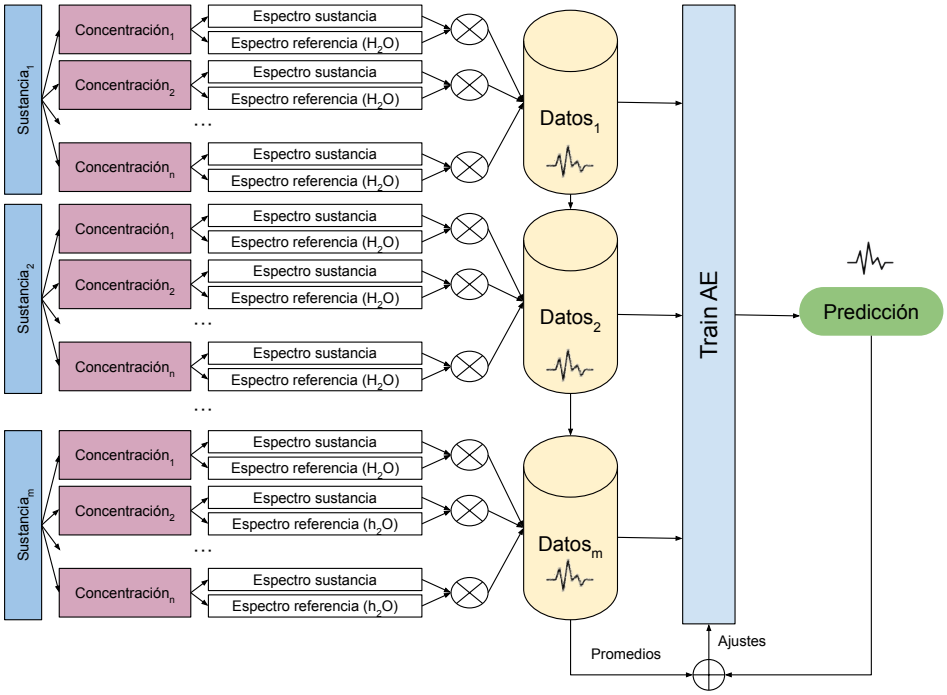

Figura 4.7: Esquema de entrenamiento según la media

Cada uno de estos bloques de datos $Datos_1 Datos_2 ... Datos_n$ serán los que se empleen para entrenar el AE convolucional y generar predicciones a partir de ellos. Además, se usarán para evaluar cómo de bien ha ido el entrenamiento. Destacar que, para evaluar cómo de bien ha funcionado el entrenamiento del modelo, se emplean los promedios obtenidos de los conjuntos de datos mencionados anteriormente. Como se puede ver en la figura 4.8, se realizan las predicciones de todos los conjuntos de datos de forma unificada.

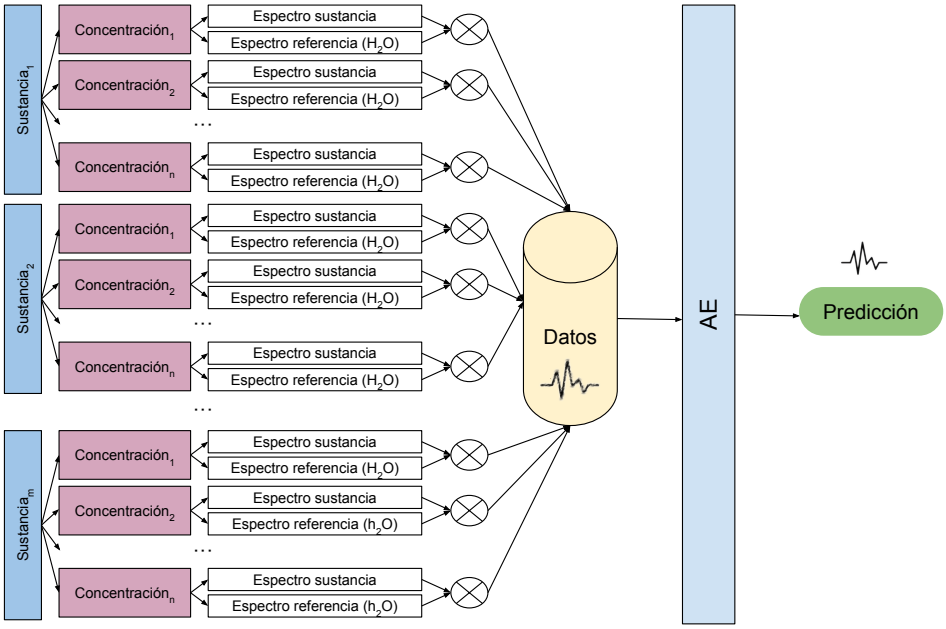

Figura 4.8: Esquema de test según la media

Una vez se han realizado todas las predicciones del AE que se acaba de entrenar, se van almacenando en una tabla externa y al término de los entrenamientos de los modelos se genera un documento con muestras filtradas de ejemplo. Por ejemplo, en la figura 4.9, se visualiza una señal de acetato-5 contra su señal filtrada mediante la agrupación por media. La señal de la derecha es la señal resultante de aplicar el AE entrenado sobre las muestras que se han preparado para test. Las métricas y análisis de resultados se discuten más adelante en la sección 5.

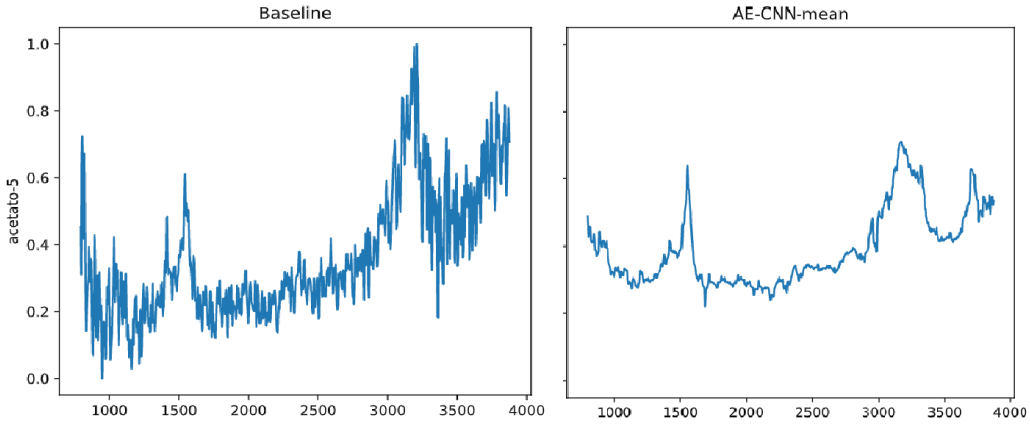

Figura 4.9: Acetato-5 (izquierda) y su filtrado mediante agrupación por media (derecha)

Las predicciones generadas por el modelo se almacenan al terminar el entrenamiento

en un conjunto de datos aparte que, posteriormente, se utilizará para poder visualizar cómo de bien está filtrando el ruido el modelo.

4.3.3. Agrupación por parejas concentración-elemento

En esta agrupación, la intención es crear parejas para el entrenamiento del AE entre los ejemplos del mismo grupo, es decir, se crean parejas para las muestras que son de la misma concentración-elemento. A continuación, cuando ya se ha entrenado el modelo, se ha de medir lo bien que ha funcionado comparando la predicción del modelo con la media total del grupo. Es decir, la fase de test sería la misma que la explicada en la sección 4.3.2 y cuyo esquema se aprecia en la figura 4.8, no así la parte del entrenamiento.

En esta aproximación, el preprocesado previo para crear el conjunto de datos de entrenamiento y de test es el mismo que se ha explicado en 4.3. Sin embargo, se generan parejas de elementos del mismo grupo. Por ejemplo, para el grupo urea-5 se selecciona como entrada para el entrenamiento una de las muestras aleatoriamente, de modo que se le asigna una salida también aleatoria dentro del mismo grupo urea-5. Se ha de recordar que las salidas de una muestra concreta para grupo concentración-elemento, es la media de todas las señales de ese mismo grupo concentración elemento. Como se puede ver en la figura 4.10, dada una concentración de una sustancia determinada se seleccionan n parejas que es un parámetro que se ha llamado de manera interna *batch* y al que se le ha asignado el valor 42. Este parámetro, marca el tamaño de los bloques de parejas que se abarcan en cada iteración.

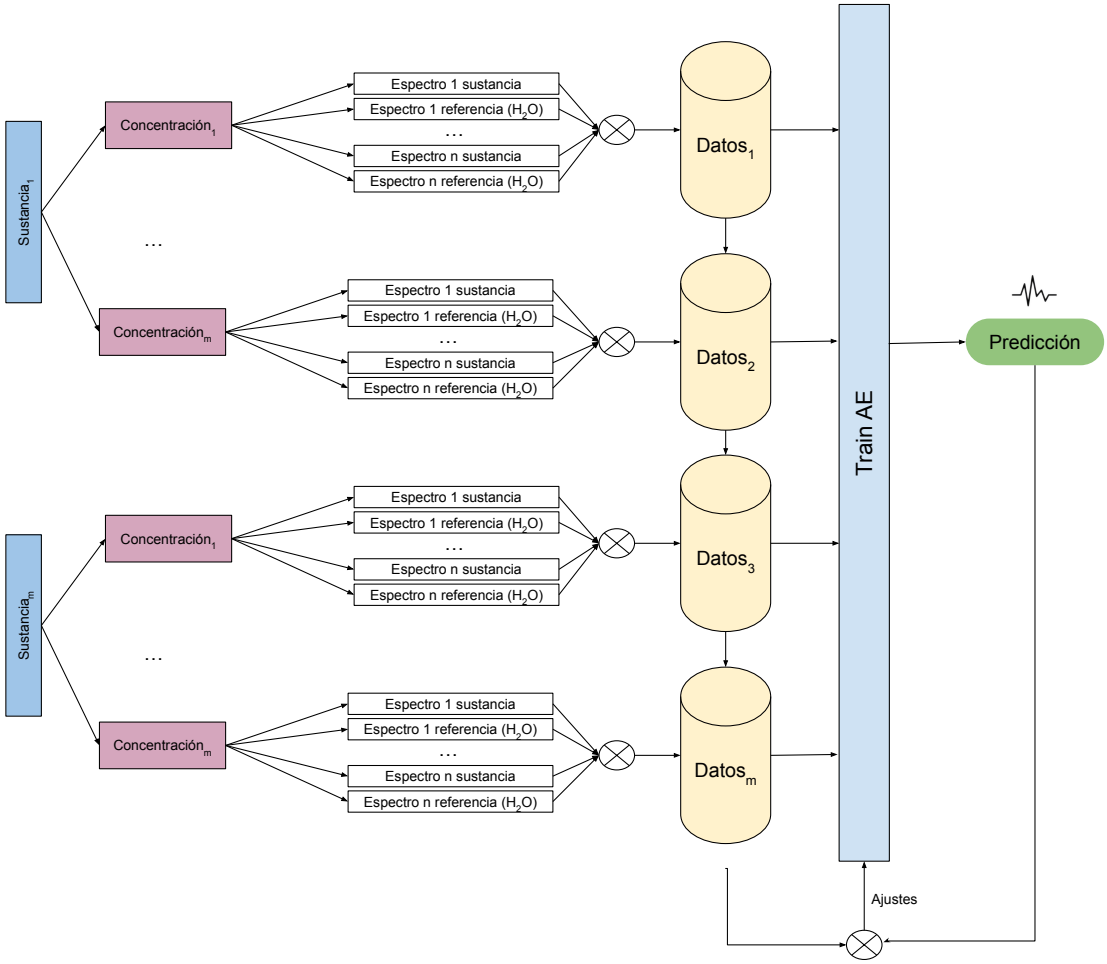

Figura 4.10: Esquema de entrenamiento por pares

Estas parejas contienen la entrada de una muestra aleatoria y una de las salidas aleatorias. Cuando el tamaño del *batch* se cumple, se devuelve el conjunto de parejas del mismo grupo concentración-elemento y se hace entrenar el AE con dicho conjunto. A continuación, se vuelve a repetir hasta que se agotan los elementos disponibles en el conjunto de entrenamiento.

Posteriormente, igual que en la aproximación anterior, se realizan las predicciones de las muestras al termino del entrenamiento del modelo y se guardan en una tabla para mostrarlo posteriormente. En la figura 4.11, se visualiza el filtrado del AE convolucional sobre una muestra de acetato-5.

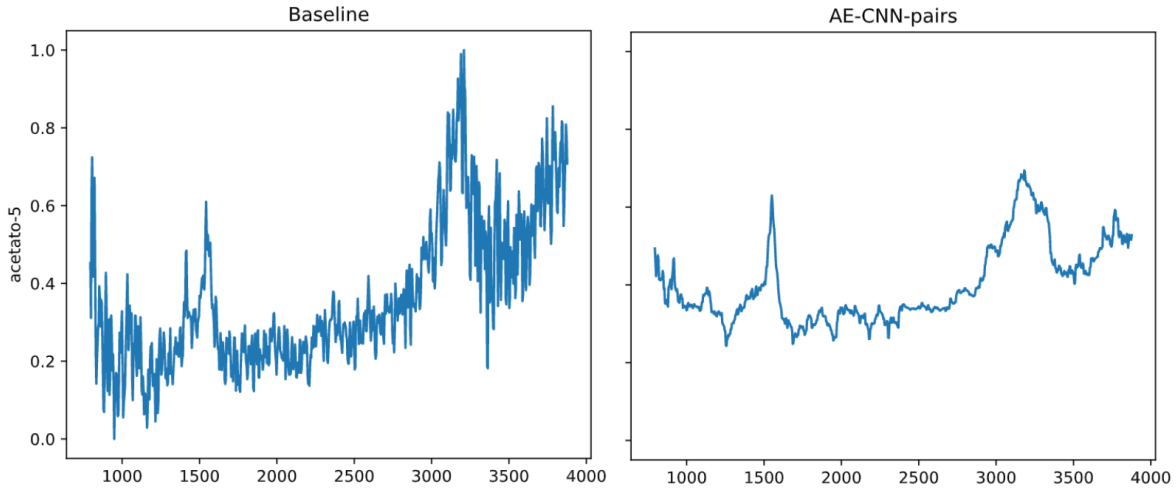

Figura 4.11: Acetato-5 (izquierda) y su filtrado mediante agrupación por pares (derecha)

A continuación, se explica la experimentación realizada y los resultados obtenidos con los conjuntos de datos anteriores. Para exponer los resultados, se han empleado diferentes gráficos y métricas que servirán para ilustrar cómo de bien han funcionado los modelos.

Capítulo 5

Experimentación

En la experimentación, se han empleado una serie de gráficos y métricas que resultan útiles para extraer conclusiones acerca de lo bien que se han ajustado los modelos propuestos al entrenamiento y cómo de bien predicen sobre el conjunto de test. Adicionalmente, también se han generado gráficas de algunas de las señales filtradas. Algunas de ellas se muestran en la sección Metodología 4.

5.1. Gráficos empleados

Con el objetivo de poder analizar los resultados obtenidos en los experimentos de una forma visual, se han seleccionado dos tipos gráficos diferentes: el gráfico de barras y el de violín. Cada uno posee unas virtudes y características que los hacen interesantes y que se detallan a continuación.

5.1.1. Gráfico de barras

El gráfico de barras es uno de los tipos de gráficos más simples que existen y consiste en una representación de un conjunto de datos por categorías de variable cualitativa y su frecuencia de aparición en una muestra.

En este caso, la métrica expresada en el eje de ordenadas es el error cuadrático medio. Esta métrica se analiza en la sección Error cuadrático medio 5.2.1.

En el caso del eje de ordenadas, se establece cada uno de los algoritmos aplicados de modo que cada barra representa el error medio cuadrático o, en inglés, Root Mean Squared Error (RMSE) obtenido tras la experimentación realizada para un algoritmo

concreto. En la figura 5.1, se muestra un ejemplo de gráfico de barras con los RMSE para cada uno de los métodos estudiados.

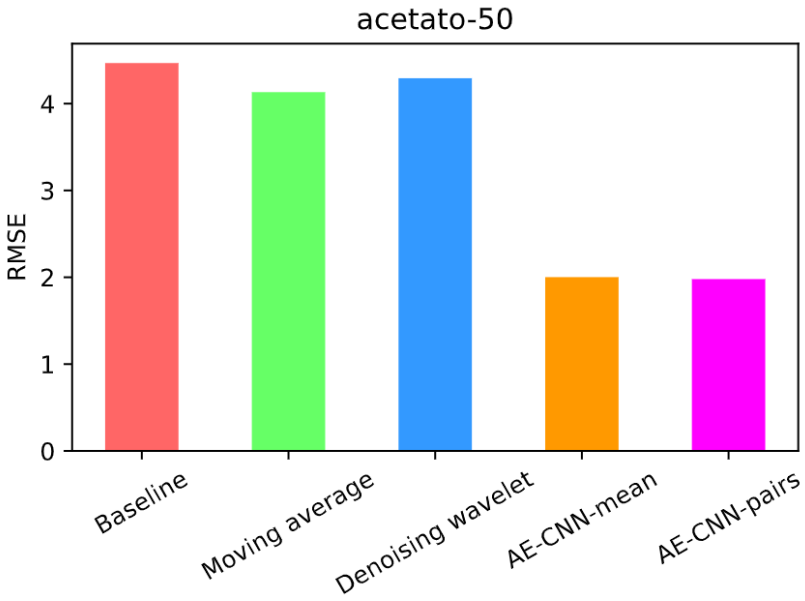

Figura 5.1: Ejemplo de gráfico de barras empleado para acetato-50.

5.1.2. Gráficos de violín

Los gráficos de violín son muy útiles para poder visualizar la distribución de los datos y su densidad de probabilidad. La forma característica de violín es gracias a que se trata de una combinación de un gráfico de cajas y bigotes junto a un diagrama de densidad. La barra gruesa del centro nos indica el rango intercuartílico y la barra negra fina que sale de la gruesa representa el 95 % de los intervalos de confianza. Adicionalmente, el punto blanco del centro representa la mediana de los datos.

En referencia al cuerpo del gráfico, es decir, la parte que rodea el diagrama de cajas, se puede apreciar la distribución de los datos. Una sección ancha del cuerpo del gráfico indica que hay una mayor probabilidad de que el valor que se obtenga de la experimentación tenga dicho RMSE.

A modo de ejemplo, en la figura 5.2 se muestra uno de los gráficos de violín obtenidos en uno de los experimentos realizados.

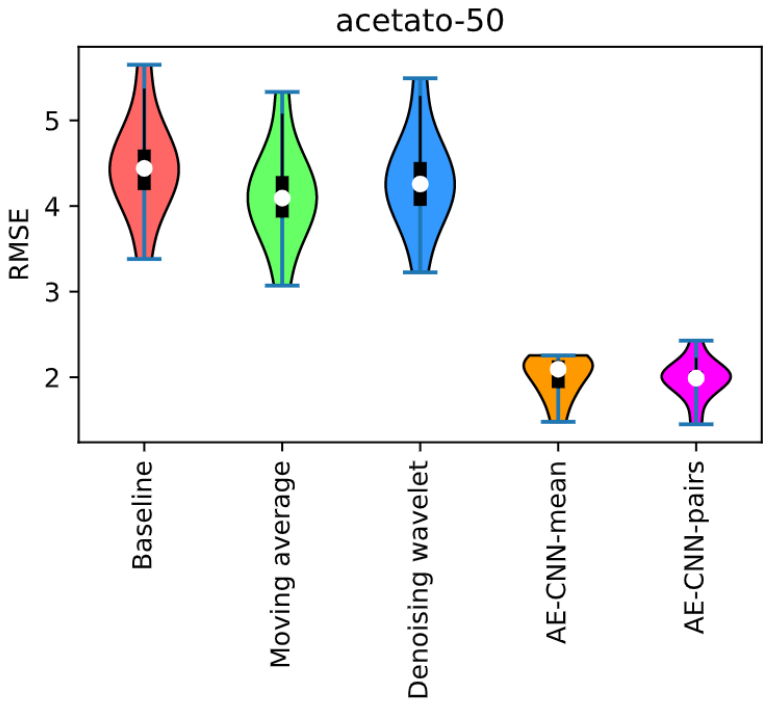

Figura 5.2: Ejemplo de gráfico de violín empleado para acetato-50.

5.2. Métricas empleadas

A continuación, se detallan las métricas que se han empleado para medir de forma cuantitativa como de bien se han comportado los modelos.

5.2.1. Error cuadrático medio

Para poder medir la cantidad de error que existen entre dos conjuntos de datos, como son las entradas de la red y sus respectivas salidas, se ha hecho uso del error cuadrático medio. Esta medida, cuya ecuación puede apreciarse en la figura 5.1, representa la raíz cuadrada de la distancia cuadrada promedio entre el valor real y el que el modelo ha predicho.

$$RMSE = \sqrt{\Sigma_{i=1}^{n} \frac{(\hat{P}_i - P_i)^2}{n}} \tag{5.1}$$

Donde $\hat{P}_1$, $\hat{P}_2$, ..., $\hat{P}_n$ representan los valores predichos y P_1, P_2, ..., P_n representan los valores reales u observados mientras que n es el número de observaciones. Es decir, esta medida nos indica como de cerca están los puntos de los datos reales y los predichos.

Esta medida siempre cumple la condición de $RMSE \geq 0$, por lo que un valor de 0 indicaría que efectivamente los datos reales y los predichos se ajustan a la perfección. Sin embargo, este hecho casi nunca se cumple en la práctica.

5.2.2. Entropía cruzada binaria

En aprendizaje automático, es necesario establecer una métrica de evaluación para que el modelo pueda aprender correctamente. El propósito de esta métrica es poder indicarle al modelo si está avanzando en la dirección correcta durante el proceso de entrenamiento y es conocida con el nombre de función de pérdida. La métrica que se ha seleccionado es la entropía cruzada binaria o en inglés *binary cross-entropy*.

Esta medida, (ecuación 5.2), compara cada una de las probabilidades predichas con el resultado real, que bien puede ser 0 o 1 en el caso de una clasificación binaria. A continuación, calcula la puntuación que penaliza las probabilidades en función de la distancia al valor esperado. Esto significa lo cerca o lejos que está del valor real.

$$C = -\frac{1}{N}\sum_{i=0}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i)) \tag{5.2}$$

Donde p_i representa la probabilidad de que la muestra i pertenezca a la clase 1 mientras que y_i representa la muestra actual.

5.2.3. Test de rangos con signo de Wilcoxon

Cuando se realiza un mismo experimento con diferentes modelos, resulta interesante poder verificar si alguno de los modelos se ha comportado mejor que otro o, por el contrario, son equivalentes. Para poder resolver este problema, se ha empleado el test de rangos con signo de Wilcoxon.

Esta medida (ecuación 5.3) determina si existen diferencias entre el rango medio de dos muestras relacionadas, como pueden ser los resultados de las evaluaciones de los modelos propuestos (Wilcoxon, 1992).

Para poder obtener la métrica en cuestión, se parte de dos suposiciones:

1. Si $z_i = y_i - x_i$, entonces los valores de z_i son independientes.

2. Los valores z_i tienen una distribución continua y simétrica respecto a una mediana

común θ.

Por lo tanto, la hipótesis nula $H_0 : \theta = 0$ aplicada a los datos obtenidos x_i, y_i en la experimentación en dos modelos diferentes nos indica que ambos resultados son iguales, es decir, ambos modelos se comportan igual (Wilcoxon, 1992).

Para verificar dicha hipótesis se ordenan los valores absolutos $|z_1|, ..., |z_n|$ y se les asigna su R_i de forma que la métrica de la prueba se obtiene según la expresión 5.3.

$$W^+ = \sum_{z_i>0} R_i \tag{5.3}$$

Donde R_i representa la suma de los rangos correspondientes a los valores positivos de z_i. La distribución obtenida W^+ nos muestra si se rechaza la hipótesis. Este valor a su vez nos ofrece un parámetro p que nos indica con que probabilidad uno de los conjuntos de datos es mejor que otro (Wilcoxon, 1992).

5.3. Validación cruzada

Cuando se entrenan modelos de DL pueden existir sesgos en el conjunto de datos con los que se trabajan. Por lo tanto, puede darse el caso de que se seleccione una partición entrenamiento-validación sesgada y afecte al resultado. Para cerciorarse de que no se produce dicha situación, el experimento se repite varias veces con diferentes particiones de los datos con las mismas proporciones de observaciones con una salida dada.

La solución que ofrece la librería *ScikitLearn* nos permite que cada una de las particiones entrenamiento-test sea una selección aleatoria del conjunto de datos disponible y además, mantiene un porcentaje equivalente del par entrada-salida en cada partición para evitar que una de las particiones contenga más pares de una clase determinada. Esta solución es conocida con el nombre de *StratifiedKFold* con la que se han generado 10 particiones ($n\_splits = 10$) y seleccionado de forma aleatoria.

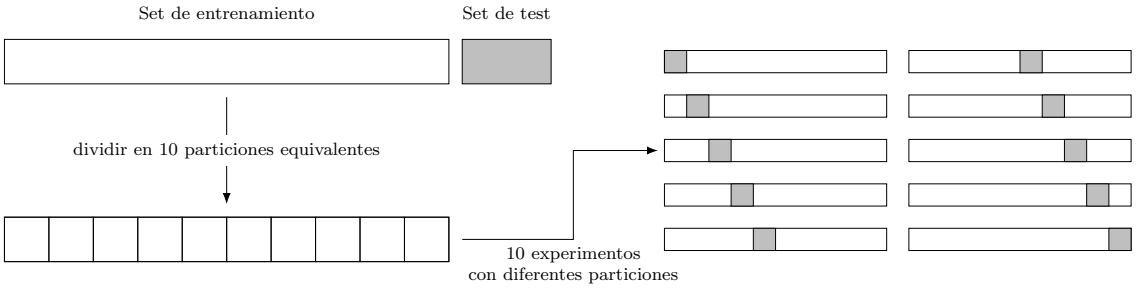

Figura 5.3: Esquema de validación cruzada 10-CV.

El objetivo principal de aplicar la validación cruzada, es evitar que el modelo se ajuste demasiado al conjunto de entrenamiento (conocido como *overfitting*). De esta manera, será capaz de generalizar mejor sobre datos que no ha visto durante la fase de entrenamiento.

5.4. Análisis de los resultados

A continuación, se detallan los experimentos realizados y los resultados obtenidos siguiendo los procedimientos explicados en la sección 4 Metodología.

La experimentación se ha ejecutado en cualquier caso para todos los algoritmos detallados en la sección 2 Estado del arte. Dicha experimentación se ha desglosado en dos: por un lado, se han evaluado los modelos ante datos normalizados y sin normalizar. Por otro lado, se han realizado experimentos con muestras ruidosas en la fase de entrenamiento con muestras menos ruidosas en la fase de test y viceversa, en ambos casos con datos normalizados.

5.4.1. Mejoras en los datos normalizados

Durante las fases iniciales del estudio, se pudo observar que los modelos de aprendizaje automático ofrecían resultados considerablemente peores que los modelos tradicionales. A la vista estaba que no eran capaces de filtrar el ruido correctamente. Tal y como se ha explicado en la sección Normalizado 0-1 4.3.1, el normalizado de los datos es una técnica que permite optimizar el entrenamiento y facilitar la convergencia de los modelos, por lo tanto, se ha aplicado un normalizado a los datos de entrada y se ha comparado con los mismos datos no normalizados.

En la figura 5.4, se puede apreciar los resultados para todas las concentraciones disponibles de la urea. Por ejemplo, las barras del grupo urea-5 representan el RMSE obtenido para cada unos de los algoritmos en el proceso de filtrado para dicho par elemento-concentración. En el caso del grupo urea-all, se trata de un grupo sintetizado que consiste en el sumatorio de los errores en cada concentración para cada algoritmo. Gracias a esto, se tiene una visión general de como se ha comportado el algoritmo para todas las concentraciones disponibles.

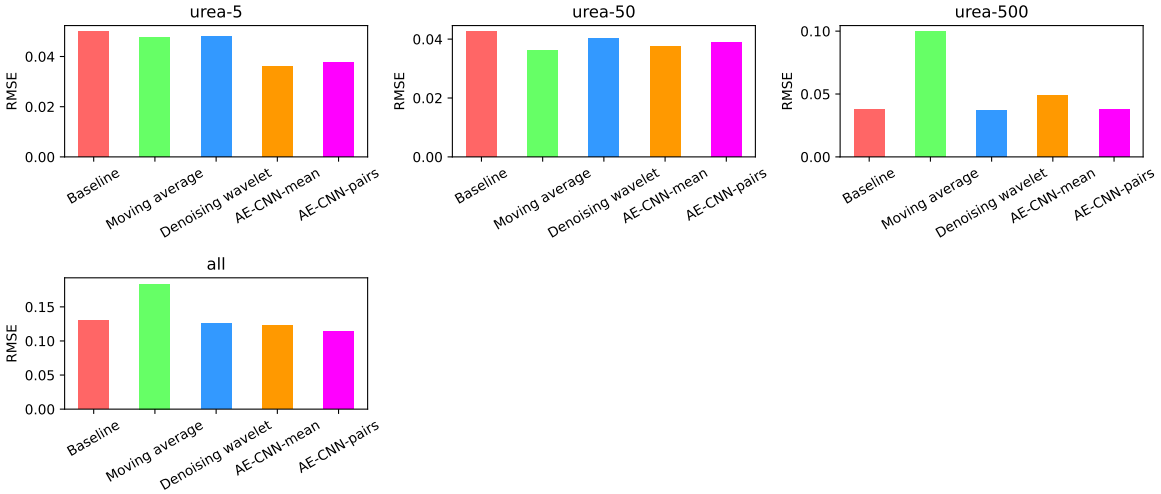

Figura 5.4: Resultados con datos no normalizados en urea.

El gráfico 5.4 ha sido obtenido a partir de los resultados devueltos por los modelos, habiendo sido entrenados y testeados únicamente con urea. La lectura del gráfico nos muestra que, en líneas generales, los modelos de aprendizaje automático han obtenido un RMSE inferior, comportándose ligeramente peor en el grupo urea-500.

Por otro lado, en la figura 5.5, se pueden observar los resultados obtenidos en las mismas condiciones que la figura 5.4 pero con el normalizado realizado en la sección 4.3.1.

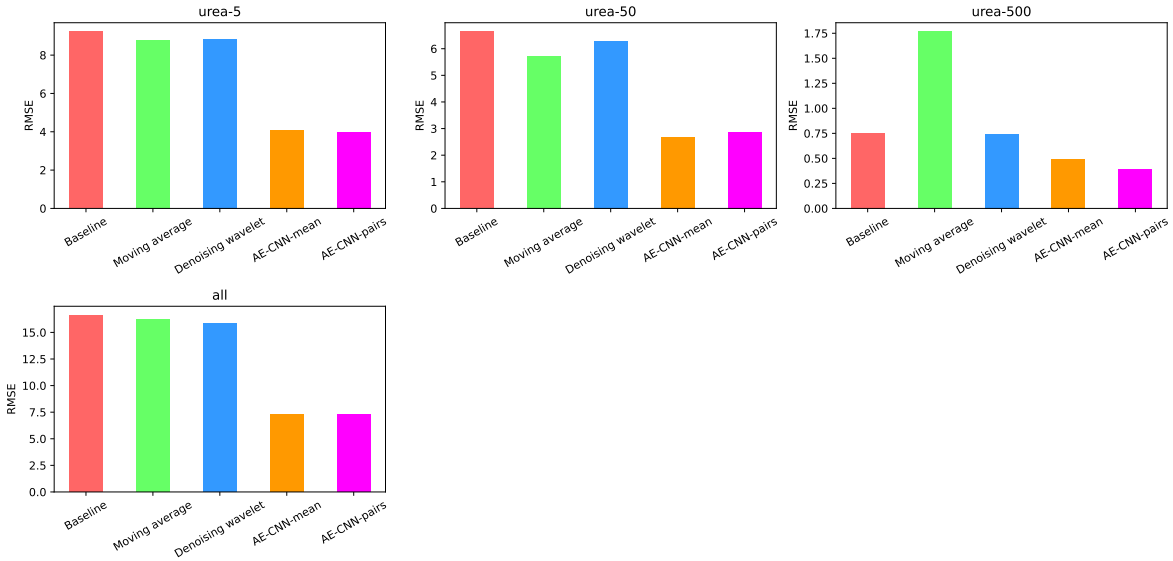

Figura 5.5: RMSE en barras con datos normalizados en urea.

En esta ocasión, cuando se introduce a la red los datos normalizados, se puede apreciar claramente que los resultados obtenidos por los modelos de aprendizaje son bastante superiores a los obtenidos por los métodos tradicionales.

Profundizando más en los resultados obtenidos y su distribución, se puede apreciar en la figura 5.6 que la distribución del RMSE obtenido para los métodos de aprendizaje automático se mantiene uniforme, mientras que para los métodos tradicionales se observa un gráfico más elongado, como es el caso de la TW para el grupo urea-50.

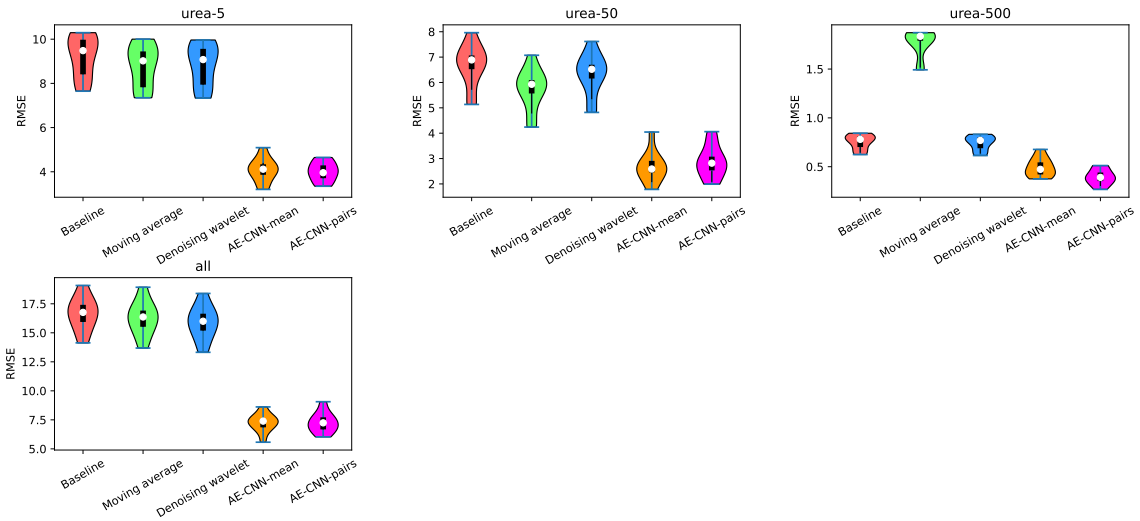

Figura 5.6: RMSE en violín con datos normalizados en urea.

No obstante, a medida que incrementa la concentración de la sustancia, la diferencia de error obtenida entre los métodos de aprendizaje automático y los tradicionales se

reduce. Sin embargo, es claramente apreciable que los métodos sugeridos en este estudio obtienen un error menor.

A continuación, se presentan las muestras filtradas por los modelos propuestos cuando se entrenan con datos normalizados.

En la figura 5.7, se puede apreciar la muestra original con ruido que se pretende filtrar y, a la derecha de la misma, los filtrados realizados por cada método estudiado.

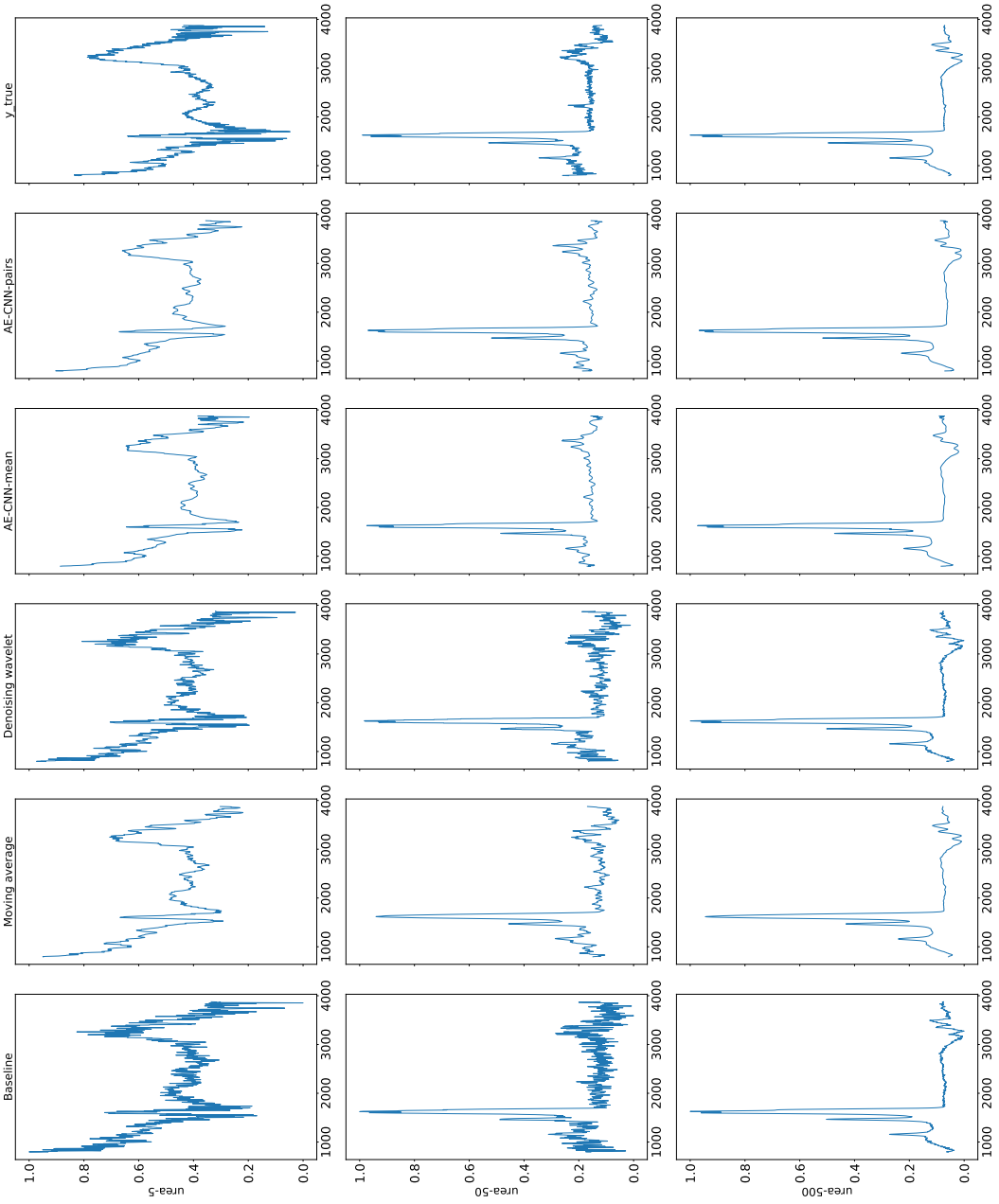

Figura 5.7: Ejemplos de filtrado con datos normalizados en urea.

Como se puede apreciar en la figura 5.7, los métodos tradicionales realizan un buen trabajo de filtrado sobre la señal original, destacando la MA por la línea tan fina que genera. Por otro lado, en cuanto a los modelos de aprendizaje automático, se puede ver claramente como se realiza el filtrado de la señal, sin embargo, generan una serie de picos y valles que no parecen estar sacados de la señal original. Por ejemplo, el grupo urea-50 para el modelo basado en la agrupación según la media 4.3.2 se observa como directamente suprime los picos ubicados entre las frecuencias situadas entre los $2000\,\mathrm{cm}^{-1}$ - $3000\,\mathrm{cm}^{-1}$.

No obstante, a la vista está que para el caso de la urea los dos modelos de aprendizaje automático propuestos filtran el ruido de la señal original. Sendos modelos y el resto de métodos también han sido aplicados al conjunto de acetato y sobre acetato y urea. En el caso del entrenamiento únicamente con acetato, los resultados son muy similares a los obtenidos a la urea, por lo tanto, se analiza continuación, los resultados obtenidos cuando se entrenan los modelos con urea y acetato.

En este caso, el RMSE obtenido en la evaluación de cada elemento y de cada concentración se puede apreciar en la figura 5.8.

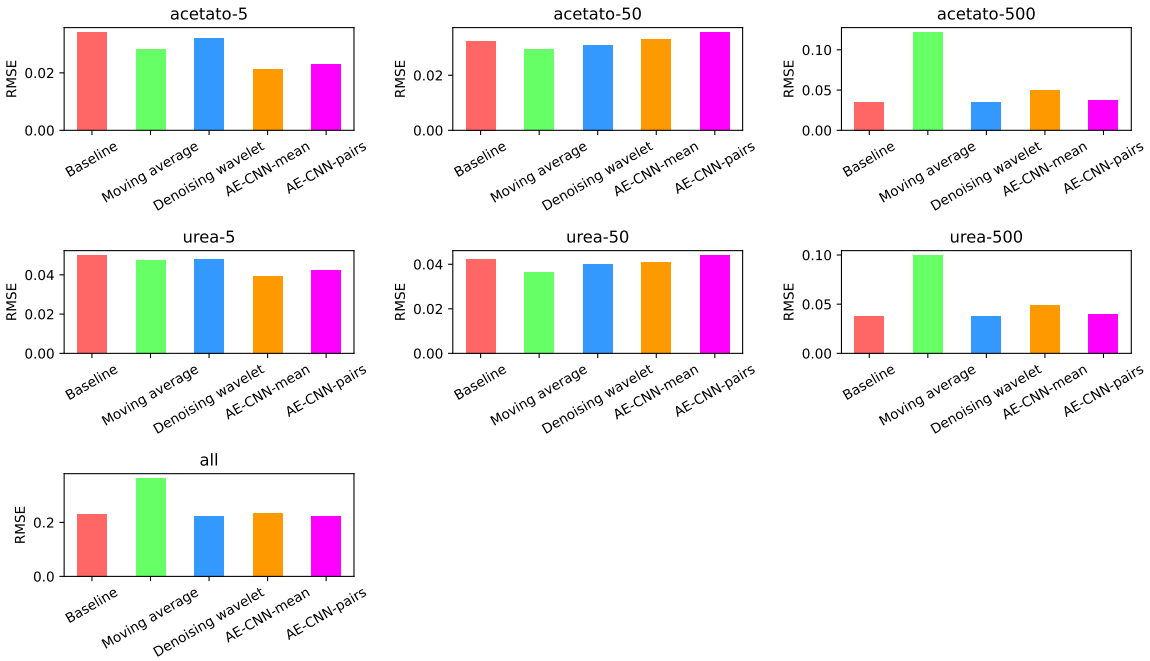

Figura 5.8: RMSE en barras con datos no normalizados en urea y acetato.

Siguiendo la línea de los resultados mostrados en la figura 5.4, se aprecia claramente que cuando no se normalizan los datos, los métodos propuestos no se comportan mejor

que los tradicionales. A continuación, se detallan los resultados obtenidos cuando se entrenan los modelos sugeridos con ambas substancias (acetato y urea) y se normalizan las entradas de las redes. En la figura 5.9, se aprecian los resultados obtenidos en todos los modelos para este caso en forma de barras.

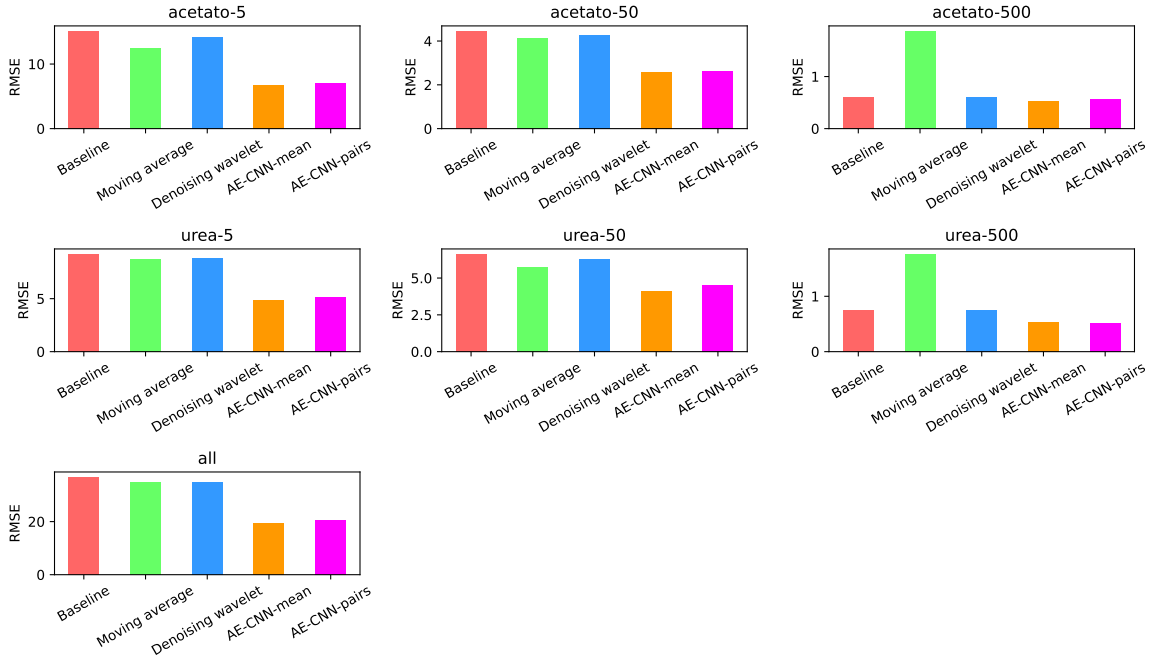

Figura 5.9: RMSE en barras con datos normalizados en urea y acetato.

Cuando normalizamos los datos y entrenamos los modelos propuestos con ambas substancias, los resultados no muestran grandes diferencias respecto a los resultados anteriores. Se puede apreciar claramente que los métodos propuestos ofrecen un RMSE inferior en comparación con los tradicionales. En cuanto al gráfico de violín 5.10, se observa un reflejo del gráfico anterior.

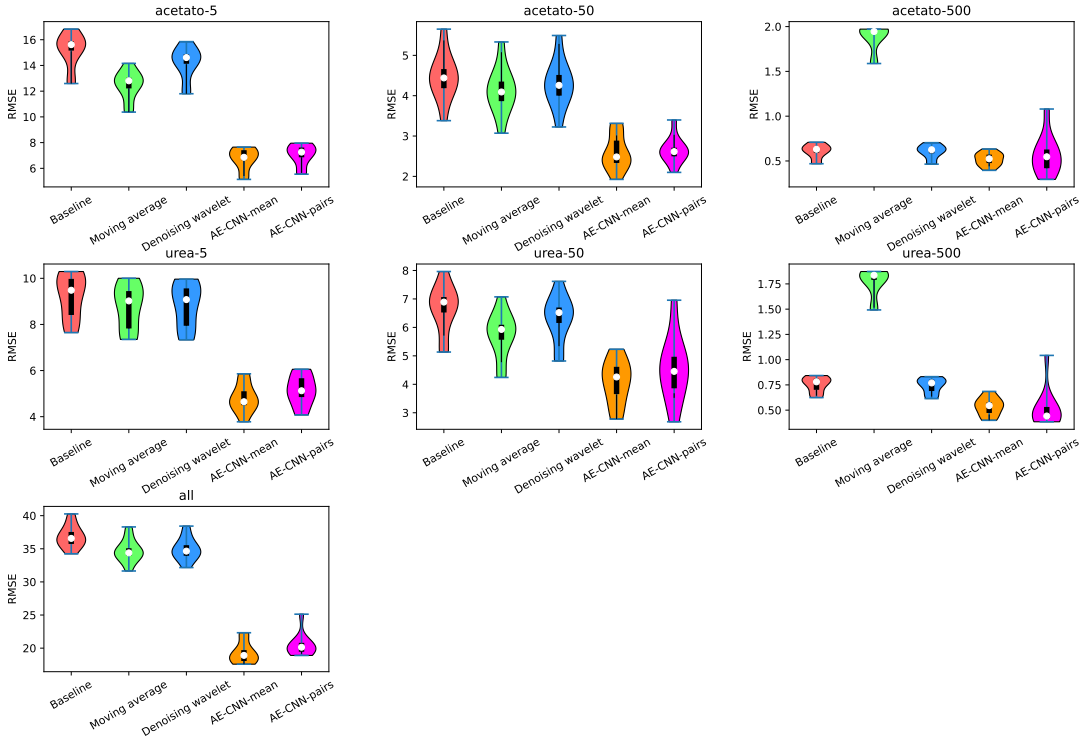

Figura 5.10: RMSE en violín con datos normalizados en urea y acetato.

Sin embargo, en urea-50 se observa como el AE convolucional basado en agrupación por pares presenta un rango de valores más amplio, generando valores atípicos que incluso superan la mediana obtenida en algunos de los métodos tradicionales.

Concluyendo con los gráficos asociados a los resultados obtenidos sobre los datos, se analiza el filtrado propiamente dicho sobre muestras reales cuando se entrenan los modelos con el conjunto completo de elementos y normalizado. En la figura 5.11, se visualiza el filtrado realizado por cada uno de los modelos sobre los diferentes grupos sustancia-concentración cuando se entrenan con urea y acetato.

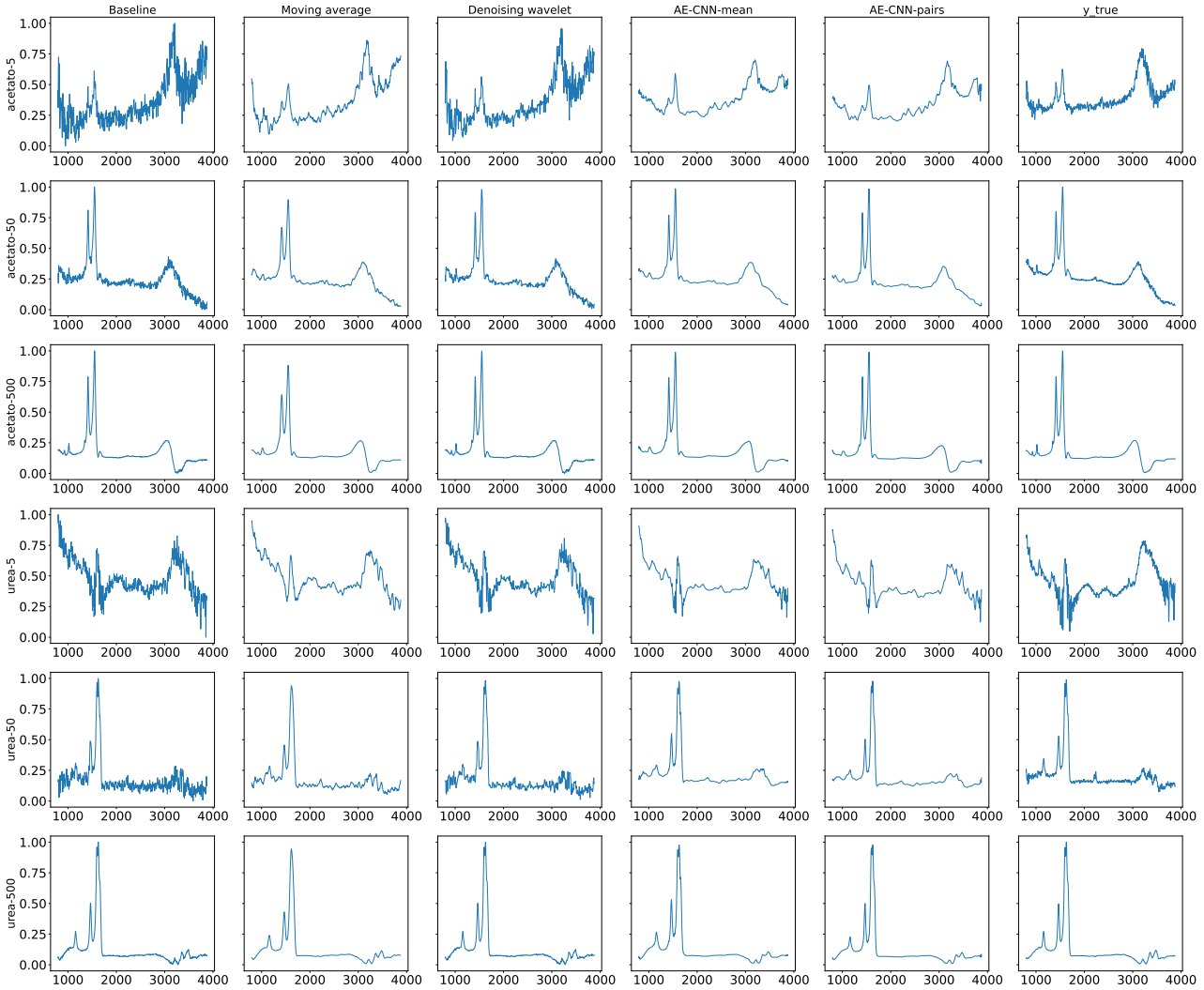

Figura 5.11: Ejemplos de filtrado con datos normalizados en urea y acetato.

Los resultados obtenidos se asemejan a los ya analizados en la figura 5.7. En el caso del acetato también se observa que los métodos de aprendizaje automático crean valles y picos en frecuencias que no existen en la señal original. Por ejemplo, para el caso de acetato-5 (primera fila de gráficos en la figura), se aprecia como en el modelo basado en agrupación según la media se genera un valle que no existe en la señal de referencia (alrededor de la banda de los $3500Hz$) y que el método de la MA tampoco contempla. Si bien se puede apreciar que la señal resultante posee una contribución de ruido inferior a las originales.

Basándose en los resultados anteriores de los datos normalizados, es interesante poder comparar los modelos 1:1 para verificar si ante el mismo conjunto de datos es mejor aplicar uno u otro o bien, los resultados son equivalentes. Para verificar dicha comparativa, se emplea el ya mencionado test de rango con signo de Wilcoxon (sección 5.2.3).

En la figura 5.12 se puede apreciar una comparativa de los algoritmos propuestos 1:1 para el filtrado de urea-50. El círculo verde indica que el algoritmo de la coordenada Y es mejor que el algoritmo de la coordenada X con una confianza $\geq 95\,\%$, en el caso de que sea amarillo indica una confianza $\geq 90\,\%$, blanco que es el mismo algoritmo y gris indica que no se está seguro de si un algoritmo es mejor que el otro.

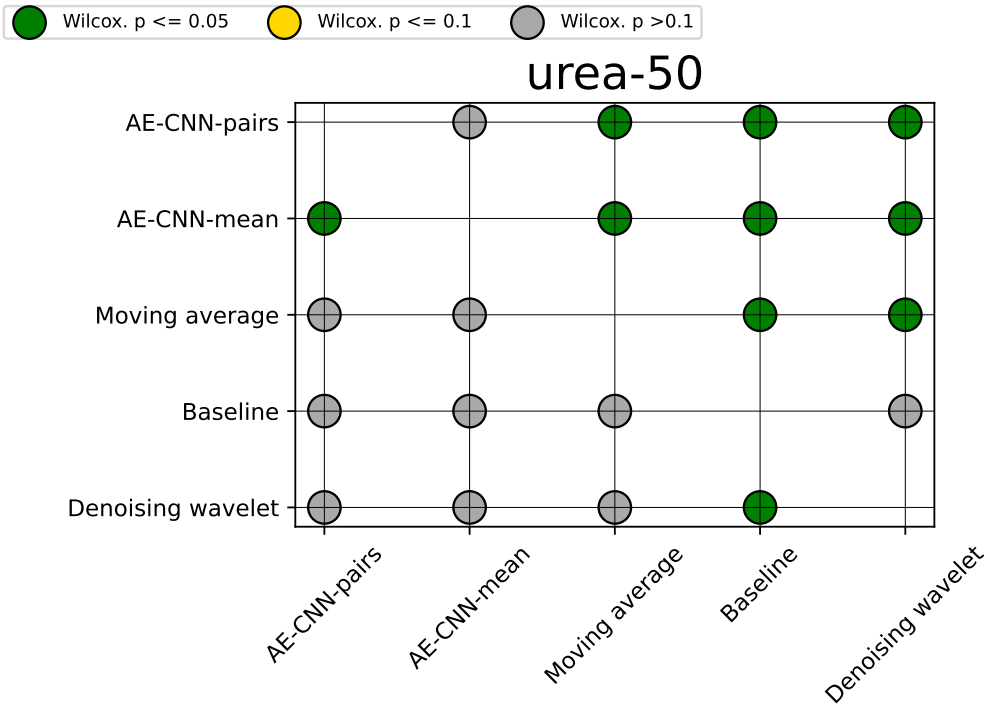

Figura 5.12: Test de Wilcoxon para urea-50.

Como se puede apreciar, los algoritmos de aprendizaje automático son mejores que todos los métodos tradicionales con una confianza mayor del 95 %. Incluso el algoritmo basado en agrupación por media, es mejor que el algoritmo basado en agrupación por pares. Por otro lado, resulta lógico que todos los algoritmos sean mejor que *baseline*, ya que es la muestra sin aplicar ningún filtro.

Este gráfico comparativo 1:1 supone una confirmación de los datos obtenidos en los gráficos 5.2 y 5.1 donde se puede ver que los métodos de aprendizaje automático ofrecen un RMSE menor y una distribución uniforme de los errores obtenidos para una validación cruzada de 10 *splits*.

No obstante, en la figura 5.13 no se observa una evidencia clara que los algoritmos de aprendizaje automático sean superiores en el filtrado de ruido que los tradicionales.

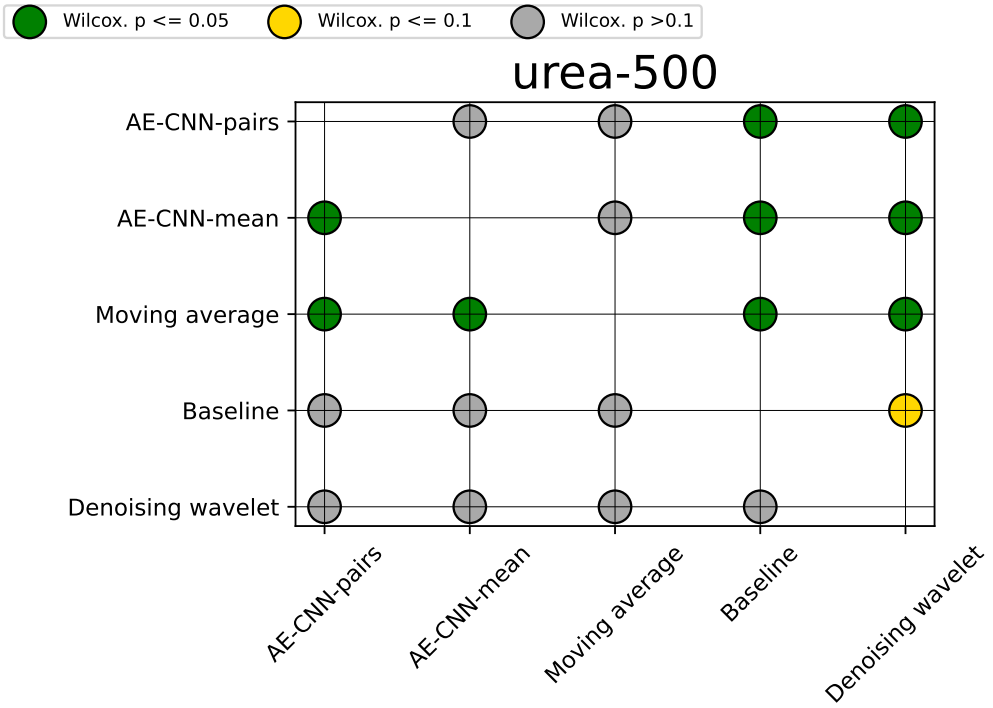

Figura 5.13: Test de Wilcoxon para urea-500.

De hecho, se puede apreciar que el test de Wilcoxon nos muestra que aplicar la técnica de la MA (2.1) es mejor que utilizar cualquiera de los dos métodos de aprendizaje automático propuestos. Esto es debido a que en concentraciones muy altas la red no es capaz de obtener el ruido de forma correcta sobre la muestra y acaba acoplando más ruido a la señal que la técnica de la MA.

Por último, en la figura 5.14 se visualizan uno a uno los algoritmos cuando son

entrenados con todos los elementos disponibles.

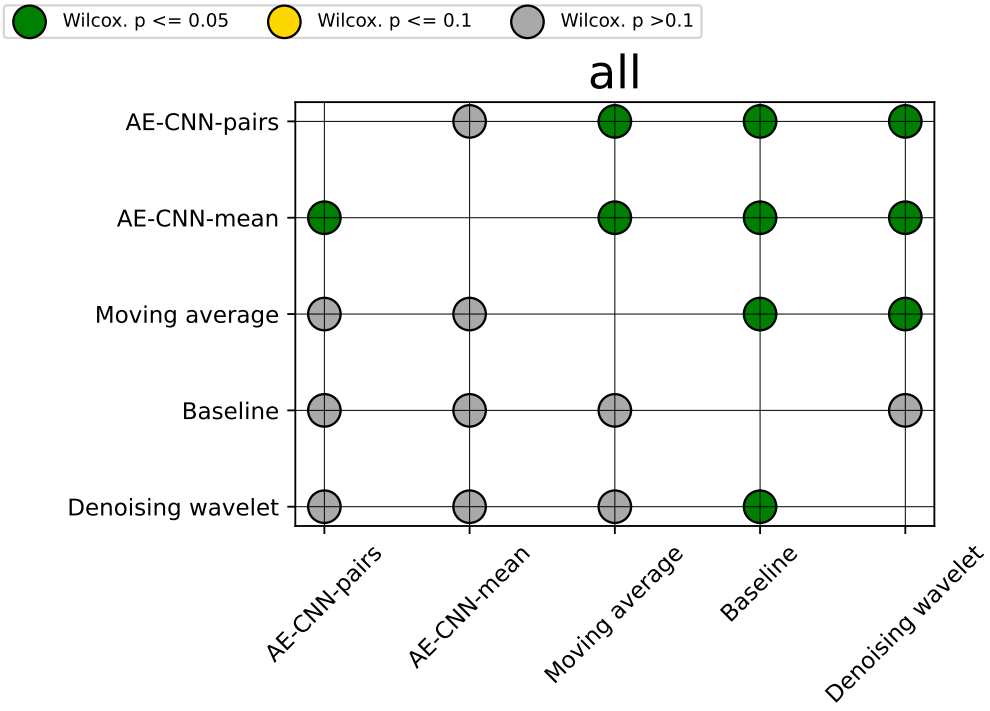

Figura 5.14: Test de Wilcoxon para todos los elementos.

En líneas generales, los métodos propuestos en este estudio son mejores a los métodos tradicionales de filtrado de ruido a nivel de tasa de error medido en la evaluación con el conjunto de datos del que se disponía. No obstante, estos resultados únicamente son posibles cuando se realiza el normalizado de las entradas entre 0 y 1.

A continuación, se detalla, de forma más breve, los resultados obtenidos por los modelos cuando han sido entrenados con más ruido en el entrenamiento y validado con menos ruido en el test y viceversa.

5.4.2. Muestras ruidosas vs. muestras menos ruidosas

Para poder efectuar esta experimentación, se extrajeron nuevos datos de urea con más y menos contribución de ruido y con las mismas concentraciones mencionadas: 5, 50 y 500. Sin embargo, en esta ocasión no se pudo ejecutar una validación cruzada debido a la falta de tiempo, por lo que no se han podido obtener los gráficos de violín.

La hipótesis de partida consistía en que partiendo de un conjunto de entrenamiento considerablemente más ruidoso que el de test, los métodos de aprendizaje automático fueran capaces de realizar un mejor filtrado que realizándolo a la inversa.

En la figura 5.15 se pueden apreciar los RMSE obtenidos para cada uno de los modelos cuando se efectúa el entrenamiento con muestras más ruidosas.

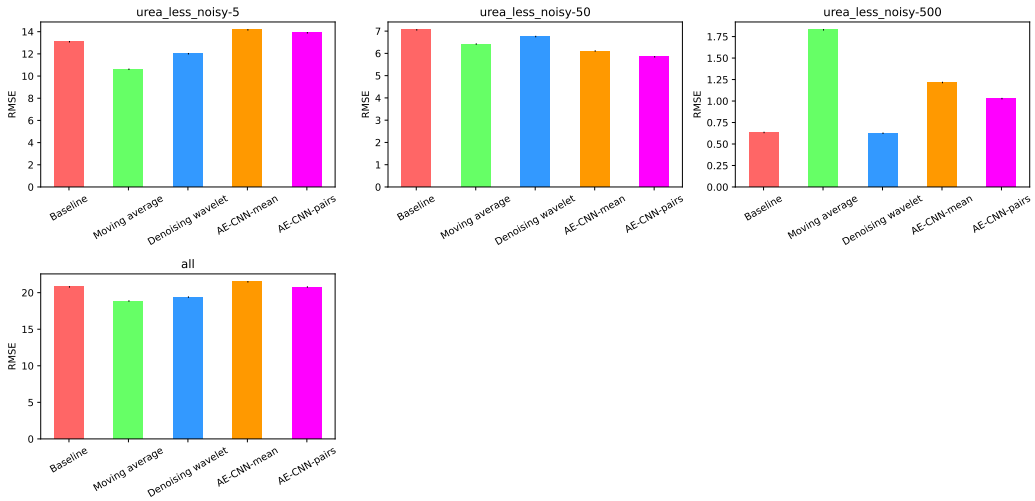

Figura 5.15: RMSE en barras con datos normalizados y ruido en el entrenamiento.

Como se puede visualizar en los gráficos, los métodos de aprendizaje automático son considerablemente peores que los métodos tradicionales. Esto puede ser debido a que las redes neuronales están entrenando con una distribución característica de los datos para, ulteriormente, evaluarlas con una distribución distinta.

A continuación, en la figura 5.16, se muestran los resultados para los modelos cuando se realiza un entrenamiento con menos ruido que el test.

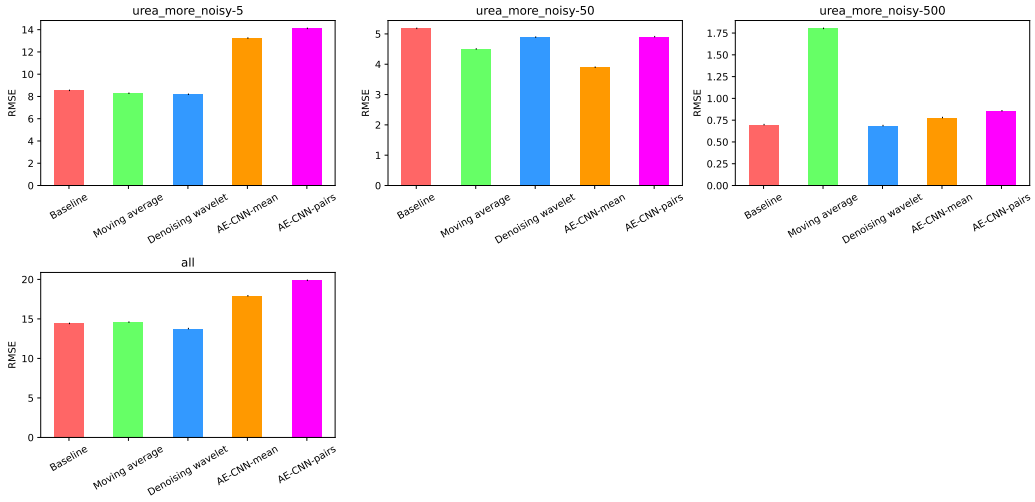

Figura 5.16: RMSE en barras con datos normalizados y ruido en el entrenamiento en test.

Tal y como se puede ver, con los datos normalizados y mayor presencia de ruido en el entrenamiento, los modelos son incluso peores que en el caso anterior. No solo es

evidente que les perjudica entrenar con un conjunto de datos menos ruidoso que en el test, sino que además, resulta sencillo deducir que alterar la distribución de los datos en la experimentación perjudica en gran medida a las redes neuronales.

Por último, se muestra a continuación como se han comportado los modelos a la hora de realizar el filtrado sobre la muestra original. En el primer caso (más ruido en el entrenamiento), se aprecia en la figura 5.17 unos ejemplos de este filtrado.

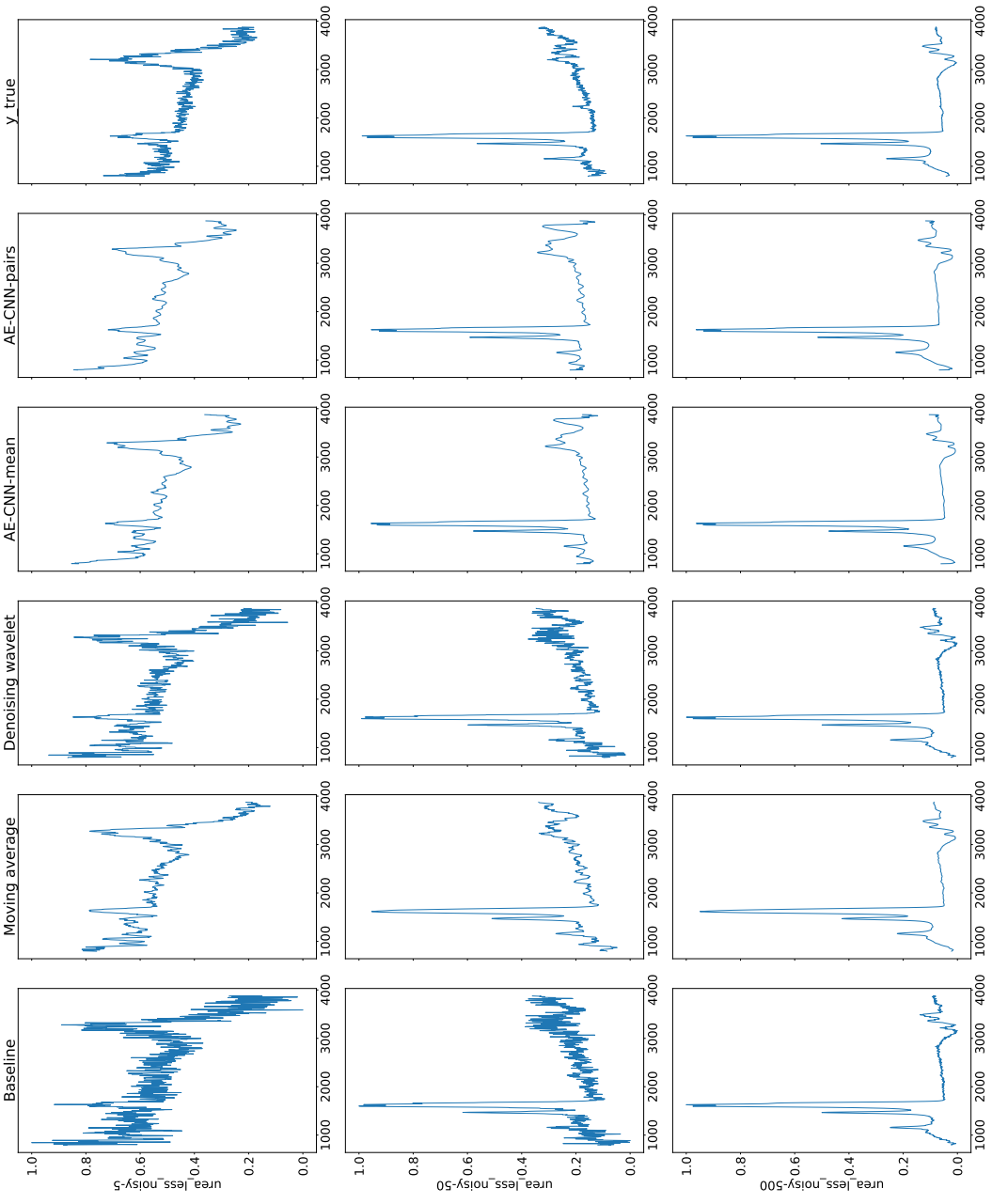

Figura 5.17: Ejemplos de filtrado con datos normalizados en urea más ruidosa en entrenamiento.

Los filtrados nos muestran que, cuando se entrena con más ruido que en el test, los filtros realizados por los modelos de aprendizaje automático no son demasiado efectivos, generando picos incongruentes con la señal original y con técnicas más simples como por ejemplo la MA. En concreto, en urea-5 (primera fila) en la banda que va desde los $2000Hz - 3000Hz$ tanto el modelo basado en agrupación según la media y por pares generan una serie de picos extraños que no se identifican en la señal original. Esto mismo, aunque en menor medida, se puede observar parar el grupo urea-50 (segunda fila de gráficos).

En el segundo caso (menos ruido en el entrenamiento), se observa en la figura 5.18 un ejemplo.

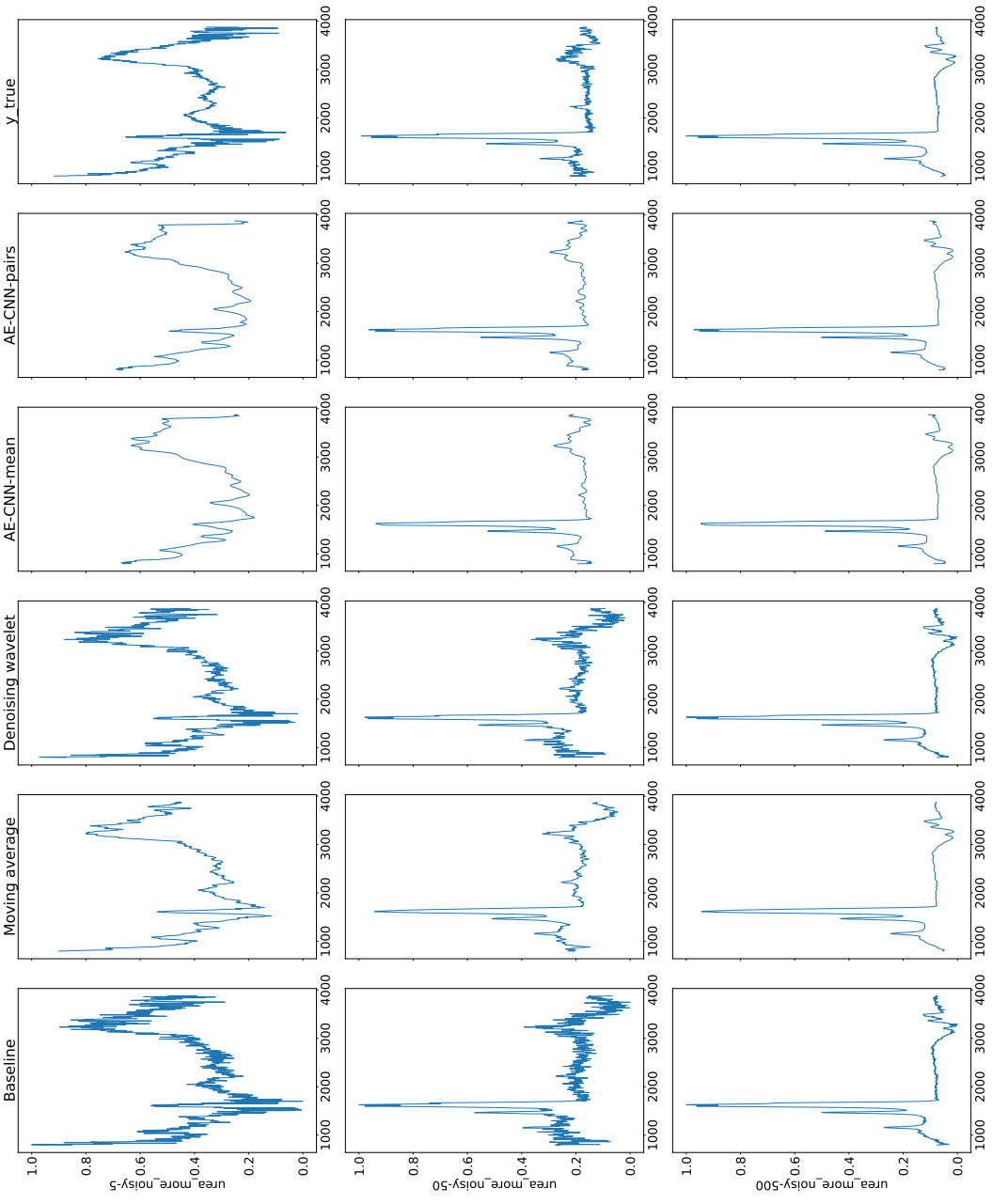

Figura 5.18: Ejemplos de filtrado con datos normalizados en urea menos ruidosa en entrenamiento.

Los filtrados realizados cuando se entrena con menos ruido en el entrenamiento, pueden considerarse iguales o incluso ligeramente peores que los métodos tradicionales, ya que las diferencias respecto a la señal que espera son mucho mayores. Por ejemplo, para el grupo urea-5, en la banda entre $3000\,\mathrm{cm}^{-1}$ - $4000\,\mathrm{cm}^{-1}$ se puede observar como los modelos de aprendizaje profundo generan una serie de picos donde ni en la MA ni en la señal esperada se puede identificar. Así mismo, este suceso se refleja también en la banda de los $2000\,\mathrm{cm}^{-1}$ - $3000\,\mathrm{cm}^{-1}$.

No obstante, se debe tener presente que en una aplicación real es posible que la red se encuentre con unos datos más ruidosos porque tengan interferencia de agua. Es por ello que, con menos ruido en el entrenamiento y normalizando los datos, la red no se ha comportado tan mal, porque ha sido capaz de eliminar el ruido del agua. El problema reside en que no se dispone de una señal real con la que validar el resultado.

Capítulo 6

Discusión

Basándose en los resultados obtenidos en la sección 5 Experimentación, queda a la vista que las redes neuronales se ven perjudicadas cuando se les introduce datos no normalizados. Bajo esta condición, los métodos tradicionales ofrecen un buen rendimiento a la hora de realizar el filtrado en todas las concentraciones, destacando sobre las redes neuronales. Sin embargo, esta diferencia no es tan grande cuando se trata con concentraciones altas. Esto es debido a que, cuando un elemento está presente en mayor concentración sobre el disolvente, la contribución de ruido sobre el espectro final es menor, por lo tanto, la cantidad de ruido a eliminar también es menor.

Por otro lado, cuando las redes neuronales reciben los datos normalizados en el rango 0-1, ofrecen muy buenos resultados tanto a nivel de métricas como cualitativo. Este hecho es apreciable en las figuras 5.10 y 5.11, ya que se puede verificar como el RMSE ofrecido por los métodos de aprendizaje automático es considerablemente inferior a los métodos tradicionales y, de la misma manera, el filtrado realizado también es correcto.

Así mismo, también es evidente que las redes neuronales no son capaces de realizar un filtrado tan correcto como en el primer caso cuando el dominio de los datos se ve alterado entre el entrenamiento y el test. El hecho de cambiar la cantidad de ruido del conjunto de datos entre entrenamiento y test, afecta a la forma que tienen los modelos de aprendizaje automático de aprender los patrones de ruido sobre los datos e inferirlos. No obstante, sería conveniente realizar la experimentación oportuna con una validación cruzada para este caso, ya que dicha prueba no se ha podido abarcar en este estudio.

Capítulo 7

Conclusiones y trabajos futuros

En nuestra hipótesis inicial, planteábamos los métodos de aprendizaje automático como una herramienta a estudiar para el filtrado de ruido presente en un espectro. A pesar de que las redes neuronales han sido ampliamente probadas en la labor del filtrado de ruido en otros tipos de datos, tales como señales e imágenes, en este estudio y los datos de partida no disponíamos de unas señales puras verdaderas (denominado en inglés *ground truth*). Por tanto, tras el análisis realizado, podemos concluir que siempre y cuando los datos se introduzcan de forma normalizada a los modelos y el dominio de los datos no se vea alterado, las redes neuronales (en nuestro estudio los autoencoders) ofrecerán un comportamiento igual o mejor a los métodos tradicionales.

No obstante, podemos considerar que el conjunto de datos de partida es un conjunto reducido y puede no ser representativo de lo que un modelo de aprendizaje automático podría encontrar en un entorno de producción real. De este modo, este trabajo deja abiertas experimentaciones adicionales, tales como incorporar nuevas sustancias químicas en nuevas concentraciones, o bien, ampliar el conjunto de datos inicial, ejecutando nuevas pruebas sistemáticas en diferentes días para incorporar en los datos una representatividad mayor del ruido involuntariamente incorporado al espectro. Es por ello que, un conjunto de datos representativo para resolver este problema es complejo y costoso de obtener, ya que depende de variables que escapan al control humano.

Bibliografía

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017a). Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017b). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

Bank, D., Koenigstein, N., and Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Bhanja, S. and Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting. *arXiv preprint arXiv:1812.05519*.

Brito, N., Souza, B., and Pires, F. (1998). Daubechies wavelets in quality of electrical power. In *8th International Conference on Harmonics and Quality of Power. Proceedings (Cat. No.98EX227)*, volume 1, pages 511–515 vol.1.

Caiafa, C. F. and Lew, S. E. (2020). ¿ qué es la inteligencia artificial?

Cichy, R. M. and Kaiser, D. (2019). Deep neural networks as scientific models. *Trends in cognitive sciences*, 23(4):305–317.

Cruz, I. B., Martínez, S. S., Abed, A. R., Ábalo, R. G., and Lorenzo, M. M. G. (2007). Redes neuronales recurrentes para el análisis de secuencias. *Revista Cubana de Ciencias Informáticas*, 1(4):48–57.

Cupertino, A. F. and Pereira, H. A. (2021). Next generation of grid-connected photovoltaic systems: modeling and control. *Design, Analysis, and Applications of Renewable Energy Systems*, pages 509–548.

Díaz, N. A., Ruiz, J. A. B., Reyes, E. F., Cejudo, A. G., Novo, J. J., Peinado, J. P., Meléndez-Valdés, F. T., and Fiñana, I. T. (2010). Espectrofometría: Espectros de absorción y cuantificación colorimétrica de biomoléculas. *Universidad de Córdoba*, pages 1–8.

Fuchs, J., Dubey, A., Lübke, M., Weigel, R., and Lurz, F. (2020). Automotive radar interference mitigation using a convolutional autoencoder. In *2020 IEEE International Radar Conference (RADAR)*, pages 315–320. IEEE.

Gallego, A.-J., Calvo-Zaragoza, J., and Rico-Juan, J. R. (2020). Insights into efficient k-nearest neighbor classification with convolutional neural codes. *IEEE Access*, 8:99312–99326.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE.

González, G. (1997). Series de fourier, transformadas de fourier y aplicaciones. *Divulgaciones matemáticas*, 5(1/2):43–60.

Guiñón, J. L., Ortega, E., García-Antón, J., and Pérez-Herranz, V. (2007). Moving average and savitzki-golay smoothing filters using mathcad. *Papers ICEE*, 2007:1–4.

Harris, D. C. (942). *Análisis químico cuantitativo (3a. ed.)*, volume 14.

Izaurieta, F. and Saavedra, C. (2000). Redes neuronales artificiales. *Departamento de Física, Universidad de Concepción Chile*.

Maino, D. G. (2013). Predicción de sistemas dinámicos con redes neuronales profundas. B.S. thesis, Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional ....

MATLAB (2022). *Wavelet Denoising (R2022a)*. The MathWorks Inc., Natick, Massachusetts.

Ongsulee, P. (2017). Artificial intelligence, machine learning and deep learning. In *2017 15th International Conference on ICT and Knowledge Engineering (ICT&KE)*, pages 1–6. IEEE.

Ordóñez, J. L. (2012). Espectro electromagnético y espectro radioeléctrico. *Manual formativo de ACTA*, (62):17–31.

Pan, Q., Zhang, L., Dai, G., and Zhang, H. (1999). Two denoising methods by wavelet transform. *IEEE Transactions on Signal Processing*, 47(12):3401–3406.

Pasti, L., Walczak, B., Massart, D., and Reschiglian, P. (1999). Optimization of signal denoising in discrete wavelet transform. *Chemometrics and intelligent laboratory systems*, 48(1):21–34.

Pérez-Ortiz, J. A. (2002). Modelos predictivos basados en redes neuronales recurrentes de tiempo discreto.

Piqué, T. M. and Vázquez, A. (2012). Uso de espectroscopía infrarroja con transformada de fourier (ftir) en el estudio de la hidratación del cemento. *Concreto y cemento. Investigación y desarrollo*, 3(2):62–71.

Rico-Juan, J. R. and Calvo-Zaragoza, J. (2015). Improving classification using a confidence matrix based on weak classifiers applied to ocr. *Neurocomputing*, 151:1354–1361.

Rojas Monsalvo, K. et al. (2013). Radiación electromagnetica.

Seyfioğlu, M. S., Özbayoğlu, A. M., and Gürbüz, S. Z. (2018). Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):1709–1723.

Staudemeyer, R. C. and Morris, E. R. (2019). Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*.

Vargas Cañas, Rubiel; Loaiza Correa, H. (2011). Filtrado de señales en espectrofotometría de absorción mediante wavelets invariantes a la traslación. *Ingeniería e Investigación*.

Wang, S.-C. (2003). Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer.

Wilcoxon, F. (1992). Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Wu, Y.-c. and Feng, J.-w. (2018). Development and application of artificial neural network. *Wireless Personal Communications*, 102(2):1645–1656.

Yu, Y., Si, X., Hu, C., and Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270.

Zhang, D. (2019). Wavelet transform. In *Fundamentals of Image Data Mining*, pages 35–44. Springer.

Zhang, G., Liu, Y., and Jin, X. (2020). A survey of autoencoder-based recommender systems. *Frontiers of Computer Science*, 14(2):430–450.